CS536: Homework 3

Keith Funkhouser

September 30th, 2015

This homework is about the language of *regular expressions*. One way to define that language is using a context-free grammar (CFG). To make things a little simpler, rather than allowing any characters as operands in regular expressions, we will restrict ourselves to only allowing letters (that way we don't have to worry about how to specifiy characters that are the same as operators or things like newlines). As usual, we will allow ϵ in our regular expressions. The operators for our language of regular expressions are:

- | means "or" (alternation)
- writing two or more things next to each other means "followed by" (catenation)
- * means "zero or more" (closure or iteration)
- + means "one or more" (positive closure)
- ( ) are used for grouping

In a regular expression, * and + have the same, highest precedence, "followed by" has middle precedence, and | has the lowest precedence. All of the operators are left associative.

1

Write an unambiguous CFG for this language of regular expressions so that parse trees correctly reflect the precedences and associativities of the operators. Use lower-case names for nonterminals and use the following terminals:

```
LTR      // any letter
EPS      // epsilon
OR       // |
STAR     // *
PLUS     // +
LPAR     // left paren
RPAR     // right paren
```

Consider the CFG defined by the following 4-tuple:

- N: {regex, cat, closure, expr}
- Σ: {LTR, EPS, OR, STAR, PLUS, LPAR, RPAR}
- P:

$$\begin{aligned} \langle regex \rangle &::= \langle regex \rangle \ \langle OR \rangle \ \langle cat \rangle \\ &\ \ | \ \ \langle cat \rangle \end{aligned}$$

$$\begin{aligned} \langle cat \rangle &::= \langle cat \rangle \ \langle closure \rangle \\ &\ \ | \ \ \langle closure \rangle \end{aligned}$$

$\langle closure \rangle$ ::= $\langle expr \rangle$ $\langle STAR \rangle$
| $\langle expr \rangle$ $\langle PLUS \rangle$
| $\langle expr \rangle$

$\langle expr \rangle$ ::= $\langle LTR \rangle$
| $\langle EPSILON \rangle$
| $\langle LPAREN \rangle$ $\langle regex \rangle$ $\langle RPAREN \rangle$

- S: `r`egex

2

Draw a parse tree for the string:

`ab+|c*df|`ϵ

Use `LTR(a)` in the parse tree to mean "the `LTR` token for the letter a" (and similarly for the other letters in the string).