Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities

Carlo Angiuli
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
cangiuli@cs.cmu.edu
https://orcid.org/0000-0002-9590-3303

Kuen-Bang Hou (Favonia)[1]
School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA
favonia@math.ias.edu
https://orcid.org/0000-0002-2310-3673

Robert Harper
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
rwh@cs.cmu.edu
https://orcid.org/0000-0002-9400-2941

Abstract

We present a dependent type theory organized around a Cartesian notion of cubes (with faces, degeneracies, and diagonals), supporting both fibrant and non-fibrant types. The fibrant fragment validates Voevodsky's univalence axiom and includes a circle type, while the non-fibrant fragment includes exact (strict) equality types satisfying equality reflection. Our type theory is defined by a semantics in cubical partial equivalence relations, and is the first two-level type theory to satisfy the canonicity property: all closed terms of boolean type evaluate to either true or false.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases Homotopy Type Theory, Two-Level Type Theory, Computational Type Theory, Cubical Sets

Related Version https://arxiv.org/abs/1712.01800

Funding This research was supported by the Air Force Office of Scientific Research through MURI grant FA9550-15-1-0053 and the National Science Foundation through grant DMS-1638352. Any opinions, findings and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of any sponsoring institution, the U.S. government or any other entity.

Acknowledgements We are greatly indebted to Steve Awodey, Marc Bezem, Evan Cavallo, Daniel Gratzer, Simon Huber, Dan Licata, Ed Morehouse, Anders Mörtberg, Andrew Pitts, Jonathan Sterling, and Todd Wilson for their contributions and advice.

1 Introduction

Martin-Löf has proposed two rather different approaches to equality in dependent type theory, in the guise of his extensional [24] and intensional [25] type theories. *Extensional type theory*,

[1] This author thanks the Isaac Newton Institute for Mathematical Sciences for its support and hospitality during the program "Big Proof" when part of work on this paper was undertaken. The program was supported by EPSRC grant number EP/K032208/1.

particularly its realization as Nuprl's computational type theory [2], is justified by meaning explanations in which closed terms are programs equipped with an operational semantics, and variables are considered to range over closed terms of their given type.

One consequence is that equations hold whenever they are true for all closed terms; for instance, $n : \mathsf{nat}, m : \mathsf{nat} \gg n + m \doteq m + n \in \mathsf{nat}$ as a judgmental equality because $N + M$ and $M + N$ compute the same natural number for any closed natural numbers N, M. Another consequence is known as *equality reflection*: the equality type $\mathsf{Eq}_A(M, N)$ has at most one element, and is inhabited if and only if $M \doteq N \in A$ judgmentally.

In contrast, in *intensional type theory*, judgmental equality is precisely β- (and at certain types, η-) equivalence, and context variables are treated as additional axioms whose form is indeterminate. The identity type $\mathsf{Id}_A(M, N)$ mediates equality reasoning; in an empty context it is inhabited by a single element if and only if $M \equiv N : A$ judgmentally, but in non-empty contexts includes additional equalities such as $n : \mathsf{nat}, m : \mathsf{nat} \vdash P : \mathsf{Id}_{\mathsf{nat}}(n + m, m + n)$, which does not hold judgmentally for variables n, m.

Traditional type theories, extensional or intensional, are constructive in the sense that they admit an interpretation of proofs as programs, often distilled into the *canonicity property* that closed elements of type bool evaluate and are judgmentally equal to either true or false. In computational type theory, this is the very definition of $M \in \mathsf{bool}$ (see Theorem 15), while in intensional type theory, canonicity can be verified by a metatheoretic argument.

Homotopy type theory [29] extends intensional type theory with a number of axioms, including Voevodsky's univalence axiom [31] and higher inductive types [23]. These axioms are justified by mathematical models interpreting types as spaces (e.g., simplicial sets [20] or fibrant objects in a model category [10]), elements of types as points, and identity types as path spaces. In such models, homotopy type theory serves as a framework for synthetic homotopy theory [29], in which higher inductive types provide concrete homotopy types (e.g., n-spheres), the rules of the identity type assert that all constructions respect paths, and univalence asserts moreover that all constructions are invariant under homotopy equivalence.

