Andrew Rutherford

CSCI 3155

Lab 6 Write-up

2. b. i.

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

2. b. ii.

A recursive descent parser of this grammar with left recursion would go into an infinite loop because the parser doesn't know when to end when parsing the nonterminal symbols.

2. b. iii.

re	::=	union
union	::=	intersect unions
unions	::=	ε | '|' intersect unions
intersect	::=	concat intersects
intersects	::=	ε | '&' concat intersect
concat	::=	not concats
concats	::=	ε | not concat
not	::=	'~' not | star
star	::=	atom stars
stars	::=	ε | '*' atom star
stars	::=	ε | '+' atom star
stars	::=	ε | '?' atom star
atom	::=	'!' | '#' | c | '.' | '('re')'

Lab 6 Inference Rules

No String

$$\frac{r = !}{r.test(s) \rightarrow false}$$

Empty String

$$\frac{r = \#\quad s = ""}{r.test(s) \rightarrow true}$$

NonEmpty String

$$\frac{r = \#\quad s \neq ""}{r.test(s) \rightarrow false}$$

Any Character

$$\frac{s = ""}{r.test(s) \rightarrow false}$$

Single Character

$$\frac{r = c\quad s = c :: t}{r.test(s) \rightarrow true}$$

Single Character Search

$$\frac{r = c\quad s = h :: t}{r.test(s) \rightarrow r.test(t)}$$

Concat

$$\frac{s = B' :: E'\quad E' = s' :: \_\quad r = concat(r_1, r_2)\quad r_1.test(s' :: t') = true\quad r_2.test(t' :: \_)}{r.test(s) \rightarrow}$$

Union

$$\frac{r = r_1 | r_2\quad b' = r_1.test(s) \;||\; r_2.test(s)}{r.test(s) \rightarrow b'}$$

Star False

$$\frac{r = r'*\quad |r'*| >= |s|}{r.test(s) \rightarrow false}$$

Star True

$$\frac{r = r'*\quad |r'*| < |s|\quad s = h :: t\quad r' = h}{r.test(s) \rightarrow true}$$

Star Recurse

$$\frac{r = r'*\quad |r'*| < |s|\quad s = h :: t\quad r' \neq L\quad B' = r.test(t)}{r.test(s) \rightarrow B'}$$

AnyChar

$$\frac{r = .\quad B' = (s \neq "")}{r.test(B) \rightarrow B'}$$

Option

$$\frac{r = re1+\quad r' = (re1)(re1*)\quad B' = r'.test(s)}{r.test(s) \rightarrow B'}$$

Intersection

$$\frac{r = re1 \,\&\, re2\quad B' = (re1.test(s) \,\&\, re2.test(s))}{r.test(s) \rightarrow B'}$$

Type Evaluation:

Type RegExp

$$\frac{}{\Gamma \vdash /\hat{}\ re\$/ : \text{Regexp}}$$

Type RegExp Test

$$\frac{e_1 : \text{RegExp} \qquad e_2 : \text{String}}{\Gamma \vdash e_1.\text{test}(e_2) : \text{Boolean}}$$