

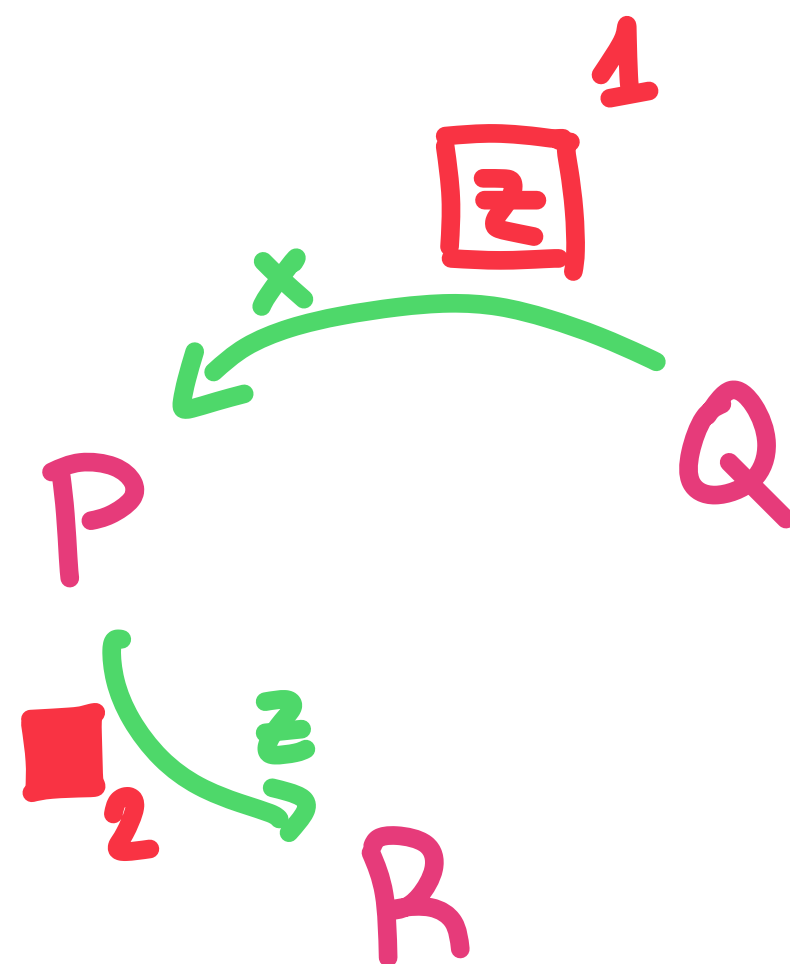
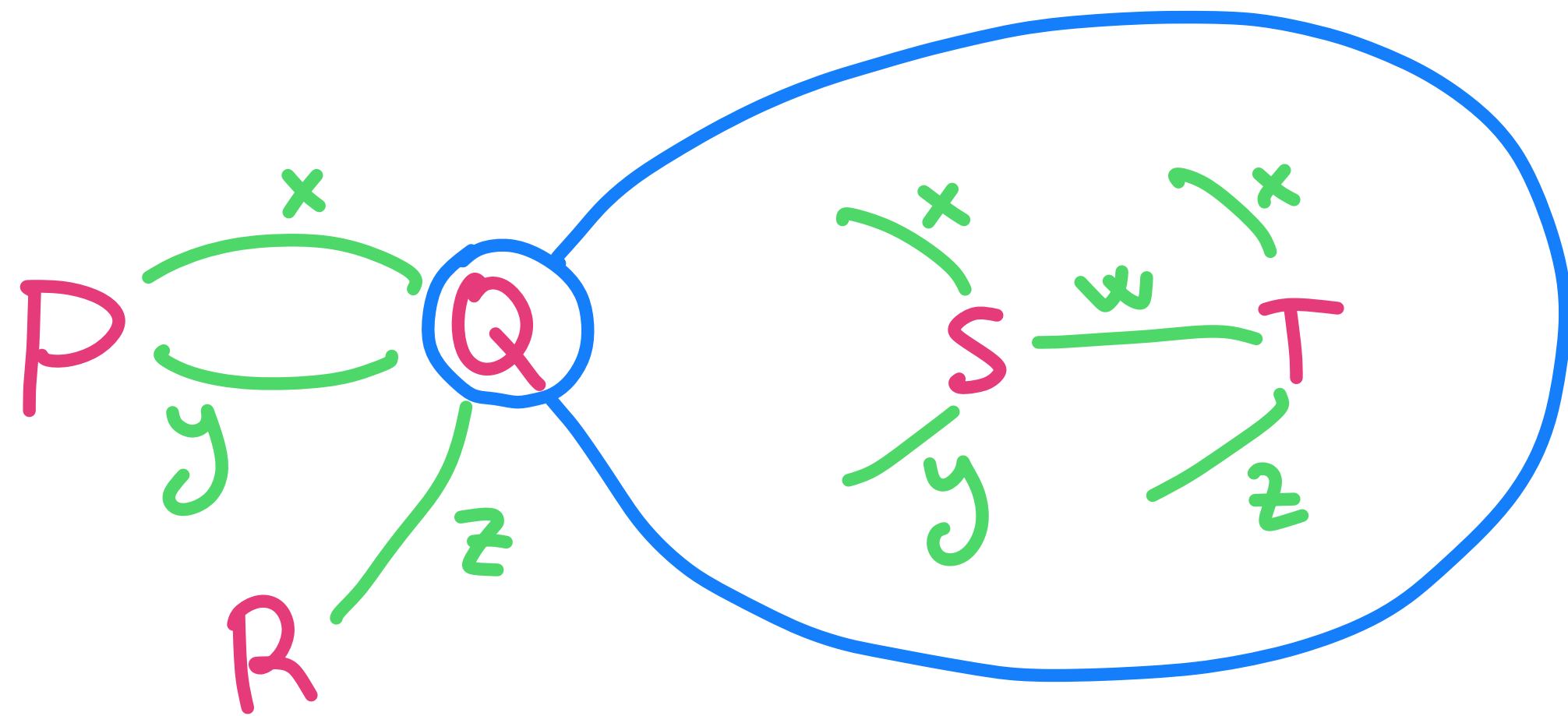
(value dependent)

