CS454 Project 1:
« Lexer Generator »

M. Barney, J. Conrad, and S. Patel

March 11, 2013

1 Introduction

Implementation of a lexer generator in Haskell.

```
module FiniteStateAutomata (Transition (),
  Node (),
  FSA (),
  newTransition,
  newNode,
  newFSA) where
data Transition a = Transition {getLabel :: a, getNode :: Node a}
data Node a = Node {isAccepting :: Bool, getTransitions :: [Transition a]}
data FSA a = FSA {getStart :: Node a}
newTransition :: (Eq a, Show a) ⇒ a → Node a → Transition a
newTransition = Transition
newNode :: (Eq a, Show a) ⇒ Bool → [Transition a] → Node a
newNode = Node
newFSA :: (Eq a, Show a) ⇒ Node a → FSA a
newFSA = FSA
```

2 Regular Expressions

In this module we give the haskell data type for a regular expression; the encoding almost exactly mirrors the definition given in the assignment.

```
module Regex (Regex (..)) where

data Regex a = Alt (Regex a) (Regex a)
    | Concat (Regex a) (Regex a)
    | Repeat (Regex a)
    | Term a
    | Empty
```

3 Module: Main.lhs

```
module Main where

main =
  putStrLn "(( .x x) helloworld)"
```