Computational Higher Type Theory

Robert Harper

Computer Science Department
Carnegie Mellon University

GDP70 Celebration
Edinburgh, UK

Happy Birthday Gordon!

Thanks

Joint work with Carlo Angiuli (CMU) and Todd Wilson (CSUF).

Thanks to Dan Licata for many discussions.

Supported by AFOSR MURI FA9550-15-1-0053.

Two Kinds of Type Theory

Two traditions in type theory, both embodied by Martin-Löf:

- Formal, or axiomatic, as in ITT and HoTT.
- Computational, or semantic, as in CMCP.

Two Kinds of Type Theory

Two traditions in type theory, both embodied by Martin-Löf:

- Formal, or axiomatic, as in ITT and HoTT.
- Computational, or semantic, as in CMCP.

HoTT is a formal type theory with

- Univalence Axiom stating that equivalences are type identifications.
- Higher Inductive Types, supporting truncation, etc.

Two Kinds of Type Theory

Two traditions in type theory, both embodied by Martin-Löf:

- Formal, or axiomatic, as in ITT and HoTT.
- Computational, or semantic, as in CMCP.

HoTT is a formal type theory with

- Univalence Axiom stating that equivalences are type identifications.
- Higher Inductive Types, supporting truncation, etc.

What is the computational content of HoTT?

Semantic Type Theory

Martin-Löf; Constable, et al

Meaning explanations define types and elements semantically:

- Computational: as programs with deterministic dynamics.
- Mathematical: using inchoate concepts of set and function.

Semantic Type Theory

Martin-Löf; Constable, et al

Meaning explanations define types and elements semantically:

- Computational: as programs with deterministic dynamics.
- Mathematical: using inchoate concepts of set and function.

Computational meaning explanation: type theory as a prog lang.

- Types are behavioral specifications.
- Types and objects are programs that execute.

Semantic Type Theory

Martin-Löf; Constable, et al

Meaning explanations define types and elements semantically:

- Computational: as programs with deterministic dynamics.
- Mathematical: using inchoate concepts of set and function.

Computational meaning explanation: type theory as a prog lang.

- Types are behavioral specifications.
- Types and objects are programs that execute.

Inverts conceptual order compared to formal type theory:

- Type theory as a theory of truth.
- Proof theory accesses the truth.

Computational Meaning Explanation

Martin-Löf: Constr. Math. and Comp. Prog.

Start with computation on closed expressions (types and terms):

- Transition: $M \longmapsto M'$, one step of execution.
- Termination: M val is canonical/complete.

Computational Meaning Explanation

Martin-Löf: Constr. Math. and Comp. Prog.

Start with computation on closed expressions (types and terms):

- Transition: $M \longmapsto M'$, one step of execution.
- Termination: M val is canonical/complete.

Define exact equality of closed types and terms:

- Type equality: $A \doteq B$ type $[\Psi]$.
- Term equality in a type: $M \doteq N \in A$ $[\Psi]$.

Computational Meaning Explanation

Martin-Löf: Constr. Math. and Comp. Prog.

Start with computation on closed expressions (types and terms):

- Transition: $M \longmapsto M'$, one step of execution.
- Termination: M val is canonical/complete.

Define exact equality of closed types and terms:

- Type equality: $A \doteq B$ type $[\Psi]$.
- Term equality in a type: $M \doteq N \in A$ $[\Psi]$.

Extend to open forms by functionality aka extensionality:

- Types: $a_1{:}A_1, \ldots, a_n{:}A_n \gg A \doteq B$ type $[\Psi]$.
- Terms: $a_1{:}A_1, \ldots, a_n{:}A_n \gg M \doteq N \in A$ $[\Psi]$.

Cubical Type Theory

BCH; CCHM; LH; LB

Bezem, et al.: model of (a version of) HoTT valued in constructive cubical sets.

- Indirect account of computational content of HoTT.
- Simplicial account appears not to be constructive.

Cubical Type Theory

BCH; CCHM; LH; LB

Bezem, et al.: model of (a version of) HoTT valued in constructive cubical sets.

- Indirect account of computational content of HoTT.
- Simplicial account appears not to be constructive.

Focus shifted to higher type theory.

Cubical Type Theory

BCH; CCHM; LH; LB

Bezem, et al.: model of (a version of) HoTT valued in constructive cubical sets.

- Indirect account of computational content of HoTT.
- Simplicial account appears not to be constructive.

Focus shifted to higher type theory.

- Judgmental treatment of identifications.

Cubical Type Theory

BCH; CCHM; LH; LB

Bezem, et al.: model of (a version of) HoTT valued in constructive cubical sets.

- Indirect account of computational content of HoTT.
- Simplicial account appears not to be constructive.

Focus shifted to higher type theory.

- Judgmental treatment of identifications.
- Cubical structure seems most natural.