π and session types
with leftovers

Uma Zlakain, UoGlasgow

UoKent, Dec 2021

π -calculus



$P, Q ::= \text{end}$
 $\quad | \text{new } x \ P$
 $\quad | P \parallel Q$
 $\quad | \text{recv } x \ y \ P$
 $\quad | \text{send } x \ y \ P$
 $\quad | \text{repl } P$

simple types

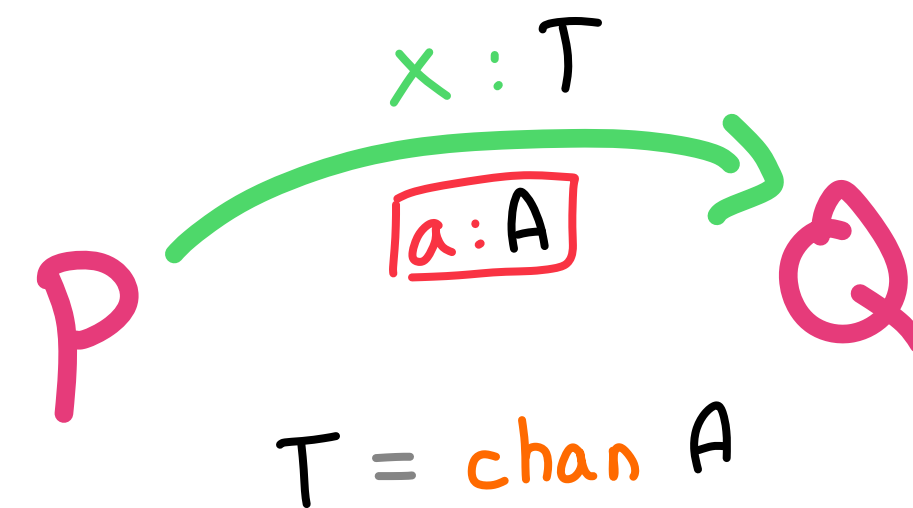
data Type : Set₁ where

pure : Set → Type

chan : Type → Type

prod : Type → Type → Type

sum : Type → Type → Type



· type safety of payload data

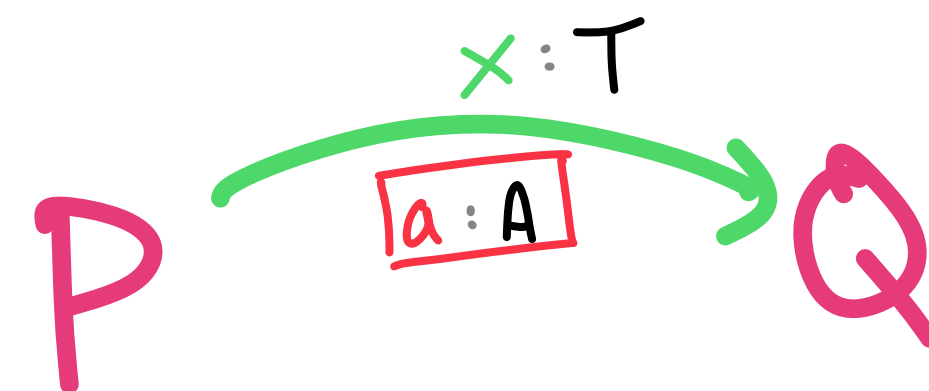
linear channel types

channels used exactly once for sending, once for receiving

data Mult : Set where

0. : Mult

1. : Mult



$$T = \text{chan}_{1,0}.A$$

data Type : Set₁ where

...
chan : Mult \rightarrow Mult \rightarrow Type \rightarrow Type
...