Despite the success of homotopy type theory as a medium for synthetic results in homotopy theory [11, 30, 14], it is believed that certain objects—famously, semi-simplicial types—cannot be constructed without reference to some notion of exact equality stricter than paths [8, 33]. Because exact equality does not respect paths, any theory with both exact equality and paths must therefore stratify types into *fibrant types* that respect paths, and *non-fibrant types* that do not. Candidate such *two-level type theories* include the Homotopy Type System (HTS) of Voevodsky [33] and the two-level type theory of Altenkirch et al. [3].

Critically, homotopy type theory and existing two-level type theories lack the aforementioned canonicity property, because the ordinary judgmental equalities of intensional type theory do not apply to uses of the univalence axiom or paths in higher inductive types. Nor are they known to satisfy the weaker *homotopy canonicity* property that for any closed $M : \mathsf{bool}$ there exists a proof $P : \mathsf{Id}_{\mathsf{bool}}(M, \mathsf{true})$ or $P : \mathsf{Id}_{\mathsf{bool}}(M, \mathsf{false})$ [32].

1.1 Contributions

We define a two-level computational type theory satisfying the canonicity property, whose fibrant types include a cumulative hierarchy of univalent universes of fibrant types, universes of non-fibrant types, dependent function, dependent pair, and path types, and whose non-fibrant types include also exact equality types with equality reflection.

Our type theory is the first two-level type theory with canonicity, and the second univalent type theory with canonicity, after the cubical type theory of Cohen et al. [17]. Like Cohen et al. [17], our type theory is inspired by a model of homotopy type theory in cubical sets

[12], and represents n-dimensional cubes as terms parametrized by n variables ranging over a formal interval. However, the fibrant fragment of our type theory differs from Cohen et al. [17] by endowing the interval with less (namely, Cartesian) structure, and defining fibrancy with a substantially different uniform Kan condition. Thus we affirmatively resolve the open question of whether Cartesian interval structure constructively models univalence [18, 22].

In the spirit of Martin-Löf's meaning explanations [24], we define the judgments of type theory as relations on programs in an untyped programming language. In Section 2, we define a λ-calculus extended by nominal constants representing elements of a formal interval object [26]. In Section 3, we define a cubical generalization of Allen's partial equivalence relation (PER) semantics of Nuprl [1], sufficient to describe non-fibrant types and their elements at all dimensions. In Section 4, we define fibrant types as non-fibrant types equipped with two Kan operations, called coercion and homogeneous composition. In Sections 5 and 6 we summarize the semantics of each type former, and provide valid rules of inference. We conclude in Section 7 with comparisons to related work.

Full details and proofs for our construction are available in our associated preprint [7]. Our type theory is currently being implemented in the RedPRL proof assistant [28], in which we have already formalized a proof of univalence (`https://git.io/vFjUQ`).

2 Programming language

We begin by defining an untyped *cubical programming language*, a call-by-name λ-calculus extended by nominal constants [26], whose terms serve as the types and elements of our cubical type theory. Names (or dimensions) $x, y, \dots$ represent generic elements of an abstract interval $\mathbb{I}$ with two constant elements (or endpoints) $0, 1$. Given any two finite sets of names Ψ, Ψ', a *dimension substitution* $\psi : \Psi' \to \Psi$ sends each name in Ψ to 0, 1, or a name in Ψ'. We write $\langle r/x\rangle : \Psi \to (\Psi, x)$ for the dimension substitution sending x to $r \in \Psi \cup \{0,1\}$ and constant on Ψ. Given $\psi : \Psi' \to \Psi$ and a term M whose free names are contained in Ψ, we write $M\psi$ for the term obtained by replacing each $x \in \Psi$ in M with $\psi(x)$.

