NAME: **ROLL NO:**

CS738: Advanced Compiler Optimizations
End Semester Examination, 2017-18 I

Max Time: 3 Hours Max Marks: 160

NOTE:

- There are total **6** questions on **5 pages**.
- Write your name and roll number on the question paper and the answer book.
- Presenting your answers properly is your responsibility. You lose credit if you can not present your ideas clearly, and in proper form. Please DO NOT come back for re-evaluation saying, "What I actually meant was . . . ".
- Be precise and write clearly. Remember that somebody has to read it to evaluate!
- **IMPORTANT: Some questions have to be answered on the sheet provided at the end of the question paper. Write you Name and Roll Number, and return it along with the answer book.**

Notations

- CFG stands for control flow graph.
- $\mathsf{IN}(S)$ denotes the program point before the statement S. $\mathsf{OUT}(S)$ denotes the program point after the statement of S.
- $\mathsf{PRED}(S)$ denotes the set of predecessors, and $\mathsf{SUCC}(S)$ denotes the set of successors of S.
- In a CFG, $x \xrightarrow{+} y$ denotes a path from node x to node y, having one or more edges. Both x and y are considered to be a part of the path.
- $\mathrm{DF}^{+}(\varphi)$ denotes the Iterated Dominance Frontier of the set of CFG nodes φ.

1. [15 + 20] **Access Range of A Variable.** A program point π is in the *access range* of a variable v if **both** the following conditions hold:

   - There is an access (definition or use) of v on some path from Entry to π

     AND

   - There is an access of v on some path from π to Exit.

   Figure **??** shows the CFG for an example program having variables A, B and C.

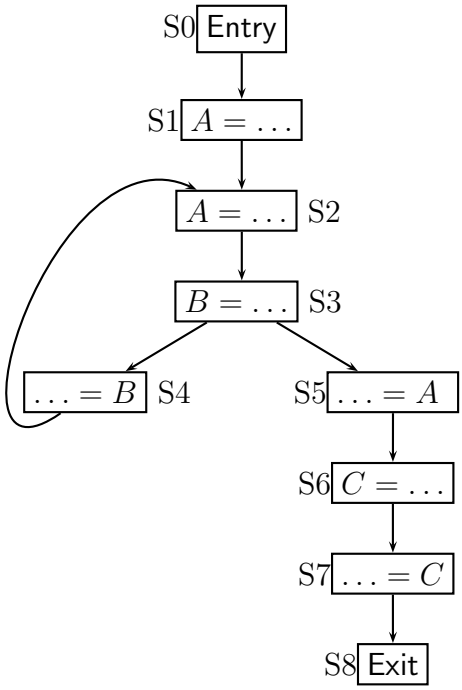


Figure 1: An Example Program

   (a) Complete the Table **??** (on the last page) to show the program points in the access ranges of variables A, B and C for the CFG in Figure **??**.

   (b) **Define data flow equations to compute** AccIN(S, v) **and** AccOUT(S, v), where

   - AccIN(S, v) is true if IN(S) is in the access range of a variable v.
   - AccOUT(S, v) is true if OUT(S) is in the access range of a variable v.

   You might find it easy to define the equations for the following auxiliary data flow quantities and then define AccIN(S, v) and AccOUT(S, v) in terms of these auxiliary quantities (this is not compulsory though!):

   - BeforeIN(v, S) is true if there is an access to variable v before IN(S). BeforeOut(v, S) is true if there is an access to the variable v before OUT(S).
   - AfterIN(v, S) is true if there is an access to variable v after IN(S). AfterOut(v, S) is true if there is a access to variable v after OUT(S).

$$\mathsf{BeforeOut}(v,S) = \begin{cases} \text{true, if } S \text{ has an access (def or use) of } v \\ \mathsf{BeforeIN}(v,S), \text{ otherwise} \end{cases}$$

$$\mathsf{BeforeIN}(v,S) = \begin{cases} \text{false, if } S \text{ is } \mathsf{Entry} \text{ block} \\ \bigvee_{P \in \mathsf{PRED}(S)} \mathsf{BeforeOut}(v,P), \text{ otherwise} \end{cases}$$

$$\mathsf{AfterIN}(v,S) = \begin{cases} \text{true, if } S \text{ has an access (def or use) of } v \\ \mathsf{AfterOut}(v,S), \text{ otherwise} \end{cases}$$

$$\mathsf{AfterOut}(v,S) = \begin{cases} \text{false, if } S \text{ is } \mathsf{Exit} \text{ block} \\ \bigvee_{BS \in \mathsf{SUCC}(S)} \mathsf{AfterIN}(v,BS), \text{ otherwise} \end{cases}$$

$$\mathsf{AccIN}(v,S) = \mathsf{BeforeIN}(v,S) \wedge \mathsf{AfterIN}(v,S)$$

$$\mathsf{AccOUT}(v,S) = \mathsf{BeforeOut}(v,S) \wedge \mathsf{AfterOut}(v,S)$$

2. [15+10] **Access Range of a Set of Variables:** A program point π is in the access range of a set of variables, φ, if both the following conditions hold:

