# Parametricity for Ordered Functional Programming

Chris Martens
Joint WIP w/ Frank Pfenning

@ Northeastern PRL
Nov 8, 2024

Two big ideas:

1. Parametricity
2. Ordered functional programming

# CONTEXT: Parametric Polymorphism

$$map : (\alpha \to \beta) \to list\ \alpha \to list\ \beta$$

# CONTEXT: Parametric Polymorphism

$$map : (\alpha \to \beta) \to list\ \alpha \to list\ \beta$$

$$\forall \alpha, \beta.$$

# CONTEXT: Parametric Polymorphism

$$map : \forall \alpha, \beta.\ (\alpha \to \beta) \to list\ \alpha \to list\ \beta$$

$$map[int][bool] : (int \to bool) \to list\ int \to list\ bool$$

# CONTEXT: Parametric Polymorphism

$$map : \forall \alpha, \beta.\ (\alpha \to \beta) \to list\ \alpha \to list\ \beta$$

$$map[int][bool] : (int \to bool) \to list\ int \to list\ bool$$

SYSTEM F

Girard '71
Reynolds '74

# Theorems for Free! (Wadler '89)

# Theorems for Free!

$$\forall \alpha \,.\, \alpha \to \alpha$$

# Theorems for Free!

$$\Lambda\alpha.\lambda x{:}\alpha.x \;:\; \forall\alpha.\,\alpha\to\alpha$$

# Theorems for Free!

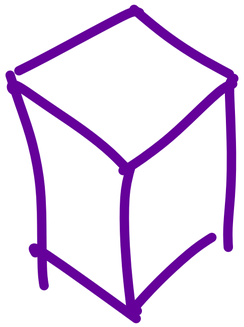
If $f : \forall \alpha . \alpha \to \alpha$

# Theorems for Free!

If $f : \forall \alpha . \alpha \to \alpha$

how can $f$ behave?

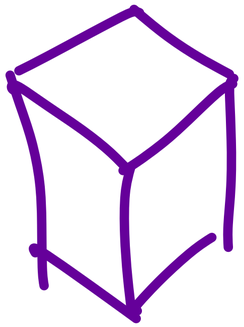
# Intuition Interlude



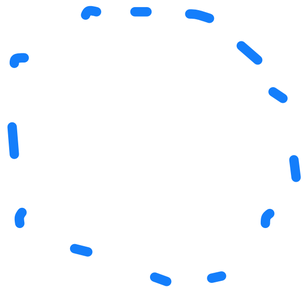
this is a

~~f~~nord

# Intuition Interlude



this is a fnord



please give me a fnord

# Intuition Interlude

only one choice!

this is a fnord

please give me a fnord

# Theorems for Free!

If $f : \forall \alpha . \alpha \to \alpha$

then for all $v : \sigma$

$(f\ v) \Downarrow v$

# Theorems for Free!

If $f : \forall \alpha . \alpha \to \alpha$

then $f \approx \Lambda \alpha . \lambda x{:}\alpha . x$

# Theorems for Free!

If $f : \forall \alpha . \alpha \to \alpha$

then $f \approx \Lambda \alpha . \lambda x{:}\alpha . x$

(exactly one canonical inhabitant

# Theorems for Free!

$\mathbb{1} \triangleq ()$

If $f : \forall \alpha . \alpha \to \alpha$

then $f \approx \Lambda \alpha . \lambda x{:}\alpha . x$

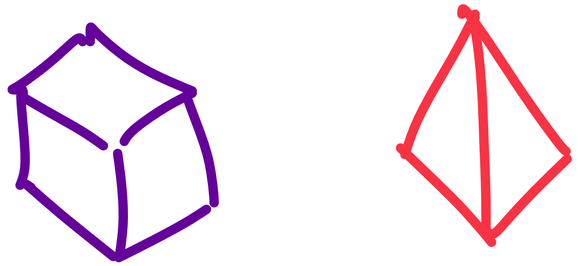
(exactly one canonical inhabitant

# Theorems for Free!

How about $\forall \alpha . \alpha \to \alpha \to \alpha$ ?