Cubical Type Theory

BCH; CCHM; LH; LB

Bezem, et al.: model of (a version of) HoTT valued in constructive cubical sets.

- Indirect account of computational content of HoTT.
- Simplicial account appears not to be constructive.

Focus shifted to higher type theory.

- Judgmental treatment of identifications.
- Cubical structure seems most natural.

Formal cubical type theories are under active development (Coquand, et al.; this work.)

Cubical Programming Language

Licata, Brunerie; Coquand, et al.

Syntax is organized cubically:

- Points correspond to ordinary terms and types.
- Lines represent identifications of elements and types.

Cubical Programming Language

Licata, Brunerie; Coquand, et al.

Syntax is organized cubically:

- Points correspond to ordinary terms and types.
- Lines represent identifications of elements and types.

Cartesian cubes are specified by a dimension context, Ψ:

- Finite set of dimension variables $x, y, z, \ldots$.

Cubical Programming Language

Licata, Brunerie; Coquand, et al.

Syntax is organized cubically:

- Points correspond to ordinary terms and types.
- Lines represent identifications of elements and types.

Cartesian cubes are specified by a dimension context, Ψ:

- Finite set of dimension variables $x, y, z, \ldots$.

Substitutions $\psi : \Psi' \to \Psi$ send $x \in \Psi$ to $\psi(x) = 0/1/x' \in \Psi'$.

Cubical Programming Language

Substitutions define the aspects of a cube E:

- Faces: $E\langle 0/x\rangle$, $E\langle 1/x\rangle$.
- Diagonals: $E\langle x', x'/x, y\rangle$.
- Symmetries: $E\langle y, x/x, y\rangle$.
- Degeneracy: silent/implicit.

$$\begin{tikzcd}
E\langle 0/x\rangle\langle 0/y\rangle \arrow[r, "E\langle 0/y\rangle"] \arrow[d, "E\langle 0/x\rangle"'] \arrow[dr, phantom, "E"] & E\langle 1/x\rangle\langle 0/y\rangle \arrow[d, "E\langle 1/x\rangle"] \\
E\langle 0/x\rangle\langle 1/y\rangle \arrow[r, "E\langle 1/y\rangle"'] & E\langle 1/x\rangle\langle 1/y\rangle
\end{tikzcd}$$

Cubical Programming Language

Conventional functional programming constructs:

- Booleans, pairs, functions.
- Lazy dynamics (weak head reduction)

Cubical Programming Language

Conventional functional programming constructs:

- Booleans, pairs, functions.
- Lazy dynamics (weak head reduction)

Unconventional functional programming constructs:

- Circle: $\mathbb{S}^1, \mathsf{base}, \mathsf{loop}_x, \mathbb{S}^1\text{-}\mathsf{elim}_{a.A}(M; M_{\mathsf{b}}, x.M_{\mathsf{l}})$.
- Negation: not_x, a type line, and glueing, $\mathsf{notel}_x(M)$.
- Kan operations: coe, hcom.

Cubical Programming Language

Conventional functional programming constructs:

- Booleans, pairs, functions.
- Lazy dynamics (weak head reduction)

Unconventional functional programming constructs:

- Circle: $\mathbb{S}^1, \mathsf{base}, \mathsf{loop}_x, \mathbb{S}^1\text{-}\mathsf{elim}_{a.A}(M; M_{\mathsf{b}}, x.M_{\mathsf{l}})$.
- Negation: not_x, a type line, and glueing, $\mathsf{notel}_x(M)$.
- Kan operations: coe, hcom.

The Kan operations are computational content of the Kan condition (cf, LB14, CCHM16).

Kan Operations

Coercion along a type line: $\mathsf{coe}_{x.A}^{r \rightsquigarrow r'}(M)$.

- Heterogeneous along line $x.A$.
- Evaluates A to effect coercion from $A\langle r/x\rangle$ to $A\langle r'/x\rangle$.

Composition: $\mathsf{hcom}_{A}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^{\varepsilon}})$.

Kan Operations

Coercion along a type line: $\mathsf{coe}^{r \rightsquigarrow r'}_{x.A}(M)$.

- Heterogeneous along line $x.A$.
- Evaluates A to effect coercion from $A\langle r/x\rangle$ to $A\langle r'/x\rangle$.

Composition: $\mathsf{hcom}^{\overrightarrow{r_i}}_{A}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_{i}})$.

- Homogeneous: within type, not line, A.

Kan Operations

Coercion along a type line: $\mathsf{coe}_{x.A}^{r \rightsquigarrow r'}(M)$.

- Heterogeneous along line $x.A$.
- Evaluates A to effect coercion from $A\langle r/x \rangle$ to $A\langle r'/x \rangle$.

Composition: $\mathsf{hcom}_A^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon})$.

