The Proof Theory of Processes with Protocols

Subashis Chakraborty

Department of Computer Science
University of Calgary

schakr@ucalgary.ca

FMCS 2012

λ-Calculus and Sequential World

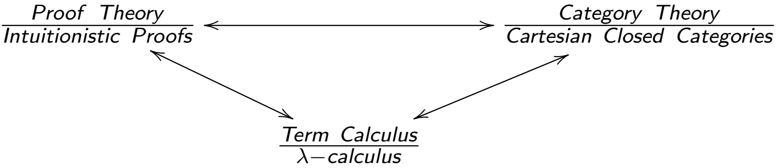


Fig : Curry-Howard-Lambek correspondence

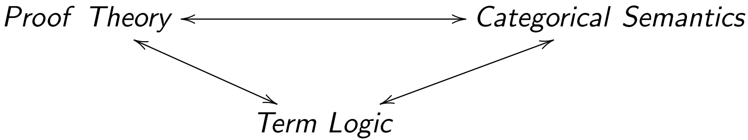
π-Calculus and Process World

- π-calculus is the λ-calculus of the process world.
    - No proof theory.
    - No type theory.
    - Message passing is sole mechanism.

So what is the proof theory of processes?

How can we model **message passing** (key ingredient of concurrent programming) in this setting?

We want:



Proof Theory of Process

Proof Theory of Process

- Two-tier logic (sequential logic and process logic) given by Robin and Pastro.
- Significant distinction between "sequential world" and "process world".
- Gives a basic language for concurrency.
- Does not allow the passing of channel names as messages.
- Cut-elimination provides the operational semantics.
- Use of the linear logic's tensor and par to bundle the channels.
- Add (sequential) message passing facility.

Sequential Logic

- Logic of a monoidal category with coproducts .
- Could be cartesian category or something weaker.
- The logic is presented as Gentzen sequents:

$$\Phi \vdash A$$

where, the antecedent Φ is an unordered list of formulas
the succedent is a single formula.

- Exchange is implicit:

$$\frac{\Phi_1, C, B, \Phi_2 \vdash A}{\Phi_1, B, C, \Phi_2 \vdash A} \text{ exchange}$$

Inference rules of Sequential Logic

$$\frac{}{\Phi \vdash A}\ \text{axiom} \qquad \frac{\Phi \vdash A \quad \Psi_1, A, \Psi_2 \vdash B}{\Psi_1, \Phi, \Psi_2 \vdash B}\ \text{subs}$$

$$\frac{\Phi, A, B \vdash C}{\Phi, A * B \vdash C}\ *_{\text{l}} \qquad \frac{\Phi \vdash A \quad \Psi \vdash B}{\Phi, \Psi \vdash A * B}\ *_{\text{r}}$$

$$\frac{\Phi \vdash A}{\Phi, I \vdash A}\ \text{I}_{\text{l}} \qquad \frac{}{\vdash I}\ \text{I}_{\text{r}}$$

$$\frac{\Phi, A \vdash C \quad \Phi, B \vdash C}{\Phi, A + B \vdash C}\ \text{coprod} \qquad \frac{\Phi \vdash A}{\Phi \vdash A + B}\ \text{inj}_{\text{l}}$$

$$\frac{}{\Phi, 0 \vdash A}\ 0 \qquad \frac{\Phi \vdash B}{\Phi \vdash A + B}\ \text{inj}_{\text{r}}$$

Tensor $= *$ Unit $= I$

Term Formation Rules of Sequential Logic

$$\frac{\textit{axiom } \mathsf{f}}{x_1 : A_1, \ldots, x_n : A_n \vdash f(x_1, \ldots, x_n) : B} \qquad \frac{\Phi \vdash t_1 : A \quad \Psi_1, w : A, \Psi_2 \vdash t_2 : B}{\Psi_1, \Phi, \Psi_2 \vdash (w \mapsto t_2)t_1 : B}$$

$$\frac{\Phi, x : A, y : B \vdash C}{\Phi, (x, y) : A * B \vdash C} \qquad \frac{\Phi \vdash t_1 : A \quad \Psi \vdash t_2 : B}{\Phi, \Psi \vdash (t_1, t_2) : A * B}$$

$$\frac{\Phi \vdash t_1 : A}{\Phi, () : I \vdash t_1 : A} \qquad \frac{}{\vdash () : I}$$

