1.a.

[ x:= y]1;

while [x>=y]2 do (

if [y>=0]3

then [y:=x]4

else [x:=y]5

);

[skip]6

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

flow^R = { (2,1), (6,2), (3,2), (4,3), (5,3), (2,5), (2,4) }

b.

kill(1) = {(x, ?), (x, 5), (x, 1)}

kill(2) = {}

kill(3) = {}

kill(4) = {(y, ?), (y, 4)}

kill(5) = {(x, ?), (x, 5), (x, 1)}

kill(6) = {}

gen(1) = {(x, 1)}

gen(2) = {}

gen(3) = {}

gen(4) = {(y, 4)}

gen(5) = {(x, 5)}

gen(6) = {}

|  |  |  |
| --- | --- | --- |
| Label | RD Entry | RD Exit |
| 1 | (x,?) (y,?) | RDEntry(1) \ kill(1) U gen(1) |
| 2 | RDExit(1) U RDExit(4) U RDExit(5) | RDEntry(2) \ kill(2) U gen(2) |
| 3 | RDExit(2) | ... |
| 4 | RDExit(3) | ... |
| 5 | RDExit(3) | ... |
| 6 | RDExit(2) | Same as above |

c.

|  |  |  |
| --- | --- | --- |
| Label | RD Entry | RD Exit |
| 1 | (x,?) (y,?) | (y,?) (x,1) |
| 2 | (y,?) (x,1) (y,4) (x,5) | (y,?) (x,1) (y,4) (x,5) |
| 3 | (y,?) (x,1) (y,4) (x,5) | (y,?) (x,1) (y,4) (x,5) |
| 4 | (y,?) (x,1) (y,4) (x,5) | (x,1) (y,4) (x,5) |
| 5 | (y,?) (x,1) (y,4) (x,5) | (y, ?) (y,4) (x,5) |
| 6 | (y,?) (x,1) (y,4) (x,5) | (y,?) (x,1) (y,4) (x,5) |

d.

|  |  |  |
| --- | --- | --- |
| Label | Kill | Gen |
| 1 | {x} | {y} |
| 2 | {} | {x}{y} |
| 3 | {} | {y} |
| 4 | {y} | {x} |
| 5 | {x} | {y} |
| 6 | {} | {} |

|  |  |  |
| --- | --- | --- |
| Label | Entry | Exit |
| 1 | LVExit(1) \kill(1) U gen(1) | LVEntry(2) |
| 2 | LVExit(2) ... | LVEntry(3) U LVEntry(6) |
| 3 | LVExit(3) ... | LVEntry(4) U LVEntry(5) |
| 4 | LVExit(4) ... | LVEntry(2) |
| 5 | LVExit(5) \kill(5) U gen(5) | LVEntry(2) |
| 6 | LVExit(6) \kill(6) U gen(6) | y |

E

|  |  |  |
| --- | --- | --- |
| Label | Entry | Exit |
| 1 | {y} | {x,y} |
| 2 | {x,y} | {x,y} |
| 3 | {x,y} | {x,y} |
| 4 | {x} | {x, y} |
| 5 | {y} | {x, y} |
| 6 | {y} | {y} |

Changing the condition to x>y will have no effect on the analysis as the RD and LV analysis only deal with the variable conditions rather than taking into account the boolean checks. RD might be useful in determining that the loop will never be entered as every reachable definition of x indicates that x = y

Q2. a

S ::= [x := a]l | [skip]l | S1 ; S2 | if [b]l …..| choosel S1…. | combineI S1 … | while [b]l do S

Init(combinel S1 ….) = l

final(combine) =

The rest can be found in lecture slides

b. 5

Property lattice L = P({-,0,+})

The state space : state = Var --> L

Top element:

Partial order:

Least upper bound :

2022 edit:

Above thing doesn’t look correct. The signed analysis is a forward must-analysis, so the least upper bound should be some set intersection n\* (like normal set union, but coalescing, such that { x >= 0 } n\* { x > 0 } = { x >= 0 } and {x >= 0 } n\* { x = 0 } = { x >= 0 }). Also then the less than operator should be similar to subset inclusion, but coalescing (and from right to left, like for AE analysis). Such that { x = 0 } ⊇\* {x <= 0, y > 0 } and also { x < 0 } ⊇\* { x<= 0, y < 0}

c.

I think we can use the coursework 1 answer for this, instead of parity(n), we can use sign(n)

(correct me if I am wrong): If sign analysis is a forward analysis and the extremal labels = {init(S)} and the initial value should be ? I thinks its like the constant propagation analysis, extremal label start with \lambda x. T

d.

if (x > 0)1 then (choose2 [x:=0]3| [x:=1]4) else (combine5 [x:=0]6| [x:=1]7) skip

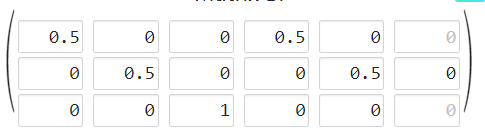
flow: (1,1,2),(1,1,5),(2,1/2,3),(2,1/2,4),(3,1,8),(4,1,8),(5,1/2,7),(6,1/2,8),(7,1/2,8), (6,½,7),(7,½,6)

flow also needs (5, 1/2, 6)

4.

1. Coursework 2 Q2
2. Concrete X =P({0,1,..,5}) Abstract X = P({0,1,2})

A = 

G = 

c.

x := 0

T(<l1,p,l2>) = U(x←0)⊗E(l1,l2)

x ?= {0,1,2,3,4,5}

T(⟨l1, *p*, l2⟩) = (1/6)[U(x←0)+ U(x←1)+…+ U(x←5)] ⊗E(l1,l2)

x := (x+1) mod 3

T(<l1,p,l2>) = U(x←(x+1) mod 3)⊗E(l1,l2)