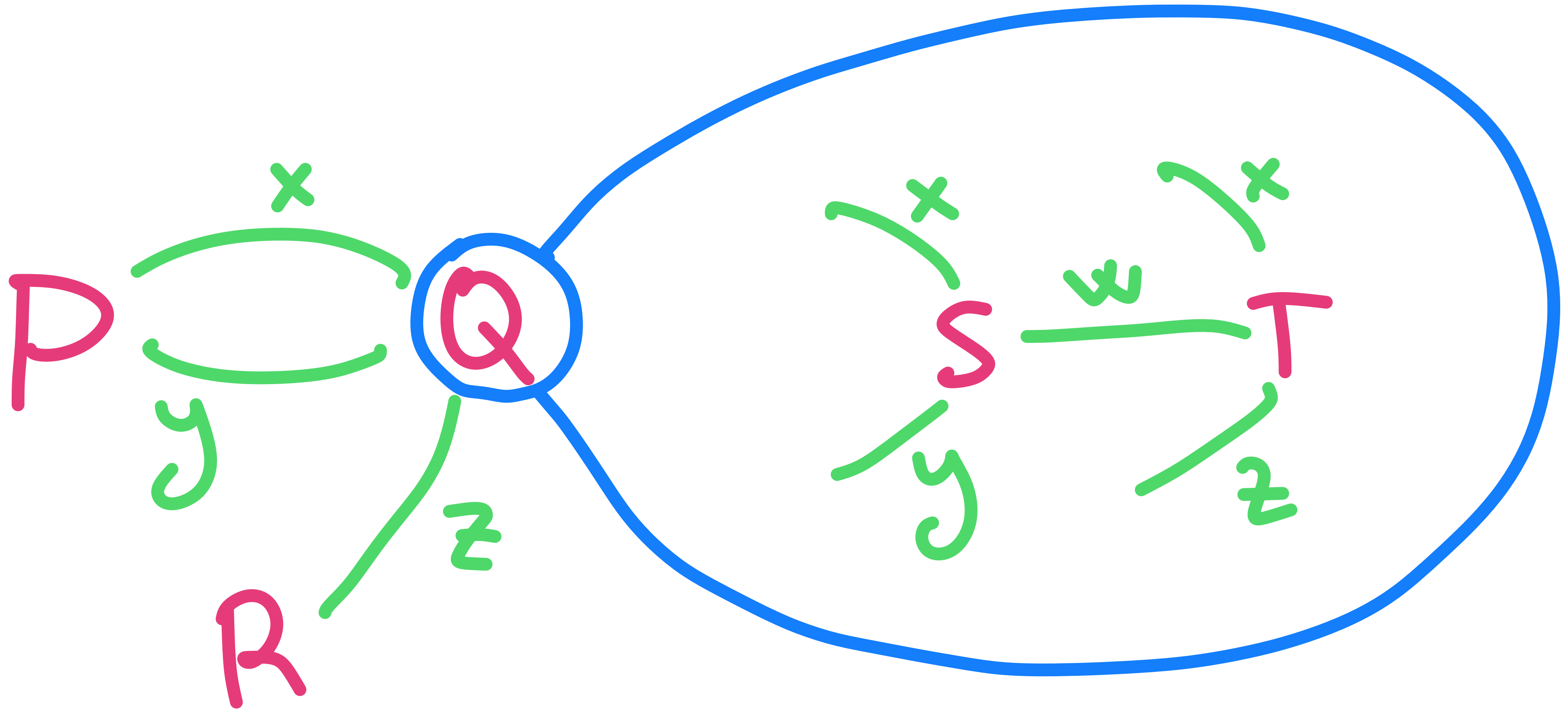
(value dependent) π and session types with leftovers

