Cubical Computational Type Theory & RedPRL

Carlo Angiuli
Evan Cavallo
(*) Favonia
Bob Harper
Dan Licata
Jon Sterling
Todd Wilson

>> redprl.org >>

2018.02.07 Cornell
Vladimir Voevodsky 1966–2017

Martin Hofmann 1965–2018
Cubical & Computational

Features

1. computational: canonicity by definition
2. key features from homotopy type theory (HoTT)
3. exact equality types
Cubical & Computational Features

1. computational: canonicity by definition
2. key features from homotopy type theory (HoTT)
3. exact equality types

Advantages for computer science:

1. Equational reasoning closer to standard mathematics
2. Equivalent (good) types share the same properties
3. Quotients and other types built with relations
4. Openness to new constructs
Computational Types

programs/
realizers

computation
Computational Types

- programs/
  - realizers

- computational type theory

- computation

- theory of computation
Computational Types

- Programs/Realizers
- Computation
- Computational type theory
- Theory of computation
- Meaning/Explanation
- Martin-Löf type theory
- Pre-mathematical in M-L's work
A Minimum Example

\[ M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M,M,M) \]
A Minimum Example

\[ M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M,M,M) \]

true val
false val
\[ \text{if}(\text{true},M,\_) \mapsto M \]
\[ \text{if}(\text{false},\_,M) \mapsto M \]
A Minimum Example

\[ M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if} (M, M, M) \]

\[
\begin{align*}
\text{bool val} & \quad \text{true val} & \quad \text{if} (\text{true}, M, _) & \Rightarrow M \\
\text{false val} & \quad \text{if} (\text{false}, _, M) & \Rightarrow M
\end{align*}
\]
A Minimum Example

M := a | bool | true | false | if(M,M,M)

bool val true val if(true,M,_) ↦ M
false val if(false,_,M) ↦ M

The Language

What are the types in canonical forms? {bool}
A Minimum Example

\[ M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M, M, M) \]

\begin{align*}
\text{bool val} & \quad \text{true val} & \quad \text{if}(\text{true}, M, _) & \Rightarrow M \\
\text{false val} & \quad \text{if}(\text{false}, _, M) & \Rightarrow M
\end{align*}

The Language

What are the types in \textit{canonical forms}? \{\text{bool}\}

What are the \textit{canonical forms} of the types?
\text{bool}: \{\text{true}, \text{false}\}
A Minimum Example

M := a | bool | true | false | if(M,M,M)

true val
false val
if(true,M,_) ↦ M
if(false,_,M) ↦ M

The Language

What are the types in canonical forms? \{bool\}

What are the canonical forms of the types?

bool: \{true, false\}

How they are equal? syntactic equality
A Minimum Example

\[ M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M,M,M) \]

- bool val
- true val
- if(true,M,_) ↦ M
- false val
- if(false,_,M) ↦ M

The Language

What are the types in canonical forms? \{bool\}

What are the canonical forms of the types?
- bool: \{true, false\}

How they are equal? syntactic equality

One Theory
A Minimum Example

\[
M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M,M,M)
\]

**types:** \{bool\} with syntactic equality

**bool:** \{true, false\} with syntactic equality
A Minimum Example

\[ M := a | \text{bool} | \text{true} | \text{false} | \text{if}(M,M,M) \]

types: \{\text{bool}\} with syntactic equality

\text{bool}: \{\text{true, false}\} with syntactic equality

A ≐ B type

A⇓A' B⇓B' and A'≈B'
A Minimum Example

\[
M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M,M,M)
\]

