Harper 1.1

type theory from a computational perspective

- Martin-Löf : Constructive mathematics and computer programming
- Constable et. al. : NuPRL

Plan

1. develop type theory starting w/ computation
   → theory of truth (based on proof)
2. contrast w formalisms
   → theory of ~~pf~~ formal proof

R! a sufficiently expressive PL is a foundation for all of mathematics

Idea: Start w a programming language

- Deterministic operational semantics / transition system
- Assume: some idea of abstract syntax w/ binding and scope — subst for vars
- forms of expressions E
- two judgment forms E val means E is fully evaluated

  Consts: tt val, ff val, Bool val

- $E \mapsto E'$ means one step of simplification of E
- Derived notion of $E \Downarrow E_0$ means $E \mapsto^* E_0'$ and E_0' val

e.g.

$$\frac{E \mapsto E'}{\text{if}(E_1;E_2)(E) \mapsto \text{if}(E_1;E_2)(E')} \qquad \left\{ \begin{array}{l} \overline{\text{if}(E_1;E_2)(tt) \mapsto E_1} \\ \quad \text{''} \quad (ff) \mapsto E_2 \end{array} \right\}$$

TYPES ARE SPECIFICATIONS OF PROGRAM BEHAVIOR

(two principal forms of judgment (expressions of knowledge))

A type, $M \in A$:
1. Behavioral (not structural)
2. both M and A here are programs

e.g. Bool type, $tt \in$ Bool, $ff \in$ Bool "true by definition"

fact not axiom: if $M \in$ Bool and $M_1, M_2 \in A$ and A type then $\text{if}(M_1, M_2)(M) \in A$

$\underline{\underline{E}}$ $\text{if}(17; \text{Garbage})(ff) \in \text{Nat}$ runs by $\mapsto$ simplifies to $17 \in \text{Nat}$

$\underline{\underline{E}}$ $\text{if}(\text{Nat}; \text{Bool})(M)$ type when $M \in$ Bool b/c any outcome for M induces a simplification to a type.

Harper 1.d/ E $\text{if}(17; tt)(M) \in \text{if}(Nat; Bool)(M)$

R Specifications/types are programs

- Key idea: type-indexed families of types aka dependent types

e.g. $seq(n)$ type when $n \in Nat$

$n : nat \gg seq$ type

a family of types indexed by a type

→ ~~the~~ HYPOTHETICAL JUDGMENT / GENERAL JUDGMENT (Martin-Löf)

$f \in n : Nat \to Seq(\ )$ (NuPRL notation)

$(\Pi n : Nat.\ seq(n))$

critical idea: FUNCTIONALITY

families of types, of elements must respect equality of indices

E $seq(2+2)$ "same as" $seq(4)$

- $seq(\text{if}(17;18)(M))$ "same as" $\text{if}(seq(17); seq(18))(M)$

$a : Bool \gg \left(Seq(\text{if}(17;18)(a)) \text{ "same as" } \text{if}(seq(17); seq(18))(a) \right)$

---

revise our principle forms of judgment

Judgments

A type $\rightsquigarrow$ $A \doteq A'$ (Exact equality of types)

$M \in A$ $\rightsquigarrow$ $M \doteq M' \in A$ (exact equality of elements) "equisatisfaction"

at type

↑ dependent!

* "$\doteq$" is the "same as" from earlier

- E not $2 \doteq 4 \in Nat$

is $2 \doteq 4 \in Nat/\text{mod } 2$

exact equality of elements is a three part judgment, not a chain of two judgments

Harper 1.3 (

intention is that if $M \doteq M' \in A$ and $A \doteq A'$ then $M \doteq M' \in A'$

Meaning Explanations aka Semantics (computational)

there exists $A_0, \ldots$ Conjunctive

1. $A \doteq A'$ means $A \Downarrow A_0$ $A' \Downarrow A_0'$ $A_0 \doteq_0 A_0'$

A_0 and A_0' are equal type-values "equal canonical types" (Martin-Löf)

E by defn $\underset{val}{Bool} =_0 \underset{val}{Bool}$ i.e. Bool type

2. $M \doteq M' \in A$ means, where A type (i.e. $A \Downarrow A_0$ $A_0 \doteq_0 A_0$),

means $M \Downarrow M_0$ & $M' \Downarrow M_0'$ & $M_0 =_0 M_0' \in A_0$ (equal values in a type value)

3. $a{:}A \gg B \doteq B'$ means if $M \doteq M' \in A$ then $B[M/a] \doteq B'[M'/a]$ ("functionality")

Check $a{:}A \gg B$ type means $M' \doteq M' \in A$ implies $B[M/a] \doteq B[M'/a]$

