Propositions as Sessions

Logical Foundations of Concurrent Computation

Dan Frumin and Jorge A. Pérez

University of Groningen

ESSLLI 2024
Aug 2nd

This Course

A bird's eye view on the logical foundations of concurrent computation.

Plan:

1. Motivation (Jorge) - Multiplicative Linear Logic (MLL) (Dan)
2. The concurrent interpretation of MLL (Jorge)
3. Cut-elimination and correctness for concurrent processes (Jorge)
4. Beyond linear resources: the !-modality and resource sharing (Dan)
5. **An alternative view of resource sharing: Bunched Implications** (Dan)

BI: The logic of Bunched Implications

The logic of Bunched Implications

So far we have seen πDILL based on intuitionistic linear logic.

- A formula signifies amount of resources.
- $A \otimes B$: one copy of A, plus one copy of B.
- Models *quantity* of resources
- Sharing is allowed through the ! modality.

In this lecture we will talk about an alternative logic for resources: BI and πBI.

The logic of Bunched Implications

$$\begin{aligned} A, B \quad ::= \quad & \mathbf{1} \mid A * B \mid A \mathrel{-\!\!*} B \mid \\ & \top \mid A \wedge B \mid A \rightarrow B \mid A \vee B \end{aligned}$$

- A formula signifies resources owned.
- $A * B$: own resources can be separated into resources denoted by A, and resources denoted by B.
- Models *ownership* (and separation) of resources.
- Sharing is allowed through intuitionistic connectives
  - $A \wedge B$: own resources satisfy both A and B

NB: Separation Logic

BI forms a basis for *separation logic*...

- $\ell \mapsto v$: the current state has the location ℓ in memory, and it stores the value v
- $P * Q$: the current state can be divided into two disjoint parts, for which P and Q hold respecively

NB: Separation Logic

BI forms a basis for *separation logic*...

- $\ell \mapsto v$: the current state has the location ℓ in memory, and it stores the value v
- $P * Q$: the current state can be divided into two disjoint parts, for which P and Q hold respecively
- $\ell \mapsto v * \ell' \mapsto v'$: the locations ℓ and ℓ' do not alias each other
- $\ell \mapsto v \wedge \ell' \mapsto v'$: aliasing is allowed

NB: Separation Logic

BI forms a basis for *separation logic*...

- $\ell \mapsto v$: the current state has the location ℓ in memory, and it stores the value v
- $P * Q$: the current state can be divided into two disjoint parts, for which P and Q hold respecively
- $\ell \mapsto v * \ell' \mapsto v'$: the locations ℓ and ℓ' do not alias each other
- $\ell \mapsto v \wedge \ell' \mapsto v'$: aliasing is allowed
- $\ell_1 \mapsto (v_1, \ell_2) * \ell_2 \mapsto (v_2, \ell_3) * \cdots * \ell_n \mapsto (v_n, \ell_o) * \ell_o \mapsto \texttt{NULL}$: a linked list without cycles

BI Proof Theory

Bunches in BI

Contexts in linear logic can be formalized as a data type:

$$\Delta ::= \emptyset \mid A, \Delta \qquad \text{or}$$
$$\Delta ::= \emptyset \mid A \mid \Delta_1, \Delta_2$$

The composition Δ_1, Δ_2 externalizes the $\otimes$ connective (c.f. left and right rules for $\otimes$). The & operator by contrast did not an “external” version.

Bunches in BI

Contexts in linear logic can be formalized as a data type:

$$\Delta ::= \emptyset \mid A, \Delta \qquad \text{or}$$
$$\Delta ::= \emptyset \mid A \mid \Delta_1, \Delta_2$$

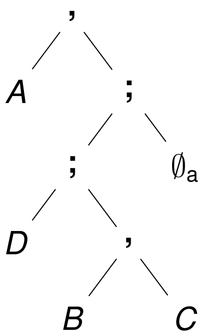
The composition Δ_1, Δ_2 externalizes the $\otimes$ connective (c.f. left and right rules for $\otimes$). The & operator by contrast did not an "external" version.

