Project 2: Grammar Analysis and Parsing

S. Patel, J. Collard, M. Barney

April 6, 2013

1 Context Free Grammar

In this section we provide the context free grammar data type.

At its heart, a grammar it consists of a list of productions, where each production consists of a constructor and two arguments; the first a paramaterized nonterminal, and the second a paramaterized right hand side.

An *RHS* is either empty, a terminal, which takes two arguments — the paramaterized object representing a terminal, and another *RHS*; or a nonterminal, which similarly takes two arguments.

```
{-# LANGUAGE FlexibleInstances #-}
module ContextFreeGrammar (Grammar, Production (..), RHS (..)) where
type Grammar nt t = [Production nt t]
data Production nt t = Production {nonterminal :: nt,
  rhs :: RHS nt t}
instance Show (Production String String) where
  show (Production nt rhs) = nt ++ " -> " ++ show rhs
data RHS nt t = Empty
  | Term t (RHS nt t)
  | NonT nt (RHS nt t)
instance Show (RHS String String) where
  show Empty = ""
  show (Term t rhs) = t ++ (show rhs)
  show (NonT nt rhs) = nt ++ (show rhs)
simpleGrammar :: Grammar String String
simpleGrammar = [a, b, c, d] where
  a = Production "A" (Term "a" Empty)
  b = Production "B" (NonT "B" Empty)
```

```
c = Production "C" (Term "a" (NonT "B" Empty))
d = Production "D" (NonT "B" (Term "a" Empty))
```

2 Scanner and Parser for context-free grammars

In this section we provide code for scanning and parsing a textual representation of a context free grammar.

The concrete representation is as follows:

```
STUFF
```

```
module ScannerAndParser where
function = ⊥
```

3 Main module

The main module puts everything together, takes an textual representation of a context-free grammar as input, scans, parses, and performs the rest of the duties that are required.

```
module Main where
import ContextFreeGrammar
import ScannerAndParser
  -- import BadHygiene
import System.Environment
main = do
  putStrLn "hello"
```