ABHISHEK PARDHI
200026

CS 335 Semester 2022–2023-II: End-Semester Exam

8–11 AM, April 25th 2023

Suggestions

- There are six pages. Answer questions exactly as asked, and be concise in your explanations.
- Your answers should be legible. Show logical steps and rough work in your computation.
- Explicitly mention ANY ASSUMPTIONS that you made for coming up with your solutions.
- Few questions require you to fill in answers in the question paper itself. Turn in YOUR QUESTION PAPER ALONG WITH YOUR SOLUTION SHEETS.

Questions

1. Answer the following questions related to parsing. **[3×4 marks]**

   (a) Show the left-factored form of the following grammar.

   $$S \to S\dagger \mid S \nabla S \mid S \Updownarrow \mid \lceil T \rceil$$
   $$T \to Ta \mid Tb \mid Tc \mid \epsilon$$

   (b) Eliminate left recursion from the following grammar.

   $$S \to Sab \mid S! \mid (T) \mid bTb$$
   $$T \to Ta \mid Tb \mid Tc \mid \epsilon$$

   (c) Consider shift-reduce bottom-up parsers which can support both left and right recursion. Do you think there are any advantages of using left recursion over right recursion in the input grammar for unrestricted input?

   (d) Why do semantic routines need to be placed at the end of productions in LR grammars?

2. Consider the following basic block and its constituent instructions. Assume all variables are of type integers, `f` is input to the basic block, and `y` is the only variable live on exit from the basic block.

```
1   a = f * 2 + 0;
2   b = a - 0;
3   c = 1 + 6;
4   d = c * b;
5   e = f * f;
6   x = e + d;
7   g = b + d;
8   h = b + d;
9   i = g * 1;
10  y = i / h;
```

(i) Apply the following optimizations to the basic block, in order: (i) Algebraic simplification (AS), (ii) Constant folding (CF), (iii) Common sub-expression elimination (CSE), (iv) Strength reduction (SR), and (v) Dead code elimination (DCE). Explain the modifications carried out by each optimization, and show the result of each transformation. The input to optimization i is the output from the optimization $i-1$.

(ii) Can you suggest *additional* optimizations on the final output from DCE.

[8 marks]

3. Consider the following assembly-like IR that makes use of a few temporaries (symbolic registers).

```
t1 = t0 + 15;
t2 = t0 * 3;
if (t1 < t2) {
  t3 = t1 * t2;
  t4 = t1 * 2;
} else {
  t3 = t1 + t2;
  t4 = t1 + t3;
}
t5 = t4 - t2;
t6 = t5 + t3;
t2 = t5 + 1;
t7 = t2 + t3;
if (t6 < t7) {
  t8 = t6;
} else {
  t8 = t7;
}
t9 = t8 * 2;
```

(i) List the temporaries that are live at each program point. Note that `t0` is the only input temporary for the given code and `t9` will be the only live value on exit.

(ii) Draw the interference graph between temporaries for the above program.

(iii) Provide a lower bound on the number of registers required by the program.

(iv) Provide a k-coloring of the interference graph, where k is from item (iii).

[5+3+2+2 marks]

4. Construct an SDT scheme that translates Roman numerals into integers. The grammar and a reference table are given below. Do not worry about numbers that the given grammar cannot generate. **[8 marks]**

$$
\begin{aligned}
\mathit{Rnum} &\rightarrow \mathit{Thousand}\ \mathit{Hundred}\ \mathit{Ten}\ \mathit{Digit} \\
\mathit{Thousand} &\rightarrow \mathrm{M} \mid \mathrm{MM} \mid \epsilon \\
\mathit{Hundred} &\rightarrow \mathit{SmallHundred} \mid \mathrm{CD} \mid \mathrm{D}\ \mathit{Smallhundred} \mid \mathrm{CM} \\
\mathit{SmallHundred} &\rightarrow \mathrm{C} \mid \mathrm{CC} \mid \epsilon \\
\mathit{Ten} &\rightarrow \mathit{SmallTen} \mid \mathrm{XL} \mid \mathrm{L}\ \mathit{SmallTen} \mid \mathrm{XC} \\
\mathit{SmallTen} &\rightarrow \mathrm{X} \mid \mathrm{XX} \mid \epsilon \\
\mathit{Digit} &\rightarrow \mathit{SmallDigit} \mid \mathrm{IV} \mid \mathrm{V}\ \mathit{SmallDigit} \mid \mathrm{IX} \\
\mathit{SmallDigit} &\rightarrow \mathrm{I} \mid \mathrm{II} \mid \mathrm{III} \mid \epsilon
\end{aligned}
$$

