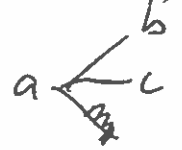
HW 19

David Pan

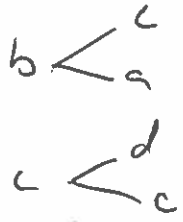
1. (a) a) No. Nobody from x loves everyone from y. loves(x,y) loves(y,x)

b) Yes. C is loved by everyone in x.



c) Yes. Everyone in x loves C in y.

d) Yes. Everyone in y is loved by somebody in x.



e) Yes. Somebody in x loves somebody in y.

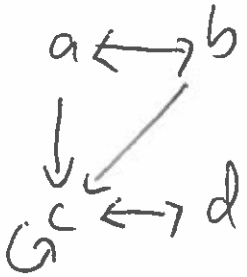
f) ~~Yes~~ No. There is nobody ~~Somebody~~ in x who does not love anyone ~~everyone~~ in y. d – c

g) ~~No~~ Yes. There is nobody ~~Somebody~~ in x who does not love anyone ~~everyone~~ in y.

(b)

Type 1	$x \geq y$	$x = y$
×	✓	✓
×	×	✓
×	✓	✓

(c)



	In	out
a	1	2
b	1	2
c	4	2
d	1	1

1. $Out = 4$
2. $In = 4$
3. $\exists (out > 0)$
4. $\exists (in > 0)$

2

1) Prove $\exists x.\ sum(x, x, ssO)$

Let $x = sO \Rightarrow sum(sO, sO, ssO)$

$sum(sO, O, sO)$ True by axiom 1.

2) Prove $\forall x.\ sum(O, x, x)$

$sum(O, O, O)$ Let $x = s^n(O)$

$sum(O, sO, sO)$

$sum(O, ssO, ssO)$

$\vdots$

$sum(O, s^n(O), s^n(O)) \Rightarrow sum(O, x, x)$

(3)

(9) -x y -y 4 -x x f -y 4

mar_ending: [redacted]

mar_so_far: [redacted]

mar(0,2) = [redacted]

mar(0,2) = [redacted]

mar(0,1) = 1

mar(0,1) = 1

mar(0,-2) = 0

mar(1,-2) = 1

mar(0,4) = 4

mar(1,4) = 4

mar(0,3) = 3

mar(3,4) = 4

mar(0,5) = 5

mar(1,5) = 5

mar(0,6) = 6

mar(5,6) = 6

mar(0,1) = 1

mar(6,1) = 6

mar(0,5) = 5

mar(6,5) = 6

(b) No. There is no multi. branched recursion. Each iteration directly follows the one before it.

(c) Yes. We break the problem down into smaller subproblems, store the solutions to these subproblems, and build a final solution from these. The storing in this case is just remembering the mar length at each point.

(d) I do not see an advantage to memoization here, unless we are trying to find the mar multiple times.

The recursion graph is the same.

4. (a)

bs(3, xs, 0, 9) mid = 4 xs[4] = 7

⇓

bs(3, xs, 0, 4) mid = 2 xs[2] = 4

⇓

bs(3, xs, 0, 2) = 1

(b) No. There is no multi-branched recursion.

(c) No. ~~There is~~ It
The problem isn't really broken down into simpler problems, instead, we're just making our array limits tighter.

(d) The memoized version will have the same recursion graph in the first call. It only helps if we search for the same value in the same list.

(e) It's useful if we are going to be searching the same value in the same list multiple times.

(f) (left + right) / 2 might exceed an int's capacity