· type safety of payload data

· two uniquely owned endpoints per channel

\Rightarrow all communication is private

\Rightarrow no communication races

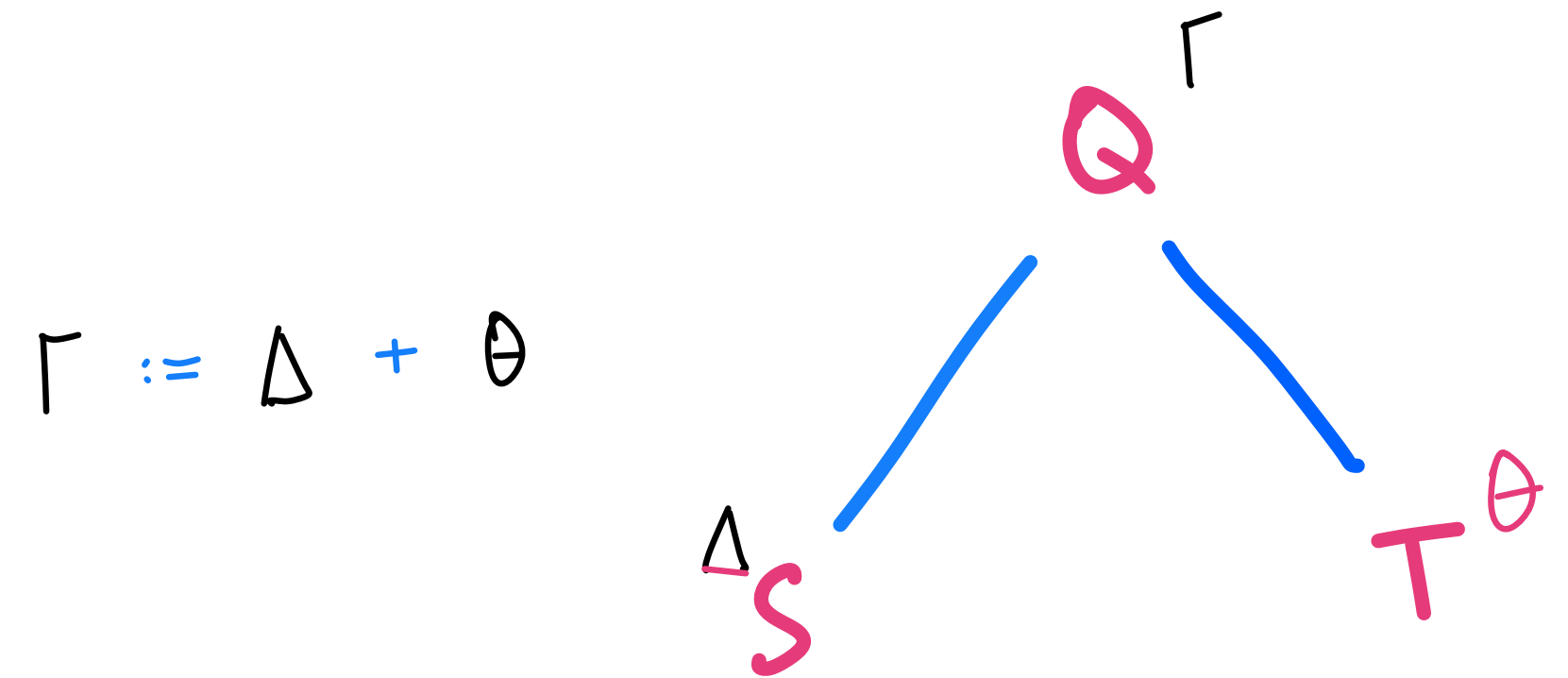
typing rules

$$\frac{\Gamma := \Delta + \theta \quad \text{Proc } \Delta \quad \text{Proc } \theta}{\text{Proc } \Gamma} \text{ comp}$$

$$\frac{\Gamma := \Delta + \theta \quad \Delta \ni \text{chan}_{0,1}.T \quad \text{Proc}(T :: \Delta)}{\text{Proc } \Gamma} \text{ recv}$$