**types:** \{\text{bool}\} \text{ with syntactic equality}

\text{bool}: \{\text{true, false}\} \text{ with syntactic equality}

\[
\begin{align*}
A & \equiv B \text{ type} \\
A & \Downarrow A' \quad B \Downarrow B' \quad \text{and} \quad A' \approx B'
\end{align*}
\]

bool \equiv \text{bool type}
A Minimum Example

M := a | bool | true | false | if(M,M,M)

types: \{bool\} with syntactic equality
bool: \{true, false\} with syntactic equality

\[ M \equiv N \in A \]
A \equiv A \text{ type, } M \downarrow M', N \downarrow N', A \downarrow A' \text{ and } M' \approx_{A'} N' \]
A Minimum Example

\[ M := a \mid \text{bool} \mid \text{true} \mid \text{false} \mid \text{if}(M,M,M) \]

types: \{\text{bool}\} with syntactic equality

\text{bool}: \{\text{true, false}\} with syntactic equality

\[ M \equiv N \in A \]

\[ A \equiv A \text{ type, } M \downarrow M', N \downarrow N', A \downarrow A' \text{ and } M' \approx_{A'} N' \]

\[ \text{false} \equiv \text{false} \in \text{bool} \]
A Minimum Example

$M := a | \text{bool} | \text{true} | \text{false} | \text{if}(M,M,M)$

types: $\{\text{bool}\}$ with syntactic equality

$\text{bool}: \{\text{true, false}\}$ with syntactic equality

$M \equiv N \in A$

$A \equiv A$ type, $M \Downarrow M'$, $N \Downarrow N'$, $A \Downarrow A'$ and $M' \approx_{A'} N'$

false $\not\equiv$ false $\in \text{bool}$

if$(true, true, \text{bool}) \equiv true \in$ if$(true, \text{bool}, \text{bool})$

$\Downarrow true$ $\Downarrow \text{bool}$
A Minimum Example

M := a | bool | true | false | if(M,M,M)

types: \{bool\} with syntactic equality
bool: \{true, false\} with syntactic equality

\[ a: A \gg M \equiv N \in B \]
\[ P \equiv Q \in A \text{ implies } M[P/a] \equiv N[Q/a] \in B \]
A Minimum Example

\[ M := a | \text{bool} | \text{true} | \text{false} | \text{if}(M,M,M) \]

Types: \{\text{bool}\} with syntactic equality

\text{bool}: \{\text{true, false}\} with syntactic equality

\[ a:A \gg M \equiv N \in B \]

\[ P \equiv Q \in A \text{ implies } M[P/a] \equiv N[Q/a] \in B \]

\[ b:\text{bool} \gg b \equiv \text{if}(b, \text{true, false}) \in \text{bool}? \]
Variables

In Nuprl and friends
variables range over closed terms
Variables

In Nuprl and friends
variables range over closed terms

In Coq, Agda, and friends
variables (generally) cannot be inductively analyzed
Variables

In Nuprl and friends
variables range over closed terms

In Coq, Agda, and friends
variables (generally)
cannot be inductively analyzed

closed reduction ⇔ vars over closed terms
open reduction ⇔ vars indeterminate
A Functional Example

\[ M := a \mid M_1 \rightarrow M_2 \mid \lambda a. M \mid M_1 \ M_2 \mid \ldots \]

\[(M_1 \rightarrow M_2) \ val \ \lambda a. M \ val \ (\lambda a. M_1)M_2 \mapsto M_1[M_2/a]\]

Another Language
A Functional Example

\[ M := a \mid M_1 \rightarrow M_2 \mid \lambda a.M \mid M_1 \ M_2 \mid ... \]

(M_1 \rightarrow M_2) \text{ val } \lambda a.M \text{ val } (\lambda a.M_1)M_2 \mapsto M_1[M_2/a](M_1 \rightarrow M_2) \text{ val}

Another Language

What are the types in canonical forms?

the least fixed point of
S.\((\{M\rightarrow N \mid M \downarrow, N \downarrow \text{ in } S\} \text{ union } ...\))

What are the canonical forms of the types?

A \rightarrow B: \{\lambda a.M\}

How they are equal?

