Assignment 3 - Parser

Saul Garcia

January 10, 2016

Objective

The objective of this assignment is to build a Syntactic Parser by using Prolog language and Definite Clause Grammars (DCG), which serves as an interpreter for a few `Unix` commands. The commands to parse are: **cal**, **cat**, **cp** and **grep**.

1. Express the syntax of the different commands by utilizing the *BNF formalism*

```
Operation
<operation> ::= "cal" <arg_calendar> | "cat" <arg_concatenate> |
                "cp" <arg_copy> | "grep" <arg_searchexpr>

Arguments calendar
<arg_calendar> ::= <arg_month> <arg_year> | <arg_year> | ""
<arg_month> ::= <integer (1...12)>
<arg_year> ::= <integer (1...9999)>

Argument Concatenate
<arg_concatenate> ::= <arg_options> <arg_files> | <arg_files>
<arg_options> ::= <special><character> | <special><character> <special><character> ...
<arg_files> ::= <files> | <files>*
<special> ::= -
<character> ::= n | b | s | u | v | e | t
<files> ::= <atom> | <atom>*
<atom> ::= <lowercase letter> | <small atom> <character>
<lowercase letter> ::= a | b | c | ... | x | y | z
<uppercase letter> ::= A | B | C | ... | X | Y | Z | _

Argument Copy
<arg_copy> ::= <arg_options>* | <arg_files>
<arg_options> ::= <special><character> | <special><character>*
<arg_files> ::= <files> | <files>*
<special> ::= -
<character> ::= r | R | f | i | p
<files> ::= <atom> | <atom>*
<atom> ::= <lowercase letter> | <small atom> <character>
<lowercase letter> ::= a | b | c | ... | x | y | z
<uppercase letter> ::= A | B | C | ... | X | Y | Z | _

Argument Search Expr
<arg_searchexpr> ::= <arg_options>* <arg_e> <files> <arg_files> |
               <arg_options>* <files> <arg_files> |
                       <files> <arg_files> | <arg_e> <files> |
                       <arg_options>* <files> | <files>
<arg_options> ::= <special><character> | <special><character>*
<special> ::= -
```

```
<character> ::= b|c|i|h|l|n|v|s|y
<arg_e> ::= <special><atom>
<arg_files> ::= <files> | <files>*
<files> ::= <atom> | <atom>*
<atom> ::= <lowercase letter> | <small atom> <character>
```

2. read_command(C): Reads a line on the current input stream and that returns the list of ascii codes it contains.

The code can be found in the document: *garciaCalderon.pl*

ASCII code

```
?- read_command(C).
|: Hello world
C = [72, 101, 108, 108, 111, 32, 119, 111, 114|...].
```

3. Prolog: Consider the commands *cal*, *cat*, *cp* and *grep*. Write the commands *send(input,file)*, *append(input,file)*.

Calendar

The Calendar command it has 3 different scenarios:

```
#Input Month and Year
?- read_command(C).
|: cal 1 99
C = calendar(1, 99).

#Input Year
?- read_command(C).
|: cal 1000
C = calendar(1000).

#Expect to get current Month and Year
?- read_command(C).
|: cal
C = calendar(1, 2016).
```

Concatenate

```
#Concatenate the options and files
?- read_command(C).
|: cat -n -b file1 file2
C = concatenate([n, b], [file1, file2]).

#One of the options do not belong to the list.
?- read_command(C).
|: cat -n -z file1 file2
false.
```

Copy

```
?- read_command(C).
|: cp -r -R file file2 target
C = copy([r, 'R'], [file, file2], target)

?- read_command(C).
|: cp -r -R file
false.
```

Search_Expr

```
?- read_command(C).
|: grep -bci -e exp file1 file2
C = search_exp([b, c, i], e, exp, [file1, file2]) .

?- read_command(C).
|: grep -e file1 file2
C = search_exp([], e, file1, [file2]).
```

Send

```
?- read_command(X).
|: cat -b file1 file2 >file
X = send(concatenate([b], [file1, file2]), file)
```

Append

```
?- read_command(X).
|: cat -b file1 file2 >>file
X = append(concatenate([b], [file1, file2]), file) .
```

4. Rewrite the code for DCG formalism. ***read_dcg(X).***

Calendar

```
?- read_dcg(X).
|: cal 1 99
X = calendar(1, 99).

?- read_dcg(X).
|: cal 2000
X = calendar(2000).

?- read_dcg(X).
|: cal
X = calendar(1, 2016).
```

Concatenate

```
#Concatenate the options and files
?- read_dcg(C).
|: cat -n -s file1 file2
C = concatenate([n, b], [file1, file2]).

#One of the options do not belong to the list.
?- read_dcg(C).
|: cat -n -z file1 file2
false.
```

Copy

```
?- read_command(C).
|: cp -r -R file file2 target
C = copy([r, 'R'], [file, file2], target)

?- read_command(C).
|: cp -r -R file
false.
```

Search_Expr

```
?- read_dcg(X).
|: grep -n -e exp file >>file
X = append(search_exp([n], e, exp, [file]), file) .

?- read_dcg(X).
|: grep -b
false.
```

Send and Append

```
?- read_dcg(X).
|: grep -b -e file1 file2 >file
X = send(search_exp([b], e, file1, [file2]), file) .

?- read_dcg(X).
|: grep -b -e file1 file2 >>file
X = append(search_exp([b], e, file1, [file2]), file)
```

Code

The code will be found in appended document.