# Theorems for Free!

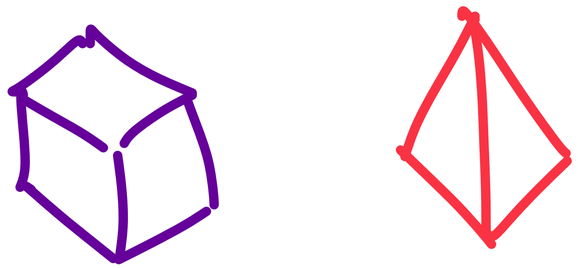
How about $\forall \alpha . \alpha \to \alpha \to \alpha$ ?



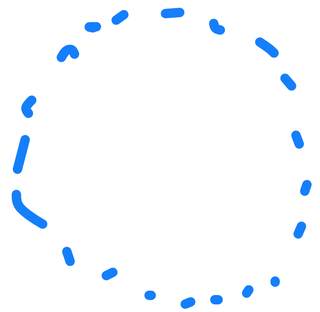
these are both fnords

# Theorems for Free!

How about $\forall \alpha . \alpha \to \alpha \to \alpha$ ?



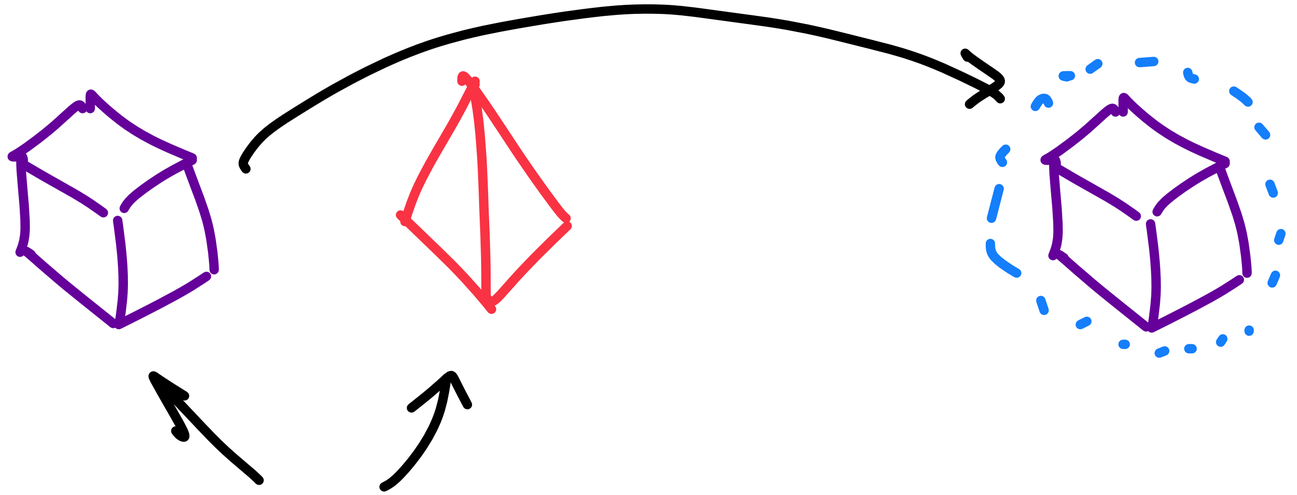
these are both fnords



please give me a fnord

# Theorems for Free!

How about $\forall \alpha . \alpha \to \alpha \to \alpha$ ?



