

CMSC 430

Midterm 1 Practice Problems

1. Attribute grammars and syntax-directed translation

- (a) What are the advantages of syntax-directed translation over context-free parsing?
- (b) What are the differences between context-free and context-sensitive parsing?
- (c) Rank the following techniques in terms of the languages (sets of strings) they can recognize: syntax-directed translation, attribute grammars, context-free grammars, context-sensitive grammars.
- (d) In an attribute grammar, what are the differences between inherited and synthesized attributes?
- (e) Write a small attribute grammar with one inherited and one synthesized attribute, labeling each.

2. Type expressions

Given the following C declarations:

```
int num(char *x, int chuk[4]);
typedef struct {
    int a, b;
} CELL;
CELL foo[100];
CELL *bar(int x, CELL y) { ... }
```

- (a) write the type expression for "num"
- (b) write the type expression for "foo"
- (c) write the type expression for "bar"

3. Syntax-directed translation

Consider the following grammar productions. Assume you have an attribute E.odd which can be set to either true, false, or unknown, and an attribute CONST.val which is the value of the constant.

E	→	CONST	{ E.odd = ?? }
		ID	{ E.odd = unknown }
		E ₁ + E ₂	{ E.odd = ?? }
		E ₁ - E ₂	{ E.odd = ?? }
		E ₁ * E ₂	{ E.odd = ?? }
		(E ₁)	{ E.odd = ?? }
		- E ₁	{ E.odd = ?? }

- (a) Add rules to the attribute grammar to calculate E.odd for each grammar production.
- (b) Provide the parse tree for the expression (5 + 2) - (4 * - x) and show the calculation for E.odd at each point.

4. Syntax-directed translation and type checking

Consider the following grammar, which generates expressions for a number of operators. Assume you have the attributes E.min and E.max which should be set to the minimum and maximum values for E, and an attribute CONST.val which is the value of the constant.

E	→	CONST	{ E.min = ?? ; E.max = ?? }
		ID	{ E.min = ID.min ; E.max = ID.max }
		E ₁ + E ₂	{ E.min = ?? ; E.max = ?? }
		E ₁ - E ₂	{ E.min = ?? ; E.max = ?? }
		E ₁ * E ₂	{ E.min = ?? ; E.max = ?? }
		(E ₁)	{ E.min = ?? ; E.max = ?? }
		- E ₁	{ E.min = ?? ; E.max = ?? }

- (a) Write the type-checking rules which calculates the range for each subexpression.
- (b) Provide the parse tree for the expression (5 + 2) - (4 * - x) and show the calculation for E.min and E.max at each point, assuming you know x.min = -3 and x.max = 10.

5. Syntax-directed translation and type checking

Consider the following grammar, which generates expressions formed by applying "+" to integer and floating point constants. When two integers are added, the result is integer, otherwise, it is a float.

$$E \rightarrow E + T \mid T$$
$$T \rightarrow \text{num} \mid \text{num} . \text{num}$$

- (a) Give a syntax-directed definition to determine the type of each subexpression. Assign each symbol an attribute "type".
- (b) Provide the parse tree for the express
 $5 + 4 + 3.2 + 1$
and show the computation E.type and T.type at each point.

6. Symbol table

Consider the following program in a lexically-scoped language such as C.

```
int x, y;
int foo( )
  { int z; { int y; { int x; } } }
int bar( )
  { int z; { int x; } { int y; // HERE } }
```

- (a) What is nested lexical scoping?
- (b) How can symbol tables handle nested lexical scoping?
- (c) How does the compiler determine where each variable encountered during compilation is actually declared when handling nested lexical scoping?
- (d) Use your answer to construct the logical state of the symbol table(s) for the example when the compiler reaches the point marked HERE.