Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

TRINITY COLLEGE DUBLIN
THE UNIVERSITY OF DUBLIN

Faculty of Engineering, Mathematics and Science
School of Computer Science and Statistics

Integrated Computer Science — Trinity Term 2017

Compiler Design I
(CS3071)

10 May, 2017 — Examination Hall — 14:00 - 16:00

Dr DM Abrahamson

<u>Instructions to Candidates</u>

Attempt question number 1 and one other question

<u>Materials permitted for this examination</u>

None

1. i Discuss the relationship between the output action symbols {label$_p$}, {jump$_p$} and {jumpf$_{p,q}$}, and use them to convert the following context-free productions into translation grammar form:

```
<statement> → WHILE <condition> DO <statement>
<statement> → REPEAT <statement> UNTIL <condition>
```

Explain the role of the function "`newl`" and demonstrate, by example, how the processing of the address field in a generated branch instruction is dependent on the relative position of a {label} and its corresponding {jump} or {jumpf} action.

[20 marks: 12 for the discussion and translation grammar; 2 for the description of NEWL; and 6 for the example]

ii Describe in detail the information that should be stored in the symbol table to record the properties of structured variables such as arrays and records, and show by example how to design a set of L-attributed translation grammar productions to cater for type and variable declarations of the form:

```
TYPE
   CourseData = RECORD
                   code, mark: INTEGER
                END;
   StudentInfo = RECORD
                    name: STRING;
                    number: INTEGER;
                    course: ARRAY [1..8] OF CourseData
                 END;
VAR
   Students: ARRAY [1..32] OF StudentInfo;
```

[20 marks: 8 for the description; and 12 for the translation grammar]

iii Consider the L-attributed translation grammar

$$
\begin{array}{lll}
\texttt{<statement>} & \rightarrow & \texttt{<expression>}_p \ \{\texttt{print}\}_p \\
\texttt{<expression>}_p & \rightarrow & + \ \texttt{<expression>}_q \ \texttt{<expression>}_r \ \{\texttt{add}\}_{q,r,p} \\
 & & \{\texttt{add}\}_{p,q,r} \quad r \leftarrow p+q \\
\texttt{<expression>}_p & \rightarrow & * \ \texttt{<expression>}_q \ \texttt{<expression>}_r \ \{\texttt{mult}\}_{q,r,p} \\
 & & \{\texttt{mult}\}_{p,q,r} \quad r \leftarrow p*q \\
\texttt{<expression>}_p & \rightarrow & \texttt{const}_q
\end{array}
$$

where `<expression>`$_p$ – synthesized $_p$, all action symbol attributes are inherited and const is a lexical token for an integer constant. Design an augmented pushdown interpreter to compute the value of arithmetic prefix expressions, and demonstrate its operation by showing the contents of the machine's stack at each step along the way while parsing the expression:

```
* 2 + 16 32
```

[20 marks: 6 for the augmented pushdown machine; 4 for the stack symbols; 4 for the replacement operations; and 6 for the demonstration]

[60 marks in total]

2. Design L-attributed translation grammar productions for compiling *conditional assignment* statements and a simple C style *for* statement. In each case, show by example the function of the action symbols and the information represented by the attributes.

[40 marks in total: 20 for the *conditional assignment* statement; and 20 for the *for* statement]

3. i What are the differences between the first, follow and selection sets in the context of LL(1) parsing? Demonstrate by example how they are computed.

[8 marks: 3 for the differences; and 5 for the demonstration]

ii In relation to error processing, describe what is meant by the term prefix property, and outline the major differences between local and global error recovery in a recursive descent parser.

[6 marks: 2 for the meaning; and 4 for differences]

iii Explain in detail the difference between synthesized and inherited attributes of non-terminal symbols.

[4 marks]

iv In relation to the top-down processing of l-attributed grammars, attribute evaluation rules for both synthesized and inherited attributes of non-terminal symbols must be in simple assignment form. Explain why this is the case, and show by example how a regular attribute evaluation rule for a synthesized or inherited attribute of a non-terminal symbol may be rewritten if necessary.

[8 marks: 5 for the explanation; and 3 for the description of the example]

v Show by example the differences between left factoring and the removal of left recursion from the productions of a context-free grammar.

[6 marks: 3 for left factoring; and 3 for left recursion]

vi Explain why the following two context-free productions are not LL(1)

```
<statement> -> if <condition> then <statement> else <statement>
<statement> -> if <condition> then <statement>
```

and demonstrate how they may be handled in a top-down parser.

[8 marks: 4 for the explanation; and 4 for the solution]

[40 marks in total]