In BI we externalize both multiplicative $*$ and additive $\wedge$ on equal footing.

Bunches in BI

Bunches are trees of formulas with two kinds of nodes:

$$\Delta \quad ::= \; A \mid \emptyset_{\mathrm{a}} \mid \emptyset_{\mathrm{m}} \mid \Delta_1 \,;\, \Delta_2 \mid \Delta_1 \,,\, \Delta_2$$



For example: $A \,,\, (D \,;\, (B \,,\, C) \,;\, \emptyset_{\mathrm{a}})$.

Bunch equivalence

$\equiv$ is the smallest congruence generated by

$$\Delta_1 \,,\, \Delta_2 \equiv \Delta_2 \,,\, \Delta_1 \qquad \Delta \,,\, \emptyset_m \equiv \Delta \qquad \Delta_1 \,,\, (\Delta_2 \,,\, \Delta_3) \equiv (\Delta_1 \,,\, \Delta_2) \,,\, \Delta_3$$

$$\Delta_1 \,;\, \Delta_2 \equiv \Delta_2 \,;\, \Delta_1 \qquad \Delta \,;\, \emptyset_a \equiv \Delta \qquad \Delta_1 \,;\, (\Delta_2 \,;\, \Delta_3) \equiv (\Delta_1 \,;\, \Delta_2) \,;\, \Delta_3$$

E.g. $A \,,\, (D \,;\, (B \,,\, C) \,;\, \emptyset_a) \equiv ((B \,,\, C) \,;\, D) \,,\, A$

Bunch equivalence

$\equiv$ is the smallest congruence generated by

$$\Delta_1 \,, \Delta_2 \equiv \Delta_2 \,, \Delta_1 \qquad \Delta \,, \emptyset_{\mathrm{m}} \equiv \Delta \qquad \Delta_1 \,, (\Delta_2 \,, \Delta_3) \equiv (\Delta_1 \,, \Delta_2) \,, \Delta_3$$

$$\Delta_1 \,; \Delta_2 \equiv \Delta_2 \,; \Delta_1 \qquad \Delta \,; \emptyset_{\mathrm{a}} \equiv \Delta \qquad \Delta_1 \,; (\Delta_2 \,; \Delta_3) \equiv (\Delta_1 \,; \Delta_2) \,; \Delta_3$$

E.g. $A \,, (D \,; (B \,, C) \,; \emptyset_{\mathrm{a}}) \equiv ((B \,, C) \,; D) \,, A$

We will work with bunches modulo $\equiv$

BI sequent calculus

Sequent: $\Gamma \vdash A$

$$\frac{\Gamma\,;A\,;B \vdash C}{\Gamma\,;A \wedge B \vdash C}$$

$$\frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1;\Gamma_2 \vdash A \wedge B}$$

BI sequent calculus

Left and right rules

$$\frac{\Gamma \,;\, A \,;\, B \vdash C}{\Gamma \,;\, A \wedge B \vdash C} \qquad \frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1 ; \Gamma_2 \vdash A \wedge B}$$

$$\frac{\Gamma ; \Gamma \vdash C}{\Gamma \vdash C} \qquad \frac{\Gamma \vdash C}{\Gamma ; \Gamma' \vdash C}$$

BI sequent calculus

Structural rules

$$\frac{\Gamma\,;A\,;B \vdash C}{\Gamma\,;A \wedge B \vdash C} \qquad \frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1\,;\Gamma_2 \vdash A \wedge B}$$

$$\frac{\Gamma\,;\Gamma \vdash C}{\Gamma \vdash C} \qquad \frac{\Gamma \vdash C}{\Gamma\,;\Gamma' \vdash C}$$

BI sequent calculus

$$\frac{\Gamma\,;(A\,,B) \vdash C}{\Gamma\,;A * B \vdash C} \qquad \frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1\,,\Gamma_2 \vdash A * B}$$

$$\frac{\Gamma\,;A\,;B \vdash C}{\Gamma\,;A \wedge B \vdash C} \qquad \frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1\,;\Gamma_2 \vdash A \wedge B}$$

$$\frac{\Gamma\,;\Gamma \vdash C}{\Gamma \vdash C} \qquad \frac{\Gamma \vdash C}{\Gamma\,;\Gamma' \vdash C}$$