- Homogeneous: within type, not line, A.
- The start r and end r' dimensions.

Kan Operations

Coercion along a type line: $\mathsf{coe}_{x.A}^{r \rightsquigarrow r'}(M)$.

- Heterogeneous along line $x.A$.
- Evaluates A to effect coercion from $A\langle r/x\rangle$ to $A\langle r'/x\rangle$.

Composition: $\mathsf{hcom}_A^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon})$.

- Homogeneous: within type, not line, A.
- The start r and end r' dimensions.
- The cap M is the starting cube.

Kan Operations

Coercion along a type line: $\mathsf{coe}_{x.A}^{r \rightsquigarrow r'}(M)$.

- Heterogeneous along line $x.A$.
- Evaluates A to effect coercion from $A\langle r/x\rangle$ to $A\langle r'/x\rangle$.

Composition: $\mathsf{hcom}_{A}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^{\varepsilon}})$.

- Homogeneous: within type, not line, A.
- The start r and end r' dimensions.
- The cap M is the starting cube.
- The tubes $\overrightarrow{y.N_i^{\varepsilon}}$ with extent $\overrightarrow{r_i}$ in dimension $\overrightarrow{y_i}$.

Kan Operations

Coercion along a type line: $\mathsf{coe}_{x.A}^{r \rightsquigarrow r'}(M)$.

- Heterogeneous along line $x.A$.
- Evaluates A to effect coercion from $A\langle r/x\rangle$ to $A\langle r'/x\rangle$.

Composition: $\mathsf{hcom}_{A}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^{\varepsilon}})$.

- Homogeneous: within type, not line, A.
- The start r and end r' dimensions.
- The cap M is the starting cube.
- The tubes $\overrightarrow{y.N_i^{\varepsilon}}$ with extent $\overrightarrow{r_i}$ in dimension $\overrightarrow{y_i}$.
- Evaluates A to define composite, which may or may not be the hcom itself.

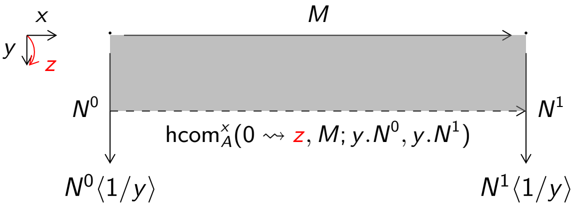
Two-Dimensional Compositions

$$
\begin{tikzcd}
{} \arrow[r, "x"] \arrow[d, "y"'] & {} \\
{} &
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge]
\cdot \arrow[r, "M"] \arrow[r, dashed, bend right, "\mathsf{hcom}^{x}_{A}(0 \rightsquigarrow 0, M; y.N^0, y.N^1)"'] \arrow[d, "N^0"'] & \cdot \arrow[d, "N^1"] \\
N^0\langle 1/y \rangle & N^1\langle 1/y \rangle
\end{tikzcd}
$$

Two-Dimensional Compositions

$$
\begin{tikzcd}
\cdot \arrow[r, "x"] \arrow[d, "y"'] & {} \\
{} &
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge, row sep=large]
\cdot \arrow[r, "M"] \arrow[d, "N^0"'] & \cdot \arrow[d, "N^1"] \\
N^0\langle 1/y \rangle \arrow[r, dashed, "\mathsf{hcom}^x_A(0 \rightsquigarrow {\color{red}1}, M; y.N^0, y.N^1)"'] & N^1\langle 1/y \rangle
\end{tikzcd}
$$

Two-Dimensional Compositions



Cubical Meaning Explanation

Explanation proceeds in stages:

- Define the canonical types and their elements at each dimension Ψ.
- Define pre-types to be cubical, ie with coherent aspects.
- Define types to be Kan pre-types.

Cubical Meaning Explanation

Explanation proceeds in stages:

- Define the canonical types and their elements at each dimension Ψ.
- Define pre-types to be cubical, ie with coherent aspects.
- Define types to be Kan pre-types.

The main criteria for a higher type system:

- All aspects of a type or element must be types or elements.
- Taking aspects must commute with evaluation.
- Equal types must have the same element equality.
- Equal types must be equally Kan.

Cubical Type Systems

A cubical type system consists of a family of per's:

Cubical Type Systems

A cubical type system consists of a family of per's:

- Canonical types: $A_0 \approx^{\Psi} B_0$.

Cubical Type Systems

A cubical type system consists of a family of per's:

- Canonical types: $A_0 \approx^{\Psi} B_0$.
- Canonical elements of a canonical type: $M_0 \approx^{\Psi}_{A_0} N_0$.

Cubical Type Systems

A cubical type system consists of a family of per's:

- Canonical types: $A_0 \approx^{\Psi} B_0$.
- Canonical elements of a canonical type: $M_0 \approx^{\Psi}_{A_0} N_0$.
- Type equality: If $A_0 \approx^{\Psi} B_0$, then $\approx^{\Psi}_{A_0}$ is $\approx^{\Psi}_{B_0}$.