A_1 \rightarrow B_1 \approx A_2 \rightarrow B_2 \text{ if } A_1 \equiv A_2 \text{ and } B_1 \equiv B_2
\lambda a.M_1 \approx_{\rightarrow \text{b}} \lambda a.M_2 \text{ if } a:A \gg M_1 \equiv M_2 \in B
Openness
Openness

Open to new constructs
Openness

Open to *new constructs*

Open to *new theories*

for the same language
Openness

Open to *new constructs*

Open to *new theories* for the same language

Open to *new proof theories* (rules in proof assistants) for the same theory
Openness

Open to new constructs
Open to new theories for the same language
Open to new proof theories (rules in proof assistants) for the same theory

Canonicity always holds
# Homotopy Type Theory

[Awodey and Warren] [Voevodsky et al] [van den Berg and Garner]

<table>
<thead>
<tr>
<th>$A$</th>
<th>Type</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a : A$</td>
<td>Element</td>
<td>Point</td>
</tr>
<tr>
<td>$f : A \to B$</td>
<td>Function</td>
<td>Continuous Mapping</td>
</tr>
<tr>
<td>$C : A \to Type$</td>
<td>Dependent Type</td>
<td>Fibration</td>
</tr>
<tr>
<td>$a =_A b$</td>
<td>Identification</td>
<td>Path</td>
</tr>
</tbody>
</table>
Homotopy Type Theory

$\bullet a$

$\bullet b$

points
Homotopy Type Theory

$p : a = b$

$q : a = b$

paths

points
Homotopy Type Theory

\[ p : a = b \]
\[ q : a = b \]
\[ h : p = q \]
Homotopy Type Theory

\[ p : a = b \]
\[ h : p = q \]
\[ q : a = b \]
Homotopy Type Theory

Tons of results in homotopy theory are mechanized through this.

In some case new proofs were discovered and inspired new results. [Anel, Biedermann, Finster, Joyal]

Also lots of works in category theory and other fields.
Key Features of HoTT

0. Based on intensional type theory

1. Identifications as paths

2. Univalence: if $e$ is an equivalence between $A$ and $B$, then $ua(e):A=B$

3. Higher inductive types: generalized inductive types with (higher) path generators
Key Features of HoTT

0. Based on intensional type theory

1. Identifications as paths

2. Univalence: if \( e \) is an equivalence between \( A \) and \( B \), then \( ua(e):A=B \)

3. Higher inductive types:
   generalized inductive types
   with (higher) path generators

Problems: 2&3 give new identifications
The Poor J Eliminator

\[ J[a.C](\text{refl-case, path}) \]

eliminator for identifications can only handle reflexivity

\[
\text{coe}(p:A=B,a:A):B
\]

\[
\text{coe}(ua(e),a) \text{ is stuck}
\]
The Poor J Eliminator

\[ J[a.C](\text{refl-case, path}) \]

eliminator for identifications

can only handle reflexivity

\[ \text{coe}(p:A=B,a:A):B \]
\[ \text{coe}(ua(e),a) \] is stuck

Solution

motive C handles paths itself
The Happy J Eliminator

each motive handles paths itself

each type has **cubical Kan structure**

[Bezem, Coquand, Huber] [Cohen, Coquand, Huber, Mörtberg]

This work:
extend Nuprl by **cubical Kan structures**

Cubical Programming
Cubical Programming

dim expr r := 0 | 1 | x
Cubical Programming

\[
dim \ expr \ r \ := \ 0 \mid 1 \mid x
\]

dimension variables (x) should not be inductively analyzable

\[
\begin{array}{cc}
0 & x \\
\bullet & \bullet
\end{array}
\]

somewhere
Cubical Programming

\[
\text{dim expr } \ r := 0 \mid 1 \mid x
\]

dimension variables (x) should not be inductively analyzable

somewhere

* new reduction *

closed in expression variables
open in dimension variables
Circle

base

loop\{x\}
Circle

\[ M := S1 \mid \text{base} \mid \text{loop}\{r\} \]
\[ \mid S1elim(a.M, M, M, x.M) \mid \ldots \]
Circle

\[ M := S1 \mid \text{base} \mid \text{loop}\{r\} \mid S1\text{elim}(a.M, M, M, x.M) \mid \ldots \]
Circle

M := S1 | base | loop{r} |
| S1elim(a.M, M, M, x.M) | ...