Geometrically, a term M with free dimension names in Ψ (henceforth, a Ψ-dimensional term) represents a $|\Psi|$-dimensional cube—a point ($|\Psi| = 0$), line ($|\Psi| = 1$), square ($|\Psi| = 2$), and so forth. Dimension substitutions are compositions of permutations, *face maps* $\langle 0/x\rangle, \langle 1/x\rangle : \Psi \to (\Psi, x)$, *diagonal maps* $\langle y/x\rangle : (\Psi, y) \to (\Psi, x, y)$, and (silent) *degeneracy maps* $(\Psi, y) \to \Psi$, and perform the corresponding geometric operation when applied to a term M. Below, we illustrate the faces of a square M in dimensions $\{x, y\}$; note that the bottom endpoint of the left face and the left endpoint of the bottom face are drawn as a single point, because $\langle 0/x\rangle\langle 1/y\rangle = \langle 1/y\rangle\langle 0/x\rangle$.

$$
y \downarrow \xrightarrow{x} \qquad
\begin{array}{ccc}
M\langle 0/x\rangle\langle 0/y\rangle & \xrightarrow{M\langle 0/y\rangle} & M\langle 1/x\rangle\langle 0/y\rangle \\
{\scriptstyle M\langle 0/x\rangle}\Big\downarrow & M & \Big\downarrow{\scriptstyle M\langle 1/x\rangle} \\
M\langle 0/x\rangle\langle 1/y\rangle & \xrightarrow[M\langle 1/y\rangle]{} & M\langle 1/x\rangle\langle 1/y\rangle
\end{array}
$$

This notion of cubes is *Cartesian* because sets of names and dimension substitutions form a free finite-product category generated by the two endpoint maps $\langle 0/x\rangle, \langle 1/x\rangle : \emptyset \to \{x\}$ [22, 9, 15]. In contrast, Cohen et al. [17] equip the interval with a De Morgan algebra structure also containing *connections* $\langle (x \wedge y)/y\rangle, \langle (x \vee y)/y\rangle : (\Psi, x, y) \to (\Psi, y)$ and *reversals* $\langle (1-y)/y \rangle : (\Psi, y) \to (\Psi, y)$. Cartesian cubes are appealing for their ubiquity and simplicity: dimensions behave like structural variables (with exchange, weakening, and contraction) and have a trivial equational theory (as opposed to De Morgan laws).

Following Martin-Löf's meaning explanations [24], we only give operational meaning to closed terms, and consider term variables to range over closed terms of their given types. However, we cannot treat dimension names as ranging only over $\{0, 1\}$—such a semantics would enforce *uniqueness of identity proofs*, by equating all lines whose boundaries coincide.

We therefore define a deterministic small-step operational semantics on terms with no free term variables, but any number of free dimension names. We write V val for values, $M \longmapsto M'$ when M takes one step of computation to M', and $M \Downarrow V$ (M evaluates to V), when $M \longmapsto^* V$ (in zero or more steps) and V val. Notably, the operational semantics are not stable under dimension substitution: because face and diagonal maps can expose new simplifications, we have neither (1) if V val then $V\psi$ val, nor (2) if $M \longmapsto^* M'$ then $M\psi \longmapsto^* M'\psi$. Consider the circle (Section 5.2), inductively generated by a point base and a line loop_x. We arrange that the faces of loop_x are base by including an operational step $(\mathsf{loop}_x)\langle 0/x \rangle = \mathsf{loop}_0 \longmapsto \mathsf{base}$. On the other hand, loop_x val because it is a constructor, contradicting (1). Maps out of the circle are determined by a point P (the image of base) and an abstracted line $x.L$ (the image of loop_x). Thus $\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(\mathsf{loop}_x; P, x.L) \longmapsto L$ but

$$\begin{aligned}(\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(\mathsf{loop}_x; P, x.L))\langle 0/x \rangle &= \mathbb{S}^1\text{-}\mathsf{elim}_{c.A\langle 0/x \rangle}(\mathsf{loop}_0; P\langle 0/x \rangle, x.L) \\ &\longmapsto \mathbb{S}^1\text{-}\mathsf{elim}_{c.A\langle 0/x \rangle}(\mathsf{base}; P\langle 0/x \rangle, x.L) \\ &\longmapsto P\langle 0/x \rangle\end{aligned}$$

where L and $P\langle 0/x \rangle$ are a priori unrelated, contradicting (2). Fortunately, most rules of the operational semantics are in fact *cubically stable*, or preserved by dimension substitutions: for instance, $(\mathsf{loop}_0)\psi \longmapsto \mathsf{base}\psi$ for all $\psi : \Psi' \to \Psi$. We write $M \longmapsto_{\square} M'$ when $M\psi \longmapsto M'\psi$ for all $\psi : \Psi' \to \Psi$, and V $\mathsf{val}_{\square}$ when $V\psi$ val for all $\psi : \Psi' \to \Psi$.