4. $a{:}A \gg N \doteq N' \in B$ means if $M \doteq M' \in A$ then $N[M/a] \doteq N'[M'/a] \in B[M/a]$ $(\doteq B[M'/a])$

(assuming $a{:}A \gg B \doteq B$)

Booleans

1. $Bool \doteq_0 Bool$ i.e. $Bool$ $type_0$ i.e. Bool is a type

names

2. $M_0 \doteq M_0' \in_0 Bool$ is the strongest relation s.t. $tt \doteq_0 tt \in Bool$ (i.e. $tt \in_0 Bool$)

$ff \doteq_0 ff \in Bool$ (i.e. $ff \in_0 Bool$)

in other words

a. stated conditions hold
b. nothing else

strongest $R \subseteq Exp \times Exp$ s.t. $R(tt, tt)$, $R(ff, ff)$

R you "must" accept this as a valid definition

Scanned by CamScanner

Harper 1.4

Prop/Fact/Claim

- if $M \in Bool$ and A type and $M_1 \in A$ and $M_2 \in A$, then $if(M_1;M_2)(M) \in A$

Pf How to prove it? key: $\in Bool$ is given by a universal property – least containing
$tt \in Bool$
$ff \in Bool$

- Fix A type, $M_1 \in A$, $M_2 \in A$
  - if $M \in Bool$ then $if(M_1;M_2)(M) \in A$ / for $M \in Bool$ means $M \Downarrow M_0$ $M_0 = tt$ or $M_0 = ff$
- Sufficient to Show (STS): $if(M_1;M_2)(tt) \in A$
  " " " $(ff) \in A$ } cuz "if" evaluates its principle arg.

a. $if(M_1;M_2)(tt) \mapsto M_1 \in A$
b. $if(M_1;M_2)(ff) \mapsto M_2 \in A$

L "head expansion" aka "reverse execution"
if $M' \in A$ and $M \mapsto M'$ then $M \in A$

Ex prove it using the defns in terms of eval to canonical form

1. Bool is inductively defined
2. typing is closed under head expansion.

R this lemma is like type preservation in retrograde, yet behavioral whereas type preservation is static.

Harper 2.1

A type System consists of

$A \doteq A'$ (w/ A type iff $A \doteq A$)

$M \doteq M' \in A$ (w $M \in A$ iff $M \doteq M \in A$)

$\not\doteq$ is a partial equivalence.

$\doteq$ we omit reflexivity.

Symmetric

transitive

if $A \doteq A'$ and $M \doteq M' \in A$ then $M \doteq M' \in A'$

we assert the existence of certain type systems - defined in terms of evaluation

Hypotheticals express functionality

$a : A \gg B$ type means B is a family that depends functionally on $a : A$

$M \doteq M' \in A$ implies $B[M/a] \doteq B[M'/a]$

A type, $a : A \gg B$ type

a presupposition

$a : A \gg N \in B$ means B is a family of elements

$M \doteq M' \in A$ implies $N[M/a] \doteq N[M'/a] \in B[M/a] \doteq B[M'/a]$

Similarly for $B \doteq B'$, $N \doteq N' \in B$

there exists a type System containing Booleans

$Bool \doteq Bool$ i.e, Bool type

$M \doteq M' \in Bool$ iff $\begin{pmatrix} M \Downarrow true \\ \text{and } M' \Downarrow true \end{pmatrix}$ or $\begin{pmatrix} M \Downarrow false \\ \text{and } M' \Downarrow false \end{pmatrix}$

Fact if $a : Bool \gg B$ type and $M_1 \in B[true/a]$ and $M_2 \in B[false/a]$ and $M \in Bool$ then $if(M_1 ; M_2)(M) \in B[M/a]$

means you prove

Pf either $M \Downarrow true$ or $M \Downarrow false$

$\therefore M \doteq true \in Bool$ by head expansion

$\therefore M \doteq false \in Bool$ by head expansion

$$\frac{M_1 \in B[true/a]}{if(M_1 ; M_2)(M) \doteq if(M_1 ; M_2)(true) \doteq M_1 \in B[true/a] \doteq B[M/a]}$$

Harper 2.2

HW ① $if(M_1; M_2)(true) \doteq M_1 \in B[^{true}/_a]$

② $if(M_1; M_2)(false) \doteq M_2 \in B[^{false}/_a]$

③ $M \doteq if(true; false)(M) \in Bool$

RTS

if $a : Bool \gg P \in B$ then $P[^M/_a] \doteq if(P[^{true}/_a], P[^{false}/_a])(M)$ | "shannon expansion" (BDD's) — binary decision diagrams. "pivot on M"

E

$\exists$ type system containing natural numbers

$Nat \doteq Nat$ — gives us an induction principle.

