NOTE: THIS IS A MARKDOWN FILE, BEST VIEWED IN THE INCLUDED PDF, OR A MARKDOWN VIEWER.

Project 2: Part 2, Scheme

By: Kevin Bohinski & Brittany Reedman

11/2/2016

CSC 435 Programming Languages

Readme File

Files

Within the zip there should be three files:

```
\-p2p2.zip
    | p2p2.scm
    | readme.md
    | readme.pdf
```

`p2p2.scm` contains our source, and `readme.md` or `readme.pdf` contains this readme.

Running the Project

- Unzip the folder
- Load p2p2.scm in your scheme (R5RS) interpreter of choice
- Run `(parse-table calc-gram)`
- Done! Output should be printed.

Input & Output

Our file requires one to invoke a function to kickstart it, there is no main function that is ran on launch. Please run `(parse-table calc-gram)` where `calc-gram` is a valid grammar per spec. From here, one should see the predict sets printed per spec.

Positives

- Project is complete and is done so without the skeleton code.
- We worked well together on this project!
- We learned so much about scheme and functional programming during implementation.
- We didn't need the additional extension!

Limitations and Bugs

- Bugs occur when the grammar is not formatted to spec.

Conventions

- We attempted to follow the conventions found in the scheme docs as well as the textbook.
- We followed the schemedoc commenting style, so one can generate a HTML page with documentation.

Abstractions

We took advantage of abstractions in two contexts, nested functions, and helper functions. By splitting up the code in chunks this way it was simpler to develop.

Data Structures

We made extensive use of lists, functions, booleans, and strings.

Lists

We used lists to store and organize data. There is significant nesting of lists in this project.

Functions

Functions power this project, as the code is distributed through them.

Booleans

Booleans are occassionally used as flags.

Strings

The grammar is represented as strings.

Algorithms

We followed the algorithms for first and follow sets mainly from the textbook. They are as follows:

First Set Algorithm

1. First(t)={t}, where t is a terminal
2. epsilon is an element of First(X)

- If X -> epsilon
- If X -> A_1 ... A_n and epsilon is an element of First(A_i) for 1 <= i <= n

1. First(alpha) is a subset of First(X) if

- X -> A_1 ... A_n alpha, and
- epsilon is an element of First(A_i) for 1 <= i <= n

Follow Set Algorithm

1. $ is an element of Follow(S)
2. First(beta) - {epsilon} is a subset of Follow(X), for each production A -> alpha X beta

3. First(A) is a subset of Follow(X), for each production A -> alpha X beta, epsilon is an element of where First(beta)

Predict Set Algorithm (from https://www.usna.edu/Users/cs/roche/courses/f11si413/c10/ff.pdf)

Predict(A) contains:

- Any token that appears as the leftmost symbol in a production rule for A.
- For every non-terminal B that appears as the leftmost symbol in a production rule for A, every token in Predict(B).
- If A goes to epsilon, every token in Follow(A).