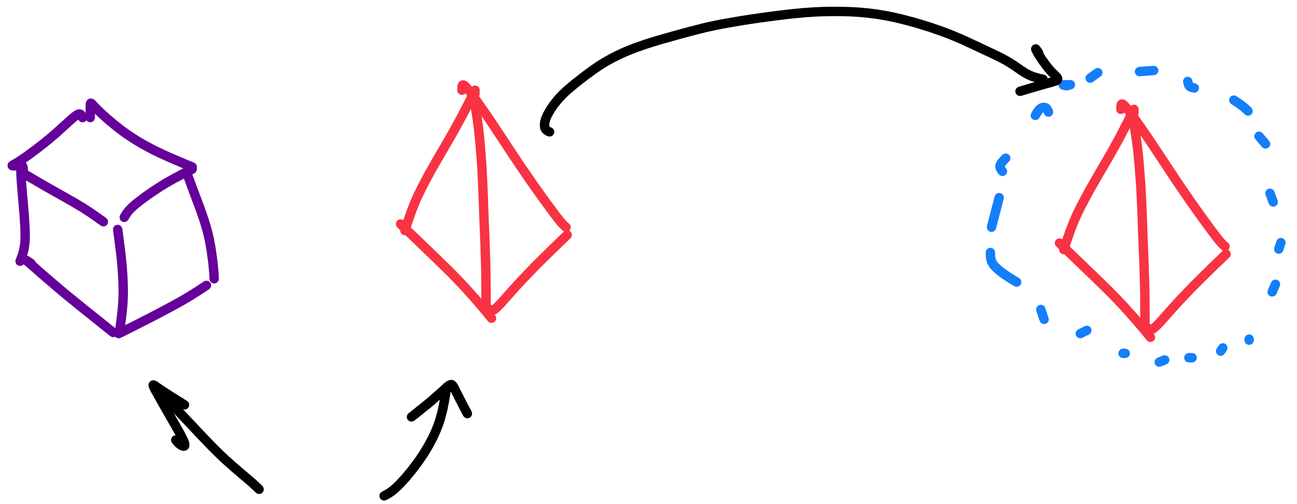
these are both fnords

please give me a fnord

choice ①

# Theorems for Free!

How about $\forall \alpha . \alpha \to \alpha \to \alpha$ ?



these are both fnords

please give me a fnord

choice ②

# Theorems for Free!

$$\forall \alpha . \alpha \to \alpha \to \alpha$$

Exactly <u>two</u> canonical inhabitants:

$\Lambda\alpha.\lambda x \lambda y.x$

$\Lambda\alpha.\lambda x \lambda y.y$

# Theorems for Free!

$\text{bool} \triangleq \forall\alpha.\ \alpha \to \alpha \to \alpha$

Exactly <u>two</u> canonical inhabitants:

$\text{true} \triangleq \Lambda\alpha.\lambda x\lambda y.x$

$\text{false} \triangleq \Lambda\alpha.\lambda x\lambda y.y$

Consider

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

how many canonical inhabitants?

Consider

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

how many canonical inhabitants?

1. $\Lambda\alpha.\lambda x.\lambda y.\ (x, y)$
2. $\Lambda\alpha.\lambda x.\lambda y.\ (y, x)$
3. $\Lambda\alpha.\ \lambda x.\lambda y.\ (x, x)$
4. $\Lambda\alpha.\Lambda\alpha.\lambda y.\ (y, y)$

# Theorems for Free!

In general: expressions

$$e : \forall \alpha . \tau$$

behave parametrically in $\alpha$.

# Canonical inhabitants

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

1. $\Lambda\alpha.\lambda x.\lambda y.\ (x, y)$
2. $\Lambda\alpha.\lambda x.\lambda y.\ (y, x)$
3. $\Lambda\alpha.\ \lambda x.\lambda y.\ (x, x)$
4. $\Lambda\alpha.\Lambda\alpha.\lambda y.\ (y, y)$

# Canonical inhabitants

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

1. $\Lambda\alpha.\lambda x.\lambda y.\ (x, y)$
2. $\Lambda\alpha.\lambda x.\lambda y.\ (y, x)$
3. $\Lambda\alpha.\ \lambda x.\lambda y.\ (x, x)$
4. $\Lambda\alpha.\Lambda\alpha.\lambda y.\ (y, y)$

what if I may not drop or copy input?

# Canonical inhabitants

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

1. $\Lambda\alpha.\lambda x.\lambda y.\ (x, y)$
2. $\Lambda\alpha.\lambda x.\lambda y.\ (y, x)$
3. ~~$\Lambda\alpha.\lambda x.\lambda y.\ (x, x)$~~
4. ~~$\Lambda\alpha.\Lambda\alpha.\lambda y.\ (y, y)$~~

what if I may not drop or copy input?

# Canonical inhabitants

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

1. $\Lambda\alpha.\lambda x.\lambda y.\ (x, y)$
2. $\Lambda\alpha.\lambda x.\lambda y.\ (y, x)$
3. $\Lambda\alpha.\ \lambda x.\lambda y.\ (x, x)$
4. $\Lambda\alpha.\Lambda\alpha.\lambda y.\ (y, y)$

What if I can't drop, copy, or reorder?