$$\frac{\Phi, A \vdash C \quad \Phi, B \vdash C}{\Phi, t_3 : A + B \vdash \left\{ \begin{array}{l} \sigma_1(x) \mapsto t_1 \\ \sigma_2(y) \mapsto t_2 \end{array} \right\} t_3 : C} \qquad \frac{\Phi \vdash t_1 : A}{\Phi \vdash \sigma_1(t_1) : A + B}$$

$$\frac{}{\Phi, t_3 : 0 \vdash \{\} t_3 : A} \qquad \frac{\Phi \vdash t_1 : B}{\Phi \vdash \sigma_2(t_1) : A + B}$$

In program logic:

$$\left\{ \begin{array}{l} \sigma_1(x) \mapsto t_1 \\ \sigma_2(y) \mapsto t_2 \end{array} \right\} t_3 =: \textit{Case } t_3 \textit{ of } \begin{array}{l} \mid \sigma_1(x) \mapsto t_1 \\ \mid \sigma_2(y) \mapsto t_2 \end{array}$$

The logic of Process

- Built on top of the sequential logic.
- A sequent takes the form:

$$\Phi \mid \Gamma \Vdash \Delta$$

where,

Φ denotes the sequential context
Γ denotes the input process types
Δ denotes the output process types

Γ, Δ are channel name process type lists

e.g $\Gamma = \alpha_1 : P_1, \cdots, \alpha_n : P_n$

Inference rules of Tensor and Par

$$\frac{\Phi \mid \Gamma, X, Y \Vdash \Delta}{\Phi \mid \Gamma, X \otimes Y \Vdash \Delta} \; \otimes_{\mathrm{l}}$$

$$\frac{\Phi \mid \Gamma \Vdash X, Y, \Delta}{\Phi \mid \Gamma \Vdash X \oplus Y, \Delta} \; \oplus_{\mathrm{r}}$$

$$\frac{\Phi \mid \Gamma_1, X \Vdash \Delta_1 \qquad \Psi \mid Y, \Gamma_2 \Vdash \Delta_2}{\Phi, \Psi \mid \Gamma_1, X \oplus Y, \Gamma_2 \Vdash \Delta_1, \Delta_2} \; \oplus_{\mathrm{l}}$$

$$\frac{\Phi \mid \Gamma_1 \Vdash \Delta_1, X \qquad \Psi \mid \Gamma_2 \Vdash Y, \Delta_2}{\Phi, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, X \otimes Y, \Delta_2} \; \otimes_{\mathrm{r}}$$

Term formation rules of Tensor and Par - Split and Fork

$$\frac{s :: \Phi \mid \Gamma, \alpha_1 : X, \alpha_2 : Y \Vdash \Delta}{\textit{split } \alpha \textit{ as } \alpha_1, \alpha_2; s :: \Phi \mid \Gamma, \alpha : X \otimes Y \Vdash \Delta} [\otimes_l]$$

$$\frac{s :: \Phi \mid \Gamma \Vdash \alpha_1 : X, \alpha_2 : Y, \Delta}{\textit{split } \alpha \textit{ as } \alpha_1, \alpha_2; s :: \Phi \mid \Gamma \Vdash \alpha : X \oplus Y, \Delta} [\oplus_r]$$

$$\frac{s_1 :: \Phi \mid \beta_1 : \Gamma_1, \alpha_1 : X \Vdash \Delta_1 \qquad s_2 :: \Psi \mid \alpha_2 : Y, \beta_2 : \Gamma_2 \Vdash \Delta_2}{\textit{fork } \alpha \textit{ as } \begin{array}{l} \mid \alpha_1 \textit{ with } \beta_1 \mapsto s_1 \\ \mid \alpha_2 \textit{ with } \beta_2 \mapsto s_2 \end{array} :: \Phi, \Psi \mid \beta_1 : \Gamma_1, \alpha : X \oplus Y, \beta_2 : \Gamma_2 \Vdash \Delta_1, \Delta_2} [\oplus_l]$$

$$\frac{s_1 :: \Phi \mid \beta_1 : \Gamma_1 \Vdash \Delta_1, \alpha_1 : X \qquad s_2 :: \Psi \mid \beta_2 : \Gamma_2 \Vdash \alpha_2 : Y, \Delta_2}{\textit{fork } \alpha \textit{ as } \begin{array}{l} \mid \alpha_1 \textit{ with } \beta_1 \mapsto s_1 \\ \mid \alpha_2 \textit{ with } \beta_2 \mapsto s_2 \end{array} :: \Phi, \Psi \mid \beta_1 : \Gamma_1, \beta_2 : \Gamma_2 \Vdash \Delta_1, \alpha : X \otimes Y, \Delta_2} [\otimes_r]$$