BI sequent calculus

$$\frac{\Delta(A \,,\, B) \vdash C}{\Delta(A * B) \vdash C} \qquad \frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1 \,,\, \Gamma_2 \vdash A * B}$$

$$\frac{\Delta(A \,;\, B) \vdash C}{\Delta(A \wedge B) \vdash C} \qquad \frac{\Gamma_1 \vdash A \qquad \Gamma_2 \vdash B}{\Gamma_1 \,;\, \Gamma_2 \vdash A \wedge B}$$

$$\frac{\Delta(\Gamma \,;\, \Gamma) \vdash C}{\Delta(\Gamma) \vdash C} \qquad \frac{\Delta(\Gamma) \vdash C}{\Delta(\Gamma \,;\, \Gamma') \vdash C}$$

$\Delta(-)$ is a bunch with a hole, $\Delta(\Gamma)$ plugs Γ into the hole. E.g. $\Delta(-) = A\,;((-)\,,C)$, and $\Delta(\Gamma) = A\,;(\Gamma\,,C)$

BI sequent calculus

$$\frac{\Delta\,,A \vdash B}{\Delta \vdash A \mathrel{-\!\!*} B} \qquad \frac{\Delta\,;A \vdash B}{\Delta \vdash A \to B}$$

BI sequent calculus

$$\frac{\Delta\,\mathbf{,}\,A \vdash B}{\Delta \vdash A \mathrel{-\!\!*} B} \qquad\qquad \frac{\Delta\,\mathbf{;}\,A \vdash B}{\Delta \vdash A \to B}$$

- Sequent calculus for BI externalizes $\wedge$ and $*$ as different connectives: **;** and **,**. Only **;** admits weakening and contraction.

BI sequent calculus

$$\frac{\Delta\,\mathbf{,}\,A \vdash B}{\Delta \vdash A \mathrel{-\!\!*} B} \qquad \frac{\Delta\,\mathbf{;}\,A \vdash B}{\Delta \vdash A \to B}$$

- Sequent calculus for BI externalizes $\wedge$ and $*$ as different connectives: **;** and **,**. Only **;** admits weakening and contraction.
- Because of that, contexts in the sequents are not lists/multisets, but *bunches*;

BI sequent calculus

$$\frac{\Delta \,, A \vdash B}{\Delta \vdash A \mathrel{-\!\!*} B} \qquad \frac{\Delta \,; A \vdash B}{\Delta \vdash A \to B}$$

- Sequent calculus for BI externalizes $\wedge$ and $*$ as different connectives: **;** and **,**. Only **;** admits weakening and contraction.
- Because of that, contexts in the sequents are not lists/multisets, but *bunches*;
- Left rules can be applied deep inside an arbitrary *bunched context*.

BI sequent calculus

Structural rules can be applied deep inside a bunched context, including the cut rule:

$$\frac{\Gamma \vdash A \qquad \Delta(A) \vdash C}{\Delta(\Gamma) \vdash C}$$

BI vs Linear Logic

- As it stands, BI and ILL are incompatible
- BI is conservative over MILL and Intuitionistic Logic

BI vs Linear Logic

- As it stands, BI and ILL are incompatible
- BI is conservative over MILL and Intuitionistic Logic
  - $A \otimes B \otimes (C \multimap D)$ is provable in MILL iff $A * B * (C \mathrel{-\!\!*} D)$ is provable in BI

BI vs Linear Logic

- As it stands, BI and ILL are incompatible
- BI is conservative over MILL and Intuitionistic Logic
  - $A \otimes B \otimes (C \multimap D)$ is provable in MILL iff $A * B * (C \mathrel{-\!\!*} D)$ is provable in BI
- BI is not conservative over MAILL
  - Distributivity of $\wedge$ over $\vee$ holds in BI: $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$
  - But not in ILL: $A \,\&\, (B \oplus C) \nvdash (A \,\&\, B) \oplus (A \,\&\, C)$

πBI calculus

πBI calculus

We would like to have a session-types interpretation of BI that is conservative over the MILL fragment of πDILL.