# Canonical inhabitants

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

1. $\Lambda\alpha.\lambda x.\lambda y.\ (x, y)$
2. ~~$\Lambda\alpha.\lambda x.\lambda y.\ (y, x)$~~
3. ~~$\Lambda\alpha.\lambda x.\lambda y.\ (x, x)$~~
4. ~~$\Lambda\alpha.\Lambda\alpha.\lambda y.\ (y, y)$~~

What if I can't drop, copy, or reorder?

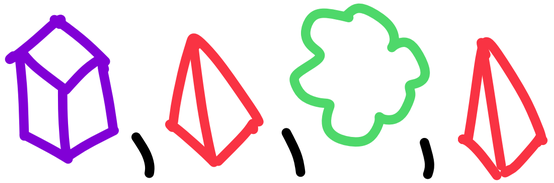
# Theorems for Free!

Wadler: $f : \forall \alpha.\ \mathrm{list}\ \alpha \to \mathrm{list}\ \alpha$

$f\ [\tau]\ l$ must return a list whose elements are drawn from $l$

$\forall \alpha.\ \text{list}\ \alpha \to \text{list}\ \alpha$

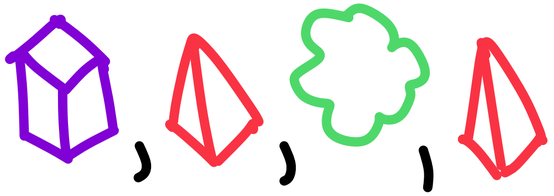
here are some fnords:



give me some fnords!

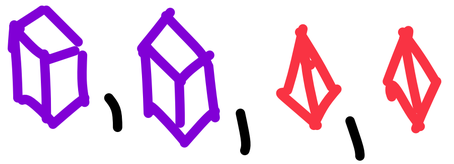
$\forall \alpha.\ \mathrm{list}\ \alpha \to \mathrm{list}\ \alpha$

here are some fnords:



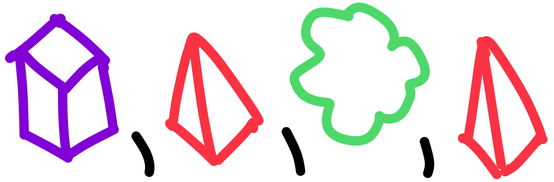
give me some fnords!

choices (sample):

1. [] (ignore input)
2.  (copy input)

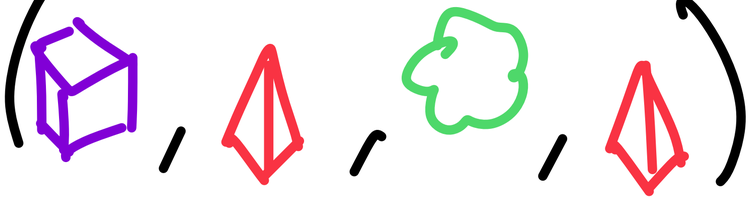
$\forall \alpha.\ \mathrm{list}\ \alpha \to \mathrm{list}\ \alpha$

here are some fnords:



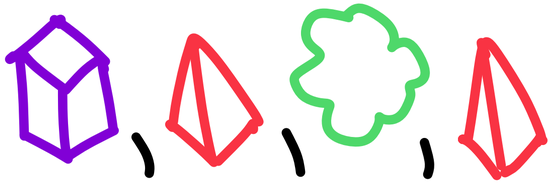
give me some fnords!
(no drop or copy)

Choices: permutations

$\pi$(  )

$\forall \alpha.\ \text{list}\ \alpha \to \text{list}\ \alpha$

here are some fnords:

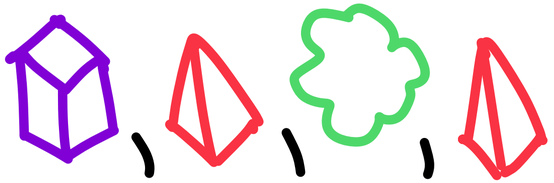


give me some fnords!
(no drop or copy
or swap)

$\Rightarrow$ exact input

$\forall \alpha.\ \text{list}\ \alpha \to \text{list}\ \alpha$

here are some fnords:



give me some fnords!

