

Remember the adequacy criterion!
 ↳ formulate up-front  natural aka
commutes w/
categorification.

e.g) STLC

$$\Gamma \vdash_{STLC} M : A^\top \iff \text{COMPOSITIONAL BISECTION.}$$

$$\Gamma \vdash_{LF_{\Sigma_{STLC}}} M : \text{el } [A^\top] \quad \text{LF context.}$$

where $\vdash_{X_1 : A_1, \dots, X_n : A_n} = \overbrace{x_1 : \text{el } [A_1], \dots, x_n : \text{el } [A_n]}$

etc.

Ex μ, ν types w/ positivity restriction |

Mar 30 2021

Recap: "intrinsic" encodings of STLC statics

$\text{ty} : \underline{\text{sort}}$. $\text{one} : \text{ty}$. $\text{two} : \text{ty}$.

$\text{prod} : \text{ty} \rightarrow \text{ty} \rightarrow \text{ty}$. $\text{arr} : \text{ty} \rightarrow \text{ty} \rightarrow \text{ty}$.

$\text{el} : \text{ty} \rightarrow \underline{\text{sort}}$. "elements of a type".

$\text{null} : \text{el}(\text{one})$. (etc)

Definitional equivalence:

$\text{eqv} : \alpha : \text{ty} \rightarrow \text{el}(\alpha) \rightarrow \text{el}(\alpha) \rightarrow \underline{\text{sort}}$.

$\text{refl} : \alpha : \text{ty} \rightarrow \text{m} : \text{el}(\alpha) \rightarrow \text{eqv } \alpha \sim \text{m}$.

$\text{sym} : \alpha : \text{ty} \rightarrow \text{m}, \text{m}' : \text{el}(\alpha) \rightarrow \text{eqv } \text{a m m}' \rightarrow \text{eqv a m' m}$.

$\text{trans} : \dots$ similar ...

↓

$\text{Pair-}\alpha_1 : \alpha_1, \alpha_2 : \text{ty} \rightarrow m_1, m_1' : \text{el}(\alpha_1) \rightarrow m_2, m_2' : \text{el}(\alpha_2) \rightarrow$
 $\text{egv } \alpha_1, m_1, m_1' \rightarrow \text{egv } \alpha_2, m_2, m_2' \rightarrow$
 $\text{egv} (\text{prod } \alpha_1 \alpha_2) (\text{Pair } m_1, m_2) (\text{Pair } m_1', m_2') .$

etc

$\text{Prod-}\beta_1 : \alpha_1, \alpha_2 : \text{ty} \rightarrow m_1 : \text{el}(\alpha_1) \rightarrow m_2 : \text{el}(\alpha_2) \rightarrow$
 $\text{egv } \alpha_1 (\text{fst } \alpha_1, \alpha_2 (\text{Pair } \alpha_1, \alpha_2, m_1, m_2)) m_1 .$

$\text{Prod-}\beta_2 : \dots \text{similar} \dots$

$\text{Prod-}\eta : \alpha_1, \alpha_2 : \text{ty} \rightarrow m : \text{el} (\text{prod } \alpha_1 \alpha_2) \rightarrow$
 $\text{egv} (\text{prod } \alpha_1 \alpha_2) m$
 $(\text{Pair } \alpha_1 \alpha_2 (\text{fst } \alpha_1, \alpha_2, m) (\text{snd } \alpha_1, \alpha_2, m)) .$

In short:

$\text{---} : \alpha_1, \alpha_2 : \text{ty} \rightarrow \text{el} (\text{prod } \alpha_1 \alpha_2) \cong (\text{el}(\alpha_1) \times \text{el}(\alpha_2))$
 i.e. elements of products are pairs of elements

$\text{arr-}\beta : a_1, a_2 : t \rightarrow m : (\text{el}(a_1) \rightarrow \text{el}(a_2)) \rightarrow m : \text{el}(a_1) \rightarrow$
 $\text{eg} \quad a_2 \text{ cap } a_1 \text{ arr } (\lambda x. \text{el}(a_1)(x, \text{arr}(x))) \downarrow m_1$
 $(m_2 m_1)$.

