1

A

i) fn: c, e; fv: x’, y, x

ii) fn: {a, b, c} fv: {z}

b. i) Ends with (v b, e) (e\_bar<e> | b(x\_3).b(x\_4).x3\_bar <x4>) | (v c,d) (c\_bar<c> | d\_bar<d>) | TNN<a>

ii) ->\* a\_bar<b> | …. | a\_bar<b> | a\_bar <c> | SWF<a,b,c,>

c.

SHOW fn((va)(P|Q)) = fn((va) P|Q):

fn( (va) (P | Q)) = fn(P)\{a} U fn(Q) = fn ( (va)P | Q)

SHOW P=Q => fn(P) = fn(Q):

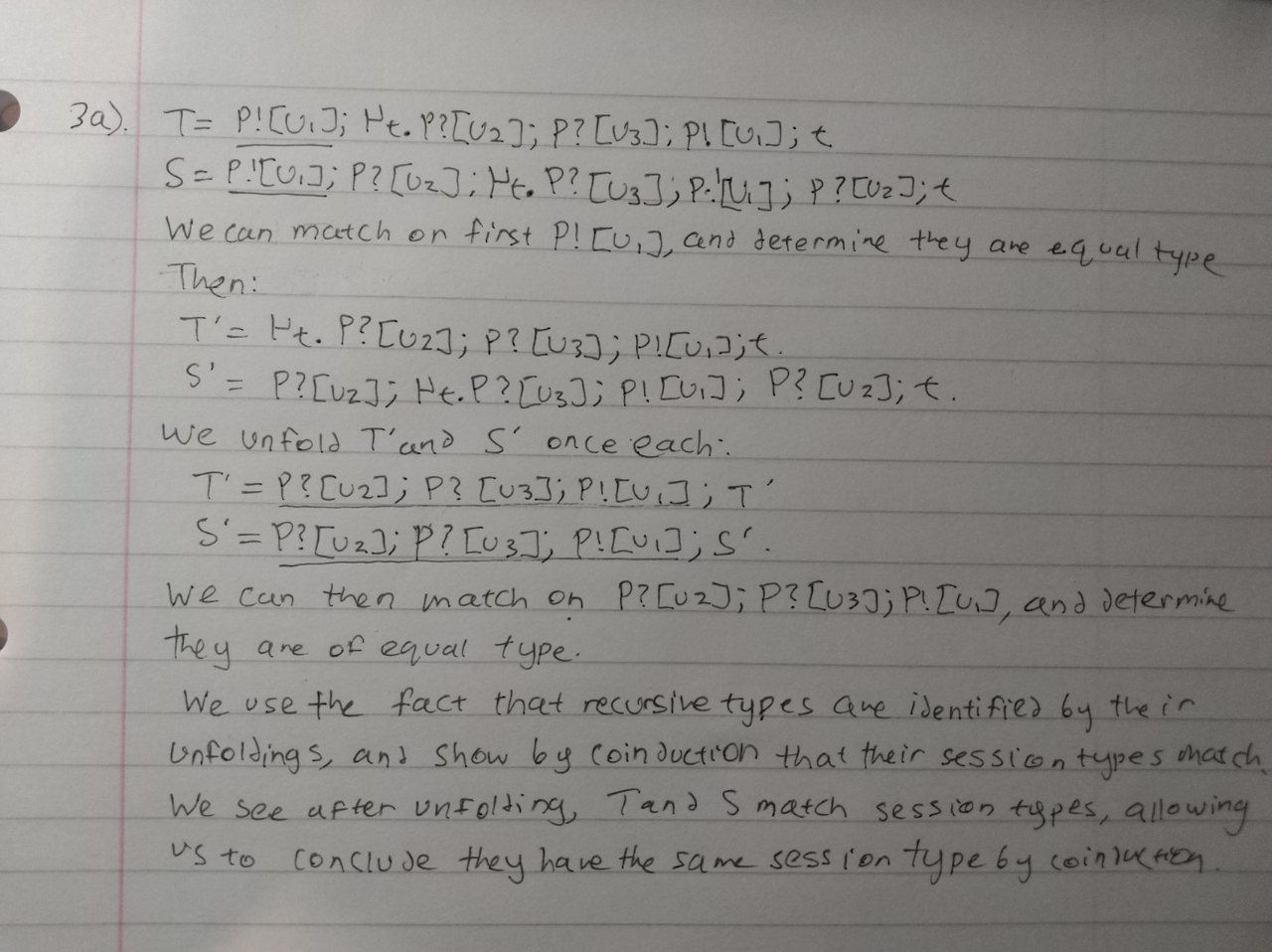
We go over the rules of structural congruence, proving them by rule. Start with the base cases (reflexivity, Associativity, Commutativity, Zero, Res Nil, Res Res , Res Par) The prior part proves the rule for Res Par. Then prove the inductive cases (rest of rules).

^ If they ask this in the exam, refer to piazza post @27. But its incredibly long so I don’t think we’ll only have to give a brief explanation

2

1. Polyadic synchronous must have continuation after output and can have multiple communication values while monadic asynchronous cannot have cont after outpus and has only one communication value.
2. [P ⊕ Q] = (v c,d) (c\_bar<d> | c(d).P | c(d).Q)

3

1. 
2. P\_Bob = Alice\_bar<”The Winter’s Tale”> . Alice(quote).

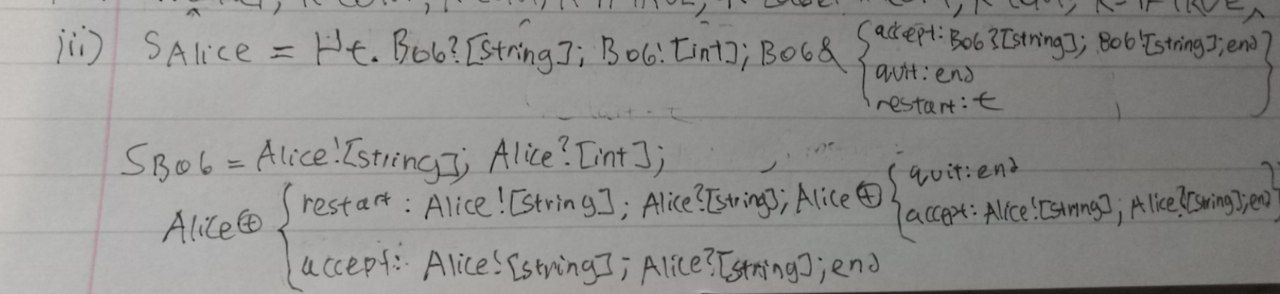
If quote > 2000

then Alice <| restart.Alice\_bar<”The Taming of the Shrew”>.Alice(quote).

if quote > 800 then Alice <| quit.0 else Alice <| accept.Alice\_bar<”addr”>.Alice(date).0

else Alice <| accept.Alice\_bar<”addr”>.Alice(date).0

1. Rules used: R-Cong, R-Comm, R-Comm, R-IfTrue, R-Label, R-Comm, R-Comm, R-IfTrue,R-Label



4.

* 1. Take participants A, B, such that +A=B, A=+B  
     S\_a = µt. B![int];B ⊕{yes: t, no: end}

S\_b = µt. A?[int];A&{yes: t, no: end}

* 1. Binary sessions have 2 participants (e.g. PAlice = Bob<x>.0 & PBob = Alice(x).0) while multiparty have more than that (e.g. PAlice = Bob<x>.0 & PBob = Alice(x).Carol<x>.0 & PCarol = Bob(x).0)