- Exercise: other combos of

| drop | copy | swap |
|---|---|---|
| √/x | √/x | √/x |

RQ: Can we generalize theorems for free to give us STRONGER theorems for WEAKER types?

RQ: Can we generalize theorems for free to give us STRONGER theorems for WEAKER types?

(spoiler: yes!)

# Theorems for Free!

How does it work?

# Theorems for Free!

How does it work?

IF $\Gamma \vdash e : \tau$

THEN $e \approx e' \in [\![\tau]\!]$

strong enough so $[\![\forall \alpha . \tau]\!]$ captures parametricity

# Theorems for Free!

How does it work?

IF $\Gamma \vdash e : \tau$ THEN $e \approx e' \in [\![\tau]\!]$

Syntax $\Longleftrightarrow$ Semantics

# Theorems for Free!

How does it work?

$\Gamma \vdash e : \tau$ $\qquad$ $e \approx e' \in [\![\tau]\!]$

I. Syntax $\Longleftrightarrow$ Semantics II.

Ordered FP $\qquad$ Relational Parametricity

Syntax:

Ordered Functional Programming

Goal of this research project:

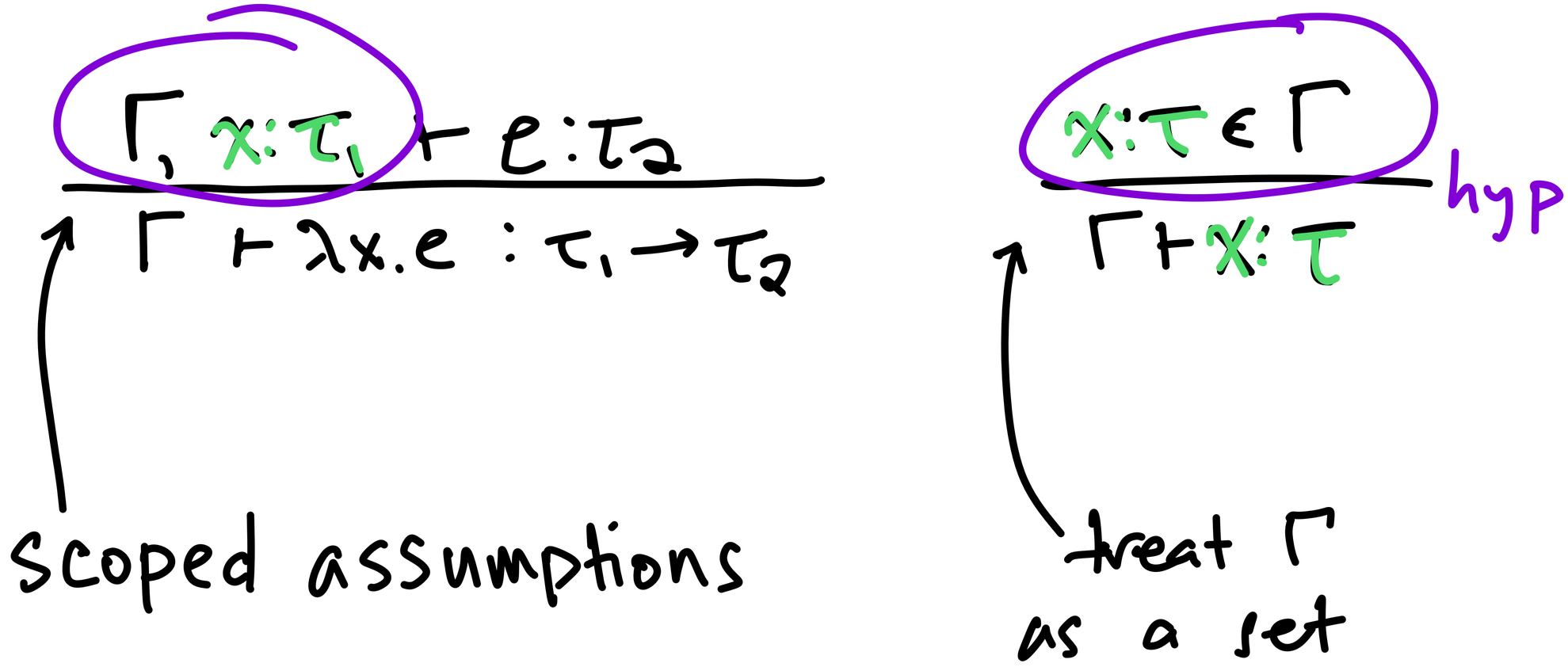
extend parametricity to type systems that capture resource use