$\text{arr-}\eta : a_1, a_2 : t \rightarrow m : \text{el}(\text{arr } a_1, a_2) \rightarrow$
 $\text{eg } (\text{prod } a_1, a_2) \text{ in } (\lambda x. \text{el}(a_1)(x, \text{arr } m x)))$.

In short :

$- : a_1, a_2 : t \rightarrow \text{el}(\text{arr } a_1, a_2) \cong (\text{el}(a_1) \rightarrow \text{el}(a_2))$
 - etc of arrows arrows of el's
 ie. formally, functions are open terms!
 (semantically, not at all).

Given intrinsic encodings of types

$$\Gamma \vdash F_{\sum_{\text{src}}}^{\text{can}} M : \text{el } A$$

wrt compositional encodings $\Gamma \dashv$.

How to formulate definitional equivalence?

Judgments-as-Types Princ. } formalistic w/
} given by LF sig's.

Introduce equivalence judgment:

$$\rightarrow \text{eq} : a : \text{ty} \rightarrow \underline{\text{el } a} \rightarrow \underline{\text{el } a} \rightarrow \text{sort}$$

Populate this family so that it is adequate
wrt (informal) formalization given earlier.

What is def'l equivalence?

1) equivalence relation

refl : $a : \text{ty} \rightarrow m : \text{ela} \rightarrow \text{eq } a \text{ } m \text{ } m$.

sym : $a : \text{ty} \rightarrow m, m' : \text{ela} \rightarrow \text{eq } a \text{ } m \text{ } m' \rightarrow \text{eq } a \text{ } m' \text{ } m$.

trans : — exercise —

2) compatibility

pair : $a_1, a_2 : \text{ty} \rightarrow m_1, m'_1 : \text{ela}_1 \rightarrow m_2, m'_2 : \text{ela}_2 \rightarrow$
 $\text{eq } a_1 \text{ } m_1 \text{ } m'_1 \rightarrow \text{eq } a_2 \text{ } m_2 \text{ } m'_2 \rightarrow$
 $\text{eq } (\text{prod } a_1 a_2) (\text{pair } m_1 m_2) (\text{pair } m'_1 m'_2)$.



$\text{lam} : \alpha_1, \alpha_2 : \text{ty} \rightarrow M_2, M'_2 : \text{el}(\alpha_1) \rightarrow \text{el}(\alpha_2) \rightarrow$
 $(x : \text{el}(\alpha_1) \rightarrow \text{eq}_{\alpha_2}(M_2 x) (M'_2 x)) \rightarrow$
 $\text{eqv} (\text{arr} \alpha_1 \alpha_2) (\text{lam } \alpha_1 \alpha_2 (\lambda_{\text{el}(\alpha_1)}(x, M_2 x))).$
 $(\text{lam } \alpha_1 \alpha_2 (\lambda_{\text{el}(\alpha_1)}(x, \underline{\underline{M'_2 x}})))$
 $\equiv_{\text{LF}} M'_2.$

3) β principle's "vs" $\text{el}(\text{arr } \alpha_1 \alpha_2)$

$\beta \rightarrow : \left\{ \begin{array}{l} \alpha_1, \alpha_2 : \text{ty} \rightarrow M_2 : \text{el}(\alpha_1) \rightarrow \text{el}(\alpha_2) \rightarrow \\ M_1 : \text{el}(\alpha_1) \rightarrow \end{array} \right.$
 $\left\{ \begin{array}{l} \text{eq}_{\alpha_2} (\text{app } \alpha_1 \alpha_2 (\text{lam } \alpha_1 \alpha_2 (\lambda_{\text{el}(\alpha_1)}(x, \underline{\underline{M_2 x}}))) M_1) \\ (M_2 M_1) \end{array} \right.$

4) η -func's

$\eta \rightsquigarrow : a_1, a_2 : \text{ty} \rightarrow m : \text{el}(\text{arr } a_1, a_2) \rightarrow$

$$\text{eq } (\text{arr } a_1, a_2) \rightsquigarrow (\lambda a_1 \ a_2 . \ \text{r}_{\text{app}(m, x)} \cdot \\ (\lambda_{\text{el}(a_1)} (x . \ \text{app } a_1 \ a_2 \ m \ x)))) .$$

The $\beta\eta$ laws establish an isomorphism
btw $\text{el}(\text{arr } a_1, a_2)$ and $\text{el}(a_1) \rightarrow \text{el}(a_2)$!