base

base val

loop{x}

S1 val
Circle

\[ M := S1 \mid base \mid loop^r \mid S1elim(a.M, M, M, x.M) \mid ... \]
Circle

base

loop\{x\}

\[
\begin{align*}
M \mapsto M'
\end{align*}
\]

\[
\text{S1elim}(a.A, M, B, x.L) \mapsto \text{S1elim}(a.A, M', B, x.L)
\]
Circle

\[ S_1\text{elim}(a.A, \text{base}, B, x._) \mapsto B \]

\[ \text{base} \]

\[ \text{loop}\{x\} \]

\[ S_1\text{val} \]

\[ M \Rightarrow M' \]

\[ S_1\text{elim}(a.A, M, B, x.L) \Rightarrow S_1\text{elim}(a.A, M', B, x.L) \]

\[ S_1\text{elim}(a.A, \text{base}, B, x._) \Rightarrow B \]
Circle

\[
\begin{align*}
&\text{S1elim}(a.A, \text{base}, B, x._) \mapsto B \\
&\text{S1elim}(a.A, \text{loop}\{x\}, _, y.L) \mapsto L<x/y> \\
&\text{M} \Rightarrow \text{M}' \\
&\text{S1elim}(a.A, \text{M}, B, x.L) \Rightarrow \text{S1elim}(a.A, \text{M}', B, x.L) \\
&\text{S1elim}(a.A, \text{base}, B, x._) \Rightarrow B \\
&\text{S1elim}(a.A, \text{loop}\{x\}, _, y.L) \Rightarrow L<x/y>
\end{align*}
\]
Kan: Coercion

\[ A^{(\emptyset/x)} \quad A^{(1/x)} \]
Kan: Coercion

\[ A^{0/x} \xrightarrow{\text{coe}\{0 \rightarrow 1\}} A^{1/x} \]

\[ (x. A, M) \]
Kan: Coercion

\[ \text{coe}^{r \rightarrow r'}(x.A, M) \in A^{r' / x} \]

\[ \text{M} \]

\[ \text{coe}^{0 \rightarrow 1}(x.A, M) \]

\[ A^{0 / x} \rightarrow A^{1 / x} \]

\[ (x.A, M) \]

\[ A^{r / x} \]
Kan: Homogeneous Comp.
Kan: Homogeneous Comp.

\[ hcom\{0\to1\}(A,M) \leftarrow [x=0\to y.N_0, \ x=1\to y.N_1] \]
Kan: Homogeneous Comp.

\[
\text{hcom}\{0 \rightarrow 1\}(A,M) \\
[x=0 \rightarrow y.N_0, x=1 \rightarrow y.N_1]
\]

\[
\text{hcom}\{r \rightarrow r'\}(A, M) \\
[\ldots, r_i = r'_i \rightarrow y.N_i, \ldots]
\]
Kan Circle

\[ \text{coe}(r \rightarrow r')(\_\cdot S1, M) \Rightarrow M \]
Kan Circle

coe\{r \to r'\}(\_ . S1, M) \Rightarrow M

hcom\{r \to r'\}(S1, M)[...] \Rightarrow fcom\{r \to r'\}(M)[...]

formal composition
Kan Circle

\[\text{coe}(r \rightarrow r')(\_ . S1, M) \Rightarrow M\]

\[\text{hcom}(r \rightarrow r')(S1, M)[...] \Rightarrow \text{fcom}(r \rightarrow r')(M)[...]\]

\[\text{fcom}(r \rightarrow r')(M)[...] \Rightarrow M\]
Kan Circle

coe\{r \to r'\}(\_ . S1, M) \Rightarrow M

hcom\{r \to r'\}(S1, M)[... ] \Rightarrow fcom\{r \to r'\}(M)[... ]

fcom\{r \to r\}(M)[... ] \Rightarrow M

r! = r' \quad r_i = r'_i (the first i)