(drop, copy, swap)

# Type system for $\lambda$ calculus

$$\frac{\Gamma, x:\tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2} \qquad \frac{x:\tau \in \Gamma}{\Gamma \vdash x : \tau}$$

# Type system for λ calculus



# Type system for $\lambda$ calculus

$$\frac{\Gamma, x:\tau_1 \vdash e:\tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2} \qquad \frac{}{x:\tau \vdash x:\tau}\ \text{hyp}'$$

alternative: articulate assumptions about $\Gamma$

# STRUCTURAL RULES

$$\frac{\Gamma \vdash J}{\Gamma, A \vdash J}\,(W)\ \text{weakening} \qquad \frac{\Gamma, A, A \vdash J}{\Gamma, A \vdash J}\,(C)\ \text{contraction}$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash J}{\Gamma_1, B, A, \Gamma_2 \vdash J}\,(X)\ \text{change}$$

AKA

# STRUCTURAL RULES

drop

$$\frac{\Gamma \vdash J}{\Gamma, A \vdash J}\,(W)$$ weakening

$$\frac{\Gamma, A, A \vdash J}{\Gamma, A \vdash J}\,(C)$$ contraction

copy

Swap

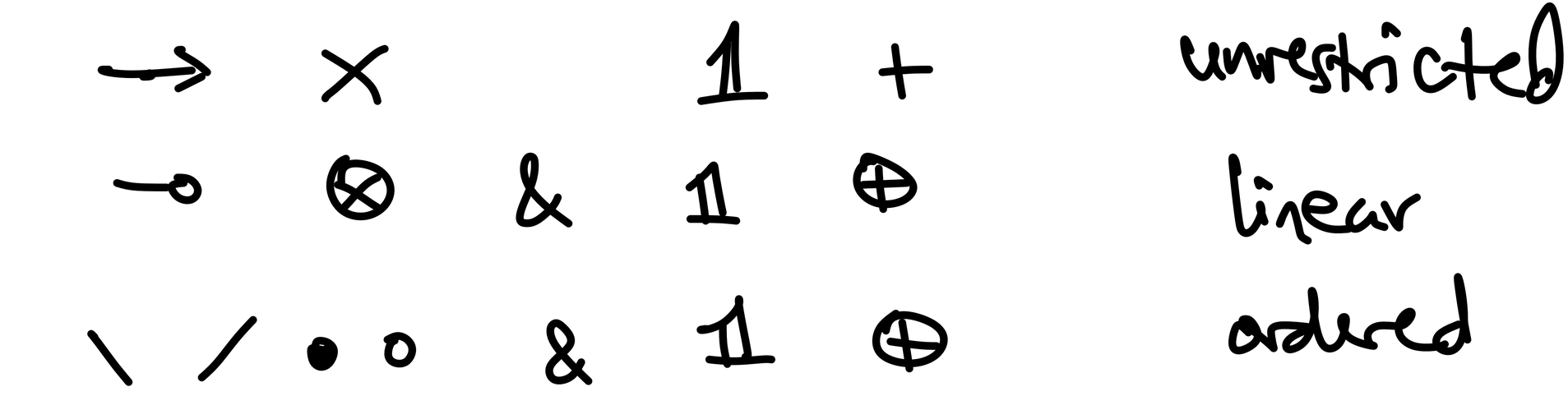
$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash J}{\Gamma_1, B, A, \Gamma_2 \vdash J}\,(X)$$ change

# SUB structural type systems

deny (or: distinguish) the use of 1 or more structural rules.

| | drop | copy | swap |
|---|---|---|---|
| unrestricted | ✓ | ✓ | ✓ |
| linear | | | ✓ |
| affine | ✓ | | ✓ |
| strict | | ✓ | |
| ordered | ✗ | ✗ | ✗ |