Cut rule-Plug

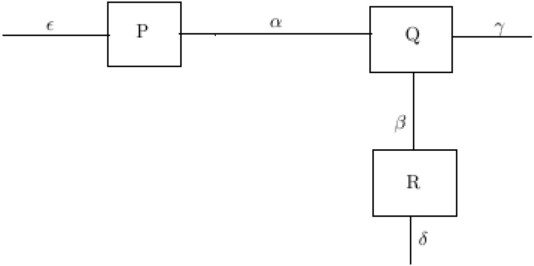
- Inference rule

$$\frac{\Phi \mid \Gamma_1 \Vdash \Delta_1, X \qquad \Psi \mid X, \Gamma_2 \Vdash \Delta_2}{\Phi, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_2}\ \mathrm{cut}$$

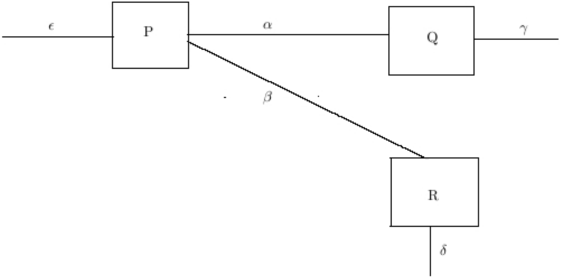
- Term formation

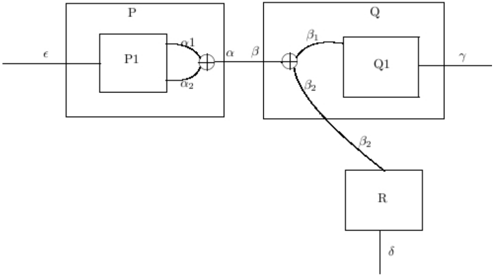
$$\frac{s :: \Phi \mid \Gamma_1 \Vdash \Delta_1, \alpha : X \qquad t :: \Psi \mid \beta : X, \Gamma_2 \Vdash \Delta_2}{\mathit{plug}\ \alpha\ \beta\ s\ t :: \Phi, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_2}\ [\mathrm{cut}]$$

Example



Example





plug α β

split α as α_1, α_2 in P1

fork β as

| β_1 with $\gamma \mapsto$ Q1

| β_2 with $\delta \mapsto$ R

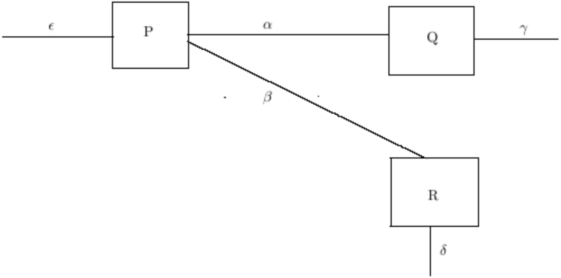
Cut Elimination in Program Logic

```
plug α β
      split α as α1, α2 in P1
      fork β as
            | β1 with γ ↦ Q1
            | β2 with δ ↦ R
```

$$\Downarrow$$

```
plug α2 β2
        plug α1 β1
             P1
             Q1
        R
```

Example



Message passing rules

$$\frac{\Phi, A \mid \Gamma, X \Vdash \Delta}{\Phi \mid \Gamma, A \circ X \Vdash \Delta}\ \circ_{\mathrm{l}}$$

$$\frac{\Phi, A \mid \Gamma \Vdash X, \Delta}{\Phi \mid \Gamma \Vdash A \bullet X, \Delta}\ \bullet_{\mathrm{r}}$$

$$\frac{\Phi \vdash A \qquad \Psi \mid \Gamma, X \Vdash \Delta}{\Phi, \Psi \mid \Gamma, A \bullet X \Vdash \Delta}\ \bullet_{\mathrm{l}}$$

$$\frac{\Phi \vdash A \qquad \Psi \mid \Gamma \Vdash X, \Delta}{\Phi, \Psi \mid \Gamma \Vdash A \circ X, \Delta}\ \circ_{\mathrm{r}}$$