We include some operational semantics rules in Fig. 1, but omit the many rules pertaining to the Kan operations (defined in Section 4), as well as rules that evaluate the principal argument of an elimination form (for example, $\mathsf{app}(M, N) \longmapsto \mathsf{app}(M', N)$ when $M \longmapsto M'$). We adopt the convention that $a, b, c, \ldots$ are term variables, $x, y, z, \ldots$ are dimension names, and r, r', r_i are dimension expressions (names x or constants $0, 1$).

3 Cubical PER semantics

Type theory is built on the judgments of typehood (and equality of types) and membership in a type (and equality of members in a type). Intensional type theories—including homotopy type theory and the cubical type theory of Cohen et al. [17]—typically define these judgments inductively by a collection of syntactic inference rules. We instead define these judgments semantically as partial equivalence relations (PERs, or symmetric and transitive relations) over terms of the language described in Section 2. Such an approach can be seen as a mathematically precise reading of Martin-Löf's meaning explanations of type theory [24], or as a relational semantics of type theory in the style of Tait [27], and is the approach adopted by Nuprl [2]. The role of inference rules is therefore not definitional, but rather to summarize desirable properties validated by the semantics.

We adopt this semantical approach for multiple reasons. By defining types as relations over programs, we ensure the constructive character of the theory; for instance, it will follow from the definitions that elements of boolean type are programs that evaluate to true or false (Theorem 15). Moreover, because the meaning of open terms is given by their closed (term)

$$
\begin{array}{rl}
(a{:}A) \to B \ \mathsf{val}_{\square} & \\
\lambda a.M \ \mathsf{val}_{\square} & \\
\mathsf{app}(\lambda a.M, N) \longmapsto_{\square} & M[N/a] \\
(a{:}A) \times B \ \mathsf{val}_{\square} & \\
\langle M, N\rangle \ \mathsf{val}_{\square} & \\
\mathsf{fst}(\langle M, N\rangle) \longmapsto_{\square} & M \\
\mathsf{snd}(\langle M, N\rangle) \longmapsto_{\square} & N \\
\mathsf{Path}_{x.A}(M, N) \ \mathsf{val}_{\square} & \\
\langle x\rangle M \ \mathsf{val}_{\square} & \\
(\langle x\rangle M)@r \longmapsto_{\square} & M\langle r/x\rangle \\
\mathsf{Eq}_{A}(M, N) \ \mathsf{val}_{\square} & \\
\star \ \mathsf{val}_{\square} & \\
\mathsf{bool} \ \mathsf{val}_{\square} & \\
\mathsf{true} \ \mathsf{val}_{\square} & \\
\mathsf{false} \ \mathsf{val}_{\square} & \\
\mathsf{if}_{b.A}(\mathsf{true}; T, F) \longmapsto_{\square} & T
\end{array}
\qquad
\begin{array}{rl}
\mathsf{if}_{b.A}(\mathsf{false}; T, F) \longmapsto_{\square} & F \\
\mathbb{S}^1 \ \mathsf{val}_{\square} & \\
\mathsf{base} \ \mathsf{val}_{\square} & \\
\mathsf{loop}_x \ \mathsf{val} & \\
\mathsf{loop}_{\varepsilon} \longmapsto_{\square} & \mathsf{base} \quad (\varepsilon \in \{0,1\}) \\
\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(\mathsf{base}; P, x.L) \longmapsto_{\square} & P \\
\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(\mathsf{loop}_x; P, y.L) \longmapsto & L\langle x/y\rangle \\
\mathsf{V}_x(A, B, E) \ \mathsf{val} & \\
\mathsf{V}_{\varepsilon}(A_0, A_1, E) \longmapsto_{\square} & A_{\varepsilon} \quad (\varepsilon \in \{0,1\}) \\
\mathsf{Vin}_x(M, N) \ \mathsf{val} & \\
\mathsf{Vin}_{\varepsilon}(M_0, M_1) \longmapsto_{\square} & M_{\varepsilon} \quad (\varepsilon \in \{0,1\}) \\
\mathsf{Vproj}_x(\mathsf{Vin}_x(M, N), F) \longmapsto & N \\
\mathsf{Vproj}_0(M, F) \longmapsto_{\square} & \mathsf{app}(F, M) \\
\mathsf{Vproj}_1(M, F) \longmapsto_{\square} & M \\
\mathcal{U}_j^{\kappa} \ \mathsf{val}_{\square} & (\kappa \in \{\mathsf{pre}, \mathsf{Kan}\})
\end{array}
$$

