Type Theory from a Computational Perspective
Lecture 1

Bob Harper
Carnegie Mellon University

July 16, 2018

1 Plan

1. Develop type theory starting with computation
   → Theory of TRUTH (based on proof)

2. Contrast with formalisms
   → Theory of PROOF (formal proof)

2 The Idea

Start with a programming language
Deterministic operational semantics
Assume: Some idea of abstract syntax with bindings and scope (ie: sub for vars)
Forms of expression `E`
Two judgement forms:

1. `E val` means `E` is fully evaluated

2. `E` $\mapsto$ `E'` means one step of simplification of `E`

Derived notion: $E \Downarrow E_0$ or V means $E \mapsto^* E_0$ `val`

$$\frac{E \mapsto E'}{if(E_1, E_2)(E) \mapsto if(E_1, E_2)(E')} \tag{1}$$

$$\frac{}{if(E_1, E_2)(\mathrm{tt}) \mapsto E_1} \tag{2}$$

$$\frac{}{if(E_1, E_2)(\mathrm{ff}) \mapsto E_2} \tag{3}$$

TYPES ARE SPECIFICATIONS OF BEHAVIOR!
Two principal forms of judgement (expression of knowledge)

A type and M $\in$ A

1. behavioral (not structural)
2. both M and A here are programs

i.e.: Bool type; tt $\in$ Bool; ff $\in$ Bool
Example:
if M $\in$ Bool and M_1,M_2 $\in$ A with A type
then if(M_1; M_2)(M) $\in$ A
Example:
if(17; (GARBAGE))(tt) $\in$ Nat, runs by simplifying to 17 $\in$ Nat
Example:
if(Nat; Bool)(M) type when M $\in$ Bool b/c any outcome for M induces a simplification to a type
Example:
if(17; tt)(M) $\in$ if(Nat, Bool)(M) !
SPECS/TYPES are programs

3 Key Idea: (Type-Indexed) Families of Types

a.k.a.: Dependent types
Example:
seq(n) type when n $\in$ Nat
n: Nat >> seq(n) type $\rightarrow$ Hypothetical/General Judgement
Family of typese indexed by a type.

But how to express a function that takes a natural number and returns a sequence of that length?

```
f ∈ n: Nat → Seq(n)
(Πn:Nat.Seq(n))
```

4 Critical Idea: <u>FUCTIONALITY</u>

Families (of types, of elements) must respect equality of indices.
Example:

```
seq(2+2) "same as" seq(4)
seq(if(17; 18)(M)) "same as" if(seq(17); seq(18))(M)
```

4.1 Judgements

$$A \text{ type} \longrightarrow A \doteq A' \text{ (exact equality of types (?))}$$
$$M \in A \longrightarrow M \doteq M' \in A \text{ (exact equality of elements (”equi-satisfaction”))}$$

Example:

$$\text{not: } 2 \doteq 4 \in Nat$$
$$\text{is: } 2 \doteq 4 \in Nat/2 \text{ (evens)}$$

Intention is thet if M $\doteq$ M' $\in$ A and A $\doteq$ A' then M $\doteq$ M' $\in$ A'

5 Meaning Explanations a.k.a: Semantics (Computational)

1. $A \doteq A'$ means $A \Downarrow A_0$ val, $A' \Downarrow A_0'$ val, A_0 val $\doteq_0$ A_0' val
   A_0 and A_0' are equal type values
   Now, by definition:
   Bool val $\doteq_0$ Bool val (i.e.: Bool type_0)

2. $M \doteq M' \in A$ where A type (i.e.: $A \Downarrow A_0$, $A_0 \doteq_0 A_0$)
   means $M \Downarrow M_0$ and $M' \Downarrow M_0'$ and $M_0 \doteq_0 M_0' \in A$
   Equal values in a type value

3. $a : A >> B \doteq B'$ means
   if $M \doteq M' \in A$ then $B[M/a] = B'[M'/a]$ "functionality"

4. $a : A >> N \doteq N' \in B$ means
   if $M \doteq M' \in A$ then $N[M/a] \doteq N'[M'/a] \in B[M/a] \doteq B[M'/a]$

5.1 Booleans

1. Bool $\doteq_0$ Bool or Bool type_0 or Bool is a type (names a type)
   Aside: Not going to say 17 type, could, but wont.

2. $M_0 \doteq M_0' \in_0$ Bool is the strongest (least,...) 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)

   (a) the stated conditions hold

   (b) nothing else!

   strongest: R $\subseteq$ Exp $\times$ Exp
   s.t. R(tt,tt) and R(ff,ff)
   You must accept this as a valid definition.

Prop/Fact/Claim:
If $M \in$ Bool and A type and $M_1 \in A$ and $M_2 \in A$,
then $\text{if}(M_1, M_2)(M) \in A$
Proof:
How to prove it? Key: Bool is least containing tt and ff
Fix A type, $M_1 \in A$, $M_2 \in A$

if $M \in \text{Bool}$ then $\text{if}(M_1, M_2)(M) \in A$
$M \in \text{Bool}$ means $M \Downarrow M_0$ s.t. $M_0 = \text{tt}$ or ff
Suffices to show both:

1. $\text{if}(M_1, M_2)(\text{tt}) \in A$
2. $\text{if}(M_1, M_2)(\text{ff}) \in A$

1. $\text{if}(M_1, M_2)(\text{tt}) \mapsto M_1 \in A$
2. $\text{if}(M_1, M_2)(\text{ff}) \mapsto M_2 \in A$

Lemma (Head Expansion or Reverse Execution):
If $M' \in A$ and $M \mapsto M'$ then $M \in A$
Can be proved using the definitions in terms of evaluation to canonical form

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