Message passing rules

$$\frac{s :: x : A, \Phi \mid \Gamma, \alpha : X \Vdash \Delta}{\color{red}{get\ x\ \alpha.s} :: \Phi \mid \Gamma, \alpha : A \circ X \Vdash \Delta}\ [\circ_{\mathrm{l}}]$$

$$\frac{s :: x : A, \Phi \mid \Gamma \Vdash \alpha : X, \Delta}{\color{red}{get\ x\ \alpha.s} :: \Phi \mid \Gamma \Vdash \alpha : A \bullet X, \Delta}\ [\bullet_{\mathrm{r}}]$$

$$\frac{\Phi \vdash t : A \quad s :: \Psi \mid \Gamma, \alpha : X \Vdash \Delta}{\color{red}{put\ t\ \alpha; s} :: \Phi, \Psi \mid \Gamma, \alpha : A \bullet X \Vdash \Delta}\ [\bullet_{\mathrm{l}}]$$

$$\frac{\Phi \vdash t : A \quad s :: \Psi \mid \Gamma \Vdash \alpha : X, \Delta}{\color{red}{put\ t\ \alpha; s} :: \Phi, \Psi \mid \Gamma \Vdash \alpha : A \circ X, \Delta}\ [\circ_{\mathrm{r}}]$$

Units

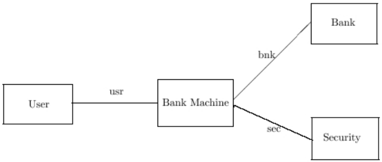
$$\frac{\Phi \mid \Gamma \Vdash \Delta}{\Phi \mid \Gamma, \top \Vdash \Delta}\ \top_{\mathrm{l}} \qquad \frac{s :: \Phi \mid \Gamma \Vdash \Delta}{\mathit{close}\ \alpha.s :: \Phi \mid \Gamma, \alpha : \top \Vdash \Delta}\ [\top_{\mathrm{l}}]$$

$$\frac{\Phi \mid \Gamma \Vdash \Delta}{\Phi \mid \Gamma \Vdash \bot, \Delta}\ \bot_{\mathrm{r}} \qquad \frac{s :: \Phi \mid \Gamma \Vdash \Delta}{\mathit{close}\ \alpha.s :: \Phi \mid \Gamma \Vdash \alpha : \bot, \Delta}\ [\bot_{\mathrm{r}}]$$

$$\frac{}{\emptyset \mid \bot \ \Vdash}\ \bot_{\mathrm{l}} \qquad \frac{}{\mathit{end}\ \alpha :: \emptyset \mid \alpha : \bot \Vdash}\ [\bot_{\mathrm{l}}]$$

$$\frac{}{\emptyset \mid \ \Vdash \top}\ \top_{\mathrm{r}} \qquad \frac{}{\mathit{end}\ \alpha :: \emptyset \mid \ \Vdash \alpha : \top}\ [\top_{\mathrm{r}}]$$

Banking-Example

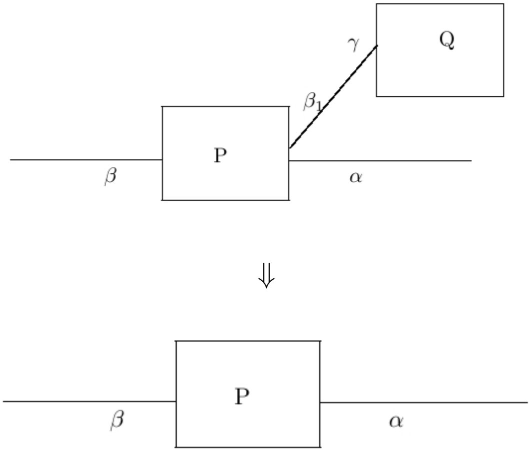


usr : Request ∘(Response ∘⊥) ⊩ bnk : Request ∘(BResponse •⊥), sec : TransID ∘(SRespose •⊥)