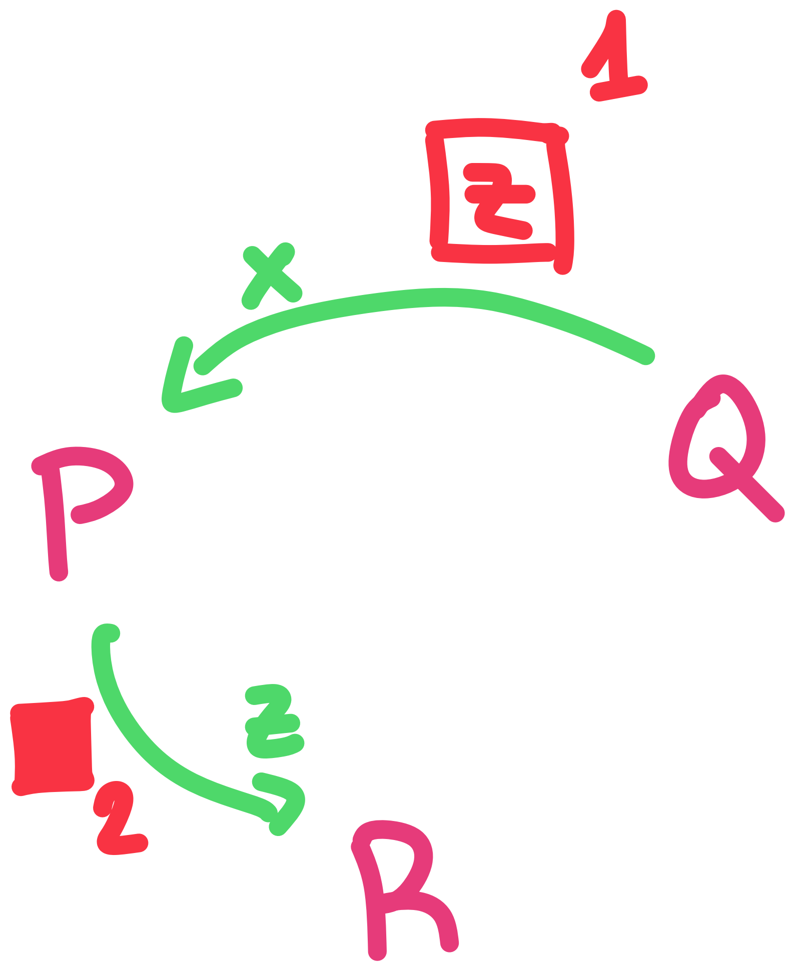
Uma Zalakain, UoGlasgow

UoKent, Dec 2021

π - calculus

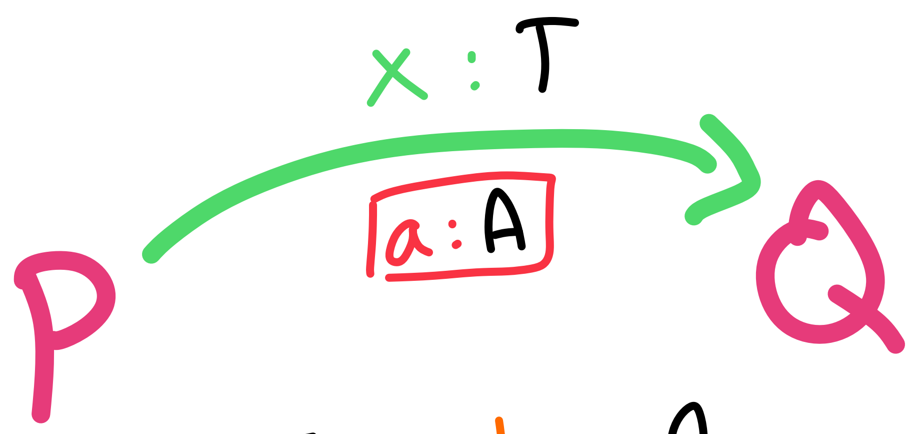


$$
\begin{aligned}
P, Q \ ::= \ & \texttt{end} \\
\mid \ & \texttt{new} \ x \ P \\
\mid \ & P \parallel Q \\
\mid \ & \texttt{recv} \ x \ y \ P \\
\mid \ & \texttt{send} \ x \ y \ P \\
\mid \ & \texttt{repl} \ P
\end{aligned}
$$



simple types

```
data Type : Set₁ where
  pure : Set  → Type
  chan : Type → Type
  prod : Type → Type → Type
  sum  : Type → Type → Type
```