(iso ~ "up to eq") .

Above encodings of def'l equivalence is
EXTENSIONIC

in that eg has no meaning other than
being some family in LF.

What is an INTRINSIC encoding?

X eg: - - -
Assume the ff axioms of def'l equivalence of LF.
congruence!

ηL $m : el(one) \vdash m \equiv \text{null} : el(one)$.

$\beta\lambda$ $a_1, a_2 : ty, m_1 : el(a_1), m_2 : el(a_2) \vdash$
 $\text{proj}_{a_1, a_2}(\text{pair } a_1, a_2, m_1, m_2)) \equiv m_1 : el(a_1)$

(tc)

With equations we can express the iso mentioned above :

$$a_1, a_2 : \text{ty} + \\ \text{el}(\text{arr } a_1, a_2) \cong \text{el}(a_1) \rightarrow \text{el}(a_2)]]$$

means

- ✓ app : $\text{el}(a_1 \rightarrow a_2) \rightarrow \text{el}(a_1) \rightarrow \text{el}(a_2)$
- ✓ lam : $(\text{el}(a_1) \rightarrow \text{el}(a_2)) \rightarrow \text{el}(\text{arr } a_1, a_2)$.
- ✓ app \circ lam \equiv id : $\text{el}(a_1) \rightarrow \text{el}(a_2)$
- ✓ lam \circ app \equiv id : $\text{el}(\text{arr } a_1, a_2)$

uniquely determines $\text{arr } a_1, a_2$!

$$\boxed{\begin{array}{c} \Gamma \times A \rightarrow B \\ \cong \\ \Gamma \rightarrow B^A \end{array}}$$

1. (Purely) syntactic encodings :
- (twelf) a logic / formalism is a signature { more gen'l
retaining decidability }
- (spec of generators)
2. Semantic encodings
- generators as above
governed by relations (equations) } more expressive
sacrifice decidability
- Later : (2) can be reduced to (1)
in an extended LF (later).

LF Type Theory (basic version)

Dependent type theory w/ Π -types +
"small"

one universe

of "small" types

over el'ts of
small types.

ty : sort

e.g) el : $a : \underline{\text{ty}} \rightarrow \underline{\text{sort}}$.
universe "large" (not small)

"el[A]" is small"

"ty is small"

law: $\frac{el(a_1) \rightarrow el(a_2)}{\chi : el(a_1) \rightarrow el(a_2)}$

small

(see Semantic LF note) \Downarrow universe \Downarrow small

kind
classes

$K ::= \underline{\text{sort}} \mid a : S \rightarrow K$

"type"

families
Sort
(judg)

$S ::= \alpha \mid a : S_1 \rightarrow S_2 \mid \cancel{AP(S, M)}$ "type"

objects
terms

$M ::= c \mid x \mid \lambda_{S_1}(x. M) \mid \cancel{AP(M_1, M_2)}$

"small"

ctx

$\Gamma ::= \cdot \mid \Gamma, x : S$

"small"

sig

$\Sigma ::= \cdot \mid \Sigma, \alpha : K$

"large"

$\Gamma \text{ ctx}$

$[\Gamma \equiv \Gamma']$

$\Sigma \text{ sig}$ $[\Sigma \equiv \Sigma']$

$\left\{ \begin{array}{l} \Gamma \vdash_{\Sigma} K \text{ cls} \quad \Gamma \vdash_{\Sigma} K \equiv K' \text{cls} \\ \Gamma \vdash_{\Sigma} S :: K \quad \Gamma \vdash_{\Sigma} S \equiv S' :: K \\ \Gamma \vdash_{\Sigma} M :: S \quad \Gamma \vdash_{\Sigma} \underline{M \equiv M'} :: S \end{array} \right\}$ mutually recursive.

April 1, 2021

The "syntactic" LF type theory:

- no equations (later)
- decidable!