Figure 1 Operational semantics, selected rules.

substitution instances, it will naturally follow that judgmental equality is extensional and that the exact equality type satisfies equality reflection.

In Allen's PER semantics of Nuprl [1], a type A is interpreted as a symmetric and transitive relation $[\![A]\!]$ on values; the judgment $M \doteq N \in A$ holds whenever $M \Downarrow M_0$, $N \Downarrow N_0$, and $[\![A]\!](M_0, N_0)$ (which we henceforth write $[\![A]\!]^{\Downarrow}(M, N)$); and $M \in A$ whenever $M \doteq M \in A$. Thus, ignoring equality, A is defined by its set of values $\{V \ \mathsf{val} \mid [\![A]\!](V, V)\}$, and the elements of A are the programs whose values are elements of that set. (We write $\in$ rather than $:$ to emphasize the semantic character of these judgments.)

We generalize Nuprl's semantics by instead interpreting types as *cubical sets*: every type has a PER of Ψ-dimensional values for every Ψ, and each $\psi : \Psi' \to \Psi$ sends its Ψ-dimensional values to its Ψ'-dimensional values. Complications arise when defining the latter functorial action. First, dimension substitutions can engender computation even on values, so the action of ψ must send V to the *value* of the program $V\psi$. Second, substitution-then-evaluation is not necessarily functorial: if $V\psi \Downarrow V'$, there is in general no relationship between the values of $V\psi\psi'$ and $V'\psi'$. Third, types are themselves programs because of dependency, and therefore suffer from the same coherence issues. We solve these issues by interpreting (Ψ-dimensional) types as *value-coherent* Ψ*-PERs on values*:

▶ **Definition 1.** A Ψ*-relation* α (resp., Ψ-relation on values) is a family of binary relations α_{ψ} for every Ψ' and $\psi : \Psi' \to \Psi$, over Ψ'-dimensional terms (resp., values). If α_{ψ} varies only in the choice of Ψ' and not ψ, we say α is *context-indexed* and write $\alpha_{\Psi'}$ for α_{ψ}.

▶ **Definition 2.** For any Ψ-relation on values α, define the Ψ-relation $\mathsf{Tm}(\alpha)(M, N)$ to hold when for all $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$, $\alpha^{\Downarrow}_{\psi_1\psi_2}$ relates pairwise $M_1\psi_2$, $M\psi_1\psi_2$, $N_1\psi_2$, and $N\psi_1\psi_2$, where $M\psi_1 \Downarrow M_1$ and $N\psi_1 \Downarrow N_1$.

A Ψ-relation α can be precomposed with a dimension substitution $\psi : \Psi' \to \Psi$, yielding a Ψ'-relation $(\alpha\psi)_{\psi'} := \alpha_{\psi\psi'}$.

▶ **Definition 3.** A Ψ-relation on values α is *value-coherent*, or $\mathsf{Coh}(\alpha)$, when for all $\psi : \Psi' \to \Psi$, if $\alpha_\psi(V, V')$ then $\mathsf{Tm}(\alpha\psi)(V, V')$.

