CSC410 Assignment 2 Question 1

Liang Chen 1005735126
Joshua Han 1005109669
Yiqi Zhi 1004892725

1.1. Solution:

Given the definition of bit vector frameworks:
L = (P(D), $\sqcap$) for some finite set D where P(D) is the powerset of D and $\sqcap$ is either set union ($\cup$) or set intersection ($\cap$);
F: P(D) $\rightarrow$ P(D)
$\forall$S $\subseteq$ D, $\exists$ some K, G $\subseteq$ D s.t. f(S) = (S $\cap$ K) $\cup$ G where f $\in$ F
WTS Live Variable Analysis is a bit vector framework

Based on the set up of Live Variable Analysis, we have:
D: the set of all variables in a program, and the number of variables in the program is finite, saying n
Data flow fact P(D): the power set of D, with size 2^n, i.e. each element in this power set P(D) is a possible set of variables in this program. Besides, since any element in P(D) is a set of some variables, it is also a subset of D, i.e. an element of P(D) $\subseteq$ D
$\sqcap$: the set union
Domain: semilattice (P(D), $\cup$, P(D))
Direction: backward
F: IN[B] = f(OUT[B]) where B is some basic block in the program, specifically speaking,

$$IN[B] = (OUT[B] / Kill[B]) \cup Gen[B]$$

OUT[B]: the set of live variables at the exit point of basic block B. Since for every point of this program, the state of live variables $\in$ P(D), we have OUT[B] $\in$ P(D), i.e. OUT[B] $\subseteq$ D
IN[B]: the set of live variables at the entry point of basic block B. Since for every point of this program, the state of live variables $\in$ P(D), we have IN[B] $\in$ P(D), i.e. IN[B] $\subseteq$ D
Kill[B]: the set of variables killed in basic block B, i.e. elements that were written/redefined in block B. Kill[B] $\in$ P(D), i.e. Kill[B] $\subseteq$ D

$$Kill[B]^c = D \backslash Kill[B]$$
$$Kill[B]^c \in P(D)$$
$$Kill[B]^c \subseteq D$$

Gen[B]: the set of variables that were read, $Gen[B] \in P(D)$, i.e. $Gen[B] \subseteq D$

Assume for a given basic block B, OUT[B] of this block is an arbitrary variable set. Based on the specified statements in this block, we always can find Kill[B] and Gen[B] Correspondingly, $Kill[B]^c$ also exists. And by set operation properties, we have:

$$(OUT[B] \setminus Kill[B]) \cup Gen[B] = (OUT[B] \cap Kill[B]^c) \cup Gen[B]$$

Thus, we have:

- Meet the definition of lattice for bit vector frameworks:
  $L = (P(D), \sqcap)$ for finite set D the set of all variables in a program, where P(D) is the powerset of D and $\sqcap$ is the set union ($\cup$) in live variable analysis (semilattice $(P(D), \cup, P(D))$)
- Meet the Transfer function's requirements:
  $F: P(D) \rightarrow P(D)$, where inputs are the set of live variables at exit of basic block and outputs are the set of live variables at the entry of basic block, both IN[B] and $OUT[B] \subseteq D$;
  We always can find Kill[B], $Kill[B]^c$ and Gen[B] for any basic block, where Kill[B], $Kill[B]^c$ and Gen[B] all $\subseteq D$;
  $IN[B] = (OUT[B] \cap Kill[B]^c) \cup Gen[B]$ holds

Q.E.D

1.2 Solution:

Given:
bit vector framework is a monotone framework
L = (P(D), $\sqcap$) for some finite set D where P(D) is the powerset of D and $\sqcap$ is either set union ($\cup$) or set intersection ($\cap$);
F: P(D) → P(D)
$\forall S \subseteq D, \exists$ some K, G $\subseteq$ D s.t. f(S) = (S ∩ K) ∪ G where f ∈ F
WTS bit vector framework is a Distributive framework
WTS $\forall$x, y , f(x $\sqcap$ y) = f(x) $\sqcap$ f(y).

Since $\sqcap$ is either a set union ($\cup$) or a set intersection ($\cap$), we want to prove by cases.

Case 1: $\sqcap$ is set union ($\cup$)

$f(x \sqcap y) = f(x \cup y)$
$f(x \cup y) = ((x \cup y) \cap K) \cup G$ for some K, G $\subseteq$ D
$f(x \cup y) = ((x \cap K) \cup (y \cap K)) \cup G$
$f(x \cup y) = ((x \cap K) \cup G) \cup ((y \cap K) \cup G)$
$f(x) = (x \cap K) \cup G$ by definition of bit vector framework
$f(y) = (y \cap K) \cup G$ by definition of bit vector framework
Plug back into original function of $f(x \cup y)$, we have
$f(x \cup y) = f(x) \cup f(y)$
$f(x \cup y) = f(x) \sqcap f(y)$
$f(x \sqcap y) = f(x) \sqcap f(y)$