•*type Request* = *PIN* ∗ Integer
•*type BResponse* = *TransID* ∗ Integer
•*data Response* = *DollarInteger* | TakeCard
•*data SResponse* = *Accept* | Deny

```
get (pin, x) usr ·
    put (pin, x) bnk;
    get (tid, y) bnk.
        close bnk;
        put tid sec;
        get srp sec ·
            case srp of
                | Accept → close sec;
                           put (Dollar y) usr;
                           end usr
                | Deny →   close sec;
                           put TakeCard usr;
                           end usr
```

Cut Elimination


Q
γ
β1
P
β
α
⇓
P
β
α

$$
\dfrac{
\dfrac{
\dfrac{}{x : A \vdash x : A}\ \mathrm{Ax}
\qquad
\dfrac{
\dfrac{
\dfrac{}{\Vdash \beta : \top}\ \top_{\mathrm{r}}
}{\alpha : \top \Vdash \beta : \top}\ \top_{\mathrm{l}}
}{\alpha : \top \Vdash \beta : \top, \beta_1 : \bot}\ \bot_{\mathrm{r}}
}{
\dfrac{x : A \mid \alpha : A \bullet \top \Vdash \beta : \top, \beta_1 : \bot}{\alpha : A \bullet \top \Vdash \beta : A \bullet \top, \beta_1 : \bot}\ \bullet_{\mathrm{r}}
}\ \bullet_{\mathrm{l}}
\qquad
\dfrac{}{\emptyset \mid \gamma : \bot \Vdash}\ \bot_{\mathrm{l}}
}{\alpha : A \bullet \top \Vdash \beta : A \bullet \top}\ \mathrm{Cut}
$$

Cut Elimination-Example-Program Logic

$$
\dfrac{
\dfrac{
\dfrac{}{x : A \vdash x : A}\;\mathrm{Ax}
\qquad
\dfrac{
\dfrac{
\dfrac{}{\mathit{end}\ \beta :: \quad \Vdash \beta : \top}\;\top_r
}{\mathit{close}\ \alpha; \mathit{end}\ \beta :: \alpha : \top \Vdash \beta : \top}\;\top_l
}{\mathit{close}\ \beta_1; \mathit{close}\ \alpha; \mathit{end}\ \beta :: \alpha : \top \Vdash \beta : \top, \beta_1 : \bot}\;\bot_r
}{\mathit{put}\ x\ \alpha; \mathit{close}\ \beta_1; \mathit{close}\ \alpha; \mathit{end}\ \beta :: x : A \mid \alpha : A \bullet \top \Vdash \beta : \top, \beta_1 : \bot}\;\bullet_l
}{\mathit{get}\ x\ \beta \cdot \mathit{put}\ x\ \alpha; \mathit{close}\ \beta_1; \mathit{close}\ \alpha; \mathit{end}\ \beta :: \alpha : A \bullet \top \Vdash \beta : A \bullet \top, \beta_1 : \bot}\;\bullet_r
\qquad
\dfrac{}{\gamma[] :: \emptyset \mid \gamma : \bot \Vdash}\;\bot_l
}{\mathit{plug}\ \beta_1\ \gamma\ \mathit{get}\ x\ \beta \cdot \mathit{put}\ x\ \alpha; \mathit{close}\ \beta_1; \mathit{close}\ \alpha; \mathit{end}\ \beta\ \mathit{end}\ \gamma :: \alpha : A \bullet \top \Vdash \beta : A \bullet \top}\;\mathrm{Cut}
$$

Program

plug β_1 γ
    get x $\beta\cdot$ put x α; close β_1; close α; end β
    end γ

Cut Elimination in Program Logic

1. plug β_1 γ
   get x β· put x α; close β_1; close α; end β
   end γ
2. get x β· plug β_1 γ
   put x α; close β_1; close α; end β
   end γ
3. get x β· put x α; plug β_1 γ
   close β_1; close α; end β
   end γ
4. get x β· put x α; close α; end β

$$\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\,\mathrm{Ax} \qquad \dfrac{\dfrac{}{\Vdash \beta : \top}\,\top_{\mathrm{r}}}{| : \top \Vdash \beta : \top}\,\top_{\mathrm{l}}}{x : A \mid \alpha : A \bullet \top \Vdash \beta : \top}\,\bullet_{\mathrm{l}}}{\alpha : A \bullet \top \Vdash \beta : A \bullet \top}\,\bullet_{\mathrm{r}}$$