   - There is an access to a variable $v \in \varphi$ on a path from Entry to π

     AND

   - There is an access to a variable $v' \in \varphi$ on a path from π to Exit.

   Note that v and v' may or may not be the same variable in φ.

   (a) Complete the Table **??** (on the last page) to show the program points in the access ranges of given sets of variables for the CFG in Figure **??**.

   (b) **Define equations to compute** $\mathsf{AccSetIN}(S, \varphi)$ **and** $\mathsf{AccSetOUT}(S, \varphi)$, where

   - $\mathsf{AccSetIN}(S, \varphi)$ is true if $\mathsf{IN}(S)$ is in the access range of the set of variable φ.
   - $\mathsf{AccSetOUT}(S, \varphi)$ is true if $\mathsf{OUT}(S)$ is in the access range of the set of variables φ.

   You can reuse any quantities defined in Question ??.

$$\mathsf{AccSetIN}(\varphi, S) = \left(\bigvee_{v \in \varphi} \mathsf{BeforeIN}(v, S)\right) \wedge \left(\bigvee_{v \in \varphi} \mathsf{AfterIN}(v, S)\right)$$
$$\mathsf{AccSetOUT}(\varphi, S) = \left(\bigvee_{v \in \varphi} \mathsf{BeforeOut}(v, S)\right) \wedge \left(\bigvee_{v \in \varphi} \mathsf{AfterOut}(v, S)\right)$$

Note that these *simple* definitions **do not work**:

$$\mathsf{AccSetIN}(\varphi, S) = \bigvee_{v \in \varphi} \mathsf{AccIN}(v, S)$$
$$\mathsf{AccSetOUT}(\varphi, S) = \bigvee_{v \in \varphi} \mathsf{AccOUT}(v, S)$$

For example, $\mathsf{OUT}(S5)$ is in the access range for the set $\{C, A\}$. but it is neither in the access range of A, nor that of C

3. [10] Prove the following statement:

   For any non-null path $p : X \xrightarrow{+} Z$ in a CFG, there exists a node $X' \in \{X\} \cup \mathrm{DF}^+(\{X\})$ on p that dominates Z. Moreover, unless X dominates every node on p, the node X' can be chosen in $\mathrm{DF}^+(\{X\})$.

   (a) X **dominates every node in** p. Clearly X dominates Z.

   (b) X **does not dominate every node in** p. Suppose the sequence of nodes in the path p is $n_0(= X), n_1, n_2, \ldots, n_k(= Z)$. Since X does not dominate all nodes in p, some of the nodes in p will be in $\mathrm{DF}^+(\{X\})$ **(WHY?)**. Let n_j be the node in $\mathrm{DF}^+(\{X\})$ such that it has the highest value of j. We claim that $X' = n_j$, i.e., n_j dominates Z. Suppose n_j does not dominate $Z = (n_k)$. Then, $\exists i, j < i \leq k$ such that n_j does not dominate n_i. Choose smallest such i. We have, parent of n_i dominated by n_j, but n_i is not (strictly) dominated by n_j. This gives us:

   $$\begin{aligned} & n_i \in \mathrm{DF}(\{n_j\}) \\ \Rightarrow \quad & n_i \in \mathrm{DF}^+(\{n_j\}) \\ \Rightarrow \quad & n_i \in \mathrm{DF}^+(\{X\}) \end{aligned}$$

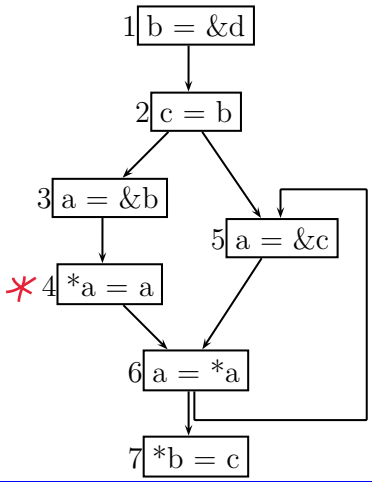
   But this contradicts the fact that j is the largest index such that $n_j \in \mathrm{DF}^+(\{X\})$.