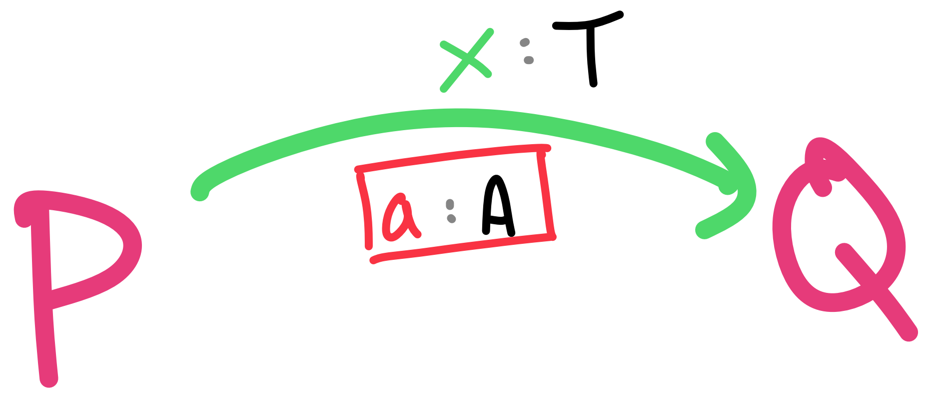
T = chan A

· type safety of payload data

linear channel types

```agda
data Mult : Set where
  0· : Mult
  1· : Mult
```

channels used *exactly* once for sending, once for receiving



$T = \mathrm{chan}_{1\cdot,\,0\cdot}\ A$

```agda
data Type : Set₁ where
  ...
  chan : Mult → Mult → Type → Type
  ...
```

- type safety of payload data
- two uniquely owned endpoints per channel
  - ⟹ all communication is private
  - ⟹ no communication races

Kobayashi, Pierce & Turner, 1996

typing rules

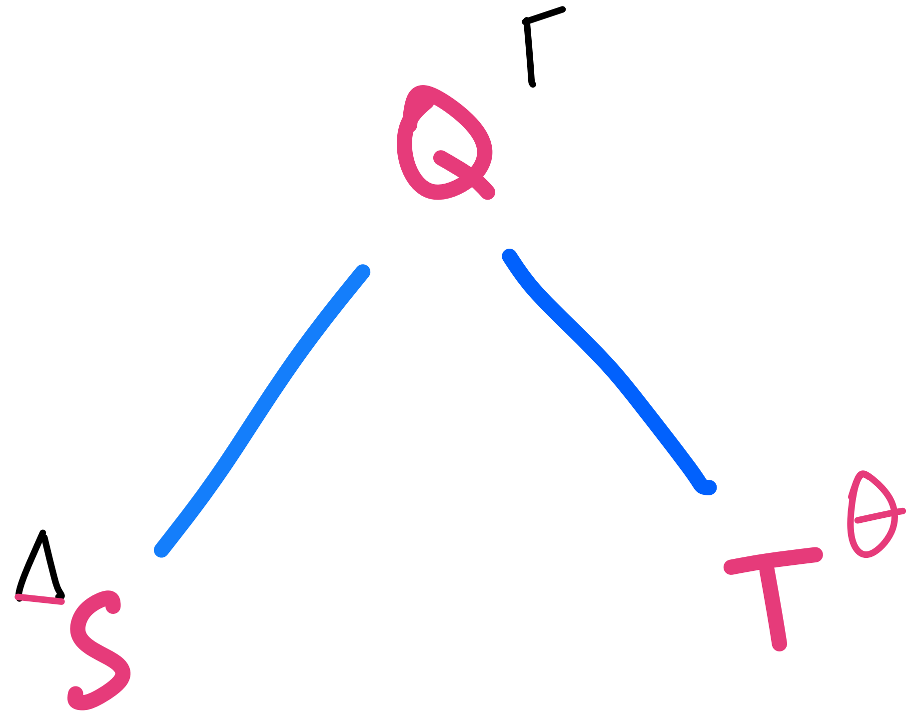
$$\frac{\Gamma := \Delta + \Theta \qquad \mathsf{Proc}\ \Delta \qquad \mathsf{Proc}\ \Theta}{\mathsf{Proc}\ \Gamma}\ \text{comp}$$

