Homework 6

Friday, November 15, 2019 7:23 PM

Q1: (i) Give the regular expressions for lexing a language consisting of whitespaces, identifiers (some letters followed by digits), numbers, operations =, < and >, and the keywords if, then and else.

(ii) Decide whether the following strings can be lexed in this language?

(a) "if y4 = 3 then 1 else 3"
(b) "if33 ifif then then23 else else 32"
(c) "if x4x < 33 then 1 else 3"

In case they can, give the corresponding token sequences. (Hint: Observe the maximal munch rule and priorities of your regular expressions that make the process of lexing unambiguous.)

A1:
(i)

keywords: if + then + else
whitespace: (_ + \n + \t)$^+$
number: 0 + [1 + 2 + .. + 9] • [0 + 1 + 2 + … + 9]*
identifier: [a-zA-Z]$^+$ • [0-9]*
operation: = + < + >

language: (keywords + whitespace + number + identifier + operation)*

(ii)

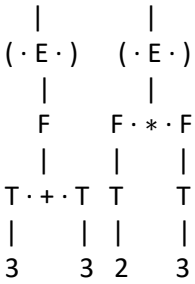
(a) (k,if), (w," "), (id,y4), (w," "), (op,=), (w," "), (n,3), (w," "), (k,then), (w," "), (n,1), (w," "), (k,else), (w," "), (n,3)
(b) (id,if33), (w," "), (id,ifif), (w," "), (k,then), (w," "), (id,then23), (w," "), (k,else), (w," "), (k,else), (w," "), (n,32)
(c) (k,if), (w," "), (id,x4x), (w," "), (op,<), (w," "), (n,33), (w," "), (k,then), (w," "), (n,1), (w," "), (k,else), (w," "), (n,3)

Q2: Suppose the grammar

E → F | F · ∗ · F | F · \ · F
F → T | T · + · T | T · - · T
T → num | (· E ·)

where E, F and T are non-terminals, E is the starting symbol of the grammar, and num stands for a number token. Give a parse tree for the string (3+3)+(2*3).

A2:



(this took way too long to do, I should have drawn it by hand :/)

Q3: Define what it means for a grammar to be ambiguous. Give an example of an ambiguous grammar.

A3: A grammar is ambiguous when there is more then one valid derivation tree for a certain string.

E --> if E then E
--> if E then E else E

with string "if x then if y then z else w"

Q4: Suppose boolean expressions are built up from

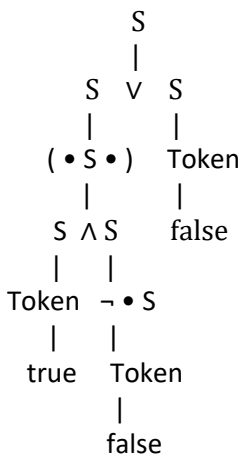
1.) tokens for true and false,
2.) the infix operations ∧ and ∨,
3.) the prefix operation ¬, and
4.) can be enclosed in parentheses.

(i) Give a grammar that can recognise such boolean expressions and
(ii) give a sample string involving all rules given in 1.-4. that can be parsed by this grammar.

A4:

S --> (• S •) | ¬ • S | S ∧ S | S ∨ S | Token
Token --> true | false

String: (true ∧ ¬false) ∨ false



Q5: Given the regular expressions

1) (ab + a) · (1 + b)
2) (aa + a)*

there are several values for how these regular expressions can recognise the strings (for 1) *ab* and (for 2) *aaa*. Give in each case all the values and indicate which one is the POSIX value.

A5:

1) Sequ(Left(Sequ(Chr(a), Chr(b))), Left(Empty)) <- POSIX compliant
   Sequ(Right(a), Right(b))

1) Stars(List(Left(Sequ(Chr(a),Chr(a))), Right(Chr(a)))) <--- POSIX compliant
2) Stars(List(Right(Chr(a)), Left(Sequ(Chr(a),Chr(a))))
3) Stars(List(Right(Chr(a)), Right(Chr(a)), Right(Chr(a))))

Q6: Parsing combinators consist of atomic parsers, alternative parsers, sequence parsers and semantic actions. What is the purpose of (1) atomic parsers and of (2) semantic actions?

A6: Atomic parsers recognize some part from the start of the input sequence and ignore the rest. Semantic actions take output processed by some parser and then apply some function to it, with the intent
of extracting some of the meaning from behind the parsed output.