$$
\dfrac{\dfrac{}{x : A \vdash x : A}\;\mathrm{Ax} \qquad \dfrac{\dfrac{}{\mathit{end}\ \beta\ ::\Vdash \beta : \top}\;\top_{\mathrm{r}}}{\mathit{close}\ \alpha;\ \mathit{end}\ \beta\ ::|: \top \Vdash \beta : \top}\;\top_{\mathrm{l}}}{\dfrac{\mathit{put}\ x\ \alpha;\ \mathit{close}\ \alpha;\ \mathit{end}\ \beta\ ::\ x : A \mid \alpha : A \bullet \top \Vdash \beta : \top}{\mathit{get}\ x\ \beta \cdot \mathit{put}\ x\ \alpha;\ \mathit{close}\ \alpha;\ \mathit{end}\ \beta\ ::\ \alpha : A \bullet \top \Vdash \beta : A \bullet \top}\;\bullet_{\mathrm{r}}}\;\bullet_{\mathrm{l}}
$$

Program

get x β $\cdot$ put x α ; close α ; end β

Process Logic with Protocols

Circular Rule

Circular rule:

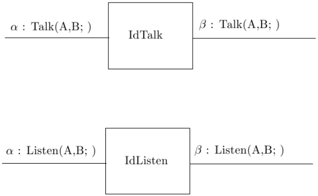
$$\dfrac{\begin{array}{|l} \forall X \qquad\qquad\qquad \Gamma, X \vdash^{f} \Delta \\ \hline \\ \dfrac{\Gamma, X \vdash \Delta}{\Gamma, F(X) \vdash \Delta}\ \mathrm{c[f]} \end{array}}{\Gamma, \mu x.F(x) \vdash \Delta}$$

Mutual Recursive Protocol

$$
\begin{array}{lll}
\textbf{protocol} & \mathrm{H}_X \to X & \\
 & \mathrm{Cons}_X : & F(X, Y) \to X \\
\text{and} & \mathrm{H}_Y \to Y & \\
 & \mathrm{Cons}_Y : & G(X, Y) \to Y
\end{array}
$$

$$
\dfrac{\dfrac{\forall X, Y \qquad \Gamma_2, Y \vdash \Delta_2 \quad \Gamma_1, X \vdash \Delta_1}{\dfrac{\Gamma_1, X \vdash \Delta_1 \quad \Gamma_2, Y \vdash \Delta_2}{\Gamma_1, F(X, Y) \vdash \Delta_1}\ \mathrm{c}_1[\_] \quad \dfrac{\Gamma_1, X \vdash \Delta_1 \quad \Gamma_2, Y \vdash \Delta_2}{\Gamma_2, G(X, Y) \vdash \Delta_2}\ \mathrm{c}_2[\_]}}{\Gamma_1, H_X \vdash \Delta_1 \qquad \Gamma_2, H_Y \vdash \Delta_2}
$$

Identity-The Copy-Cat Strategy



protocol $Talk(A, B;) \rightarrow C =$

$\#response$: $B \bullet D \rightarrow C$

and $Listen(A, B;) \rightarrow D =$

$\#listen$: $A \circ C \rightarrow D$

drive *IdTalk* on α by

$\#response$: put $\#response$ on β; get b β; put b α; *IdListen*

and *IdListen* on α by

$\#listen$: put $\#listen$ on β; get a α; put a β; *IdTalk*

Proof

protocol $Talk(A, B;) \rightarrow C =$

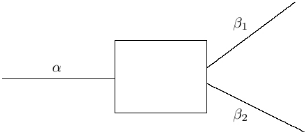
$\qquad \#response\colon\ B \bullet D \rightarrow C$