$$\frac{\Gamma := \Delta + \theta \quad \theta := \Xi + \Psi \quad \Delta \ni \text{chan}_{1,0}.T \quad \Xi \ni T \quad \text{Proc } \Psi}{\text{Proc } \Gamma} \text{ send}$$

context splits



How do we get resources to where they need to be?

data $_ := _ + _ : \text{Mult} \rightarrow \text{Mult} \rightarrow \text{Mult} \rightarrow \text{Set}$ where

zero : $0. := 0. + 0.$
 left : $1. := 1. + 0.$
 right : $1. := 0. + 1.$

- top-down resource distribution
- not ergonomic for embedded DSLs

leftover typing

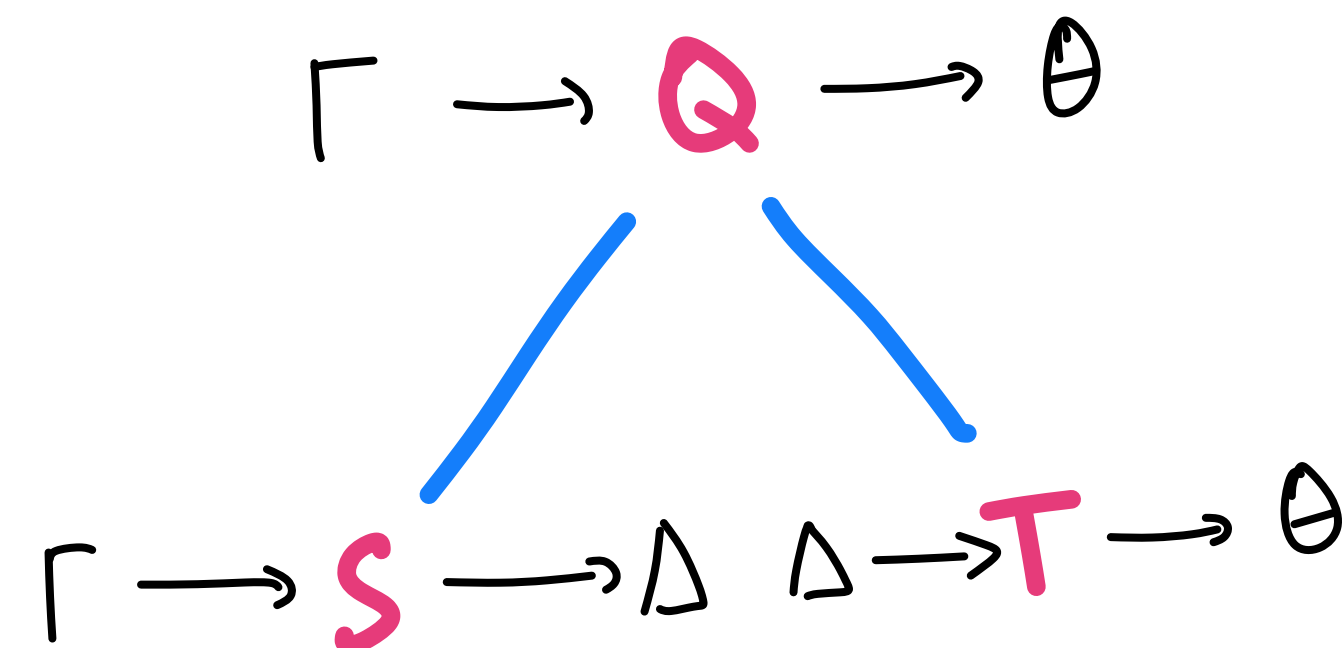
$\Gamma \vdash P \triangleright \Theta$
 input context leftover context