- We are "forced to interpret" $*$ as output, $-\!*$ as input.
- In fact, we will also treat $\wedge$ as output and $\rightarrow$ as input.
- The main difference will be in treatment of structural rules for $\wedge$.

πBI calculus: additives and multiplicatives

$$\frac{\Pi(y : A \,,\, x : B) \vdash P :: z : C}{\Pi(x : A * B) \vdash x(y).P :: z : C}$$

$$\frac{\Delta_1 \vdash P :: y : A \qquad \Delta_2 \vdash Q :: x : B}{\Delta_1 \,,\, \Delta_2 \vdash \overline{x}[y].(P \mid Q) :: x : A * B}$$

πBI calculus: additives and multiplicatives

$$\frac{\Pi(y : A \,\mathbf{,}\, x : B) \vdash P :: z : C}{\Pi(x : A * B) \vdash x(y).P :: z : C} \qquad \frac{\Pi(y : A \,\mathbf{;}\, x : B) \vdash P :: z : C}{\Pi(x : A \wedge B) \vdash x(y).P :: z : C}$$

$$\frac{\Delta_1 \vdash P :: y : A \qquad \Delta_2 \vdash Q :: x : B}{\Delta_1 \,\mathbf{,}\, \Delta_2 \vdash \overline{x}[y].(P \mid Q) :: x : A * B} \qquad \frac{\Delta_1 \vdash P :: y : A \qquad \Delta_2 \vdash Q :: x : B}{\Delta_1 \,\mathbf{;}\, \Delta_2 \vdash \overline{x}[y].(P \mid Q) :: x : A \wedge B}$$

πBI calculus: explicit structural rules

$$\frac{\Pi(x_1 : A \,;\, x_2 : A) \vdash P :: z : C}{\Pi(x : A) \vdash P :: z : C}$$

$$\frac{\Pi(\emptyset_a) \vdash P :: z : C}{\Pi(x : A) \vdash P :: z : C}$$

πBI calculus: explicit structural rules

$$\frac{\Pi(x_1 : A \,;\, x_2 : A) \vdash P :: z : C}{\Pi(x : A) \vdash \rho[x \mapsto x_1, x_2].P :: z : C}$$

$$\frac{\Pi(\emptyset_a) \vdash P :: z : C}{\Pi(x : A) \vdash \rho[x \mapsto \emptyset].P :: z : C}$$

Spawn

Spawn construct:

- $\rho[x \mapsto y, z].P$: spawn two copies of processes on x, bind them to y, z, and proceed as P
- $\rho[x \mapsto \emptyset].P$: kill the process on x, and proceed as P
- General form: $\rho[\sigma].P$ where $\sigma : \mathit{Name} \xrightharpoonup{\text{fin}} \wp(\mathit{Name})$
  - $\forall x, y \in \mathrm{dom}(\sigma).\ x \neq y \implies \sigma(x) \cap \sigma(y) = \emptyset$
  - $\forall x \in \mathrm{dom}(\sigma).\ \sigma(x) \cap \mathrm{dom}(\sigma) = \emptyset$

πBI calculus: explicit structural rules

$$\frac{\Pi(\Delta^{(1)}\,;\,\Delta^{(2)}) \vdash P :: z : C}{\Pi(\Delta) \vdash \rho[x \mapsto x_1, x_2 \mid x \in \Delta].P :: z : C}$$

$$\frac{\Pi(\emptyset_a) \vdash P :: z : C}{\Pi(\Delta) \vdash \rho[x \mapsto \emptyset \mid x \in \Delta].P :: z : C}$$