$$\frac{\Gamma := \Delta + \Theta \qquad \Delta \ni \mathsf{chan}_{0\cdot,1\cdot}\, T \qquad \mathsf{Proc}\,(T :: \Delta)}{\mathsf{Proc}\ \Gamma}\ \text{recv}$$

$$\frac{\Gamma := \Delta + \Theta \qquad \Theta := \Xi + \Psi \qquad \Delta \ni \mathsf{chan}_{1\cdot,0\cdot}\, T \qquad \Xi \ni T \qquad \mathsf{Proc}\ \Psi}{\mathsf{Proc}\ \Gamma}\ \text{send}$$

context splits

$\Gamma := \Delta + \Theta$



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

```
data _:=_+_ : Mult → Mult → Mult → Set where
  zero  : 0· := 0· + 0·
  left  : 1· := 1· + 0·
  right : 1· := 0· + 1·
```

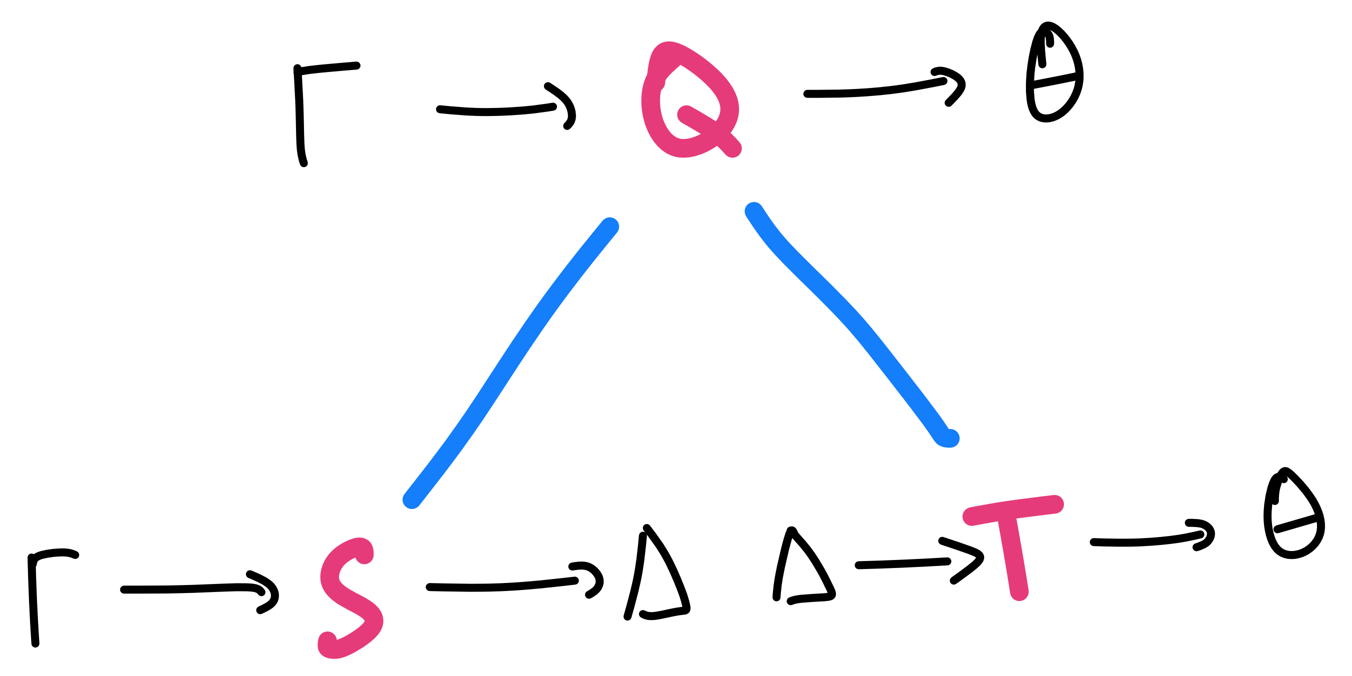
- top-down resource distribution
- not ergonomic for embedded DSLs