$M \doteq M' \in Nat$ is strongest s.t.

either $M \Downarrow 0$, $M' \Downarrow 0$

or $M \Downarrow succ(n)$, $M' \Downarrow succ(n')$

w/ $n \doteq n' \in Nat$

Nat is given axiomatically

$\overline{0 \; Nat}$ $\qquad$ $\overline{succ(M) \; Nat}$

notice: M is anything, needn't be a Nat.

consider $fix(a, succ(a)) \mapsto \underbrace{succ(fix(a, succ(a)))}_{\omega} \mapsto$!!

COINDUCTION

ω, a point at infinity, inhabits the greatest solution to specification:

if $M \doteq M' \in Nat$ then $M \Downarrow 0, M' \Downarrow 0$ or

$M \Downarrow succ(N)$ $M' \Downarrow succ(N')$ $N \doteq N' \in CoNat$

Check HW $\omega \notin Nat$

if $M \mapsto M'$, $rec(M_0; a, b. M_1)(M)$

$\mapsto rec(M_0; a, b. M_1)(M')$

$R(0) \mapsto M_0$

$R(succ(M)) \mapsto M_1[M, R(M)/a, b]$

predicate (checking if base case) — result of recursive call.

Fact suppose $a : Nat \gg B\,type$ $\quad M_0 \in B[^0/_a]$ $\quad a : Nat, b : B \gg M_1 \in B[^{succ(a)}/_a]$

then $M \in Nat$ then $R(M) \in B[^M/_a]$

Pf ① $M \Downarrow 0$ $\quad M \doteq 0 \in Nat$ $\quad M_0 \in B[^0/_a] \doteq B[^M/_a]$

$R(M) \doteq R(0) \doteq M_0$

i.e. $R(M) \in B[^M/_a]$

② $M \Downarrow succ(N)$ $\quad$ IH: $R(N) \in B[^N/_a]$

$\longrightarrow$ HW finish proof, a lot like conditionals

Scanned by CamScanner

Harper 2.3

R type systems are about specifying protocols to be obeyed

Products

$A_1 \times A_2 \doteq A_1' \times A_2'$ iff $A_1 \doteq A_1'$ $A_2 \doteq A_2'$

$M \doteq M' \in A_1 \times A_2$ iff $M \Downarrow \langle M_1, M_2 \rangle$ $M_1 \doteq M_1' \in A_1$
$M' \Downarrow \langle M_1', M_2' \rangle$ $M_2 \doteq M_2' \in A_2$

Fact ← not dfn
Suppose A_1 type A_2 type
if $M \in A_1 \times A_2$ then $M \cdot 1 \in A_1$ and $M \cdot 2 \in A_2$

(where $\dfrac{M \mapsto M'}{M \cdot i \mapsto M' \cdot i}$ $\dfrac{}{\langle M_1, M_2 \rangle \cdot 1 \mapsto M_1}\langle i = 1,2 \rangle$)

we know $M \Downarrow \langle M_1, M_2 \rangle$ $M \cdot 1 \mapsto^* \langle M_1, M_2 \rangle \cdot 1 \mapsto M_1 \in A_1$
with $M_1 \in A_1$

Fact (A_1, A_2 types)
if $M_1 \in A_1$ then $\langle M_1, M_2 \rangle \cdot 1 \doteq M_1 \in A_1$
no requirement on M_2
$\langle M_1, M_2 \rangle \cdot 1$

$\dfrac{}{\lambda a.M \text{ val}}$

$\dfrac{M \mapsto M'}{ap(M, M_1) \mapsto ap(M', M_1)}$

$\dfrac{}{ap(\lambda a.M_2, M_1) \mapsto M_2[M_1/a]}$

Functions

$A_1 \to A_2 \doteq A_1' \to A_2'$ iff $A_1 \doteq A_1'$ and $A_2 \doteq A_2'$

$M \doteq M' \in A_1 \to A_2$ iff $M \Downarrow \lambda a.M_2$ $M' \Downarrow \lambda a.M_2'$ $a : A_1 \gg M_2 \doteq M_2' \in A_2$

Fact if $M \in A_1 \to A_2$ and $M_1 \in A_1$ then $ap(M, M_1) \in A_2$

Pf (HW)

Fact if $M, M' \in A_1 \to A_2$ and $a : A_1 \gg ap(M, a) \doteq ap(M', a) \in A_2$ then $M \doteq M' \in A_1 \to A_2$ "function extensionality"

Pf (HW)

this fact is verified by the rule

$$\frac{\Gamma \vdash M : A_1 \to A_2 \quad \Gamma \vdash M_1 : A_1}{\Gamma \vdash ap(M, M_1) : A_2}$$

the rule is "protocol" or "syntax"

Q what is the quantifier complexity of $M \doteq M' \in Nat \to Nat$?