resources subtracted
as you write your program

$\frac{\text{Proc } \Gamma \triangleright \Delta \quad \text{Proc } \Delta \triangleright \Theta}{\text{Proc } \Gamma \triangleright \Theta} \text{ comp}$

$\frac{\Gamma \ni \text{chan}_{0,1}. T \triangleright \Delta \quad \text{Proc } (T :: \Delta) \triangleright (0[T] :: \Theta)}{\text{Proc } \Gamma \triangleright \Theta} \text{ recv}$

$\frac{\Gamma \ni \text{chan}_{1,0}. T \triangleright \Delta \quad \Delta \ni T \triangleright \Theta \quad \text{Proc } \Theta \triangleright \Psi}{\text{Proc } \Gamma \triangleright \Psi} \text{ send}$



Machie, 2008
 Allais, 2018

Π with leftovers : a mechanisation in Agda

- leftover typing for the linear Π -calculus
- defined on usage algebras: partial ternary relations that are
 - deterministic - associative - minimal neutral element
 - cancellative - commutative(encompasses shared, graded and linear types)
- framing, generalised weakening, subject reduction

session types

$$S = ? (Pure \mathbb{N}) . \oplus \begin{cases} End \\ !(Chan \text{ End } (! (Pure \mathbb{B}) . End)) . S \end{cases}$$

- type safety of payload data
- two uniquely owned endpoints per channel
 - \Rightarrow all communication is private
 - \Rightarrow no communication races
- if a channel advances is according to its type

data Type : Set₁ where
pure : Set \rightarrow Type
chan : Session \rightarrow Session \rightarrow Type
sum : Type \rightarrow Type \rightarrow Type

data Action : Set where
? : Action
! : Action

data Session : Set₁ where
End : Session
[]i_ : Action \rightarrow Type
 \rightarrow Session
 \rightarrow Session

CPS encoding

$$\Gamma \vdash_{ST} P \iff [\Gamma]_t \vdash_L [P]_p$$

$$[\text{chan } (?T.S) (!T.\bar{S})]_t \triangleq \text{chan}_{1,1}. ([T]_t \times [\text{chan } S \bar{S}]_t)$$

typing rules

$$\frac{\Gamma := \Delta + \Theta \quad \text{Proc } \Delta \quad \text{Proc } \Theta}{\text{Proc } \Gamma} \text{ comp}$$

$$\frac{\Gamma, x:S, y:T \vdash P}{\Gamma, x: ?T.S \vdash \text{recv } x y P}$$

$$\frac{\Gamma \ni ?T \triangleright \Delta \quad \text{Proc}(T :: \Delta)}{\text{Proc } \Gamma} \text{ recv}$$

$$\frac{\Gamma, x:S, y:O.[T] \vdash P}{\Gamma, x: !T.S, y:T \vdash \text{send } x y P}$$

$$\frac{\Gamma \ni !T \triangleright \Delta \quad \Delta \ni T \triangleright \Xi \quad \text{Proc } \Xi}{\text{Proc } \Gamma} \text{ send}$$

context splits

$$\text{pure } A := \text{pure } A + \text{pure } A$$

$$\frac{S_L := T_L + R_L \quad S_R := T_R + R_R}{\text{chan } S_L S_R := \text{chan } T_L T_R + \text{chan } R_L R_R}$$

$$\text{S} := \text{S} + \text{End} \quad \text{left}$$

$$\text{S} := \text{End} + \text{S} \quad \text{right}$$

leftover typing

Proc $\Gamma \triangleright \Delta [\delta]$

$\underbrace{\Gamma}_{\text{input context}} \quad \underbrace{\Delta}_{\text{leftover context}} \quad \underbrace{[\delta]}_{\text{output cover}}$