Spawn: reductions

$$\dfrac{\emptyset_a \vdash P :: x : A \qquad \dfrac{\Pi(x_1 : A \,;\, x_2 : A) \vdash Q :: z : C}{\Pi(x : A) \vdash \rho[x \mapsto x_1, x_2].Q :: z : C}}{\Pi(\emptyset_a) \vdash (\nu x)(P \mid \rho[x \mapsto x_1, x_2].Q) :: z : C}$$

$$\dfrac{\emptyset_a \vdash P[x_2/x] :: x_2 : A \qquad \dfrac{\emptyset_a \vdash P[x_1/x] :: x_1 : A \qquad \Pi(x_1 : A \,;\, x_2 : A) \vdash Q :: z : C}{\Pi(\emptyset_a \,;\, x_2 : A) \vdash (\nu x_1)(P[x_1/x] \mid Q) :: z : C}}{\Pi(\emptyset_a \,;\, \emptyset_a) \vdash (\nu x_2)\big(P[x_2/x] \mid (\nu x_1)(P[x_1/x] \mid Q)\big) :: z : C}$$

Spawn: reductions

$$(\nu x)(P \mid_x \rho[x \mapsto x_1, x_2].\,Q) \longrightarrow (\nu x_2)\big(P[x_2/x] \mid_{x_2} (\nu x_1)(P[x_1/x] \mid_{x_1} Q)\big)$$

Spawn: reductions

$$\dfrac{\Delta \vdash P :: x : A \qquad \dfrac{\Pi(x_1 : A\,;\, x_2 : A) \vdash Q :: z : C}{\Pi(x : A) \vdash \rho[x \mapsto x_1, x_2].Q :: z : C}}{\Pi(\Delta) \vdash (\nu x)(P \mid \rho[x \mapsto x_1, x_2].Q) :: z : C}$$

$$\dfrac{\Delta^{(2)} \vdash P[x_2/x] :: x_2 : A \qquad \dfrac{\Delta^{(1)} \vdash P[x_1/x] :: x_1 : A \qquad \Pi(x_1 : A\,;\, x_2 : A) \vdash Q :: z : C}{\Pi(\Delta^{(1)}\,;\, x_2 : A) \vdash (\nu x_1)(P[x_1/x] \mid Q) :: z : C}}{\Pi(\Delta^{(1)};??\Delta^{(2)}) \vdash (\nu x_2)\big(P[x_2/x] \mid (\nu x_1)(P[x_1/x] \mid Q)\big) :: z : C}$$

Spawn: reductions

$$(\nu x)(P \mid_x \rho[x \mapsto x_1, x_2].Q)$$

$$\longrightarrow$$

$$\rho[y \mapsto y_1, y_2 \mid y \in \mathrm{fn}(P) \setminus \{x\}].(\nu x_2)\big(P^{(2)} \mid_{x_2} (\nu x_1)(P^{(1)} \mid_{x_1} Q)\big)$$

Spawn: structural congruence

$$\dfrac{\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 : A \vdash P :: u : C}{y : A \vdash \rho[y \mapsto y_1, y_2, y_3].P :: u : C}}{x : B \,;\, y : A \vdash \rho[x \mapsto \emptyset].\rho[y \mapsto y_1, y_2, y_3].P :: u : C}$$

Spawn: structural congruence

$$\dfrac{\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 : A \vdash P :: u : C}{y : A \vdash \rho[y \mapsto y_1, y_2, y_3].P :: u : C}}{x : B \,;\, y : A \vdash \rho[x \mapsto \emptyset].\rho[y \mapsto y_1, y_2, y_3].P :: u : C}$$

$$\equiv$$

$$\dfrac{\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 : A \vdash P :: u : C}{x : B \,;\, y_1 : A \,;\, y_2 : A \,;\, y_3 : A \vdash \rho[x \mapsto \emptyset].P :: u : C}}{x : B \,;\, y : A \vdash \rho[y \mapsto y_1, y_2, y_3].\rho[x \mapsto \emptyset].P :: u : C}$$

$$\rho[x \mapsto \emptyset].\rho[y \mapsto y_1, y_2, y_3].P \equiv \rho[y \mapsto y_1, y_2, y_3].\rho[x \mapsto \emptyset].P$$

$$\rho[\sigma_1].\rho[\sigma_2].P \equiv \rho[\sigma_2].\rho[\sigma_1].P$$

Spawn: merge

$$\dfrac{\dfrac{y_1 : A \,\mathbf{;}\, y_2 : A \,\mathbf{;}\, y_3 : A \vdash P :: u : C}{y : A \vdash \rho[y \mapsto y_1, y_2, y_3].P :: u : C}}{x : B \,\mathbf{;}\, y : A \vdash \rho[x \mapsto \emptyset].\rho[y \mapsto y_1, y_2, y_3].P :: u : C}$$