leftover typing

$$\underbrace{\Gamma}_{\text{input context}} \vdash P \triangleright \underbrace{\Theta}_{\text{leftover context}}$$

resources subtracted as you write your program

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



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

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

Mackie, 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
  - cancellative
  - associative
  - commutative
  - minimal neutral element

  (encompases shared, graded and linear types)
- framing, generalised weakening, subject reduction

Zalakain & Dardha, 2021

session types

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

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

```
data Type : Set₁ where
  pure : Set → Type
  chan : Session → Session → Type
  sum  : Type → Type → Type

data Action : Set where
  ? : Action
  ! : Action

data Session : Set₁ where
  End : Session
  _[_];_ : Action → Type
           → Session
           → Session
```

CPS encoding

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

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

Dardha, 2012

typing rules

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

$$\frac{\Gamma, x : S, y : T \vdash P}{\Gamma, x : ?T.S \vdash \mathsf{recv}\ x\ y\ P} \quad \blacktriangleright \quad \frac{\Gamma \ni ?T \triangleright \Delta \quad \mathsf{Proc}\,(T :: \Delta)}{\mathsf{Proc}\ \Gamma}\ \text{recv}$$

$$\frac{\Gamma, x : S, y : 0 \cdot [T] \vdash P}{\Gamma, x : !T.S, y : T \vdash \mathsf{send}\ x\ y\ P} \quad \blacktriangleright \quad \frac{\Gamma \ni !T \triangleright \Delta \quad \Delta \ni T \triangleright \Xi \quad \mathsf{Proc}\ \Xi}{\mathsf{Proc}\ \Gamma}\ \text{send}$$

context splits

$$\frac{}{\mathsf{pure}\ A := \mathsf{pure}\ A + \mathsf{pure}\ A}$$

$$\frac{S_L := T_L + R_L \qquad S_R := T_R + R_R}{\mathsf{chan}\ S_L\ S_R := \mathsf{chan}\ T_L\ T_R + \mathsf{chan}\ R_L\ R_R}$$

$$\frac{}{S := S + \mathsf{End}}\ \text{left} \qquad \frac{}{S := \mathsf{End} + S}\ \text{right}$$

leftover typing

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

```
data Shape : Set where
  pure : Shape
  chan : Shape

data Type : Shape → Set₁ where
  pure : Set → Type pure
  chan : Session → Session → Type chan

data Cover : Set where
  used   : Cover
  unused : Cover

data SCover : Shape → Set where
  pure : SCover pure
  chan : Cover → Cover → SCover chan

data SCovered : Cover → Session → Set₁ where
  unused : SCovered unused s
  used   : SCovered used End

data TCovered : ∀{s} → SCover s → Type s → Set₁ where
  pure : TCovered pure t
  chan : SCovered c₁ s₁
       → SCovered c₂ s₂
       → TCovered (chan c₁ c₂) (chan s₁ s₂)
```

```
data Proc_▹_[_] : ∀{s} → All Type s → All Type s → All SCover s → Set₁ where

  comp : Proc Γ ▹ Δ[δ] → Pointwise TCovered δ Δ
       → Proc Δ ▹ Θ[σ] → Pointwise TCovered σ Θ
       → Proc Γ ▹ Θ[δ+σ]
```

automated:
∀ δ Γ → Dec (Pointwise TCovered δ Δ)

sum types?
recursion?

dependent types!