\hline
fcom\{r \to r'\}(M)[..., r_i = r'_i \to y . N_i, ... ] \Rightarrow N_i <r'/y>
Kan Circle

\[\text{coe}\{r \rightarrow r'\}(\_ . S1, M) \Rightarrow M\]

\[\text{hcom}\{r \rightarrow r'\}(S1, M)[...] \Rightarrow \text{fcom}\{r \rightarrow r'\}(M)[...]\]

\[\text{fcom}\{r \rightarrow r\}(M)[...] \Rightarrow M\]

\[r! = r' \quad r_i = r'_i \quad (\text{the first } i)\]

\[\text{fcom}\{r \rightarrow r'\}(M)[..., r_i = r'_i \rightarrow y.N_i, ...] \Rightarrow N_i<r'/y>\]

\[r! = r' \quad r_i! = r'_i \quad \text{for all } i\]

\[\text{fcom}\{r \rightarrow r'\}(M)[...] \text{ val}\]
Kan Circle

S1elim needs to handle fcom
Kan Circle

\( r \neq r' \quad r_i \neq r'_i \)

\[
\text{S1elim}(a.A, fcom\{r \rightarrow r'}(M)[...], B, x.L) \\
\Rightarrow \text{com}\{r \rightarrow r'}(y.A[fcom\{r \rightarrow y}(...)].../a], \\
\text{S1elim}(M, B, x.L))[...]
\]

\text{S1elim}(\text{composition}) \Rightarrow \text{composition}(\text{S1elim})
Cubical Stability

Dimension subs. do not commute with evaluation!
Cubical Stability

Dimension subssts. do not commute with evaluation!

\[
\text{S1elim}(a.A, \text{loop}\{x\}, B, y.L) \quad \xrightarrow{\quad} \quad L<x/y> \\
\quad \xrightarrow{\quad} \quad <0/x> \\
\quad \xrightarrow{\quad} \quad L<0/y>
\]
Cubical Stability

Dimension subssts. do not commute with evaluation!

\[ \text{S1elim}(a.A, \text{loop}\{x\}, B, y.L) \quad \xrightarrow{\text{L}\langle x/y \rangle} \quad \text{S1elim}(a.A, \text{base}, B, y.L) \quad \xrightarrow{\text{L}\langle 0/y \rangle} \]

Dimension substs. do not commute with evaluation!
Cubical Stability

Dimension subssts. do not commute with evaluation!

\[ S_1\text{elim}(a.A, \text{loop}\{x\}, B, y.L) \quad \xrightarrow{\downarrow \{0/x\}} \quad L\langle x/y \rangle \]

\[ S_1\text{elim}(a.A, \text{base}, B, y.L) \quad \xrightarrow{\downarrow \{0/x\}} \quad B \xrightarrow{=?=?} L\langle 0/y \rangle \]

Restrict our theory to only cubically stable terms
Cubical Type Theory

stability: consider every substitution
Cubical Type Theory

stability: consider every substitution

\[ A \doteq B \text{ type } [\Psi] \]

under any dim substitution \( \psi \)... \( A\psi \) and \( B\psi \) stably* eval to \( A' \) and \( B' \) which stably* recognize the same stable* values and have stably* equal Kan structures

(see our arXiv and POPL papers)
Cubical Type Theory

stability: consider every substitution

\[
A \equiv B \text{ type } [\Psi]
\]

under any dim substitution \(\psi\)...
A\(\psi\) and B\(\psi\) stably* eval to A' and B' which stably* recognize the same stable* values and have stably* equal Kan structures

\[
M \equiv N \in A [\Psi]
\]

A \(\equiv\) A type, A\(\downarrow\)A', M and N stably* eval to M' and N', A' stably* views N' and M' as the same value

(see our arXiv and POPL papers)
Our arXiv Papers

Part 1: stability
Part 2: dependent types
Part 3: univalence and equality
Part 4: cubical inductive types
RedPRL

a proof assistant based on the new type theory

still nascent, changing everyday

http://redprl.org
Conclusion

We extended Nuprl semantics by cubical Kan structures which justify key features of HoTT.

We also built RedPRL as a prototype.