Lab 6

Yuxiang Wang

Principles of Programming Languages

CSCI 3155-Fall 2017

3. Part B.

(I.) Exercise. In your write-up, give a refactored version of the regrammar from ?? that eliminates ambiguity in BNF (not EBNF). Use the following template for the new non-terminal names:

ANS:

re ::= union

union ::= union '|' intersect | intersect

intersect ::= intersect '&' concat | concat

concat ::= concatnot | not

not ::= '~' not | star

star ::= star'*' | star'?' | star'+' | atom

atom ::= c | '#' | '!' | '.' |'('re')'

Lab 6

(II.) Exercise. Explain briefly why a recursive descent parser following your grammar with left recursion would go into an infinite loop.

ANS:

If parsing the nonterminal symbols, the parser wouldn’t know what time should stop, that mean left side cannot match anything because its stuck in a infinite loop when the grammar is left recursive. The right side match on nothing.

(III.) Exercise. In your write-up, give a refactored version of the re grammar that re- places left-associative binary operators with n-ary versions using EBNF using the following template:

ANS:

re ::= union

union ::= intersect { ‘|’ intersect }

intersect ::= concat { '&' concat }

concat ::= not { concatnot }

not ::= ‘~’ { ‘*’star }

star ::= atom { '*' | '+' | '?' }

atom ::= '!' | '#' | '.' | c | '('re')'

(IV.) In your write-up, give the full refactored grammar in BNF without left recursion and new non-terminals like unions for lists of symbols. You will need to introduce new terminals for intersects and so forth.

ANS:

re ::= union

union ::= intersect unions

unions ::= epsilon | '|' intersect unions

intersect ::= concat intersects

intersects ::= epsilon | '&' concat intersects

concat ::= not concats

concats ::= epsilon | notconcats

not ::= star | '~' not

nots ::= episilion | ‘~’ nots

star ::= atom stars

stars ::= epsilon | '*' atom stars | ʹ+ʹ atom stars | ʹ?ʹ atom stars

atom ::= 'c' | '#' | '!' | '.' | '('re')'

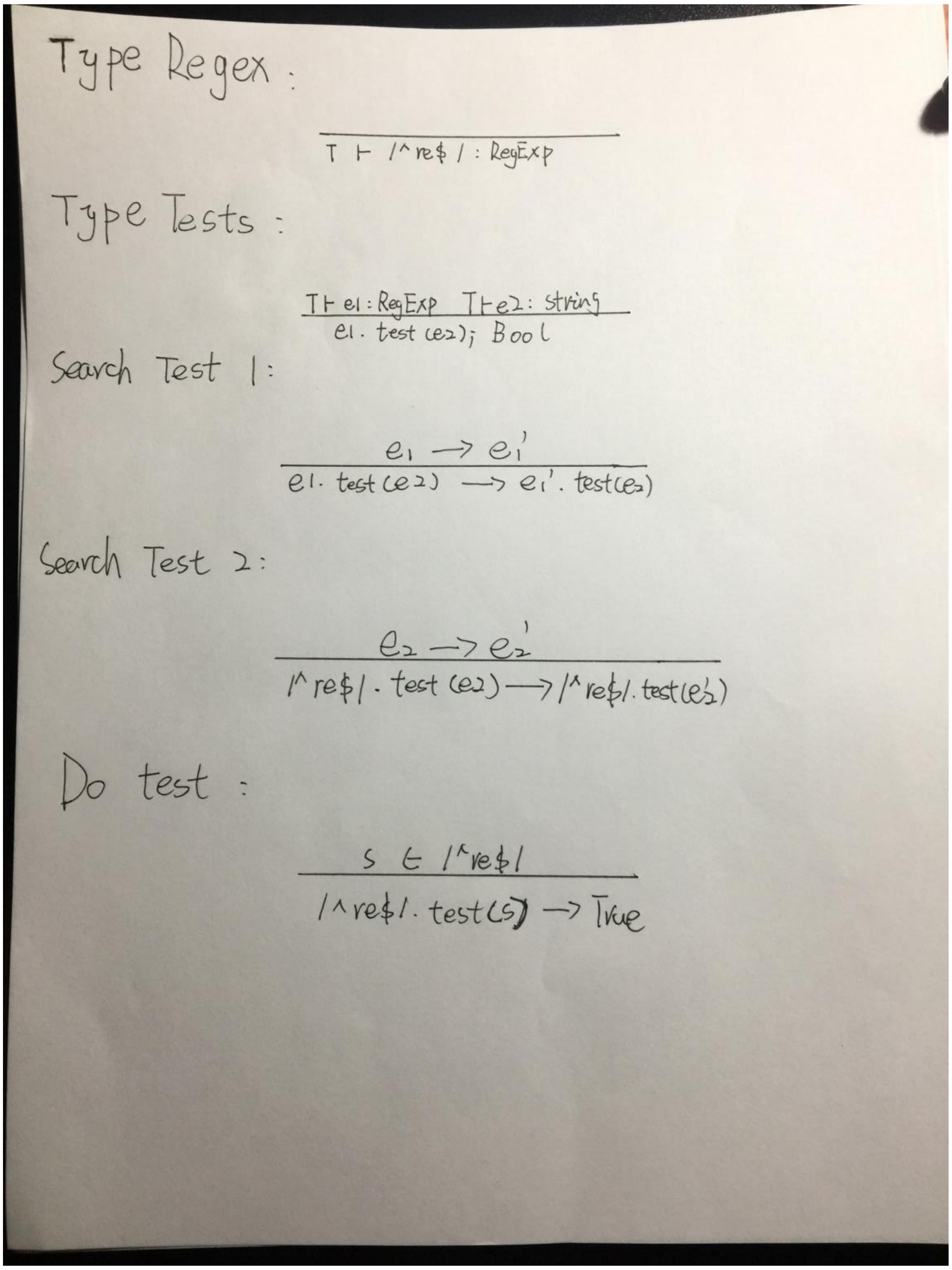
Part C.

Lab 6

I. In your write-up, give typing and small-step operational semantic rules for regular expression literals and regular expression tests based on the informal specification given above. Clearly and concisely explain how your rules enforce the constraints given above and any additional decisions you made.

ANS:

Lab 6



Type Regex:

$$\frac{}{T \vdash /{}^\wedge re\$/ : RegExp}$$

Type Tests:

$$\frac{T \vdash e1 : RegExp \quad T \vdash e2 : string}{e1 . test(e2); \ Bool}$$

Search Test 1:

$$\frac{e_1 \rightarrow e_1'}{e1 . test(e2) \rightarrow e_1' . test(e_2)}$$

Search Test 2:

$$\frac{e_2 \rightarrow e_2'}{/{}^\wedge re\$/ . test(e2) \rightarrow /{}^\wedge re\$/ . test(e_2')}$$

Do test:

$$\frac{s \in /{}^\wedge re\$/}{/{}^\wedge re\$/ . test(s) \rightarrow True}$$