# Connectives:



# Ordered Conjunction: Fuse ($\bullet$)

$$\frac{\Delta \vdash e_1 : \tau_1 \qquad \Delta' \vdash e_2 : \tau_2}{\Delta\ \Delta' \vdash (e_1, e_2) : \tau_1 \bullet \tau_2}\ \bullet I$$

$$\frac{\Delta \vdash e : \tau_1 \bullet \tau_2 \qquad \Delta_L (x : \tau_1)(y : \tau_2) \Delta_R \vdash e' : \tau}{\Delta_L \Delta\ \Delta_R \vdash \text{split } e \text{ as } (x, y).e' : \tau}\ \bullet E$$

Lists that keep their order

$$\text{list}\ \alpha = \mathbb{1} \oplus (\alpha \bullet (\text{list}\ \alpha))$$

$$\text{revlist}\ \alpha = \mathbb{1} \oplus (\alpha \circ (\text{list}\ \alpha))$$

$$\text{reverse} : \forall\alpha.\ \text{revlist}\ \alpha \setminus \text{list}\ \alpha$$

TFF: Can only be the reverse fn

# Adjoint formulations combining logics

| | $m = 1$ | $m = 0$ |
|---|---|---|
| $\tau_1 \times_m \tau_2$ | $\tau_1 \otimes \tau_2$ | $\tau_1 \bullet \tau_2$ |
| $\tau_1 \xrightarrow[m]{} \tau_2$ | $\tau_1 \multimap \tau_2$ | $\tau_2 \setminus \tau_1$ |

(Reed '09, ..., Jang/Roshal/Pfenning/Pientka '24)

$$\forall \alpha.\ \alpha \xrightarrow{m} \alpha \xrightarrow{m} \alpha \times_m \alpha$$

When $m = 1$:

$\lambda x \lambda y.(x, y)$ ✓

$\lambda x.\lambda y.(y, x)$ ✓

When $m = 0$:

$\lambda x.\lambda y.(x, y)$ only choice

# SEMANTICS:

# Relational Parametricity

# Logical Equivalence

When does $e \sim e' \in [\![\tau]\!]$ ?

# Logical Equivalence

When does $e \sim e' \in [\![\tau]\!]$ ?

Main idea (Reynolds '83):

interpret types ($[\![\tau]\!]$) as relations on terms.

# Logical Equivalence

When does $e \sim e' \in [\![\tau]\!]$ ?

Main idea (Reynolds '83):

interpret types ($[\![\tau]\!]$) as relations on terms.

of potentially different types!

# Logical Equivalence

When does $e \sim e' \in [\![\tau]\!]$ ?

$: \tau_1$ $: \tau_2$ $= \text{some}$

$R : \text{Rel}(\tau_1, \tau_2)$

# Logical Equivalence

$v \sim v' \in [\![\forall \alpha.\tau]\!]$ iff (polymorphic terms)

for all types $\sigma, \sigma'$
and relations $R : \mathrm{Rel}(\sigma, \sigma')$

$$v[\sigma] \approx v'[\sigma'] \in [\![R/\alpha]\tau]\!]$$

# Logical Equivalence

$v \sim v' \in [\![ R ]\!]$ 当且仅当 (relations)

$(v, v') \in R$

# Logical Equivalence

$v \sim v' \in [\![\tau]\!]$ defined inductively on $[\![\tau]\!]$

# Logical Equivalence

$e \approx e' \in [\![\tau]\!]$ expressions

$v \sim v' \in [\![\tau]\!]$ values

# Logical Equivalence

$e \approx e' \in [\![\tau]\!]$ iff (Expressions)

$$\left.\begin{array}{l} e \mapsto^* v \\ e' \mapsto^* v' \end{array}\right\} \text{and} \quad v \sim v' \in [\![\tau]\!]$$

(or both $e$ & $e'$ diverge)

# Logical Equivalence

$f \sim f' \in [\![ \tau_1 \to \tau_2 ]\!]$ iff (functions)

for all $v \sim v' \in [\![ \tau_1 ]\!]$

$(f\ v) \approx (f'\ v') \in [\![ \tau_2 ]\!]$

# Logical Equivalence

$v \sim v' \in [\![\tau_1 \times \tau_2]\!]$ iff (pairs)

$\exists v_1, v_2, v_1', v_2'$ s.t.