Cubical Type Systems

A cubical type system consists of a family of per's:

- Canonical types: $A_0 \approx^{\Psi} B_0$.
- Canonical elements of a canonical type: $M_0 \approx^{\Psi}_{A_0} N_0$.
- Type equality: If $A_0 \approx^{\Psi} B_0$, then $\approx^{\Psi}_{A_0}$ is $\approx^{\Psi}_{B_0}$.

Extend to general closed expressions by evaluation:

Cubical Type Systems

A cubical type system consists of a family of per's:

- Canonical types: $A_0 \approx^{\Psi} B_0$.
- Canonical elements of a canonical type: $M_0 \approx^{\Psi}_{A_0} N_0$.
- Type equality: If $A_0 \approx^{\Psi} B_0$, then $\approx^{\Psi}_{A_0}$ is $\approx^{\Psi}_{B_0}$.

Extend to general closed expressions by evaluation:

- $A \sim^{\Psi} B$ iff $A \longmapsto^{*} A_0$ and $B \longmapsto^{*} B_0$ and $A_0 \approx^{\Psi} B_0$.

Cubical Type Systems

A cubical type system consists of a family of per's:

- Canonical types: $A_0 \approx^{\Psi} B_0$.
- Canonical elements of a canonical type: $M_0 \approx^{\Psi}_{A_0} N_0$.
- Type equality: If $A_0 \approx^{\Psi} B_0$, then $\approx^{\Psi}_{A_0}$ is $\approx^{\Psi}_{B_0}$.

Extend to general closed expressions by evaluation:

- $A \sim^{\Psi} B$ iff $A \longmapsto^* A_0$ and $B \longmapsto^* B_0$ and $A_0 \approx^{\Psi} B_0$.
- $M \sim^{\Psi}_{A} N$ iff $M \longmapsto^* M_0$, $N \longmapsto^* N_0$, $A \longmapsto^* A_0$, and $M_0 \approx^{\Psi}_{A_0} N_0$.

Pre-Types: Coherent Aspects

Pre-types A pretype $[\Psi]$ must have coherent aspects:

Pre-Types: Coherent Aspects

Pre-types A pretype $[\Psi]$ must have coherent aspects:

- Let $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$.

Pre-Types: Coherent Aspects

Pre-types A pretype $[\Psi]$ must have coherent aspects:

- Let $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$.
- Let $A\psi_1 \longmapsto^* A_1$ val, and $A_1\psi_2 \longmapsto^* A_2$ val, and $A\psi_2\psi_1 \longmapsto^* A_{12}$ val.

Pre-Types: Coherent Aspects

Pre-types A pretype $[\Psi]$ must have coherent aspects:

- Let $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$.
- Let $A\psi_1 \longmapsto^* A_1$ val, and $A_1\psi_2 \longmapsto^* A_2$ val, and $A\psi_2\psi_1 \longmapsto^* A_{12}$ val.
- Require:

$$
\begin{tikzcd}
A \arrow[r, Rightarrow, "\psi_1"] \arrow[d, Rightarrow, "\psi_1\psi_2"'] & A_1 \arrow[d, Rightarrow, "\psi_2"] \\
A_{12} \arrow[r, phantom, "\approx^{\Psi_2}"] & A_2
\end{tikzcd}
$$

Pre-Types: Coherent Aspects

Pre-types A pretype $[\Psi]$ must have coherent aspects:

- Let $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$.
- Let $A\psi_1 \longmapsto^* A_1$ val, and $A_1\psi_2 \longmapsto^* A_2$ val, and $A\psi_2\psi_1 \longmapsto^* A_{12}$ val.
- Require:

$$
\begin{tikzcd}
A \arrow[r, Rightarrow, "\psi_1"] \arrow[d, Rightarrow, "\psi_1\psi_2"'] & A_1 \arrow[d, Rightarrow, "\psi_2"] \\
A_{12} \arrow[r, phantom, "\approx^{\Psi_2}"] & A_2
\end{tikzcd}
$$

Similarly for exact equality of types and of elements: substitute-then-evaluate is functorial.

Pre-Types and Types

A pretype $[\Psi]$ is cubical: its values have coherent aspects:

- If $\psi : \Psi' \to \Psi$ and $M \approx^{\Psi'}_{A\psi} N$, then $M \doteq N \in A\psi\ [\Psi']$.

Pre-Types and Types

A pretype $[\Psi]$ is cubical: its values have coherent aspects:

- If $\psi : \Psi' \to \Psi$ and $M \approx^{\Psi'}_{A\psi} N$, then $M \doteq N \in A\psi\ [\Psi']$.