data Shape : Set where
 pure : Shape
 chan : Shape

data Cover : Set where
 used : Cover
 unused : Cover

data Type : Shape \rightarrow Set₁ where
 pure : Set \rightarrow Type pure
 chan : Session \rightarrow Session \rightarrow Type chan

data SCover : Shape \rightarrow Set where
 pure : SCover pure
 chan : Cover \rightarrow Cover \rightarrow SCover chan

data SCover : Cover \rightarrow Session \rightarrow Set, where
 unused : SCover unused s
 used : SCover used End

data TCovered : $\forall \{s\} \rightarrow$ SCover s \rightarrow Type s \rightarrow Set, where
 pure : TCovered pure t
 chan : SCovered c₁ s₁
 \rightarrow SCovered c₂ s₂
 \rightarrow TCovered (chan c₁ c₂) (chan s₁ s₂)

data Proc_▷[.]: $\forall \{s\} \rightarrow$ All Type s \rightarrow All Type s \rightarrow All SCover s \rightarrow Set, where

comp : Proc $\Gamma \triangleright \Delta[\delta] \rightarrow$ Pointwise TCovered $\delta \Delta$ \rightarrow Proc $\Delta \triangleright \theta[\sigma] \rightarrow$ Pointwise TCovered $\sigma \theta$ \rightarrow Proc $\Gamma \triangleright \theta[\delta + \sigma]$

automated:
 $\forall \delta \Gamma \rightarrow$ Dec (Pointwise TCovered $\delta \Delta$)

new : Proc (t₁ : Γ) ▷ (t₂ : Δ) [c₁ : δ]
 \rightarrow TCovered c₁ t₂
 \rightarrow Proc Γ ▷ Δ[δ]

sum types?
recursion?

dependent types!

(value) dependent π types

data $Type : Set_1$ where

...
 $prod : (t : Type) \rightarrow ([t] \rightarrow Type) \rightarrow Type$
 ...

$Ctx \triangleq List (\Sigma Type [_])$

$\frac{\Gamma := \Delta + \theta \quad \Delta \ni chan_{0,1}.T \quad ((t : [T]) \rightarrow Proc ([T, t] :: \Delta))}{Proc \Gamma}$

$[_]: Type \rightarrow Set$

$[pure \ A] = A$

$[chan \ _ \ _] = Top$

$[prod \ t \ f] = \sum [t] ([_]. f)$

leverages host's dependent types
 \Rightarrow must be intrinsically typed

(value) dependent session types

$\llbracket _ \rrbracket : \text{Type} \rightarrow \text{Set}$

$\llbracket \text{pure } A \rrbracket = A$

$\llbracket \text{chan } _ \rrbracket = \text{Top}$

- for free:
 - branching and selection
 - structural label (Thiemann & Vasconcelos, 2019)
 - dependent recursion

