Any comments/feedback welcome...

Q1.

A) Property Lattice *D*: Var\* -> P(Label\* × Var\*), σ1 <= σ2 iff ∀x. σ1(x) \subseteq σ2(x), (σ1 lub σ2) (x) = σ1(x) U σ2(x), T = λx. Label\* × Var\*, ⊥= λx. Φ

Transfer function f\_[x:=a]l(σ) = σ[x -> {(l, y) | y \in fv(a)}], f\_[skip]l(σ) = σ, f\_[b]l(σ) = σ

Flow = flow(S\*) (forward, definition see slides)

Extremal Label: {init(S\*)}, value = ⊥

Alternate Answer:

Property Lattice D : P(Var\*) x Lab\*

L should be P(Var x Var x Lab) (since question says give at each point which variable depends on which variable and the assignment label)

The lower than operator is set inclusion.

Least upper bound is set union.

Kill [(x := a)l] = { (x, v, l’) | v <- FV(a), [(B)l’] in S\* } (or {(x, ?, 0)} if label is in init(S\*)

Kill (skip) = kill([(b)l]) = empty set

Gen ([(x:=a)l]) = { (x, v, l) | v <- FV(a) }

Gen (skip) = gen (b) = empty

RdEntry and exit defined like for Reaching definition analysis

B)

[x := y]1

While [x >= y]2 do (

If [y >= x]3 then [y := x]4 else [x := y]5

)

[skip]6

Flow = {(1, 2), (2, 3), (2, 6), (3, 4), (3, 5), (4, 2), (5, 2)}, flowR = reversed

C)

|  |  |  |  |
| --- | --- | --- | --- |
| l | F\_l | DAentry(l) | DAexit(l) |
| 1 | λσ.σ[x -> {(1, y)}] | ⊥ | F1(DAentry(1)) |
| 2 | λσ. σ | lub(DAexit(1,4,5)) | DAentry(2) |
| 3 | λσ. σ | DAexit(2) | DAentry(3) |
| 4 | λσ.σ[y -> {(4, x)}] | DAexit(3) | F4(DAentry(4)) |
| 5 | λσ.σ[x -> {(5, y)}] | DAexit(3) | F5(DAentry(5)) |
| 6 | λσ. σ | DAexit(2) | DAentry(6) |

D)

Transfer function f\_[x:=a]l(σ) preserves entries under x which label is not an assignment (in other words, preserve all (l, y) derived from boolean exprs)

f\_[b]l(σ)(x) = {σ(x) U {(l, y) | y \in fv(b), x =/= y} if x \in fv(b); σ(x) otherwise}

A

In my version, Lattice is P(Var x Var x Lab)

Eg for statement labelled with one DA exit(1) = DAentry(1)U(x,y,1)

I.e. transfer function is :  
f\_[x:=a]l(σ) = σ[{(x, y, l) | y \in fv(a)}],

For meet analysis, the only change will be the transfer function on [b].

f\_[b]l(σ) = {(x,y,l)| x,y is FV(b), x != y}

Edit:

In above case, don’t you also need to change the kill function so that on kill(x:=a) includes also {(y,x,l’) | y in FV(a) ^ [B]l’ in S\*} ? So when you assign the boolean expression meeting is not preserved? (since you reassigned the variable after all)

Q2  
a) 12 labels in total, notice Herbert missed a bracket in (fn x => (x x)), result is fn z => 1

B) Denote ρ = λx. ⊥, ρ1 = ρ[f -> [fz, ρ]], ρ2 = ρ1[x -> [fy, ρ1]], ρ3 = ρ1[y -> [fy, ρ1]], ρ4 = ρ1[y -> [fz, ρ]]

eval(ρ, S) = eval(ρ1, ..11) since eval(ρ, fz) = [fz, ρ]

To solve eval(ρ1, ..11), need to solve eval(ρ1, ..9) and eval(ρ1, ..10)

To solve eval(ρ1, ..9), since eval(ρ1, fx) = [fx, ρ1], eval(ρ1, fy) = [fy, ρ1], eval(ρ1, ….9) = eval(ρ2, 5)

