

Midterm Exam
CS 410, Fall 1997

October 16, 1997

There are 7 problems on the exam, with 100 points total available. There are 7 pages to the exam, including this one; make sure you have all of them. Don't forget to put your name at the top of the exam. Please read over the whole test before beginning. Good luck!!!

We use the following conventions for grammars on the exam, except where noted:

- epsilon, the empty string is represented by ϵ
- terminals are represented by lower-case letters and punctuation symbols
- non-terminals are represented by upper-case letters

	value	grade
Problem 1	10 pts.	
Problem 2	15 pts.	
Problem 3	15 pts.	
Problem 4	15 pts.	
Problem 5	18 pts.	
Problem 6	15 pts.	
Problem 7	12 pts.	
TOTAL:	100 pts.	

1

Problem 1 [10 pts]

Write a regular expression for non-parenthesized arithmetic expressions that have only $+$ (plus) as an operator, and a as an operand. Some example strings in the language:

$a + a$ a $a + a + a + a$

Problem 2 [15 pts]

$A \rightarrow AA$
 $\quad | a$
 $\quad | \epsilon$

Part A [5 pts]. What is the language generated by the above grammar?

Part B [10 pts]. Show that the grammar is ambiguous.

2

Problem 3 [15 pts]

Using subset construction, create a DFA equivalent to the following NFA. Show your work.

	a	b	ϵ
1	{2}	{3}	\emptyset
2	{2}	\emptyset	{3}
3	{2,4}	{3}	\emptyset
4	\emptyset	\emptyset	\emptyset

3

Problem 4 [15 pts]

A shift-reduce conflict arises when attempting to create a SLR parser for the following grammar. Show what it is. I.e., explain exactly the situation where the parser wouldn't be able to decide between shift and reduce. The set of terminals is: $\{\text{id}, +, (,)\}$.

$$\begin{array}{l} P \rightarrow E(E) \\ \quad | E \\ E \rightarrow \text{id} + E \\ \quad | \text{id}(E) \\ \quad | \text{id} \end{array}$$

4

Problem 5 [18 pts]

$L \rightarrow L+O$
 $| O$
 $O \rightarrow (E)$
 $| a$
 $E \rightarrow E+a$
 $| a$

Part A [3 pts.] What is the maximum number of levels of parentheses in a sentence generated by the grammar above?

Part B [15 pts.] Make a LL(1) parse table for the grammar. Show your work.

Problem 6 [15 pts total]

For the grammar below, write semantic rules for computing the semantic attribute `S.moreInside` whose value can only be true or false. `S.moreInside` is true for a sentence generated from S if that sentence has more a's inside parentheses than outside, false otherwise.

Here are some examples of sentences and values for `moreInside`:

sentence	<code>S.moreInside</code>
<code>a + a</code>	false
<code>a + (a + a)</code>	true
<code>a + a + (a)</code>	false
<code>a + (a)</code>	false
<code>(a + a + a) + a + (a + a)</code>	true

Hint: Create additional semantic attributes (besides `moreInside`) to help you compute the value of `moreInside`. In particular you are allowed to have more than one semantic attribute per non-terminal.

The grammar:

$S \rightarrow L$

$L \rightarrow L_1 + O$

$L \rightarrow O$

$O \rightarrow (E)$

$O \rightarrow a$

$E \rightarrow E_1 + a$

$E \rightarrow a$

Problem 7 [12 pts total]

Consider the following partial Cool program:

```
class C is
  x : A;
  y : B;
  foo(y : D, z : E) : F is
    let x : G, y : H ← <expr1> in
      <expr2>
  end; -- end method foo
end; -- end class C
```

Notes: <expr1> and <expr2> are placeholders for specific expressions. Assume that classes A-B and D-H are defined elsewhere in the program.

Part A [5 pts.] When the semantic analyzer processes <expr1> what object identifiers are visible (i.e., valid to use in the expression)? In answering the question give the name of each such object identifier and its type. (Note: this question concerns only object identifiers, not classes or methods.)

Part B [5 pts.] Answer the same question for <expr2>.

Part C [2 pts.] At the point that the semantic analyzer processes <expr2> how many levels of scope are on the symbol table? Put another way, how many times have we called `enter_scope` without a corresponding call to `exit_scope`.

Hint: the top-level scope is the global scope, where all the class identifiers are defined (this will count as the first one).