Emily Howell
Email: eth35@case.edu
Problem Set - 4
ID: 3401809
Course: CSDS 337 - Compiler Design
Term: Spring 2021
Instructor: Dr. Vipin Chaudhary
Due Date: 15^{th} April, 2021

Number of hours delay for this Problem Set: 0
Cumulative number of hours delay so far: 72

I discussed this homework with: Ally Dinhofer

Problem 1 - 10 points
Suppose that we have a production $A \rightarrow BCD$. Each of the four nonterminals A , B, C, and D have two attributes: s is a synthesized attribute, and i is an inherited attribute. For each of the sets of rules below, tell whether (i) the rules are consistent with an S-attributed definition (ii) the rules are consistent with an L-attributed definition, and (iii) whether the rules are consistent with any evaluation order at all?

a $A : s = B : i + C : s.$

b $A : s = B : i + C : s$ and $D : i = A : i + B : s.$

c $A : s = B : s + D : s.$

d $A : s = D : i, \quad B : i = A : s + C : s, \quad C : i = B : s,$ and $D : i = B : i + C : i.$

Solution:

a This is L-attributed because it uses inherited attributes but only from the left sibling and not from the right sibling. Is not S-attributed because it uses the inherited attribute of B.

b This is neither L-attributed nor S-attributed because it uses the inherited attribute of a non-left sibling non terminal.

c This is S-attributed because A's synthesized attribute is formed from the synthesized attributes of its children. Because it is S-attributed it is also L-attributed.

d The rule B.i=A.s+C.s violates L-attributed definition because B's inherited attribute is dependent on the values from its right sibling C. Because there are inherited attributes this is also not S-attributed.

Problem 2 - 10 points
Construct the DAG for the expression $((x + y) - ((x + y) * (x - y))) + ((x + y) * (x - y))$

Solution: For my own visualization I am going to break this expression down step by step and use letters to simplify this expression.

$((x + y) - ((x + y) * (x - y))) + ((x + y) * (x - y))$
if $A = x + y$
$(A - (A * (x - y))) + (A * (x - y))$
if $B = x - y$
$(A - (A * B)) + (A * B)$
if $C = A * B$

$(A - C) + C$
if $D = A - C$
$D + C$
if $E = A - C$
E

So the rules that we are left with are $A = x + y$, $B = x - y$, $C = A * B$, $D = A - C$, and $E = A - C$ which results in the following DAG for the expression:

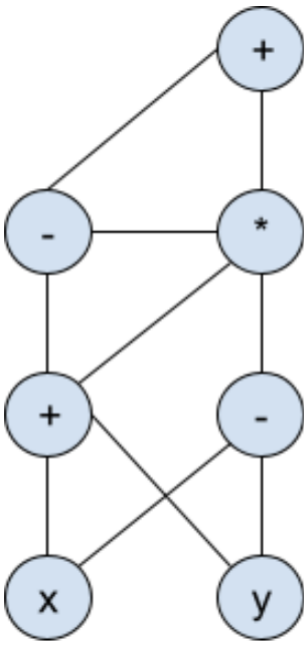

Figure 1: DAG for the above expression

Problem 3 - 10 points
Translate the arithmetic expression $a + (b + c)$.

a A syntax tree.

b Quadruples.

c Triples.

d Indirect triples.

Solution:

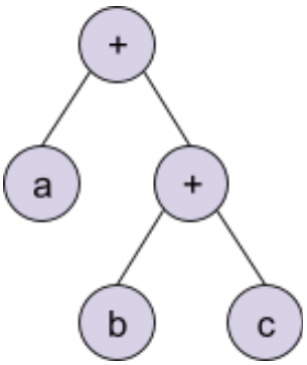

Figure 2: Syntax tree of the expression

a

	Operation	Arg 1	Arg 2	Result
0	+	b	c	t1
1	+	a	t1	t2

Figure 3: Quadruples of the expression

b

	Operation	Arg 1	Arg 2
0	+	b	c
1	+	a	(0)

Figure 4: Triples of the expression

c

	Operation	Arg 1	Arg 2
0	+	b	c
1	+	a	(0)

	Instruction
0	(0)
1	(1)

Figure 5: Indirect triples of the expression

d

Problem 4 - 10 points
A real array $A[i; j; k]$ has index i ranging from 1 to 4, j ranging from 0 to 4, and k ranging from 5 to 10. Reals take 8 bytes each. If A is stored row-major, starting at byte 0, find the location of:

a $A[3; 4; 5]$

b $A[1; 2; 7]$

c $A[4; 3; 9]$.

Repeat the above if A is stored in column-major order.

Solution:
Row-major:
$((i-1)*5*6+j*6+(k-5))*8$

a $((3-1)*5*6+4*6+(5-5))*8 = 672$

b $((1-1)*5*6+2*6+(7-5))*8 = 112$

c $((4-1)*5*6+3*6+(9-5))*8 = 896$

Column-major:
$((i-1)+j*4+(k-5)*5*4)*8$

a $((3-1)+4*4+(5-5)*5*4)*8 = 144$

b $((1-1)+2*4+(7-5)*5*4)*8 = 384$

c $((4-1)+3*4+(9-5)*5*4)*8 = 760$

Problem 5 - 20 points
Add rules to the syntax-directed definition of Fig. 6 for the following control-flow constructs:

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$
$S \rightarrow$ **if** (B) S_1	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \rightarrow$ **while** (B) S_1	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

Figure 6: Rules to the syntax-directed definition

- A repeat-statement **repeat** S **while** B.