and $Listen(A, B;) \rightarrow D =$

$\qquad \#listen :\ A \circ C \rightarrow D$

$$\dfrac{\begin{array}{c} \forall C, D \qquad\qquad C \Vdash Talk(A, B;) \qquad D \Vdash Listen(A, B;) \\ \hline \\ \dfrac{\dfrac{\dfrac{\dfrac{}{x : B \vdash x : B} \quad \dfrac{}{\alpha : D \Vdash \beta : Listen(A, B;)}}{x : B \mid \alpha : B \bullet D \Vdash \beta : B \bullet Listen(A, B;)}}{\alpha : B \bullet D \Vdash \beta : B \bullet Listen(A, B;)}}{\alpha : B \bullet D \Vdash \beta : Talk(A, B;)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{y : A \vdash y : A} \quad \dfrac{}{\alpha : C \Vdash \beta : Talk(A, B;)}}{y : A \mid \alpha : C \Vdash \beta : A \circ Talk(A, B;)}}{\alpha : A \circ C \Vdash \beta : A \circ Talk(A, B;)}}{\alpha : A \circ C \Vdash \beta : Listen(A, B;)} \end{array}}{\alpha : Talk(A, B;) \Vdash \beta : Talk(A, B;) \qquad \alpha : Listen(A, B;) \Vdash \beta : Listen(A, B;)}$$

Message Passing with Prototcols-Example



protocol $Talk(A, B;) \to C =$

$\#response$: $B \bullet D \to C$

and $Listen(A, B;) \to D =$

$\#listen$: $A \circ C \to D$

drive $Recieve_C$ on α by

$\#response$: put $\#response$ on β_1; put $\#response$ on β_2;

get b_1 β_1; get b_2 β_2; put (b_1, b_2) α; $Response_D$

and $Response_D$ on α by

$\#listen$: put $\#listen$ on β_1; put $\#listen$ on β_2; get a α;

put a β_1; put a β_2; $Recieve_C$

Proof

protocol $\mathit{Talk}(A, B;) \rightarrow C =$

$\quad \#\mathit{response}\text{: } B \bullet D \rightarrow C$

and $\mathit{Listen}(A, B;) \rightarrow D =$

$\quad \#\mathit{listen} : A \circ C \rightarrow D$

$$
\dfrac{\begin{array}{c}
\forall C, D \qquad C \Vdash Talk(A,B;), Talk(A,B;) \qquad D \Vdash Listen(A,B;), Listen(A,B;) \\[1em]
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{b_1 : B \vdash b_1 : B} \quad \dfrac{}{b_2 : B \vdash b_2 : B}}{b_1 : B, b_2 : B \vdash (b_1, b_2) : B * B} \quad \dfrac{}{\alpha : D \Vdash \beta_1 : Listen(A,B;), \beta_2 : Listen(A,B;)}}{b_1 : B, b_2 : B \mid \alpha : B * B \bullet D \Vdash \beta_1 : Listen(A,B;), \beta_2 : Listen(A,B;)}}{b_1 : B \mid \alpha : B * B \bullet D \Vdash \beta_1 : Listen(A,B;), \beta_2 : B \bullet Listen(A,B;)}}{\alpha : B * B \bullet D \Vdash \beta_1 : B \bullet Listen(A,B;), \beta_2 : B \bullet Listen(A,B;)}}{\alpha : B * B \bullet D \Vdash \beta_1 : Talk(A,B;), \beta_2 : Talk(A,B;)}
\quad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a_1 : A \vdash a_1 : A} \quad \dfrac{\dfrac{}{a_2 : A \vdash a_2 : A} \quad \dfrac{}{\alpha : C \Vdash \beta_1 : A \circ Talk(A,B;), \beta_2 : A \circ Talk(A,B;)}}{a_2 : A \mid \alpha : C \Vdash \beta_1 : A \circ Talk(A,B;), \beta_2 : A \circ Talk(A,B;)}}{a_1 : A, a_2 : A \mid \alpha : C \Vdash \beta_1 : A \circ Talk(A,B;), \beta_2 : A \circ Talk(A,B;)}}{a_1 : A \mid \alpha : C \Vdash \beta_1 : A \circ Talk(A,B;), \beta_2 : A \circ Talk(A,B;)}}{\dfrac{\alpha : AC \Vdash \beta_1 : A \circ Talk(A,B;), \beta_2 : A \circ Talk(A,B;)}{\alpha : AC \Vdash \beta_1 : Listen(A,B;), \beta_2 : Listen(A,B;)}}
\end{array}}{\alpha : Talk(A, B * B;) \Vdash \beta_1 : Talk(A,B;), \beta_2 : Talk(A,B;) \qquad \alpha : Listen(A, B * B;) \Vdash \beta_1 : Listen(A,B;), \beta_2 : Listen(A,B;)}
$$

Thank You