To solve eval(ρ2, ..5), since eval(ρ2, x3) = eval(ρ2, x4) =[fy, ρ1], eval(ρ2, ..5) = eval(ρ3, y7) = [fy, ρ1], so eval(ρ1, ..9) = [fy, ρ1]

Also, eval(ρ1, f10) = [fz, ρ], so eval(ρ1, ..11) = eval(ρ4, y7) = [fz, ρ]. Overall, eval(ρ, S) = [fz, ρ]

C) (Use U for union, < for \subseteq, => for implication)

C\*[S] = C\*[fz] U C\*[..11] U {C(2) < r(f), C(11) < r(f)}

C\*[fz] = {{fz} < C(2)} U C\*[1] (which is empty)

C\*[..11] = C\*[..9] U C[f 10] U {{fx} < C(9) => C(10) < r(x), {fx} < C9 => C5 < C11, {fy} < C9 => C10 < r(y), {fy} < C9 => C7 <= C11, {fz} < C9 => C10 < r(z), {fz} < C9 => C1 < C11}

… (similar pattern until everything is done)

D) (not too sure) W = [c6, c8, c2], D[c6] = {fx}, D[c8] = {fy}, D[c2] = {fz}, D[otherwise] = emptyset

Solution:

C(1), r(z) = empty set

C(2), c(7), c(10), c(11),c(12),r(y) = fz

rest = fy

Q3

A)

Concrete Domain *P*([-999, 999]), Abstract domain *P*({-, 0, +})

T = {-, 0, +}, ⊥ = Φ, a <= b iff a \subseteq b, a lub b = a union b

Define s(n) {+: if n > 0; 0 if n = 0; - if n < 0}, s’(n) = {[1, 999] if n = +; {0} if n = 0; [-999, -1] if n < 0}

α(X) = {s(n) | n \in X}

γ(X) = U{s’(k) | k \in X}

B)

Abstract Domain: P({-, 0, +}) × P({1, 2, 3})

Definition of lattice follows homomorphism (omit here)

Define d(n) = {1 if n \in [-9, 9]; 2 if n \in [-99, -10] U [10, 99]; 3 if n \in [-999, -100] U [100, 999]},

d’(k) = {[-9, 9] if k = 1; [-99, -10] U [10, 99] if k = 2; [-999, -100] U [100, 999] if k = 3 }

α'(X) = (α(X), {d(n) | n \in X})

γ’((X1, X2)) = (γ(X1), {d’(k) | k \in X2})

C)

f#((X1, X2)) = (P(X1), X2) where P(X) = {X if X not contain -; (X \ {-}) U {+} otherwise}

D) Anyone knows what “calculations” Herbert asks us to show here?

E) ~~Got stuck with U(x <- x + 1 mod 3) for F4, since x \in {0, 1, 3}, if x = 1 what would happen?~~

Given x \in {0, 1, 2} instead:

6 labels, flow = {(1, 1, 2), (2, 1, 3), (2, 1, 6), (3, 1/3, 4), (3, 2/3, 5), (4, 1, 2), (5, 1, 2), (6, 1, 6)}

Operators:

F1 = U(x <- 0) = 3x3, first column 1

P2 = P(true) = I

P2⊥ = P(false) = 0

F3 = I

F4 = U(x <- x + 1 mod 3) = 3x3, I3 rotate right by 1

F5 = U(x <- x – 1 mod 3) = 3x3, I3 rotate left by 1

F6 = I

T = U(x <- 0) ⊗ E(1, 2) + P(true) ⊗ E(2, 3) + P(false) ⊗ E(2, 6) + 1/3 \* I ⊗ E(3, 4) + 2/3 \* I ⊗ E(3, 5) + U(x <- x + 1 mod 3) ⊗ E(4, 2) + U (x <- x – 1 mod 3) ⊗ E(5, 2) + I ⊗ E(6, 6)

Dimension = 18x18 operator (18=3(domain size) \* 6(labels count))