A type is a Kan pre-type:

Pre-Types and Types

A pretype $[\Psi]$ is cubical: its values have coherent aspects:

- If $\psi : \Psi' \to \Psi$ and $M \approx^{\Psi'}_{A\psi} N$, then $M \doteq N \in A\psi\ [\Psi']$.

A type is a Kan pre-type:

- Supports coercion and composition.

Pre-Types and Types

A pretype $[\Psi]$ is cubical: its values have coherent aspects:

- If $\psi : \Psi' \to \Psi$ and $M \approx^{\Psi'}_{A\psi} N$, then $M \doteq N \in A\psi\ [\Psi']$.

A type is a Kan pre-type:

- Supports coercion and composition.
- Certain equational requirements are met.

Kan Conditions for Coercion

For any $\psi : (\Psi', x) \to \Psi$, if

$$M \in A\psi\langle r/x \rangle \ [\Psi'],$$

then

$$\mathsf{coe}_{x.A\psi}^{r \rightsquigarrow r'}(M) \in A\psi\langle r'/x \rangle \ [\Psi'].$$

Kan Conditions for Coercion

For any $\psi : (\Psi', x) \to \Psi$, if

$$M \in A\psi\langle r/x\rangle \ [\Psi'],$$

then

$$\mathsf{coe}_{x.A\psi}^{r \rightsquigarrow r'}(M) \in A\psi\langle r'/x\rangle \ [\Psi'].$$

For any $\psi : (\Psi', x) \to \Psi$, if

$$M \in A\psi\langle r/x\rangle \ [\Psi'],$$

then

$$\mathsf{coe}_{x.A\psi}^{r \rightsquigarrow r}(M) \doteq M \in A\psi\langle r/x\rangle \ [\Psi'].$$

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,
- $N_i^{\varepsilon} \doteq N_j^{\varepsilon'} \in A\psi\ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,
- $N_i^\varepsilon \doteq N_j^{\varepsilon'} \in A\psi\ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')
- $N_i^\varepsilon\langle r/y\rangle \doteq M \in A\psi\ [\Psi' \mid r_i = \varepsilon]$ (all i and ε)

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,
- $N_i^{\varepsilon} \doteq N_j^{\varepsilon'} \in A\psi\ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')
- $N_i^{\varepsilon}\langle r/y \rangle \doteq M \in A\psi\ [\Psi' \mid r_i = \varepsilon]$ (all i and ε)

then

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,
- $N_i^\varepsilon \doteq N_j^{\varepsilon'} \in A\psi\ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')
- $N_i^\varepsilon\langle r/y\rangle \doteq M \in A\psi\ [\Psi' \mid r_i = \varepsilon]$ (all i and ε)

then

- $\mathsf{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}) \in A\psi\ [\Psi']$.

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,
- $N_i^\varepsilon \doteq N_j^{\varepsilon'} \in A\psi\ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')
- $N_i^\varepsilon\langle r/y\rangle \doteq M \in A\psi\ [\Psi' \mid r_i = \varepsilon]$ (all i and ε)

then

- $\mathsf{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}) \in A\psi\ [\Psi']$.
- $\mathsf{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r, M; \overrightarrow{y.N_i^\varepsilon}) \doteq M \in A\psi\ [\Psi']$.

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi\ [\Psi']$,
- $N_i^\varepsilon \doteq N_j^{\varepsilon'} \in A\psi\ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')
- $N_i^\varepsilon\langle r/y\rangle \doteq M \in A\psi\ [\Psi' \mid r_i = \varepsilon]$ (all i and ε)

then

- $\mathrm{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}) \in A\psi\ [\Psi']$.
- $\mathrm{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r, M; \overrightarrow{y.N_i^\varepsilon}) \doteq M \in A\psi\ [\Psi']$.
- $\mathrm{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}) \doteq N_i^\varepsilon\langle r'/y\rangle \in A\psi\ [\Psi']$ if $r_i = \varepsilon$.

Kan Conditions for Composition

For any $\psi : \Psi' \to \Psi$, if

- $M \in A\psi \ [\Psi']$,
- $N_i^\varepsilon \doteq N_j^{\varepsilon'} \in A\psi \ [\Psi', y \mid r_i = \varepsilon, r_j = \varepsilon']$ (all i, j, ε, and ε')
- $N_i^\varepsilon\langle r/y\rangle \doteq M \in A\psi \ [\Psi' \mid r_i = \varepsilon]$ (all i and ε)

then

- $\mathsf{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}) \in A\psi \ [\Psi']$.
- $\mathsf{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r, M; \overrightarrow{y.N_i^\varepsilon}) \doteq M \in A\psi \ [\Psi']$.
- $\mathsf{hcom}_{A\psi}^{\overrightarrow{r_i}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}) \doteq N_i^\varepsilon\langle r'/y\rangle \in A\psi \ [\Psi']$ if $r_i = \varepsilon$.