Case 2: $\sqcap$ is set intersection ($\cap$)

$f(x \sqcap y) = f(x \cap y)$
$f(x \cap y) = ((x \cap y) \cap K) \cup G$ for some K, G $\subseteq$ D
$f(x \cap y) = ((x \cap K) \cap (y \cap K)) \cup G$
$f(x \cap y) = ((x \cap K) \cup G) \cap ((y \cap K) \cup G)$
$f(x) = (x \cap K) \cup G$ by definition of bit vector framework
$f(y) = (y \cap K) \cup G$ by definition of bit vector framework
Plug back into original function of $f(x \cap y)$, we have
$f(x \cup y) = f(x) \cap f(y)$
$f(x \cup y) = f(x) \sqcap f(y)$
$f(x \sqcap y) = f(x) \sqcap f(y)$

Thus, we draw the conclusion that bit vector frameworks are distributive in both cases.

Q.E.D

1.3 Solution

Counterexample of an unreal-used analysis which is distributive but not bit-vector framework (A.K.A: PMS):

Possible maximum number of assignment statements executed in a program with no loops (i.e. no while/ for loops)

- Definition:
  We say the PMS is the maximum possible number of assignment statements that the program could execute along a path from entry to exit.
- Dataflow fact D:
  {0, 1, …, n}. i.e. n is the total number of assignment statements in the given program, a finite non-negative natural number.
- Meet operation ⊓: max, return the maximum value between the two operands (i.e. for input x and y, if x >= y then x ⊓ y returns x, or y otherwise)
- Lattice: L = (D, max)
- Direction: forward
- Initialization: IN[B] = 0
- Boundary: OUT[Entry] = 0
- Transfer function:
  Input IN[B]: before entering block B, there are multiple possible paths that can reach this point. For each path, the program will execute a different number of assignment statements, input would be the maximum number of assignment statements being executed amongst all these different paths
  Output OUT[B]: maximum possible number of assignment statements executed when exiting block B

  OUT[B] = IN[B] + 1 if the statement in block B is an assignment
  OUT[B] = IN[B] otherwise
  IN[B] = max (all OUT[P]) where P are all predecessors of B

WTS:
- Transfer function is monotonic
- this framework is distributive but not a bit-vector one

Proof 1:
WTS Transfer function is monotomic
ETS:

$x \sqsubseteq y \Longrightarrow f(x) \sqsubseteq f(y)$

Since we have: $x \sqsubseteq y \Leftrightarrow x \sqcap y = x$
Also we defined: $x \sqcap y = max(x, y)$
We gained: $x \sqcap y = max(x, y) = x$, i.e. $\sqsubseteq = \geq$
Thus, for lattice $L = (D, max)$, $\sqsubseteq = \geq$

Now back to the proof, we are given $x \sqsubseteq y$ and thus have: $x \geq y$
By the definition of transfer function:
if in this node an assignment is executed, $f(x) = x + 1$; $f(y) = y + 1$; and thus $f(x) \geq f(y)$ holds;
if in this node no assignment is executed, $f(x) = x$; $f(y) = y$; and thus $f(x) \geq f(y)$ still holds;

We can say: if $x \geq y$, then $f(x) \geq f(y)$.
Thus, we have:
$x \sqsubseteq y \Longrightarrow f(x) \sqsubseteq f(y)$

Proof 2:
WTS this framework is distributive but not a bit-vector one
We can tell from the transfer function that the transfer function in our analysis is not a set operation (inseparable) and thus CAN NOT be expressed as the format of bit vector transfer function, which is obviously not a bit vector framework.
However, we still need to prove it is distributive
ETS: $f(x \sqcap y) = f(x) \sqcap f(y)$

Case 1: assignment statement is executed in block B
$x \sqcap y = max(x, y)$
$f(x \sqcap y) = f(max(x, y)) = max(x, y) + 1$
$f(x) = x + 1$
$f(y) = y + 1$
$f(x) \sqcap f(y) = max(f(x), f(y)) = max(x + 1, y + 1) = max(x, y) + 1$
Thus, $f(x \sqcap y) = f(x) \sqcap f(y)$

Case 2: no assignment statement is executed in block B
$x \sqcap y = max(x, y)$
$f(x \sqcap y) = f(max(x, y)) = max(x, y)$
$f(x) = x$
$f(y) = y$

$f(x) \sqcap f(y) = \max(f(x), f(y)) = \max(x, y)$
Thus, $f(x \sqcap y) = f(x) \sqcap f(y)$

From the two cases shown above, we proved $f(x \sqcap y) = f(x) \sqcap f(y)$ always hold, i.e. it is distributive framework.

(Note: there are also some other real-life analyses such as truly-live variables and possibly-uninitialized variables, which are IFDS (interprocedural, finite, distributive, subset problems) but not bit-vector problems as well)

Q.E.D