4. [10 + 25] Perform **Flow Sensitive May Points-to** analysis for the program below.

   (a) Give Local Data Flow information for each block in terms of May sets. [Do not write the generic equation, but instantiate it for the current statement.]

   (b) Show result of each iteration of the analysis (Maximum **3** iterations only).

Assume conservative *Must* information for Kill.



Flow Function: *x = y

$$May_{gen} = \{p \to p' \mid x \to p \in May_{IN}, y \to p' \in May_{IN}\}$$
$$May_{kill} = \bigcup_{p' \in Vars} \{p \to p' \mid x \to p \in Must_{IN}\}$$

#	May_{GEN}	May_{KILL}
1	$b \to d$	$b \to any$
2	$c \to p$ $\mid b \to p \in MayIN$	$c \to any$
3	$a \to b$	$a \to any$
4	$p \to p'$ $\mid a \to p \in MayIN$ $\mid a \to p' \in MayIN$	
5	$a \to c$	$a \to any$
6	$a \to p$ $\mid a \to p' \in MayIN$ $\mid p' \to p \in MayIN$	$a \to any$
7	$p \to p'$ $\mid b \to p \in MayIN$ $\mid c \to p' \in MayIN$	

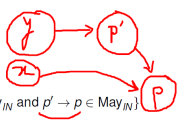
Local Data Flow Information

Flow Function: *x = y

$$May_{gen} = \{p \to p' \mid x \to p \in May_{IN}, y \to p' \in May_{IN}\}$$
$$May_{kill} = \bigcup_{p' \in Vars} \{p \to p' \mid x \to p \in Must_{IN}\}$$
$$Must_{gen} = \{p \to p' \mid x \to p \in Must_{IN}, y \to p' \in Must_{IN}\}$$
$$Must_{kill} = \bigcup_{p' \in Vars} \{p \to p' \mid x \to p \in May_{IN}\}$$

Weak update!! + Strong

Flow Function: x = *y



$$May_{gen} = \{x \to p \mid y \to p' \in May_{IN} \text{ and } p' \to p \in May_{IN}\}$$
$$May_{kill} = \bigcup_{p \in Vars} \{x \to p\}$$

#	May_{IN}	May_{OUT}
1	ϕ	$b \to d$
2	$b \to d$	$b \to d, c \to d$
3	$b \to d, c \to d$	$a \to b, b \to d, c \to d$
4	$a \to b, b \to d, c \to d$	* $a \to b, b \to bd, c \to d$
5	$b \to d, c \to d$	$a \to c, b \to d, c \to d$
6	$a \to bc, b \to bd, c \to d$	$a \to bd, b \to bd, c \to d$
7	$a \to bd, b \to bd, c \to d$	$a \to bd, b \to bd, c \to d, d \to d$

Result of Analysis for Iteration No. 1

bd → d split into b → d, d → d

#	May_{IN}	May_{OUT}
1	ϕ	$b \to d$
2	$b \to d$	$b \to d, c \to d$
3	$b \to d, c \to d$	$a \to b, b \to d, c \to d$
4	$a \to b, b \to d, c \to d$	$a \to b, b \to bd, c \to d$
5	$\mathbf{a \to bd, b \to bd, c \to d}$	$\mathbf{a \to c, b \to bd, c \to d}$
6	$a \to bc, b \to bd, c \to d$	$a \to bd, b \to bd, c \to d$
7	$a \to bd, b \to bd, c \to d$	$a \to bd, b \to bd, c \to d, d \to d$

Result of Analysis for Iteration No. 2, Changes shown in bold

#	May_{IN}	May_{OUT}
1	ϕ	$b \to d$
2	$b \to d$	$b \to d, c \to d$
3	$b \to d, c \to d$	$a \to b, b \to d, c \to d$
4	$a \to b, b \to d, c \to d$	$a \to b, b \to bd, c \to d$
5	$a \to bd, b \to bd, c \to d$	$a \to c, b \to bd, c \to d$
6	$a \to bc, b \to bd, c \to d$	$a \to bd, b \to bd, c \to d$
7	$a \to bd, b \to bd, c \to d$	$a \to bd, b \to bd, c \to d, d \to d$

Result of Analysis for Iteration No. 3 = No. 2

5. [10 + 20] Consider a simple programming language with only two primitive types: **int** and **bool**. The language specification does not require a variable to be declared before use. However, the language requires that for a program to be valid, every variable must be assigned a *unique* type during compilation.