Definition 1 captures the idea that types vary with dimension substitutions (for example, $\mathbb{S}^1\text{-}\mathsf{elim}_{c.\mathcal{U}_j^{\mathsf{Kan}}}(\mathsf{loop}_x; A, x.B)$ under $\langle 0/x\rangle$), Definition 2 lifts Ψ-relations on values to arbitrary terms by substitution-then-evaluation, and Definition 3 defines functoriality of that lifting.

▶ Remark. Writing $\mathbb{C}$ for the category of finite sets of names and dimension substitutions, a value-coherent context-indexed PER determines a functor $\mathbb{C}^{\mathsf{op}} \to \mathbf{Set}$, and a value-coherent Ψ-PER determines a functor $(\mathbb{C}/\Psi)^{\mathsf{op}} \to \mathbf{Set}$.

3.1 Judgments

We define the judgments of our type theory relative to a value-coherent context-indexed PER of types, each of which gives rise to another PER. In the style of Allen [1] and recently, Anand and Rahli [4], we present this data in a single relation.

▶ **Definition 4.** A *cubical type system* is a relation $\tau(\Psi, A_0, B_0, \varphi)$ over Ψ-dimensional values A_0, B_0, and binary relations φ over Ψ-dimensional values, satisfying:

- Functionality: if $\tau(\Psi, A_0, B_0, \varphi)$ and $\tau(\Psi, A_0, B_0, \varphi')$ then $\varphi = \varphi'$.
- PER-valuation: if $\tau(\Psi, A_0, B_0, \varphi)$ then φ is a PER.
- Symmetry: if $\tau(\Psi, A_0, B_0, \varphi)$ then $\tau(\Psi, B_0, A_0, \varphi)$.
- Transitivity: if $\tau(\Psi, A_0, B_0, \varphi)$ and $\tau(\Psi, B_0, C_0, \varphi)$ then $\tau(\Psi, A_0, C_0, \varphi)$.
- Value-coherence: $\mathsf{Coh}(\{(\Psi, A_0, B_0) \mid \tau(\Psi, A_0, B_0, \varphi)\})$.

The first three components of τ define a Ψ-PER for every Ψ, which we write τ^Ψ. If $\mathsf{Tm}(\tau^\Psi)(A, B)$, then the fourth component of τ assigns a Ψ-PER to A, B sending each $\psi : \Psi' \to \Psi$ to the relation φ^ψ where $\tau^\Downarrow(\Psi', A\psi, B\psi, \varphi^\psi)$. We write this Ψ-PER $[\![A]\!]$; it is unique by functionality, and independent from the choice of B by symmetry and transitivity.

For the remainder of this section, fix a cubical type system τ. We start by defining the closed judgments relative to τ: when are A and B equal Ψ-dimensional types, and when are M and N equal Ψ-dimensional elements of A?

▶ **Definition 5.** $A \doteq B\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$ holds when $\mathsf{Tm}(\tau^\Psi)(A, B)$ and $\mathsf{Coh}([\![A]\!])$. We write $A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$ for $A \doteq A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$.

▶ **Definition 6.** $M \doteq N \in A\ [\Psi]$, presupposing[2] $A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$, when $\mathsf{Tm}([\![A]\!])(M, N)$. We write $M \in A\ [\Psi]$ for $M \doteq M \in A\ [\Psi]$.

We extend the judgments to open terms by functionality: an open type (resp., elements) is a map sending equal elements of the context to equal closed types (resp., elements). The open judgments must be defined simultaneously, by induction on the length of the context.

▶ **Definition 7.** $(a_1 : A_1, \ldots, a_n : A_n)\ \mathsf{ctx}\ [\Psi]$ when $A_1\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$, $a_1 : A_1 \gg A_2\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$, $\ldots$, and $a_1 : A_1, \ldots, a_{n-1} : A_{n-1} \gg A_n\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$.

[2] A presupposition is a fact that must be established before a judgment can be sensibly considered. Here, it does not make sense to demand $\mathsf{Tm}([\![A]\!])(M, N)$ unless $[\![A]\!]$ is known to exist by $A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$.