Two forms:

- declarative - requires an ^{extensional} equivalence checker
- canonical - no equiv checker - subst. is tricky.
"hereditary substitution"

$$\frac{\Gamma \vdash_{\Sigma} A : K' \quad \Gamma \vdash_{\Sigma} K \equiv K'}{\Gamma \vdash_{\Sigma} A : K} \quad (\text{invariance})$$

$$\frac{\Gamma \vdash_{\Sigma} M : A' \quad \Gamma \vdash_{\Sigma} A \equiv A' : K}{\Gamma \vdash_{\Sigma} M : A} \quad (\text{invariance})$$

→ Rules are no longer syntax - directed!

eg) $\frac{\Gamma \vdash_{\Sigma} M : x : A_1 \rightarrow A_2 \quad \Gamma \vdash_{\Sigma} M_1 : B_1}{\Gamma \vdash_{\Sigma} \alpha(M, M_1) : [M_1/x]A_2}$

(I) $x : A_1 \rightarrow A_2 \equiv x : B_1 \rightarrow A_2 : \text{SNT}$

(II) $A_1 \equiv B_1$

$$\frac{\Gamma \vdash_{\Sigma} M : x:A_1 \rightarrow A_2 \quad \Gamma \vdash_{\Sigma} M_1 : A_1}{\Gamma \vdash_{\Sigma} \text{ap}(M, M_1) : [M_1/x]A_2}$$

$$\frac{\Gamma, x:A_1 \vdash_{\Sigma} M_2 : A_2 \quad \Gamma \vdash_{\Sigma} A_1 : \text{sort}^{\text{typ}}}{\Gamma \vdash_{\Sigma} \lambda_{A_1}(x.M_2) : x:A_1 \rightarrow A_2}$$

Invariants / Admissibilities

- If $\Gamma \vdash_{\Sigma} M : A$ then $\Gamma \vdash_{\Sigma} A : \text{sort}$
- If $\Gamma \vdash_{\Sigma} A : K$ then $\Gamma \vdash_{\Sigma} K \text{ kind}$
- If $\Gamma \vdash_{\Sigma} J$ and $\Delta \vdash_{\Sigma} \delta : F$ then $\Delta \vdash_{\Sigma} \hat{f}(J)$.
- If $\Gamma \vdash_{\Sigma} J$ and $\Delta \vdash_{\Sigma} \delta : F$ then $\Delta \vdash_{\Sigma} \approx$
(weakening, contraction, exchange, subst.)

$$\frac{}{\Gamma \vdash \sum, \alpha : K, \Sigma' \quad \alpha : K}$$

$$\frac{\Gamma \xrightarrow{x:A} \Gamma' \vdash x : A}{\text{NR}}$$

$$\frac{\Gamma \vdash \Sigma \ M : x : A_1 \rightarrow A_2}{\Gamma \vdash \Sigma \ M \equiv \lambda_{A_1}(x. \alpha^*(M, x)) : x : A_1 \rightarrow A_2} (\hookrightarrow)$$

$$\frac{\Gamma \vdash \Sigma \ A_1 \equiv A'_1 : \text{sort} \quad \Gamma, x : A_1 \vdash \Sigma \ A_2 \equiv A'_2 : \text{sort}}{\Gamma \vdash \Sigma \ x : A_1 \rightarrow A_2 \equiv x : A'_1 \rightarrow A'_2 : \text{sort}}$$

R, S, T for all three equiv's.

Nasty Pitfall

(?) If $\Gamma \vdash_{\Sigma} x : A_1 \rightarrow A_2 \equiv x : A'_1 \rightarrow A'_2 : \underline{\text{soft}}$,
then $\Gamma \vdash_{\Sigma} A_1 \equiv A'_1$ and $\Gamma, x : A'_1 \vdash A_2 \equiv A'_2 : \underline{\text{soft}}$.

TRANSITIVITY! (screws any direct proof)

ETC.

AUTOMATH
1968

de Bruijn

Is type checking decidable?