Constraints limit applicable substitutions; conditions can be vacuous.

Defining Booleans

The Booleans are defined as a higher inductive type.

Defining Booleans

The Booleans are defined as a higher inductive type.

- Innocent of its status as a set.

Defining Booleans

The Booleans are defined as a higher inductive type.

- Innocent of its status as a set.
- Canonical hcom's are values.

Defining Booleans

The Booleans are defined as a higher inductive type.

- Innocent of its status as a set.
- Canonical hcom's are values.
- (Could also have strict version.)

Defining Booleans

The Booleans are defined as a higher inductive type.

- Innocent of its status as a set.
- Canonical hcom's are values.
- (Could also have strict version.)

The dynamics of the conditional accounts for

- true and false, as usual.
- Canonical hcom's.

Defining Booleans

The Booleans are defined as a higher inductive type.

- Innocent of its status as a set.
- Canonical hcom's are values.
- (Could also have strict version.)

The dynamics of the conditional accounts for

- true and false, as usual.
- Canonical hcom's.

The following "rules" are theorems, not definitions.

Boolean Dynamics

$$\frac{}{\mathsf{bool}\ \mathsf{val}} \qquad \frac{\overrightarrow{r_i} = x_1, \ldots, x_{i-1}, \varepsilon, r_{i+1}, \ldots, r_n}{\mathsf{hcom}^{\overrightarrow{r_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^\varepsilon_i}) \longmapsto N^\varepsilon_i \langle r'/y \rangle}$$

$$\frac{r = r'}{\mathsf{hcom}^{x_1,\ldots,x_n}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^\varepsilon_i}) \longmapsto M} \qquad \frac{}{\mathsf{true}\ \mathsf{val}} \qquad \frac{}{\mathsf{false}\ \mathsf{val}}$$

$$\frac{r \neq r'}{\mathsf{hcom}^{x_1,\ldots,x_n}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^\varepsilon_i})\ \mathsf{val}}$$

Boolean Dynamics

$$\frac{M \longmapsto M'}{\mathsf{if}_{a.A}(M; T, F) \longmapsto \mathsf{if}_{a.A}(M'; T, F)} \qquad \frac{}{\mathsf{if}_{a.A}(\mathsf{true}; T, F) \longmapsto T}$$

$$\frac{}{\mathsf{if}_{a.A}(\mathsf{false}; T, F) \longmapsto F}$$

$$\frac{r \neq r' \quad H = \mathsf{hcom}^{x_1,\ldots,x_n}_{\mathsf{bool}}(r \rightsquigarrow z, M; \overrightarrow{y.N_i^\varepsilon})}{\begin{array}{c}\mathsf{if}_{a.A}(\mathsf{hcom}^{x_1,\ldots,x_n}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^\varepsilon}); T, F) \\ \longmapsto \\ \mathsf{com}^{x_1,\ldots,x_n}_{z.A[H/a]}(r \rightsquigarrow r', \mathsf{if}_{a.A}(M; T, F); \overrightarrow{y.\mathsf{if}_{a.A}(N_i^\varepsilon; T, F)})\end{array}}$$

$$\frac{}{\mathsf{coe}^{r \rightsquigarrow r'}_{x.\mathsf{bool}}(M) \longmapsto M}$$

Canonical Booleans

A CTS has booleans if bool $\approx^{\Psi}$ bool and $\approx^{\Psi}_{\text{bool}}$ is least s.t.

Canonical Booleans

A CTS has booleans if bool $\approx^{\Psi}$ bool and $\approx^{\Psi}_{\text{bool}}$ is least s.t.

- true $\approx^{\Psi}_{\text{bool}}$ true,

Canonical Booleans

A CTS has booleans if $\mathsf{bool} \approx^{\Psi} \mathsf{bool}$ and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- $\mathsf{true} \approx^{\Psi}_{\mathsf{bool}} \mathsf{true}$,
- $\mathsf{false} \approx^{\Psi}_{\mathsf{bool}} \mathsf{false}$, and

Canonical Booleans

A CTS has booleans if $\mathsf{bool} \approx^{\Psi} \mathsf{bool}$ and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- $\mathsf{true} \approx^{\Psi}_{\mathsf{bool}} \mathsf{true}$,
- $\mathsf{false} \approx^{\Psi}_{\mathsf{bool}} \mathsf{false}$, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_{i}}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P^{\varepsilon}_{i}})$ when

Canonical Booleans

A CTS has booleans if $\mathsf{bool} \approx^{\Psi} \mathsf{bool}$ and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- $\mathsf{true} \approx^{\Psi}_{\mathsf{bool}} \mathsf{true}$,
- $\mathsf{false} \approx^{\Psi}_{\mathsf{bool}} \mathsf{false}$, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_i}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P^{\varepsilon}_i})$ when
  - $r \neq r'$,

