Project 2: Grammar Analysis and Parsing

S. Patel, J. Collard, M. Barney

April 6, 2013

```haskell
{-# LANGUAGE FlexibleInstances #-}
module ContextFreeGrammar 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))
```

1 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 = ⊥
```

```
module Main where
import ContextFreeGrammar
import ScannerAndParser
main = putStrLn "hello"
```