This course wants us to have wrist arthritis...

Q1

a) fn = {b, ~~c~~ e, f}; fv = {z, x}; ii) fn = {a, b, f, d}; fv = {y, y’, z}

b) Denote Q = !D<a,a,a> | !K<a>, should get (v b)Q or (v b)(-a<b> | Q) or (v b)(-a<b> | -a<b> | Q) or …

Anyone knows how to formally describe this? And how to properly show non-determinism in reduction?

C) I) One extra rule P === Q => -u<a>.P === -u<a>.Q (Cong Out)

ii) For MS pi-calculus, the (Comm) rule should be replaced by:

-a<b>.P | a(x).Q -> P|Q{b/x}

(Use structural induction over definition of MS reduction relation, anyone knows if this is ok?)

Denote A(P, Q) stands for P -> Q and fn(Q) <= fn(P)

Base case (Comm): P = -a<b>.P1 | a(x).P2; Q = P1 | P2{b/x},

fn(P) = {a, b} U fn(P1) U fn(P2)

fn(Q) = fn(P1) U fn(P2{b/x}) = fn(P1) U fn(P2) U {a} by theorem 2

Thus by subset relation, fn(Q) <= fn(P)

Inductive case (Par): Assume P, Q satisfies A(P, Q), take arb process R

fn(P | R) = fn(P) U fn(R), fn(Q | R) = fn(Q) U fn(R), by subset relation, fn(Q | R) <= fn(P | R)

Inductive case (Res): Assume P, Q satisfies A(P, Q), take arb variable x

fn((v x) P) = fn(P) \ {x}; fn((v x) Q) = fn(Q) \ {x}, by subset relation, …

Inductive case (~~Cong~~ Struct): Assume P, Q satisfies A(P, Q), take arb processes P’ Q’ s.t. P === P’ and Q === Q’,

By theorem 1, fn(P’) = fn(P) and fn(Q’) = fn(Q), thus fn(Q’) <= fn(P’)

Overall, by structural induction over monadic synchronous pi-calculus' reduction relation, we conclude that if P → Q then fn(P) ⊇ fn(Q).

Q2

A) Expand T and S by 1 layer, then use subtyping with coinduction

B) P\_bob = -A<”Bob Smith”>.A ◁ positive. A(date). A ◁ negative. 0

Ii) wtf this is a fat derivation

Q3

A) Anyone... For methodology see slides p8 in mpst, for difference I can only think of multiparty session type inspects >= 2 participants where binary session type types binary session (lol)

B)

I) Ga = μt. B + {orange: C?[bool]; t banana: C?[bool]; t}

Gb = μt. A & {orange: D![nat]; t banana: D![nat]; t}

Gc = μt. A![bool].t

Gd = μt. B?[nat].t

ii) Pa = μX. If true then B ◁ orange. C(xb). X else B ◁ banana. C(xb). X

Alternative: Pa = uX. B <| orange . C(xb). X. Since this has type ut. B sel {orange: C(xb). t} which is basically a subtype of Ga select above (there’s one less branch to derive like this, but fml)

Pb = μX. A ▷ { orange: -d<1>.X apple: -d<1>.X }

Pc = μX. -A<true>.X

Pd = μX. -B(xn). X

iii) A :: PA | B :: PB | C :: PC | D :: PD

-> eventually u get back to where u start (no matter orange or banana u picked)

iv) MTY -> 4 derivations, have fun