"$\forall M_1 \doteq M_1' \in Nat$
$\exists P_1 \doteq P_1' \in Nat$
$ap(M, M_1) \doteq ap(M', M_1') \in Nat \to N$
where $ap(M, M_1) \Downarrow P$
$ap(M', M_1') \Downarrow P'$

answer $\forall\exists$ or Π_2^0

Harper 2.4

Dependent products

$a : A_1 \times A_2 \doteq a : A_1' \times A_2'$

iff $A_1 \doteq A_1'$ $a : A_1 \gg A_2 \doteq A_2'$

A_2 is a family

$M \doteq M' \in a : A_1 \times A_2$ iff

$M \Downarrow \langle M_1 ; M_2 \rangle$ $M' \Downarrow \langle M_1' ; M_2' \rangle$ $M_1 \doteq M_1' \in A_1$

$M_2 \doteq M_2' \in A_2[M_1/a] \doteq A_2[M_1'/a]$

Dependent Functions

$a_1 : A_1 \to A_2 \doteq a : A_1' \to A_2'$ iff $A_1 \doteq A_1'$

$a : A_1 \gg A_2 \doteq A_2'$

$M \doteq M' \in a : A_1 \to A_2$ iff $M \Downarrow \lambda a. M_2$ $M' \Downarrow \lambda a_1 M_2'$

$a : A_1 \gg M_2 \doteq M_2' \in A_2(a)$

if $M_1' \doteq M_1' \in A_1$ then $M_2[M_1/a] \doteq M_2[M_1'/a] \in A_2[M_1/a]$

$\doteq A_2[M_1'/a]$

Fact ① if $M \in a : A_1 \times A_2$ then

$fst(M) \in A_1$ and $snd(M) \in A_2[fst(M)/a]$

② if $M \in a : A_1 \to A_2$ and $M_1 \in A_1$ then $ap(M, M_1) \in A_2[M_1/a]$

$(\Sigma a : A_1, A_2)$ $(\Pi_a : A_1, A_2)$

Pf (HW)

Bool, Nat, $a_1 : A \times A_2$ $a : A_1 \to A_2$ are inherently computational

Harper 3.1

So far we're developing computational type theory
types are behavioral specifications,

(exact) type equality $A \doteq A'$ (A type means $A \doteq A$)
(exact) member equality $M \doteq M' \in A$ ($M \in A$ means $M \doteq M \in A$) Syntax: $M \doteq_A M$

$\longrightarrow$ expressions do not intrinsically "have" a type, the same program can satisfy many specs.

Definitions of types

Bool
Nat } inductive types

1 aka unit exercise $\langle\rangle$ val ("void" in C-like languages)

$A_1 \times A_2$ products

$A_1 \to A_2$ functions

0 aka void $\mathrm{case}\{\}(M)$

$A_1 + A_2$ $1 \cdot M$, $2 \cdot M$, $\mathrm{case}\{a_1 . M_1 ; a_2 . M_2\}(M)$

$a{:}A_1 \times A_2$ dependent product aka Σ-type $\Sigma a{:}A_1, A_2$: to defn $a{:}A_1 \times A_2$, must already have A_1 defnd
$A_2[{}^{M}/a]$ for every $M_1 \in A_1$

$a{:}A_1 \to A_2$ dependent function aka Π-type $\Pi a{:}A_1, A_2$

A_1
$A_2[M_1]$ $A_2[M_2]$ ...

a type theory is a theory of computation (program specification)

Brouwer: as a means to give a notion of truth for logical propositions. "PROPOSITIONS AS TYPES" "SEMANTIC CORRESPONDENCE"

logic	types	
$\top^*$	unit, 1	
$\bot^*$	void, 0	
$(\varphi_1 \wedge \varphi_2)^*$	$\varphi_1^* \times \varphi_2^*$	
$(\varphi_1 \vee \varphi_2)^*$	$\varphi_1^* + \varphi_2^*$	(NB) } constructive
$(\varphi_1 \supset \varphi_2)^*$	$\varphi_1^* \to \varphi_2^*$	impl/arrow
$(\forall a{:}A . \varphi_a)^*$	$a{:}A \to \varphi_a^*$	dependent arrow
$(\exists a{:}A . \varphi_a)^*$	$a{:}A \times \varphi_a^*$	(NB) } constructive existence
$(M =_A M)^*$	next page	

what does this mean? { φ prop $\longmapsto$ specification/problem $\longmapsto$ φ^* type
φ true $\longmapsto$ solution/satisfier/realizer $\longmapsto$ φ^* inhabited, i.e., there is $M \in \varphi^*$ and $M \doteq M' \in \varphi^*$

the notion of type is primary and more extensive than "mere" logic.