Canonical Booleans

A CTS has booleans if bool $\approx^{\Psi}$ bool and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- true $\approx^{\Psi}_{\mathsf{bool}}$ true,
- false $\approx^{\Psi}_{\mathsf{bool}}$ false, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_i}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P^{\varepsilon}_i})$ when
  - $r \neq r'$,
  - $M \doteq O \in \mathsf{bool}\ [\Psi]$,

Canonical Booleans

A CTS has booleans if $\mathsf{bool} \approx^{\Psi} \mathsf{bool}$ and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- $\mathsf{true} \approx^{\Psi}_{\mathsf{bool}} \mathsf{true}$,
- $\mathsf{false} \approx^{\Psi}_{\mathsf{bool}} \mathsf{false}$, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_i}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P^{\varepsilon}_i})$ when
  - $r \neq r'$,
  - $M \doteq O \in \mathsf{bool}\ [\Psi]$,
  - $N^{\varepsilon}_i \doteq N^{\varepsilon'}_j \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon, x_j = \varepsilon']$ for all $i, j, \varepsilon, \varepsilon'$,

Canonical Booleans

A CTS has booleans if $\mathsf{bool} \approx^{\Psi} \mathsf{bool}$ and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- $\mathsf{true} \approx^{\Psi}_{\mathsf{bool}} \mathsf{true}$,
- $\mathsf{false} \approx^{\Psi}_{\mathsf{bool}} \mathsf{false}$, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N_i^{\varepsilon}}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P_i^{\varepsilon}})$ when
  - $r \neq r'$,
  - $M \doteq O \in \mathsf{bool}\ [\Psi]$,
  - $N_i^{\varepsilon} \doteq N_j^{\varepsilon'} \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon, x_j = \varepsilon']$ for all $i, j, \varepsilon, \varepsilon'$,
  - $N_i^{\varepsilon} \doteq P_i^{\varepsilon} \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon]$ for all i, ε, and

Canonical Booleans

A CTS has booleans if $\mathsf{bool} \approx^{\Psi} \mathsf{bool}$ and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- $\mathsf{true} \approx^{\Psi}_{\mathsf{bool}} \mathsf{true}$,
- $\mathsf{false} \approx^{\Psi}_{\mathsf{bool}} \mathsf{false}$, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_i}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P^{\varepsilon}_i})$ when
  - $r \neq r'$,
  - $M \doteq O \in \mathsf{bool}\ [\Psi]$,
  - $N^{\varepsilon}_i \doteq N^{\varepsilon'}_j \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon, x_j = \varepsilon']$ for all $i, j, \varepsilon, \varepsilon'$,
  - $N^{\varepsilon}_i \doteq P^{\varepsilon}_i \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon]$ for all i, ε, and
  - $N^{\varepsilon}_i\langle r/y\rangle \doteq M \in \mathsf{bool}\ [\Psi \mid x_i = \varepsilon]$ for all i, ε.

Canonical Booleans

A CTS has booleans if bool $\approx^{\Psi}$ bool and $\approx^{\Psi}_{\mathsf{bool}}$ is least s.t.

- true $\approx^{\Psi}_{\mathsf{bool}}$ true,
- false $\approx^{\Psi}_{\mathsf{bool}}$ false, and
- $\mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', M; \overrightarrow{y.N^{\varepsilon}_i}) \approx^{\Psi,x}_{\mathsf{bool}} \mathsf{hcom}^{\overrightarrow{x_i}}_{\mathsf{bool}}(r \rightsquigarrow r', O; \overrightarrow{y.P^{\varepsilon}_i})$ when
  - $r \neq r'$,
  - $M \doteq O \in \mathsf{bool}\ [\Psi]$,
  - $N^{\varepsilon}_i \doteq N^{\varepsilon'}_j \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon, x_j = \varepsilon']$ for all $i, j, \varepsilon, \varepsilon'$,
  - $N^{\varepsilon}_i \doteq P^{\varepsilon}_i \in \mathsf{bool}\ [\Psi, y \mid x_i = \varepsilon]$ for all i, ε, and
  - $N^{\varepsilon}_i \langle r/y \rangle \doteq M \in \mathsf{bool}\ [\Psi \mid x_i = \varepsilon]$ for all i, ε.

Generally, values of positive type include compositions in higher dimensions.

Not as a Line of Types

Define not_x as a line of types between bool and bool.

Not as a Line of Types

Define not_x as a line of types between bool and bool.

- Given by negation (swapping) as a (strict) equivalence.

Not as a Line of Types

Define not_x as a line of types between bool and bool.

- Given by negation (swapping) as a (strict) equivalence.
- Example of univalence principle.

Not as a Line of Types

Define not_x as a line of types between bool and bool.