Roman numeral	Decimal value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

5. Consider the following grammar with 2 missing productions.

$$S \rightarrow aS \mid \ldots(1) \quad (1)$$
$$A \rightarrow \ldots(2) \mid \epsilon \quad (2)$$
$$X \rightarrow cS \mid \epsilon \quad (3)$$
$$Y \rightarrow dS \mid \epsilon \quad (4)$$
$$Z \rightarrow eS \quad (5)$$
$$(6)$$

We know the FIRST and FOLLOW sets for the grammar. Reconstruct the grammar by filling in the TWO missing productions. **[6 marks]**

Symbol	FIRST	FOLLOW
S	FIRST(S) = {a,b,c,d,e}	{\$} ∪ FOLLOW(X) ∪ FOLLOW(Y) ∪ FOLLOW(Z)
A	FIRST(A) = {c,d,e,ϵ}	{b}
X	FIRST(X) = {c,ϵ}	FIRST(Y)/ϵ ∪ FIRST(Z)
Y	FIRST(Y) = {d,ϵ}	FIRST(Z)
Z	FIRST(Z) = {e}	FOLLOW(A)
a	{a}	FIRST(S)
b	{b}	FOLLOW(S)
c	{c}	FIRST(S)
d	{d}	FIRST(S)
e	{e}	FIRST(S)

6. Consider the following grammar.

$$S \rightarrow aAd \quad (1)$$
$$S \rightarrow bBd \quad (2)$$
$$S \rightarrow aBe \quad (3)$$
$$S \rightarrow bAe \quad (4)$$
$$A \rightarrow c \quad (5)$$
$$B \rightarrow c \quad (6)$$

[8+5+5+2 marks]

(i) Show the construction of the LR(1) canonical collection of items in your ANSWER SHEET. You do not need to draw the LR(1) automaton.

(ii) Fill in the LR(1) parsing table given below. Add more rows to the table if your collection requires more states. Similarly, ignore additional rows if your collection has fewer states. Use the production RULE NUMBERS AS GIVEN ABOVE for parsing table entries.

	Action						Goto		
	a	b	c	d	e	$	S	A	B
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									

(iii) Fill in the LALR(1) parsing table given below. Add more rows to the table if your collection requires more states. Similarly, ignore additional rows if your collection has fewer states.

	Action						Goto		
	a	b	c	d	e	$	S	A	B
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

(iv) Briefly explain whether the above grammar is (i) LR(1) and (ii) LALR(1)?

7. Show an equivalent SSA-form for the following sequence of instructions. Do not forget to include ϕ-functions if needed. [6 marks]

```
            b = 0;
            d = 0;
            a = 1;
            i = ...;
Loop:       if (i > 0) {
              if (a < 0) {
                b = i;
              } else {
                b = 0;
              }
              i = i - 1;
              if (i < 0) {
                i = 0;
                goto Break;
              }
              if (b == 0) {
                goto Break;
              } else {
                a = a + d;
              }
              goto Loop;
            }
Break:      x = a;
            y = b;
```

8. Consider the intermediate code given below. [2+2+2+2 marks]

```
            i = 1
            j = 1
            t1 = 5 * i
            t2 = t1 + j
            t3 = 4 * t2
            t4 = t3
            a[t4] = -1
            j = j + 1
            if j <= 5 goto (3)
            i = i + 1
            if i < 5 goto (2)
```

The symbols `a`, `i`, and `j` refer to variables in the program, while names like `t1` refer to temporaries.

(i) Identify the basic blocks using instruction numbers. Draw the control flow graph.

(ii) Write the quadruple and triple form of the code snippet.