"construction" means a program inhabits a type (Brouwer)

fact, ~~axiom~~ of choice

logic { $\forall a{:}A\ \exists b{:}B\ R(a,b)$ "R is total"
$\supset$ $\exists f{:}A \to B\ \forall a{:}A\ R(a, f(a))$ "choice function f" (the function f is a witness)

a fact
tt { $F ::$ $(a{:}A \to (b{:}B \times R(a,b))) \to (f{:}A \to B \times (a{:}A \to R(a, f(a))))$

$f \triangleq \lambda a. (F(a) \cdot 1)$

Harper 3.2

How to interpret equality as a type?

<u>Equality types</u>

$$Eq_A(M_1, M_2) \doteq Eq_{A'}(M_1', M_2') \quad \text{iff} \quad A \doteq A' \quad M_1 \doteq M_1' \in A \quad M_2 \doteq M_2' \in A$$

$$M \in Eq_A(M_1, M_2) \quad \text{iff} \quad M \Downarrow * \quad \text{and} \quad M_1 \doteq M_2 \in A$$ "subsingleton"

<u>E</u> $Eq_{Nat}(2,3)$ uninhabited

$Eq_{Nat}(2, 1+1)$ inhabited

<u>R</u> an equation is at most true

$*$ is the trivial inhabitant

$$\exists M \in Eq_A(M_1, M_2) \quad \text{iff} \quad M_1 \doteq M_2 \in A \quad \text{where} \quad M \Downarrow *$$

$$(M_1 =_A M_2)^* = Eq_A(M_1, M_2)$$ "works"

<u>HW</u> axiomatize inductively that $Eq_A(-,-)$ is the least reflexive relation on A

① $* \in Eq_A(M, M)$ whenever $M \in A$

② TS: $Eq_A(M,N) \longrightarrow R(M,N)$ "equality induction"

STS: R is reflexive $a : A \gg \_ \in R(a,b)$

Formalisms – formal type theory is inductively defined by rules for deriving

$\Gamma \vdash A \text{ type}$ $\quad$ $\Gamma \vdash A \equiv A$ $\quad$ "definitional equality"

$\Gamma \vdash M : A$ $\quad$ $\Gamma \vdash M \equiv M' : A$ $\quad$ " "

$$\frac{}{\Gamma, x : A, \Gamma' \vdash x : A}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash A \equiv A'}{\Gamma \vdash M : A'}$$

$$\frac{\Gamma \vdash A_1 \text{ type} \quad \Gamma \vdash A_2 \text{ type}}{\Gamma \vdash A_1 \times A_2 \text{ type}}$$

$$\frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2 \rangle \; A_1 \times A_2}$$

$$\frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash M \cdot i : A_i} \langle i = 1,2 \rangle$$

etc.

✓ have it

$$\frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2 \rangle \cdot k \equiv M_k : A_k}$$

?, might have it

$$\frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \langle M \cdot 1, M \cdot 2 \rangle \equiv M : A_1 \times A_2}$$

a design requirement: all judgments ought to be <u>decidable</u>

- type checking
- def'n'l equivalence (for <u>some</u> choice of def'n eq)

Structural properties of entailment

derivation

composition of rules

$\Gamma \vdash A \equiv A'$

Formalism is a useful approximation to the truth.

* DERIVATIONS ARE INHERENTLY COMPUTABLY ENUMERABLE
* Via formal correspondence to formal logics, type-checking and derivation checking is the same thing.

Harper 3.3

$\underline{\underline{R}}$ formal logic has no opinion about two props being equal.

How to ~~axiomatize~~ formalize equality?

$\underline{\text{first cut}}$ "equality reflection"

$$\frac{\Gamma \vdash M : Eq_A(M_1, M_2)}{\Gamma \vdash M_1 \equiv M_2 : A} \ (ER)$$ is problematic , $$\frac{\Gamma \vdash M_1 \equiv M_2 : A}{\Gamma \vdash \text{"}refl_A(M_1)\text{"} : Eq_A(M_1, M_2)}$$ } less common in proof assistants.

you need "proof search", which threatens/compromises decidability.

ETT := extensional type theory

Harper 4.1

$\underline{\underline{E}}$ last time

Spse $a, b : A$, $c : Eq_A(a,b) \gg C$ type

if $a : A \gg P \in C[a, a, * / a, b, c]$

and $M \in Eq_A(M_1, M_2)$

then $P[M_1/a] \in C[M_1, M_2, M / a, b, c]$

} equality is the least reflexive relation.