Spawn: merge

$$\dfrac{\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 : A \vdash P :: u : C}{y : A \vdash \rho[y \mapsto y_1, y_2, y_3].P :: u : C}}{x : B \,;\, y : A \vdash \rho[x \mapsto \emptyset].\rho[y \mapsto y_1, y_2, y_3].P :: u : C}$$

$$\longrightarrow$$

$$\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 \vdash P :: u : C}{x : B \,;\, y : A \vdash \rho[x \mapsto \emptyset, y \mapsto y_1, y_2, y_3].P :: u : C}$$

$$\rho[x \mapsto \emptyset].\rho[y \mapsto y_1, y_2, y_3].P \longrightarrow \rho[x \mapsto \emptyset, y \mapsto y_1, y_2, y_3].P$$

Spawn: merge

$$\dfrac{\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 : A \vdash P :: u : C}{y : A \vdash \rho[y \mapsto y_1, y_2, y_3].\,P :: u : C}}{x : B \,;\, y : A \vdash \rho[x \mapsto \emptyset].\,\rho[y \mapsto y_1, y_2, y_3].\,P :: u : C}$$

$$\longrightarrow$$

$$\dfrac{y_1 : A \,;\, y_2 : A \,;\, y_3 \vdash P :: u : C}{x : B \,;\, y : A \vdash \rho[x \mapsto \emptyset, y \mapsto y_1, y_2, y_3].\,P :: u : C}$$

$$\rho[x \mapsto \emptyset].\,\rho[y \mapsto y_1, y_2, y_3].\,P \longrightarrow \rho[x \mapsto \emptyset, y \mapsto y_1, y_2, y_3].\,P$$

$$\rho[\sigma_1].\,\rho[\sigma_2].\,P \longrightarrow \rho[\sigma_1 \ltimes \sigma_2].\,P$$

Spawn: merge

$$\begin{bmatrix} x \mapsto \emptyset \\ y \mapsto \{y_1, y_2, y_3\} \end{bmatrix} \ltimes \begin{bmatrix} y_2 \mapsto \emptyset \\ y_3 \mapsto \{y_4, y_5\} \\ z \mapsto z_1 \end{bmatrix} = \begin{bmatrix} x \mapsto \emptyset \\ y \mapsto \{y_1, y_4, y_5\} \\ z \mapsto z_1 \end{bmatrix}$$

Spawn: merge

$$(\sigma_1 \ltimes \sigma_2)(x) = \begin{cases} \sigma_2[\sigma_1(x)] \cup (\sigma_1(x) \setminus \mathrm{dom}(\sigma_2)) & x \in \mathrm{dom}(\sigma_1) \\ \sigma_2(x) & x \notin \mathrm{dom}(\sigma_1) \wedge x \notin \mathrm{im}(\sigma_1) \\ \bot & \text{otherwise} \end{cases}$$

Spawn typing

With merge we can get spawns $\rho[\sigma].P$ to go beyond just weaking/contraction.
To type those intermediate spawns we have $\sigma : \Delta_1 \rightsquigarrow \Delta_2$

$$[x \mapsto \{x_1, \ldots, x_n\} \mid x \in \Delta] : \Pi(\Delta) \rightsquigarrow \Pi\left(\Delta^{(1)} \mathbin{;} \cdots \mathbin{;} \Delta^{(n)}\right)$$

