Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Fundamentals of Computation

Date: Thursday 23rd May 2013
Time: 09:45 - 11:45

Please answer any THREE Questions from the FOUR Questions provided.

Use a SEPARATE answerbook for each SECTION.

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
Section A

1. a) Consider the language of all words over the alphabet \{a, b, c\} whose first letter, if it exists, is the same as its last letter. Give a description of this language via the following means.

i) using a regular expression (2 marks)
ii) using a DFA (2 marks)
iii) using a grammar (2 marks)
iv) using the language of set theory (2 marks)

b) Give a DFA for the language of all words over the alphabet \{a, b, c\} that match the regular expression \(ab^*c^*|a^*b^*c\). (8 marks)

c) Are the following languages over the alphabet \{a, b, c\} regular? Justify your answer.

i) The language of all words whose number of symbols is divisible by three. (2 marks)

ii) The language of all words such that

- the number of \(a\)s in the word
- minus the number of \(c\)s in the word

is divisible by three. (2 marks)
2. a) Consider the following grammar. The underlying alphabet is \( \{a, b, c\} \), there is only one non-terminal symbol, \( S \), and the production rules are \( S \rightarrow aS \mid cS \mid cbS \mid bcS \mid \epsilon \).

i) Show that this grammar is ambiguous by giving a word with two parse trees. Make sure you also draw the word’s parse trees. (4 marks)

ii) Describe the language generated by the grammar. (2 marks)

iii) Give an unambiguous grammar for this language. Give an argument for your grammar being unambiguous. (4 marks)

b) Consider the regular expression \((a|b)^*ab(a|b)^*\) over the alphabet \(\{a, b\}\).

i) Describe the language consisting of all words that do not match the given regular expression without referring to that regular expression. (2 marks)

ii) Give a DFA for the language from (i). (2 marks)

c) Consider the language of balanced strings of brackets \((, )\), and square brackets \([, ]\).

Such strings must have as many opening as closing brackets of each kind, when reading the string the number of closing brackets of either kind seen so far must not exceed the number of corresponding opening brackets and each closing bracket must match the rightmost opening bracket that has not yet been matched.

i) Give a context-free grammar for this language. (3 marks)

ii) Give parse trees for the strings \(([[]])\) and \(((()][]))\). (3 marks)
3. a) State the principle of structural induction for finite lists (2 marks)

b) Using structural induction, show that (++, []) is a monoid; i.e. the append operator ++ is associative, and that the empty list [] is both a left and right identity for ++. Append (++ is defined as:

\[
\begin{align*}
(++) &:: [a] \rightarrow [a] \rightarrow [a] \\
[] ++ ys &= ys \\
(x:xs) ++ ys &= x: (xs ++ ys)
\end{align*}
\]

(5 marks)

c) Use structural induction to show that

\[
\text{map } f \circ \text{map } g = \text{map } (f \circ g)
\]

Why do we need to use induction with \( \bot \) to prove this equality?

\[
\begin{align*}
\text{map} &:: (a \rightarrow b) \rightarrow [a] \rightarrow [b] \\
\text{map } f \ [] &= [] \\
\text{map } f \ (x:xs) &= f \ x : \text{map } f \ xs
\end{align*}
\]

(3 marks)

d) Explain the following concepts: computability, uncomputability, decidability, and semi-decidability. For each class of problem, give an example problem which is in the class. (2 marks)

e) Show that the Halting Problem is undecidable; you need not give a full account of the universal program. (5 marks)

f) Show that the problem of determining that a program halts is semi-decidable. (3 marks)
4. a) Explain the following concepts: pre-condition, post-condition, partial correctness, and loop-invariant with reference to the while language. 

b) Consider the following while program:

\[ r := 1; \text{ while } (x > 0) \text{ do } (r := r*y; x := x-1) \]

Show that for some \( n \) and \( m \):

\[ y = n \land n^x \times r = m \]

is a loop-invariant for the while loop.

c) What is the value of \( r \) when the program from the previous part terminates? Using the loop invariant, prove your hypothesis.

d) Explain why we use “Big-O” notation when discussing algorithmic efficiency.

e) Using induction or otherwise, show that the factorial function \( \text{fac} \):

\[
\begin{align*}
\text{fac} \ 0 &= 1 \\
\text{fac} \ (n+1) &= (n+1) \times \text{fac} \ n
\end{align*}
\]

has time-complexity \( O(n) \).

f) Why does the problem of binary search have a lower bound of \( O(\log n) \)?