$\underline{\underline{Pf}}$ $M \doteq * \in Eq_A(M_1, M_2)$ and $M_1 \doteq M_2 \in A$

$$C[M_1, M_2, M] \doteq C[M_1, M_2, *]$$
$$\ni$$
$$P[M_1]$$

$\underline{\underline{??}}$

$\underline{\text{Fact}}$ the only element of $Eq_A(M,N)$ is $*$ if $M \in Eq_A(M_1, M_2)$ then $M \doteq *$, $* \in Eq_{Eq_A(M_1,M_2)}(M, A)$

"uniqueness of equality evidence."

Scanned by CamScanner

Harper 4/2

Examining formalisms ("abstract implementations") for type theory

- inductively def'nd by rules for deriving judgments of the ff forms

$\Gamma \vdash M : A$ $\qquad \Gamma \vdash M \equiv M' : A$

$\Gamma \vdash A$ type $\qquad \Gamma \vdash A \equiv A'$

"def'nl equivalence" — "calculational equivalence"

↑ beta-like rules

$$\frac{\Gamma, a{:}A_1 \vdash M : A_2}{\Gamma \vdash \lambda a{:}A_1 . M : A_1 \to A_2}$$

$$\frac{\Gamma, a{:}A_1 \vdash M_1 : A_2 \qquad \Gamma \vdash M_2 : A_1}{\Gamma \vdash (\lambda a{:}A_1 . M_1)(M_2) \equiv M_1[M_2/a] : A_2[M_2/a]} \leftarrow \text{"beta rule"}$$

$$\frac{\Gamma \vdash M_1 : a{:}A_2 \to A \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash M_1 M_2 : A[M_2/a]}$$

Criteria

1. decidability (type checking)
2. computational adequacy

Idea: formalism is "just" a means of deriving truths about programs

① Define erasure of formal terms, types $|M|$, $|A|$ $\qquad |\lambda a{:}A . M| = \lambda a . |M|$

(more or less trivial)

② Soundness (w.r.t. erasure)

if $\Gamma \vdash M : A$ then $|\Gamma| \gg |M| : |A|$

$\Gamma \vdash M_1 \equiv M_2 : A$ $\qquad |\Gamma| \gg |M_1| \doteq |M_2| \in |A|$

$\Gamma \vdash A$ type

$\Gamma \vdash A_1 \equiv A_2$

fundamental theorem of logical relations

$\subseteq$ if $M : Bool$ then $|M| \Downarrow true$ or $|M| \Downarrow false$

closed, i.e., the gamma is missing.

$\subseteq$ Consistency

Scanned by CamScanner

Harper 4.3

Soundness tells us that proofs have a computational content, you can do program extraction

- we'd like to internalize computation as defnl equiv.

Canonicity: if $M : Bool$, then $M \equiv true : Bool$ or $M \equiv false : Bool$

(separate them: Martin Löf 1973) "internal completeness"

How to formalize equality?

E: $a : Nat \gg a + a \doteq 2 \times a \in Nat$

$\lambda a. a + a = \lambda a\, 2 \times a \in Nat \to Nat$

- ① ETT - equality, reflection, uniqueness

expressive, not decidable

ITT (Martin Löf): formalize identity as least reflexive relation, id_A

$$\frac{\Gamma \vdash A\ type \quad \Gamma \vdash M_1 : A \quad \Gamma \vdash M_2 : A}{\Gamma \vdash Id_A(M_1, M_2)\ type}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash refl_A(M) : Id_A(M, M)}$$

$$\frac{\Gamma, a, b : A, c : Id_A(a,b) \vdash C\ type \quad \Gamma \vdash P : Id_A(M,N) \quad \Gamma, a : A \vdash Q : C[a, a, refl_A(m)]}{\Gamma \vdash J(a,b,c.C)(a.Q)(P) : C[M,N,P]}$$

$$J(a,b,c.C)(a.Q)(refl_A(m)) \equiv Q[M/a]$$

Soundness

Idea: $|Id_A(M,N)| \triangleq Eq_{|A|}(|M|,|N|)$

$|refl_A(M)| \triangleq *$

$|J \ldots| \triangleq |Q|[|M|/a]$

in Martin-Löf, you can find a proof

$a : Nat \vdash \underline{\ \checkmark\ } : Id_{Nat}(a+a, 2 \times a)$

but Not $Id_{Nat \to Nat}(\lambda a. a+a, \lambda a\, 2 \times a)$

because as functions, $\lambda a. a+a \not\equiv \lambda a. 2 \times a$

failure of function extensionality

Harper 4.4

what to do?