data Session : Set₁ where

End : Session

$_ \llbracket _ \rrbracket ; _ : \text{Action}$

$\rightarrow (t : \text{Type})$

$\rightarrow (\llbracket t \rrbracket \rightarrow \text{Session})$

$\rightarrow \text{Session}$

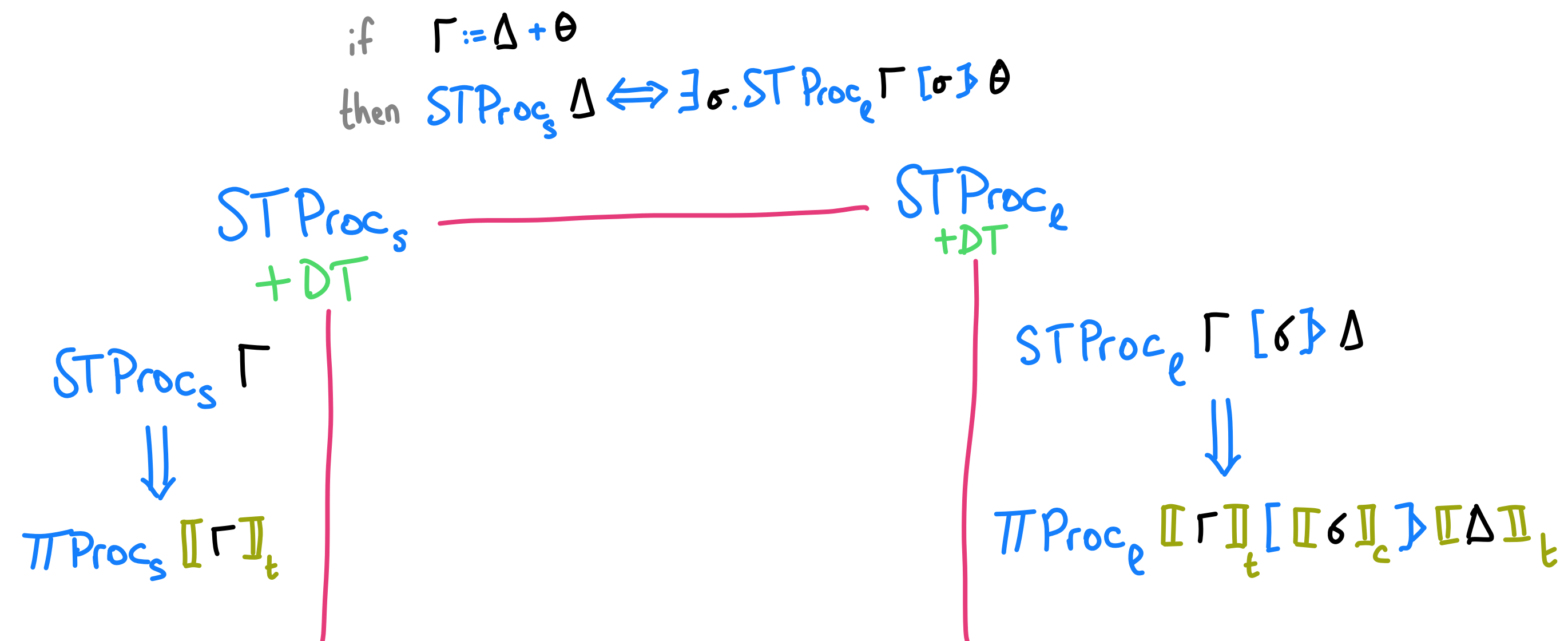
- avoid the question of "traffic dependent" types (McBride, 2015)

metatheory

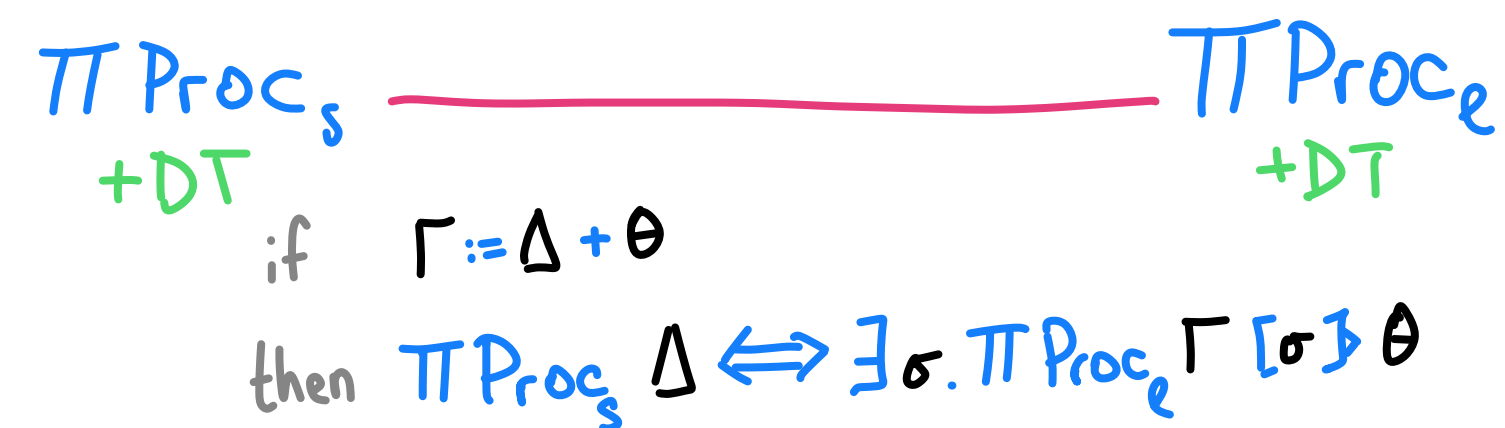
CONTEXT SPLITS

LEFTOVERS

SESSION TYPES



LINEAR Π TYPES



thank you!