

# CA215 Languages and Computability

## Autumn 2008

Attempt **three** questions. All questions carry equal marks.

Q 1.

(i) Define the constraints on  $\alpha$  and  $\beta$  in a finite-state rewrite rule of the form  $\alpha \rightarrow \beta$ .

(ii) Consider the following grammar:

$\langle V_t = \{\text{kick, talk, sing, er, s, table, laptop}\},$

$V_n = \{\text{PL, N, V}\},$

$P = \{$

$\text{PL} \rightarrow \text{N s}$

$\text{N} \rightarrow \text{V er}$

$\text{N} \rightarrow \text{table}$

$\text{N} \rightarrow \text{laptop}$

$\text{V} \rightarrow \text{kick}$

$\text{V} \rightarrow \text{talk}$

$\text{V} \rightarrow \text{sing}\},$

$S = \text{PL}\rangle$

Write down the strings in the language permitted by this grammar.

(iii) Draw the finite-state network that accepts this language.

(iv) Given the configuration  $(q_0, \text{talkers})$ , trace the execution of the DFA in (iii).

Q 2.

(i) Define the constraints on  $\alpha$  and  $\beta$  in a context-free rewrite rule of the form  $\alpha \rightarrow \beta$ .

(ii) Consider the following Type 2 grammar rules:

S  $\rightarrow$  NP VP  
NP  $\rightarrow$  DET N  
NP  $\rightarrow$  PN  
VP  $\rightarrow$  V  
VP  $\rightarrow$  V NP  
DET  $\rightarrow$  some  
N  $\rightarrow$  person  
PN  $\rightarrow$  john  
V  $\rightarrow$  phones

Write down the strings in the language permitted by this grammar. Show their derivations using trees.

(iii) Show which of the rules in the grammar could be rules in an FSG. Explain why.

(iv) Which rules in the set in (ii) make this grammar a Type 2 grammar? Explain why.

Q 3.

(i) Define the constraints on  $\alpha$  and  $\beta$  in a context-sensitive rewrite rule of the form  $\alpha \rightarrow \beta$ .

(ii) Consider the following ruleset:

- $N \rightarrow lmn$
- $N \rightarrow lNMn$
- $nM \rightarrow Mn$
- $mM \rightarrow mm$

What are the shortest three strings in this language?

(iii) Show the derivation of the 2nd and 3rd shortest strings using trees;

(iv) Show the derivation of the 2nd and 3rd shortest strings using string manipulation.

(v) Does it make a difference in which order you apply the rules? Explain your answer.

Q 4.

(i) Define the constraints on  $\alpha$  and  $\beta$  in a Type 0 grammar rewrite rule of the form  $\alpha \longrightarrow \beta$ .

(ii) Describe in your own words what a Turing machine consists of, and what it is useful for.

(iii) Give a formal definition of a Turing machine in terms of the 5-tuple  $M = (Q, \Sigma, \Gamma, q_0, \delta)$ .

(iv) Construct a Turing machine which erases the input string, assuming unary input. Comment on the approach you have taken compared to other possible solutions.

(v) For the initial configuration  $(q_0, \underline{1}11\#)$ , show how the Turing machine you provided in (iv) copes with that input tape.

Q 5.

The following are expressions in the *untyped*  $\lambda$ -calculus. Reduce the expressions as far as possible. Remember that abstraction is right-associative and that application is left-associative.

- (i)  $(\lambda p.pqp) a$
- (ii)  $(\lambda p.qb) b$
- (iii)  $((\lambda p.(\lambda q.(qpq))) a) b$
- (iv)  $\lambda p.\lambda q.\lambda r.pqr a b c$
- (v)  $(\lambda q.((\lambda p.qrps) x)) y$
- (vi)  $(\lambda x.(\lambda y.xy)) y x$
- (vii) NOT FALSE
- (viii) AND TRUE FALSE
- (ix) IFTHENELSE FALSE loop exit

where

- TRUE :=  $(\lambda x.(\lambda y.(x)))$
- FALSE :=  $(\lambda x.(\lambda y.(y)))$
- NOT :=  $(\lambda p.(p \text{ FALSE } \text{ TRUE}))$
- AND :=  $(\lambda p.(\lambda q.(p q \text{ FALSE})))$
- IFTHENELSE :=  $(\lambda p.(\lambda x.(\lambda y.(p x y))))$