A common approach is to add an axiom

Function Extensionality $\mathrm{FunExt}: (a:A_1 \to \mathrm{Id}_{A_2}(ap(M,a), ap(N,a))) \to \mathrm{Id}_{a:A_1\to A_2}(M,N)$

$\mathrm{FunExt}(H) : \mathrm{Id}_{a:A_1\to A_2}(M,N)$

H premise

Strength in Weakness

ITT cannot prove (internally) that there is only one identification "groupoid model"

ok mathematically, but it ruins canonicity

$J(a,b,c.C)(a.Q)(\mathrm{FunExt}(H)) \equiv ???$

doesn't make sense

YET $|(\mathrm{FE}(H))| \doteq *$

R: OTT, a proof theory by McBride

Idea along similar lines called UNIVALENCE

— make up for a deficiency. there aren't enough equality proofs

motivation: it is common to informally "identify structures up to isomorphism"

e.g. $B\times A \cong A\times B$ by swap

add an axiom $\mathrm{UA}(\text{"swap"}) : \mathrm{Id}_{\mathcal{U}}(A\times B, B\times A)$

univalence axiom

"universe" type of types

model using "simplicial sets"

for specific choices of A, B there can be many such isomorphisms.

① Id is a mechanism for interchangability

② Stipulate that equivalent types are identical.

---

① $J\_\_\ (\mathrm{FunExt}(E)) \equiv ?$ ruins canonicity

② $J\_\_\ (\mathrm{UA}(E)) \equiv ?$

case 1, has a computational interpretation

case 2, ??? what would be computational interpretation of univalence?

arper 5.1

recall $Id_A(M_1, M_2)$ identity/identification

(generic in A) $refl_A(M)$

$J(a,b,c.\,C)\,(a.\,P)\,(Q)$ $[M_1, M_2, Q / a,b,c]$

$Q : Id_A(M_1, M_2)$

aitin Lof 1972 if $- Id_A(M,N)$ then $M = N\ A$

1. manually adding function extensionality
2. adding $UA(E) : Id_{\mathcal{U}}(A,B)$ when $E . Equal(A,B)$ ← HoTT formalism
3. adding induction w/ identification

D $tr[a.F]$ transport of an a-indexed family of types

$$tr[a.F](P : id_A(M,N)) : F[M/a] \to F[N/a] \triangleq J(a,b,\ldots,\underbrace{F(a) \to F(b)}_{c(a,b)})(a.\ id_{F[a]})(P)$$

Problem: consider the "identity family" a.a

$tr[a.a]\,(P : id_{\mathcal{U}}[A,B]) : A \longrightarrow B$

~~this is a coercion function~~

cf. $\dfrac{M : A \quad A \equiv B}{M : B}$ $\dfrac{M \in A \quad A \doteq B}{M \in B}$

$\dfrac{P : id_{\mathcal{U}}(A,B) \quad M : A}{tr(M) : B}$

ea: identification as "proof relevant equality"

$tr[a.a](ua[E])$ gets stuck

where E would be $Equiv(A,B)$, $f : A \to B$, ...

Harper 5,2

Martin Löf ninja move (MLNM) to recover computation from HoTT

- adding axioms to a type theory is suspicious, where axioms are elements of a type

1. Gentzen vs. Hilbert
   ↑
   entailment is prior to implication $\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}$ (logical consequence)

2. Eilenberg & MacLane : maps are prior to functions $\frac{f : C \times A \longrightarrow B}{\ulcorner f \urcorner : C \longrightarrow B^A}$

   where $\ulcorner f \urcorner$ is the homset

here, bring out $\underbrace{\text{judgmental structure of identifications}}_{\text{paths}}$, then you can internalize them

→ Id as an inductive type (generic in A)

→ path A internalizes paths (depends on A)

we have to split these up to get computation

MLNM Judgment is prior to inhabitants of type.

paths between types and within types

N coe : coercion

Where do we get paths between types

1. induced by paths between elements of index types

   $a : A \vdash F$ type

   path within A will be preserved by F

   $\underbrace{M \xrightarrow{P} N}_{\text{within } A}$ induces $\underbrace{F[M] \xrightarrow{F[P]} F[N]}_{\text{multiverse of all types}}$

2. univalence where swap is a path

   $Bool \times Nat \xrightarrow{swap} Nat \times Bool$

   - $swap \triangleq ua(\text{swap is equivalence})$
   - paths induce coercion

     $coe(swap) : Bool \times Nat \longrightarrow Nat \times Bool$

$((Bool \times Nat) \to C) \xrightarrow{swap \to C} ((Nat \times Bool) \to C)$

- swap is a type
- $coe(swap \to C)$

Harper 5.3

$\text{Bool} \times \text{Nat} \xrightarrow{\text{"swap"}} \text{Nat} \times \text{Bool}$ expresses
① swap is a type
② its left endpoint is Bool × Nat
its right endpoint is Nat × Bool