$$[x \mapsto \emptyset \mid x \in \Delta_1] : \Pi(\Delta_1 \mathbin{;} \Delta_2) \rightsquigarrow \Pi(\Delta_2) \qquad \frac{\sigma_1 : \Delta_0 \rightsquigarrow \Delta_1 \qquad \sigma_2 : \Delta_1 \rightsquigarrow \Delta_2}{(\sigma_1 \ltimes \sigma_2) : \Delta_0 \rightsquigarrow \Delta_2}$$

$$[x \mapsto \emptyset, y \mapsto \{y_1, y_2, y_3\}] : \Pi(x : B \mathbin{;} y : A) \rightsquigarrow \Pi(y_1 : A \mathbin{;} y_2 : A \mathbin{;} y_3 : A)$$

Spawn typing

With merge we can get spawns $\rho[\sigma].P$ to go beyond just weaking/contraction.
To type those intermediate spawns we have $\sigma : \Delta_1 \rightsquigarrow \Delta_2$

$$\frac{\sigma : \Delta_1 \rightsquigarrow \Delta_2 \qquad \Delta_2 \vdash P :: z : C}{\Delta_1 \vdash \rho[\sigma].P :: z : C}$$

Additives vs multiplicatives in BI

.. database example..

Meta-theoretical properties of πBI

- If $\Delta \vdash P :: x : A$ and $P \equiv Q$, then $\Delta \vdash Q :: x : A$

Meta-theoretical properties of πBI

- If $\Delta \vdash P :: x : A$ and $P \equiv Q$, then $\Delta \vdash Q :: x : A$
- If $\Delta \vdash P :: x : A$ and $P \longrightarrow Q$, then $\Delta \vdash Q :: x : A$

Meta-theoretical properties of πBI

- If $\Delta \vdash P :: x : A$ and $P \equiv Q$, then $\Delta \vdash Q :: x : A$
- If $\Delta \vdash P :: x : A$ and $P \longrightarrow Q$, then $\Delta \vdash Q :: x : A$
- Given a empty bunch Σ (only composed of $\emptyset_{\mathrm{m}}$ and $\emptyset_{\mathrm{a}}$) such that $\Sigma \vdash P :: z : A$ with $A \in \{\mathbf{1}_m, \mathbf{1}_a\}$, then either
  - $P \longrightarrow -$, or
  - $P \equiv \overline{z}\langle\rangle$, or $P \equiv \rho[\emptyset].\overline{z}\langle\rangle$

Meta-theoretical properties of πBI

- If $\Delta \vdash P :: x : A$ and $P \equiv Q$, then $\Delta \vdash Q :: x : A$
- If $\Delta \vdash P :: x : A$ and $P \longrightarrow Q$, then $\Delta \vdash Q :: x : A$
- Given a empty bunch Σ (only composed of $\emptyset_{\mathrm{m}}$ and $\emptyset_{\mathrm{a}}$) such that $\Sigma \vdash P :: z : A$ with $A \in \{\mathbf{1}_m, \mathbf{1}_a\}$, then either
  - $P \longrightarrow -$, or
  - $P \equiv \overline{z}\langle\rangle$, or $P \equiv \rho[\emptyset].\overline{z}\langle\rangle$
- If $\Delta \vdash P :: x : A$, then P is weakly normalizing

Meta-theoretical properties of πBI

- If $\Delta \vdash P :: x : A$ and $P \equiv Q$, then $\Delta \vdash Q :: x : A$
- If $\Delta \vdash P :: x : A$ and $P \longrightarrow Q$, then $\Delta \vdash Q :: x : A$
- Given a empty bunch Σ (only composed of $\emptyset_{\mathrm{m}}$ and $\emptyset_{\mathrm{a}}$) such that $\Sigma \vdash P :: z : A$ with $A \in \{\mathbf{1}_m, \mathbf{1}_a\}$, then either
  - $P \longrightarrow -$, or
  - $P \equiv \overline{z}\langle\rangle$, or $P \equiv \rho[\emptyset].\overline{z}\langle\rangle$
- If $\Delta \vdash P :: x : A$, then P is weakly normalizing
  - $\exists S.\, P \longrightarrow^* S \not\longrightarrow -$