- Given by negation (swapping) as a (strict) equivalence.
- Example of univalence principle.

The term $\mathsf{notel}_x(M) \in \mathsf{not}_x\ [\Psi, x]$ is a line in not_x:

$$
\begin{tikzcd}[row sep=small, column sep=small]
{} \arrow[r, "x"] \arrow[d, "y"'] & {} \\
{}
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[r, "\mathsf{bool}"] \arrow[d, mapsto, "\mathsf{not}(-)"'] & \cdot \arrow[d, mapsto, "\mathsf{id}"] \\
\mathsf{bool} \arrow[r, dashed, "\mathsf{not}_x"'] & \mathsf{bool}
\end{tikzcd}
$$

Not as a Line of Types

Define not_x as a line of types between bool and bool.

- Given by negation (swapping) as a (strict) equivalence.
- Example of univalence principle.

The term $\mathsf{notel}_x(M) \in \mathsf{not}_x\ [\Psi, x]$ is a line in not_x:

$$\begin{tikzcd}
\cdot \arrow[r, "\mathsf{bool}"] \arrow[d, mapsto, "\mathsf{not}(-)"'] & \cdot \arrow[d, mapsto, "\mathsf{id}"] \\
\mathsf{bool} \arrow[r, dashed, "\mathsf{not}_x"'] & \mathsf{bool}
\end{tikzcd}$$

$x \rightarrow$, $y \downarrow$

Not as a Line of Types

Define not_x as a line of types between bool and bool.

- Given by negation (swapping) as a (strict) equivalence.
- Example of univalence principle.

The term $\mathsf{notel}_x(M) \in \mathsf{not}_x\ [\Psi, x]$ is a line in not_x:

$$\begin{tikzcd}
{} \arrow[r, "x"] \arrow[d, "y"'] & {} & \cdot \arrow[rr, "\mathsf{bool}"] \arrow[d, mapsto, "\mathsf{not}(-)"'] & & \cdot \arrow[d, mapsto, "\mathsf{id}"] \\
{} & & \mathsf{bool} \arrow[rr, dashed, "\mathsf{not}_x"'] & & \mathsf{bool}
\end{tikzcd}$$

Cf. CCHM gluing of equivalences to a line of types.

Computational Higher Type Theory

Canonicity Theorem: closed points of bool evaluate to true or false.

Validates higher-dimensional type theory from a computational, even Brouwerian, perspective.

Computational Higher Type Theory

Canonicity Theorem: closed points of bool evaluate to true or false.

Validates higher-dimensional type theory from a computational, even Brouwerian, perspective.

Huber has since proved canonicity for the CCHM theory, relative to weak head reduction.

Whither Proof Theory?

Validates expected formal rules where applicable.

- NuPRL rules for given constructs are valid.
- LB14 rules for Kan cubical type theories are valid.

Whither Proof Theory?

Validates expected formal rules where applicable.

- NuPRL rules for given constructs are valid.
- LB14 rules for Kan cubical type theories are valid.

May be seen as abstract cubical extensional realizability interpretation.

- Abstract = open-ended: Church's Law not accepted.
- Extensional = full universal properties for usual type constructors.
- Realizability for certain formal theories, but that's not the point.

Ongoing and Future Work

Full account of univalence.

Ongoing and Future Work

Full account of univalence.

- Glueing for composition of types not applicable (diagonals).
- Canonical composites define a line of (Kan) types.
- Line of types $\mathsf{ua}_x(E)$ for each equivalence E.
- Ensures equivalence between identifications and equivalences.

Ongoing and Future Work

Full account of univalence.

- Glueing for composition of types not applicable (diagonals).
- Canonical composites define a line of (Kan) types.
- Line of types $\mathsf{ua}_x(E)$ for each equivalence E.
- Ensures equivalence between identifications and equivalences.

Implementation in Sterling's RedPRL (`redprl.org`).

- NuPRL-like refinement rules.
- Extended with names for cubical programs.

References

Stuart F Allen, Mark Bickford, Robert L Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and Evan Moran.
Innovations in computational type theory using Nuprl.
Journal of Applied Logic, 4(4):428–469, 2006.

Carlo Angiuli and Robert Harper.
Computational higher type theory II: Dependent cubical realizability.
Preprint, June 2016.

Carlo Angiuli, Robert Harper, and Todd Wilson.
Computational higher type theory I: Abstract cubical realizability.
Preprint, April 2016.

Marc Bezem, Thierry Coquand, and Simon Huber.
A model of type theory in cubical sets.
In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26, pages 107–128, 2014.

Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg.
Cubical type theory: a constructive interpretation of the univalence axiom.
(To appear), January 2016.

Daniel R. Licata and Guillaume Brunerie.
A cubical type theory, November 2014.
Talk at Oxford Homotopy Type Theory Workshop.