- A for-loop **for** $(S_1; B; S_2)S_3$.

Solution:

- $S \rightarrow repeat \quad S1 \quad while \quad B$
  $S1.next = newlabel()$
  $B.true = newlabel()$
  $B.false = S.next$
  $S.code = label(B.true)$
  . || $S1.code$
  . || $label(S1.next)$
  . || $B.code$

- A for-loop **for** $(S_1; B; S_2)S_3$. $S \rightarrow for(S1; B; S2)S3$
  $S1.next = newlabel()$
  $B.true = newlabel()$
  $B.false = S.next$
  $S2.next = S1.next$
  $S3.next = newlabel()$
  $S.code = S1.code$
  . || $lable(S1.next)$ || $B.code$
  . || $lable(B.true)$ || $S3.code$
  . || $label(S3.next)$ || $S2.code$
  . || $gen('goto', S1.next)$

Problem 6 - 10 points
Translate the following expressions using the goto-avoiding translation scheme:

a if $(a == b \quad \&\& \quad c == d \quad || \quad e == f) \quad x == 1;$

b if $(a == b \quad || \quad c == d \quad || \quad e == f) \quad x == 1;$

c if $(a == b \quad \&\& \quad c == d \quad \&\& \quad e == f) \quad x == 1;$

Solution:

a ifFalse a==b goto L3
if c==d goto L2
L3: ifFalse e==f goto L1
L2: x == 1
L1:

b if a==b goto L2
if c==d goto L2
ifFalse e==f goto L1
L2: x == 1
L1:

c ifFalse a==b goto L1
ifFalse c==d goto L1
ifFalse e==f goto L1
L2: x == 1
L1:

Problem 7 - 10 points
The C code to compute Fibonacci numbers recursively is shown below. Suppose that the activation record for f includes the following elements in order: (return value, argument n, local s, local t); there will normally be other elements in the activation record as well. The questions below assume that the initial call is $f(5)$.

```
int f(int n) {
    int t, s;
    if (n < 2) return 1;
    s = f(n-1);
    t = f(n-2);
    return s+t;
}
```

a Show the complete activation tree.

b What does the stack and its activation records look like the first time $f(1)$ is about to return?

Solution:

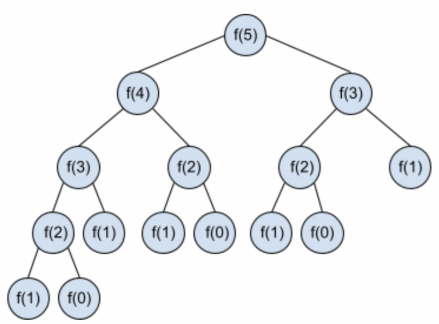

Figure 7: The activation tree for the above program

a

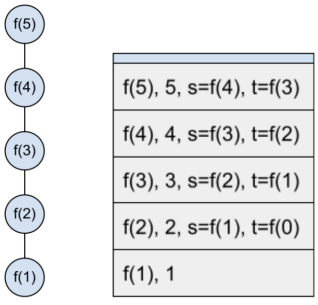

Figure 8: Stack and the activation record for the first time f(1) is about to return

b

Problem 8 - 10 points
In a language that passes parameters by reference, there is a function $f(x; y)$ that does the following:
$x = x + 1; \quad y = y + 2; \quad return \quad x + y;$
If a is assigned the value 3, and then $f(a; a)$ is called, what is returned?

Solution: $f(a; a)$:

$x = x + 1; \quad \rightarrow \quad a = a + 1 = 3 + 1 = 4$
$y = y + 2; \quad \rightarrow \quad a = a + 2 = 4 + 2 = 6$
$return \quad x + y; \quad \rightarrow \quad a + a = 6 + 6 = 12$

Therefore, since the language that passes parameters by reference, f(a;a) where a = 3 will return 12.

Problem 9 - 10 points
The C function f is defined by:

```
int f(int x, *py, **ppz) {
    **ppz += 1; *py += 2; x += 3; return x+**py+**ppz;
}
```

Variable a is a pointer to b ; variable b is a pointer to c , and c is an integer currently with value 4. If we call $f(c; b; a)$, what is returned?

Solution: $f(c; b; a)$

$**ppz+ = 1; \quad \rightarrow \quad c = 4 + 1 = 5$
$*py+ = 2; \quad \rightarrow \quad c = 5 + 2 = 7$
$x+ = 3; \quad \rightarrow \quad x = 4 + 3 = 7$
$returnx + *py + **ppz; \quad \rightarrow \quad x + c + c = 7 + 7 + 7 = 21$

Some notes about the above, because c is passing into int x and not the pointer to the value of c then the value of c is passed and when x+=3 it is not altering the value of c but rather just the value of x. However, when **ppz and *py are incremented the value of c is also incremented because those are pointers to the actual memory location where c is stored.

Therefore, f(c;b;a) where $a \rightarrow b \rightarrow c = 4$ will return 21.