$v = (v_1, v_2)$ $\quad v_1 \sim v_1' \in [\![\tau_1]\!]$

$v' = (v_1', v_2')$ $\quad v_2 \sim v_2' \in [\![\tau_2]\!]$

Example: $f : \forall\alpha.\, \alpha \to \alpha$

Free thm: for all $v, \tau$, if $\cdot \vdash v : \tau$,

$f[\tau]\, v \Downarrow v$ (or diverges)

Example: $f : \forall\alpha.\, \alpha \to \alpha$

Free thm: for all $v, \tau$, if $\cdot \vdash v : \tau$,

$f[\tau]\, v \Downarrow v$ (or diverges)

$f : \forall\alpha.\, \alpha \to \alpha$

To get the free theorem:

- Fix arbitrary $\sigma_0$, $V_0$
- Apply Parametricity: $f \sim f \in [\![\forall\alpha.\, \alpha \to \alpha]\!]$
- Unroll defn: for <u>all</u> $\sigma, \sigma', R : \mathrm{Rel}(\sigma, \sigma')$ ...
- Pick $\sigma = \sigma' = \sigma_0$, $R = \{(V_0, V_0)\}$
- Then, $f\,[\sigma_0]\,[V_0]$ <u>must</u> eval to $V_0$.

This example differentiates modes:

$$\forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$$

This example differentiates modes:

$f : \forall \alpha.\ \alpha \to \alpha \to \alpha \times \alpha$

$f\,[\sigma]\ v_1\ v_2 \Downarrow (w_1, w_2)$

$(v_1, v_1) \in R$

$(v_2, v_2) \in R$

$w_1 = v_1$ or $v_2$

$w_2 = v_1$ or $v_2$

The trick: equivalence tracks resourse use

$$v \sim v' \in [\![\tau]\!] @ \omega$$

The trick: equivalence tracks resourse use

$$v \sim v' \in [\![\tau]\!] @ \omega$$

open terms!

The trick: equivalence tracks resourse use

$$v \sim v' \in [\![\tau]\!] @ \omega$$

e.g.: $(v_1, v_2) \sim (v_1', v_2') \in [\![\tau_1 \bullet \tau_2]\!] @ \omega_1 \cdot \omega_2$

iff $v_1 \sim v_1' \in [\![\tau_1]\!] @ \omega_1$

$v_2 \sim v_2' \in [\![\tau_2]\!] @ \omega_2$

The trick: equivalence tracks resourse use

worlds $\omega ::= \varepsilon \mid \omega_1 \cdot \omega_2$

where

$\omega \cdot \varepsilon \equiv \omega \equiv \varepsilon \cdot \omega$ (unit)

$\omega_1 \cdot (\omega_2 \cdot \omega_3) \equiv \omega_1 \cdot (\omega_2 \cdot \omega_3)$ (assoc)

(AKA monoid)

# Easy to modify the algebraic structure!

worlds $\omega ::= \varepsilon \mid \omega_1 \cdot \omega_2$

where

$$\omega \cdot \varepsilon \equiv \omega \equiv \varepsilon \cdot \omega \quad \text{(unit)}$$

$$\omega_1 \cdot (\omega_2 \cdot \omega_3) \equiv \omega_1 \cdot (\omega_2 \cdot \omega_3) \quad \text{(assoc)}$$

$$\omega_1 \cdot \omega_2 \equiv \omega_1 \cdot \omega_2 \quad \text{(comm)}$$

# Related Work:

"Relational Parametricity for a Polymorphic Linear Lambda Calculus"

Zhao, Zhang, Zdancewic,
APLAS 2010 (ZZZ10)

# Project status:

- Proofs mostly done

- Implementation! (SNAX language)

- Mixing modes?

- No recursive types or fixed points

- Applications/cool examples?

# Project status:

- Proofs mostly done

- Implementation! (SNAX language)

- Mixing modes?

- No recursive types or fixed points

- Applications/cool examples?

thanks! ♡ c.martens@northeastern.edu