R Paths are traced out by dim variables that "range over unit interval [0,1]"

$\text{swap}_x \ \text{type} \ [x]$ (subst. gets you endpoints)

$\text{swap}_x\langle 0/x \rangle \doteq \text{Bool} \times \text{Nat}$

$\text{swap}_x\langle 1/x \rangle = \text{Nat} \times \text{Bool}$

$\text{swap}_0 \xrightarrow{\text{swap}_x} \text{swap}_1$ where $\text{swap}_0 \mapsto \text{Bool} \times \text{Nat}$ and $\text{swap}_1 \mapsto \text{Nat} \times \text{Bool}$

for $x \in [0,1] \subseteq \mathbb{R}$, "a line of types" $\text{swapEl}_x(N)$ is a line of elements,

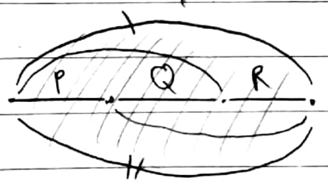
what are the elements of swap_x? "heterogenous lines" $\text{swapEl}_x(N) \in \text{swap}_x \ [x]$ if $N \in A_2 \times A_1 \ [x]$

$(\text{swapEl}_x(N))\langle 0/x \rangle \doteq \text{swapEl}_0(N\langle 0/x \rangle)$ where $\text{swapEl}_0(N) \mapsto ap(\text{swapFn}, N)$ and $\text{swapEl}_1(N) \mapsto N$

where there is some path $N_0 —— N_1$ for $N_0 = ap(\text{swapfn}(N_0))$

General Setup

① lines induce coercions
② must be able to compose lines.



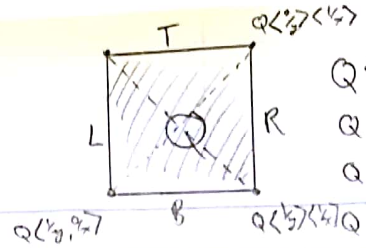
I and II need to be equal/equivalent.
meaning I can be continuously deformed into II

D Kan Condition: a program for performing composition that gives us all of above structure

Scanned by CamScanner

5.4 Harper

Cartesian Cube Setup



$Q\langle {}^0/_x\rangle \doteq L$

$Q\langle {}^1/_x\rangle \doteq R$

$Q\langle {}^0/_y\rangle \doteq T$

$Q\langle {}^1/_y\rangle\langle {}^1/_x\rangle\ Q\langle {}^1/_y\rangle \doteq B$

A type $[x_1, \ldots, x_n]$

$M \in A$ $[x_1, \ldots, x_n]$

$x \not\approx 0$

Subject to COHERENCE requirements

- to be cartesian we need to have DIAGONAL from $Q\langle {}^0/_y\rangle\langle {}^1/_x\rangle$ to $Q\langle {}^1/_y\rangle\langle {}^0/_x\rangle$

---

swap case

$\mathrm{Coe}^{0 \rightsquigarrow 1}_{x.\mathrm{swap}_x} \in \mathrm{Swap}_x\langle {}^0/_x\rangle \longrightarrow \mathrm{Swap}_x\langle {}^1/_x\rangle$

general arrow case

$\mathrm{Coe}^{0 \rightsquigarrow 1}_{x.(A_1 \to A_2)}\big(M \in A_1\langle {}^0/_x\rangle \longrightarrow A_2\langle {}^0/_x\rangle\big) \in A_1\langle {}^1/_x\rangle \to A_2\langle {}^1/_x\rangle$

$\longmapsto \lambda a_1 \in A_1\langle {}^1/_x\rangle.\ \mathrm{Coe}^{0 \rightsquigarrow 1}_{x.A_2}\Big(M\big(\mathrm{Coe}^{1 \rightsquigarrow 0}_{x.A_1}(a_1)\big)\Big)$

Coercions := $\{0 \rightsquigarrow 1,\ 1 \rightsquigarrow 0,\ 0 \rightsquigarrow x,\ x \rightsquigarrow 0,\ 1 \rightsquigarrow x,\ x \rightsquigarrow 1,\ x \rightsquigarrow y\}$

$\mathrm{Coe}^{0 \rightsquigarrow 0}_{y.B}$ —— $\mathrm{Coe}^{0 \rightsquigarrow x}_{y.B}(M)$ —— $\mathrm{Coe}^{0 \rightsquigarrow 1}_{y.\mathrm{Swap}_x}(M)$

where $\mathrm{Coe}^{0 \rightsquigarrow 0}_{y.B} \doteq M$ evals to something derived from M itself (secret to why swapEl_x ide…

R For dependent types, need $\mathrm{Coe}^{1 \rightsquigarrow x}_{A_1}(-)$