Computational Type Theory

Bob Harper
Carnegie Mellon University

July 18, 2018

1 Recap

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 $M \doteq M \in A$)
Expressions do tnot intrinsically have a type; the same program can satisfy many spec's.
$M \in \texttt{Nat} \to \texttt{Nat}$
$M \in \texttt{Primes} \to \texttt{Primes}$
$\lambda a.a \doteq \lambda a.|a| \in \texttt{Nat} \to \texttt{Nat}$
$\lambda a.a \neq \lambda a.|a| \in \texttt{Int} \to \texttt{Int}$

1.1 Definitions of types

`Bool` - Inductive type
`Nat` - Inductive type
$A_1 \times A_2$ - Products
$A_1 \to A_2$ - Functions
0 a.k.a. `Void` - $\texttt{case}\{\}(M)$
$A_1 + A_2$ - $1 \cdot M, 2 \cdot M, \texttt{case}\{a_1.M_1; a_2.M_2\}(M)$
$a : A_1 \times A_2$ - Dependent product - Σ type
$a : A_1 \to A_2$ - Dependent function - Π type
Note: To define $a : A_1 \times A_2$ or $a : A_1 \to A_2$ first need to determine A_1 then can determine A_2.

1.2 Types

`Bool - true, false`
`Nat - 0,` $\texttt{succ}(M)$
$\texttt{Bool}\times\texttt{Nat}$
$\texttt{Bool} \to \texttt{Nat}$
$(\texttt{Nat} \to \texttt{Nat}) \to \texttt{Nat}$

2 Type Theory: A Theory of Computation (Computer Specification)

Brouwer as a means to give a notion of truth for logical propositions: "Propositions as Types" or "Semantic Correspondence"
Type is primary and more extensive than mere logic.
$\top^*$ - 1 a.k.a. Unit
$\bot^*$ - 0 a.k.a. Void
$(\varphi_1 \wedge \varphi_2)^*$ - $\varphi_1^* \times \varphi_2^*$
$(\varphi_1 \vee \varphi_2)^*$ - $\varphi_1^* + \varphi_2^*$ (NB)
$(\varphi_1 \supset \varphi_2)^*$ - $\varphi_1^* \to \varphi_2^*$
$(\forall x : A.\varphi_a)^*$ - $a : A \to \varphi_a^*$
$(\exists a : A.\varphi_a)^*$ - $a : A \times \varphi_a^*$ (NB)

$$(\forall a : A.\exists b : B.R(a, b)) \supset (\exists f : A \to B.\forall a : A.R(a, f(a))) \text{ true}$$

$$(a : A \to (b : B \times R(a, b))) \to (f.A \to B \times (a : A \to R(a, f(a)))$$

$$f \stackrel{\Delta}{=} \lambda a.(F(a) \cdot 1)$$

2.1 How to interpret equality as a type?

2.1.1 Equality Types

$\mathtt{Eq}_A(M_1, M_2) \doteq \mathtt{Eq}_{A'}(M_1', M_2') \iff A \doteq A'; M_1 \doteq M_1' \in A; M_2 \doteq M_2' \in A$
$M \in \mathtt{Eq}_A(M_1, M_2) \iff M \Downarrow \star$ and $M_1 \doteq M_2 \in A$
$(M_1 =_A M_2)^* = \mathtt{Eq}_A(M_1, M_2)$

<u>Exercise</u>: Show that $\mathtt{Eq}_A(-, -)$ is the least reflexive relation on A

1. $\star \in \mathtt{Eq}_A(M, M)$ whenever $M \in A$
2. TS: $\mathtt{Eq}_A(M, N) \to R(M, N)$
   STS: R is reflexive, $a : A >> \_ \in R(a, a)$

2.2 Formalisms

formal type theory is inductively defined by rules for deriving

$\Gamma \vdash A$ `type` $\quad \Gamma \vdash A \equiv A'$

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

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

$$\frac{\Gamma \vdash A_1 \ \mathtt{type} \ \Gamma \vdash A_2 \ \mathtt{type}}{\Gamma \vdash A_1 \times A_2 \ \mathtt{type}}$$

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

$$\frac{\Gamma \vdash M_1 : A_1 \; \Gamma \vdash M_2 : A_2}{\Gamma \vdash < M_1; M_2 >: A_1 \times A_2}$$

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

$$\frac{\Gamma \vdash M_1 : A_1 \; \Gamma \vdash M_2 : A_2}{\Gamma \vdash < M_1; M_2 > \cdot i \equiv M_i : A_i} \checkmark$$

$$\frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash < M \cdot 1; M \cdot 2 = M : A_1 \times A_2} ?$$

2.2.1 Design Requirements

All judgements should be decidable

- type checking
- defined equivalence (for some choice of def eq)

Structural properties of entailment
Formalism is a useful approximation to the truth.

2.3 How to axiomatize/formalize equality?

2.3.1 First Cut: equality reflection

$$\frac{\Gamma \vdash M : \mathtt{Eq}_A(M_1, M_2)}{\Gamma \vdash M_1 \equiv M_2 : A} (\mathrm{ER})$$

Threatens/compromises decidability

$$\frac{\Gamma \vdash M_1 \equiv M_2 : A}{\Gamma \vdash \mathtt{refl}_A(M_1) \mathtt{Eq}_A(M_1, M_2)}$$