   The following table describes the operations permitted by the language, the corresponding type restrictions on the argument variables and the result of the operation.

Operation	Example	Type Restrictions on Args	Result Type
==	x == y	x and y have to be of the same type (int or bool)	bool
+	x + y	x and y both are int	int
cint	cint(x)	cast to int: x can be of any type	int
cbool	cbool(x)	cast to bool: x can be of any type	bool

   The language has following constants (each constant has a fixed type):

Const Class	Constants	Type
int_constant	$-\infty \ldots -2, -1, 0, 1, 2, \ldots \infty$	int
bool_constant	false, true	bool

   The language has the following constructs and the restrictions:

   - *assignment* statements: Both sides of an assignment must be of the *same* type.
   - Conditionals (*if-else*): The condition of the *if-else* is a single variable of type bool.

   Here are couple of sample programs, one valid and other invalid.

```
// PROGRAM 1
c = n == e;
e = 5;
if (c) {
  d = e + 3;
} else {
  d = cint(c) - 5;
}
// c: bool
// n, d, e: int
```

```
// PROGRAM 2
c = n==e;
e = false;
if (c) {
  d  = c + 3;
} else {
  d  = cint(e) - 5;
}
// invalid program
// c has to be both bool and int!
// n, e: bool    d: int
```

   (a) Design a simple (as-simple-as-possible for you) syntax (context free grammar) for the language described.

   (b) Design a **flow-insensitive** type system to infer (or validate) types of variables for programs written in the above syntax.

   Many variations possible.

6. [10 + 5 + 15 + 5] Perform *Interprocedural Available Expressions* analysis for the following program.

```
main() {
    call p()
    c * d
    call q()
}
```

```
p() {
    a * b
    if(...) then
        call q()
    else
        call q()
    c = 5
}
```

```
q() {
    if(...) then
        a = 5
    else
        b = 6
}
```

Show the following steps to arrive at the answer:

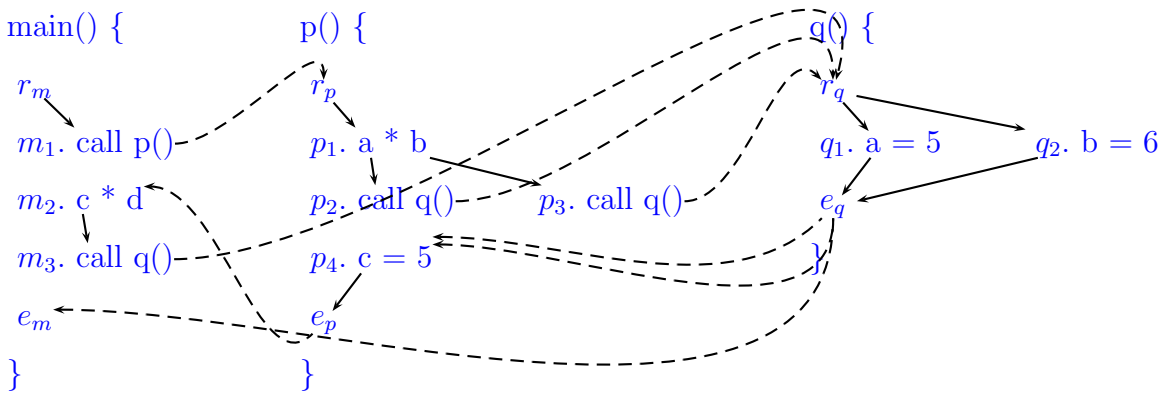
(a) Draw the inter-procedural flow graph (any one of the two representations discussed in class). Clearly label the nodes, and distinguish the E^0 (intra-procedural) and E^1 (inter-procedural) edges.

(b) For each procedure p, show the constraints for $\phi(r_p, n_p)$ for each node n_p in the CFG of the procedure. Recall that r_p is the entry node for procedure p. Assume the existence of flow functions f_0, f_1, id etc for the underlying data flow framework.

(c) Give the fixed-point solution for $\phi(r_p, n_p)$.

(d) List the program points where each of the expressions `a * b` and `c * d` is available.

(a) Inter-Procedural FG:



Solid lines are E^0 edges, dashed lines are E^1 edges.

