EASTERN UNIVERSITY, SRI LANKA

DEPARTMENT OF MATHEMATICS

SPECIAL DEGREE EXAMINATION IN COMPUTER SCIENCE 2010/2011 (March 2014)

CS 408: Compiler Design

Answer all questions

This paper has 6 questions in a total of 4 pages

Time allowed: Three Hours

1. Regular expressions are traditionally used to write the specifications for a lexical analyser of a programming language compiler.

   (a) State in detail what a compiler is. [10%]

   (b) State what is meant by *language* and define *regular expression.* [10%]

   (c) List the three constructs of regular expressions (other than terminal symbols and epsilon) and describe their properties (purpose , corresponding NFA fragment and their precedence order). [15%]

   (d) Give two strings of length five to nine for the regular expression "((ba)*a | a(b)*)*", one starting with "a" and the other starting with "b". [10%]

   (e) Propose a suitable regular expression to validate the **structure** of an IP address in dotted decimal format, (for example both 192.168.56.003, 192.168.56.300 have valid format). [15%]

   (f) Discuss about validating an IP address using a regular expression and propose a regular expression that can verify the validity and structure of an IP address, assuming that each numerical part of an IP address ranges from 0 to 255. [20%]

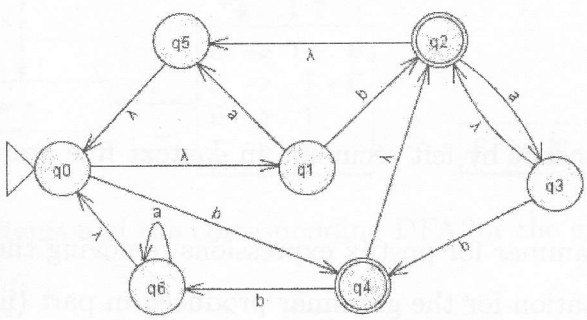
   (g) Construct an NFA for the regular expression "ab*((b|d)| c*)".

2. Finite automata are the right choice to realise the objectives of Lexical analysis.

   (a) State the objectives of Lexical analysis and explain how they can be achieved using NFA and/or DFA. [20%]

   (b) Define ϵ-*closure* and *move* operation in relation to converting NFA to DFA. [20%]

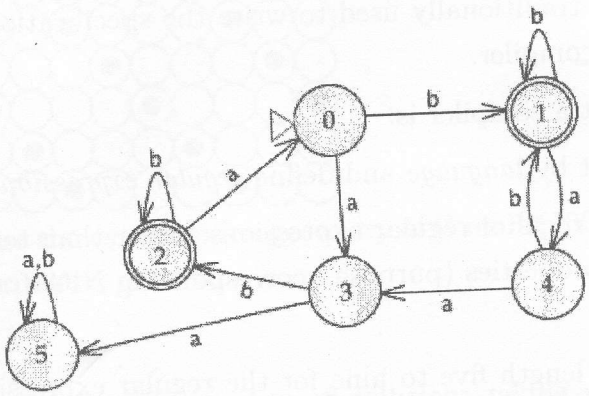
   (c) Convert the following NFA into DFA: [25%]



Note: Here $q0$ is start node and $q2$,$q4$ are accepting nodes; a and b are terminal symbols; epsilon transitions are denoted by λ.

(d) A DFA constructed from an NFA may not be minimal. State what is meant by mini DFA.

(e) Minimise the following DFA:



3. Like regular expressions, context-free grammars describe sets of strings.

(a) State what is meant by context-free grammar and describe its usage in syntax analys

(b) Given below is a context-free grammar where a, b, c are terminals; S and D are non_term

$S \rightarrow aDc$

$D \rightarrow DD \mid a \mid b \mid c$

Answer the following:

i. State what is meant by ambiguous grammar and a method to show a gram ambiguous. Verify your method to check whether the above grammar is ambigu not.

ii. State the underlying reasons for ambiguity in the above grammar.

(c) Given below is a context free grammar for postfix expressions:

$E \rightarrow E\ E\ +$

$E \rightarrow E\ E\ *$

$E \rightarrow num$

i. State what is meant by left recursion in context free grammar and describe a to remove it.

ii. Rewrite the grammar for postfix expressions removing the left recursion.

iii. Do left-factorisation for the grammar produced in part (ii).

4. LL(1) is a predictive parser commonly used for Syntax analysis.

(a) State the basic idea of predictive parsing. [15%]

(b) Consider the following grammar:

$Z \rightarrow X\ Y\ Z \mid d$

$Y \rightarrow c \mid \epsilon$

$X \rightarrow Y \mid a$

here `X`, `Y` and `Z` are non-terminals; `a`, `c` and `d` are terminals.

i. Convert the grammar suitable for constructing LL(1) parse table and construct the `First` and `Follow` sets. [35%]

ii. Construct the parse table for the above LL(1) grammar. [20%]

iii. Identify and state the reasons for the overlaps in the parse table. [15%]

iv. Using the parsing table parse the string *cacd*. [15%]

5. SLR parser is also a predictive parser which can handle more grammars than LL(1) parser.

(a) State the steps for constructing the SLR parsing table from a context free grammar. [20%]

(b) Describe the *shift* and *go* actions associated with an SLR parser. [20%]

(c) Construct the SLR parsing table for the following grammar using the augmented DFA given in Figure 1:

$S \rightarrow E\ \$$

$E \rightarrow T + E \mid T$

$T \rightarrow x$

here `S` is the starting symbol for this grammar; `E` & `T` are non_terminals and `+` & `x` are terminal symbols.

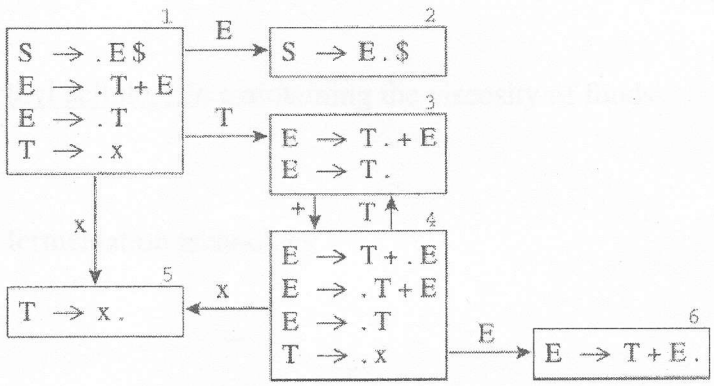


Figure 1: The set of LR(1) items and the corresponding DFA for the grammar given in Question 5, part (c)

(d) Parse the following strings and state its validity using the table you have constructed in part (d).

(i) `x + x + x $` (ii) `+ x + x $` [20%]

6. Generally the compilation process is divided into several phases with well-defined interfaces

   (a) List the common phases of a compiler and describe their function.

   (b) Other than compilers for compiling source code of programming languages, describe practical situations where compiler designing techniques can be used.

   (c) State what a symbol table is and describe two ways of implementing symbol tables compiler.

   (d) Some programming languages have shared name space. State what is meant be name s and advantages / disadvantages of having a shared name space.

   (e) The simplest way to execute a program is interpretation. State how interpretation ca done after lexical and syntax analysis phases.