Brandon Zink
Cameron Connor

Lab 6 Write Up

i. re ::= union
union ::= intersect ‘|’ intersect | intersect
intersect ::= intersect & concat | concat
concat ::= intersect concat | concat
not ::= ~ not | star
star ::= start * | star + | star ? | atom
atom ::= c|+|#|&|*|’|’|(re)

ii. The BNF grammar can have infinite unions that can cause an infinite loop. Also, ‘re’ in atom can call the top again and cause infinite parenthesis.

iii. re ::= union
union ::= intersect {‘|’ intersect}
intersect ::= concat {‘&’ concat}
concat ::= not {not}
not ::= ‘~’ not | star
star ::= atom {‘*’|’+’|’?’}
atom ::= ‘!’|’#’|c|’.’|’(‘re’)’

iv. re ::= union
union ::= intersect unions
intersect ::= concat intersect
concat ::= not concat
not ::= star not | ~ not
star ::= atom sym1 | sym1
atom ::= sym2 | re

Sym1 = {‘*’|’+’|’?’}
Sym2 = {‘!’|’#’|c|’.’|’(‘re’)’|

v.

$$\frac{v_1 = /\backslash re\$/ \quad v_2 = str}{(M, v1.test(v2)) \text{-->} (M, b)} DoReg$$

$$\frac{(M, e1) \text{ --> } (M, e1')}{(M, e1.test(e2)) \text{ --> } (M', e1'.test(e2))} SearchTest1$$

$$\frac{e1 = re1 \quad (M, e2) \text{-->} (M, e2')}{(M, re1.test(e2)) \text{ --> } (M, re1.test(e2))} SearchTest2$$

$$\frac{}{\Gamma \perp /\backslash re\$/ : RegExp} TypeRegExpVal$$

$$\frac{\Gamma \perp e1 : RegExp \quad \Gamma \perp e2 : str}{\Gamma \perp e1.test(e2) : bool} TypeTest$$