Since we are tracking **two** expressions, all data flow values will be pairs. The first component of the pair is for $a * b$ and the second for $c * d$. composition of function pairs will be pairwise composition, i.e.:

$$\langle f_1, g_1 \rangle \circ \langle f_2, g_2 \rangle \equiv \langle f_1 \circ g_1, f_2 \circ g_2 \rangle$$

(b) Constraints:

$$\begin{aligned}
\phi(r_m, m_1) &= \langle id_L, id_L \rangle \\
\phi(r_m, m_2) &= \phi(r_p, e_p) \circ \phi(r_m, m_1) \\
\phi(r_m, m_3) &= \langle id_L, f_1 \rangle \circ \phi(r_m, m_2) \\
\phi(r_m, e_m) &= \phi(r_q, e_q) \circ \phi(r_m, m_3)
\end{aligned}$$

$$\begin{aligned}
\phi(r_p, p_1) &= \langle id_L, id_L \rangle \\
\phi(r_p, p_2) &= \langle f_1, id_L \rangle \circ \phi(r_p, p_1) \\
\phi(r_p, p_3) &= \langle f_1, id_L \rangle \circ \phi(r_p, p_1) \\
\phi(r_p, p_4) &= \phi(r_q, e_q) \circ \phi(r_p, p_2) \bigwedge \phi(r_q, e_q) \circ \phi(r_p, p_3) \\
\phi(r_p, e_p) &= \langle id_L, f_0 \rangle \circ \phi(r_p, p_4)
\end{aligned}$$

$$\begin{aligned}
\phi(r_q, q_1) &= \langle id_L, id_L \rangle \\
\phi(r_q, q_2) &= \langle id_L, id_L \rangle \\
\phi(r_q, e_q) &= \langle f_0, id_L \rangle \circ \phi(r_q, q_1) \bigwedge \langle f_0, id_L \rangle \circ \phi(r_q, q_2)
\end{aligned}$$

(c) Solutions:

$$\begin{aligned}
\phi(r_m, m_1) &= \langle id, id \rangle \\
\phi(r_m, m_2) &= \langle f_0, f_0 \rangle \\
\phi(r_m, m_3) &= \langle f_0, f_1 \rangle \\
\phi(r_m, e_m) &= \langle f_0, f_1 \rangle
\end{aligned}$$

$$\begin{aligned}
\phi(r_p, p_1) &= \langle id, id \rangle \\
\phi(r_p, p_2) &= \langle f_1, id \rangle \\
\phi(r_p, p_3) &= \langle f_1, id \rangle \\
\phi(r_p, p_4) &= \langle f_0, id \rangle \\
\phi(r_p, e_p) &= \langle f_0, f_0 \rangle
\end{aligned}$$

$$\begin{aligned}
\phi(r_q, q_1) &= \langle id, id \rangle \\
\phi(r_q, q_2) &= \langle id, id \rangle \\
\phi(r_q, e_q) &= \langle f_0, id \rangle
\end{aligned}$$

(d) Availability:

- $a * b$: $\mathsf{OUT}(p_1)$, $\mathsf{IN}(p_2)$, $\mathsf{IN}(p_3)$
- $c * d$: $\mathsf{OUT}(m_2)$, $\mathsf{IN}(m_3)$, $\mathsf{OUT}(m_3)$

Table 1: Access Ranges for Variables. **T** denotes true, **F** denotes false.

Program Point	Is π In Access Range Of		
π	Variable A?	Variable B	Variable C
IN($S1$)	F	F	F
OUT($S1$)	T	F	F
IN($S2$)	T	T	F
OUT($S2$)	T	T	F
IN($S3$)	T	T	F
OUT($S3$)	T	T	F
IN($S4$)	T	T	F
OUT($S4$)	T	T	F
IN($S5$)	T	F	F
OUT($S5$)	F	F	F
IN($S6$)	F	F	F
OUT($S6$)	F	F	T

Table 2: Access Ranges for Sets of Variables. **T** denotes true, **F** denotes false.

Program Point	Is π In Access Range Of		
π	Set {A, B}?	Set {B, C}	Set {C, A}
IN($S1$)	F	F	F
OUT($S1$)	T	F	T
IN($S2$)	T	T	T
OUT($S2$)	T	T	T
IN($S3$)	T	T	T
OUT($S3$)	T	T	T
IN($S4$)	T	T	T
OUT($S4$)	T	T	T
IN($S5$)	T	T	T
OUT($S5$)	F	T	T
IN($S6$)	F	T	T
OUT($S6$)	F	T	T

Write your roll number at the top of this page and submit along with the answer-sheet.