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 2018**
BA (Mod) Computer Science and Language
Year 3 Annual Examinations

Compiler Design I
(CS3071)

Friday 4th May 2018 | **Goldsmith Hall (main)** | **14.00-16.00**

DR DM Abrahamson

Instructions to Candidates:

Attempt question 1 and one other question

Materials permitted for this examination:

None

1. Consider the following context free grammar for arithmetic expressions with starting non-terminal symbol <E>:

```
<E> → <E> + <T>
<E> → <T>
<T> → <T> * <P>
<T> → <P>
<P> → ( <E> )
<P> → IDENT
<P> → CONST
```

i. Extend the grammar to include subtraction, division and exponentiation (where exponentiation is right associative and has higher precedence than multiplication and division). [6 Marks]

ii. Produce an equivalent LL(1) grammar by left-factoring and by eliminating left recursion from the productions obtained in part i above. [6 Marks]

iii. Compute the selection set for each of the productions in the LL(1) grammar. [8 Marks]

iv. Convert the grammars obtained in parts i and ii above into attributed translation grammars for an interpreter by adding attributes and attribute evaluation rules to the productions so that the starting non-terminal symbol of each grammar will have a single synthesized attribute whose value is that of the expression being parsed. [12 Marks]

v. Show that both grammars produce the same translation by drawing fully decorated derivation-trees for the expression `A/B-2↑3↑2` where variables `A` and `B` have values `1024` and `4` respectively, and ↑ is the (right-associative) exponentiation operator. [8 Marks]

vi. Convert the grammars obtained in parts i and ii above into attributed translation grammars for a compiler by adding action symbols ({add}, {mult}, etc), attributes and attribute evaluation rules to the productions so that the starting non-terminal symbol of each grammar will have a single synthesized attribute pointing to the symbol-table entry which describes at compile time where, at run time, the value of the expression being parsed is to be found (in a register, in memory, etc). [12 Marks]

vii. Show by example that although the structure of derivation trees for arithmetic expressions parsed using the LL(1) grammar obtained in part vi above suggests the arithmetic operators "+", "–", "*" and "/" are right associative, the translation maintains their normal left to right associativity. [8 Marks]
[Total 60 Marks]

2. Describe the information that should be maintained in the symbol table at compile time to record the properties of multi (eg one and two) dimensional arrays, and design L-attributed translation grammar productions to parse variable declarations of the general form:

```
VAR
   data: ARRAY [1..16] OF INTEGER;
   s1, s2: ARRAY [1..8, 1..32] of STRING;
```

Explain in detail the function of the action symbols, and sketch the symbol table's contents when compiling the variable declarations shown above.

[40 Marks: 8 for the information; 16 for the grammar; 8 for the action symbols; and 8 for the symbol table]

3 i. Explain the relationship between the first, follow and selection sets in the context of LL(1) parsing. [6 Marks]

ii. Describe the prefix property and outline the major differences between local and global error recovery. [6 Marks: 2 for the prefix property; and 4 for the differences]

iii. Discuss the relationship between the output action symbols {label$_p$}, {jump$_p$} and {jumpf$_{p,q}$}, and use them to convert the context free grammar production for a conditional assignment statement:

```
<statement> → <ident>:=<condition>?<expression1>:<expression2>
```

into an L-attributed translation grammar production.

[28 Marks; 8 for the discussion; and 20 for the L-attributed production]

[Total 40 Marks]