(value) dependent π types

data $\mathsf{Type} : \mathsf{Set}_1$ where
  $\dots$
  $\mathsf{prod} : (t : \mathsf{Type}) \to ([\![t]\!] \to \mathsf{Type}) \to \mathsf{Type}$
  $\dots$

$[\![\_]\!] : \mathsf{Type} \to \mathsf{Set}$
$[\![\mathsf{pure}\ A]\!] = A$
$[\![\mathsf{chan}\ \_\ \_\ \_]\!] = \mathsf{Top}$
$[\![\mathsf{prod}\ t\ f]\!] = \Sigma\ [\![t]\!]\ ([\![\_]\!] \cdot f)$

$\mathsf{Ctx} \triangleq \mathsf{List}\ (\Sigma\ \mathsf{Type}\ [\![\_]\!])$

$$\frac{\Gamma := \Delta + \Theta \qquad \Delta \ni \mathsf{chan}_{0,1}\ T \qquad ((t : [\![T]\!]) \to \mathsf{Proc}\ ((T, t) :: \Delta))}{\mathsf{Proc}\ \Gamma}$$

leverages host's dependent types
⇒ must be intrinsically typed

Padovani & Ciccone, 2020

(value) dependent session types

$[\![\_]\!] : \mathsf{Type} \to \mathsf{Set}$

$[\![\, \mathsf{pure}\ A \,]\!] = A$

$[\![\, \mathsf{chan}\ \_ \,]\!] = \top$

data Session : Set_1 where
  End : Session
  $\_[\_];\_$: Action
    $\to (t : \mathsf{Type})$
    $\to ([\![t]\!] \to \mathsf{Session})$
    $\to \mathsf{Session}$

- for free:
  - branching and selection
  - structural label dependent recursion (Thiemann & Vasconcelos, 2019)
- avoid the question of "traffic dependent" types (McBride, 2015)

metatheory

CONTEXT SPLITS | **LEFTOVERS**

if $\Gamma := \Delta + \Theta$

then $\mathsf{STProc}_s\ \Delta \iff \exists\sigma.\mathsf{STProc}_\ell\ \Gamma\ [\sigma\rangle\ \Theta$

SESSION TYPES

$$
\begin{tikzcd}
\mathsf{STProc}_s + \mathsf{DT} \arrow[r, dash] \arrow[d, dash] & \mathsf{STProc}_\ell + \mathsf{DT} \arrow[d, dash] \\
\pi\mathsf{Proc}_s + \mathsf{DT} \arrow[r, dash] & \pi\mathsf{Proc}_\ell + \mathsf{DT}
\end{tikzcd}
$$

$$\mathsf{STProc}_s\ \Gamma \Downarrow \pi\mathsf{Proc}_s\ [\![\Gamma]\!]_t$$

$$\mathsf{STProc}_\ell\ \Gamma\ [\sigma\rangle\ \Delta \Downarrow \pi\mathsf{Proc}_\ell\ [\![\Gamma]\!]_t\ [\,[\![\sigma]\!]_c\rangle\ [\![\Delta]\!]_t$$

LINEAR π TYPES

if $\Gamma := \Delta + \Theta$

then $\pi\mathsf{Proc}_s\ \Delta \iff \exists\sigma.\pi\mathsf{Proc}_\ell\ \Gamma\ [\sigma\rangle\ \Theta$

thank you!