1. Reduce t.c. to eqn. checking

2. Give eqn. check alg (fix Σ)

key idea:
skeletons {

$$\begin{array}{l} K \xrightarrow[K_0]{} K' [\Gamma] \\ A \xrightarrow[A_0]{} A' \downarrow K [\Gamma] \quad \left\{ \begin{array}{l} \text{"canonical"} \\ \text{neutral} \end{array} \right. \\ C \xrightarrow[C_0]{} C' \uparrow K [\Gamma] \\ M \xrightarrow[M_0]{} M' \downarrow A [\Delta] \\ u \xrightarrow[u_0]{} u' \uparrow A [\Gamma] \end{array}$$

(Harper-Pfenning paper)

Skeleton

A^o skeleton of A

simple types !

$$\alpha^o = \alpha \quad (x : A_i \rightarrow A_j)^o = A_i^o \rightarrow A_j^o$$

$$\boxed{A(M)^o = A^o} \quad \text{drop indices in families}$$

$$\text{eg)} \quad \text{el}(\Gamma_A)^o = \text{el}.$$

Algorithm is driven by the skeleton
of the dep. type !

Alternative approach:

1. Formulate canonical LF, which is syntax-directed.
 - no such things as a non-canonical form
 - ∴ no need for equality checkings!
2. Substitution must be reformulated as HEREDITARY SUBST.
 - $[M/x]_A^N \quad [M/x]_{A^o}^N \quad \boxed{\begin{array}{l} [\lambda(x.M)y] \\ \alpha p(y, N) \end{array}}$
 - $\text{etc. } \Gamma \text{ need types! } \Pi$
 - $\alpha p(\lambda x.M, N)$
 - Not canonical!

Canonical STLC (preliminary example)

$$\Gamma ::= \cdot \mid C_i \times^\uparrow A$$

$$\Gamma \vdash M \downarrow A \quad M \text{ is canonical at type } A.$$

$$\Gamma \vdash U \uparrow A \quad U \text{ is neutral of type } A$$

$$\frac{}{\Gamma \vdash \leftrightarrow \perp 1} \quad \frac{\Gamma \vdash M_1 \downarrow A_1 \quad \Gamma \vdash M_2 \downarrow A_2}{\Gamma \vdash \langle M_1, M_2 \rangle \downarrow A_1 \times A_2}$$

$$\frac{C_i \times^\uparrow A_1 \vdash M_2 \downarrow A_2}{\Gamma \vdash \lambda_{A_1}(x, M_2) \downarrow A_1 \rightarrow A_2}$$

variable
types

(*)

$$\frac{\Gamma \vdash U \uparrow 2}{\Gamma \vdash U \downarrow 2} \rightsquigarrow$$

$$\frac{\Gamma \vdash U \uparrow X}{\Gamma \vdash U \downarrow X}$$

$$\frac{}{\Gamma, x \uparrow A \vdash x \uparrow A}$$

$$\frac{}{\Gamma \vdash y, n \uparrow 2}$$

$$\frac{\Gamma \vdash u \uparrow A_1 \times A_2}{\Gamma \vdash u \cdot i \uparrow A_i} \quad (\text{rule 2})$$

$$\frac{\Gamma \vdash u \uparrow A_1 \rightarrow A_2 \quad \Gamma \vdash m \downarrow A_1}{\Gamma \vdash \alpha p(u, m) \uparrow A_2}$$

*(Ex Give B. only
normal form)*

$$\text{check } 0) \sum_i x \uparrow A_i \rightarrow A_2 \dots \forall x \downarrow A_1 \rightarrow A_2.$$

$$2) \quad \sum_i x \uparrow A_i \rightarrow A_2 \vdash \Delta_{A_1} (y \cdot \alpha p(x, y)) \downarrow A_2$$

$$\sum_i x \uparrow A_i \rightarrow A_2, \underline{y \uparrow A_1} \vdash x \uparrow A_1 \rightarrow A_2$$

$$y \downarrow A_1$$

$$y \uparrow A_1$$

Define " $[M/x]N$ " so that

$$\frac{\Gamma \vdash M \downarrow A \quad \Gamma, x \uparrow A \vdash N \downarrow B}{\Gamma \vdash [r^A/x]N \downarrow B}$$

$$[M/x]_A^{\exists} N = N^1 \quad \underline{[M/x]_A^u u = \begin{cases} N \\ u' \end{cases}}$$

$[\forall/x]_A^M N$, $[\forall/x]_A^u u$ are unproblematic!

Next: define, extend to LF.

Next²: semantic LF.