▶ **Definition 8.** $a_1 : A_1, \ldots, a_n : A_n \gg B \doteq B'$ $\mathsf{type_{pre}}$ $[\Psi]$, presupposing $(a_1 : A_1, \ldots, a_n : A_n)$ ctx $[\Psi]$, when for any $\psi : \Psi' \to \Psi$, $N_1 \doteq N_1' \in A_1\psi$ $[\Psi']$, $N_2 \doteq N_2' \in A_2\psi[N_1/a_1]$ $[\Psi']$, $\ldots$, and $N_n \doteq N_n' \in A_n\psi[N_1, \ldots, N_{n-1}/a_1, \ldots, a_n]$ $[\Psi']$, when

$$B\psi[N_1, \ldots, N_n/a_1, \ldots, a_n] \doteq B'\psi[N_1', \ldots, N_n'/a_1, \ldots, a_n] \ \mathsf{type_{pre}} \ [\Psi'].$$

Under the same hypotheses, $a_1 : A_1, \ldots, a_n : A_n \gg M \doteq M' \in B$ $[\Psi]$ when

$$M\psi[N_1, \ldots, N_n/a_1, \ldots, a_n] \doteq M'\psi[N_1', \ldots, N_n'/a_1, \ldots, a_n] \in B\psi[N_1, \ldots, N_n/a_1, \ldots, a_n] \ [\Psi'].$$

Given the distinct roles of term variables and dimension names in Definition 8, it is natural for our judgments to separate the contexts $(a_1 : A_1, \ldots, a_n : A_n)$ and Ψ. In RedPRL, we utilize a single mixed context of terms and dimensions, as do Cohen et al. [17].

▶ Remark. Allen's PER semantics are an instance of our semantics, in the case that types are constant presheaves and terms have no free dimension names. If M, N, A, and B have no free dimensions, then $A \doteq B$ $\mathsf{type_{pre}}$ $[\Psi]$ if and only if $\tau^{\Downarrow}(\Psi', A, B, [\![A]\!]_{\Psi'})$ for all Ψ', and $M \doteq N \in A$ $[\Psi]$ if and only if $([\![A]\!]_{\Psi'})^{\Downarrow}(M, N)$ for all Ψ'.

3.2 Properties of Judgments

The main result of this paper is the construction of a cubical type system closed under a variety of type formers. However, many global properties of judgments hold in *any* cubical type system. For instance, equality judgments are all symmetric, transitive, and closed under dimension substitution (if $\mathcal{J}$ $[\Psi]$ and $\psi : \Psi' \to \Psi$, then $\mathcal{J}\psi$ $[\Psi']$). Open judgments satisfy the hypothesis (if $(\Gamma, a : A, \Gamma')$ ctx $[\Psi]$ then $\Gamma, a : A, \Gamma' \gg a \in A$ $[\Psi]$) and weakening rules. Equal types have the same elements (if $A \doteq B$ $\mathsf{type_{pre}}$ $[\Psi]$ and $M \doteq N \in A$ $[\Psi]$ then $M \doteq N \in B$ $[\Psi]$).

To prove $M \in A$ $[\Psi]$ in a particular cubical type system, we must compare the definition of $[\![A]\!]$ with the evaluation behavior of all dimension substitution instances of M. When all instances of M begin to evaluate in lockstep, it suffices to consider only M itself (Lemma 9); otherwise, it suffices to show that the instances of M become coherent up to equality at A, after some number of steps (Lemma 10).

▶ **Lemma 9** (Head expansion). *If $M' \in A$ $[\Psi]$ and $M \longmapsto^*_{\square} M'$, then $M \doteq M' \in A$ $[\Psi]$.*

▶ **Lemma 10.** *Suppose that M is a Ψ-dimensional term, and we have a family of terms $\{M_\psi\}$ for each $\psi : \Psi' \to \Psi$ such that $M\psi \longmapsto^* M_\psi$. If $M_\psi \doteq (M_{\mathsf{id}_\Psi})\psi \in A\psi$ $[\Psi']$ for all ψ, then $M \doteq M_{\mathsf{id}_\Psi} \in A$ $[\Psi]$.*

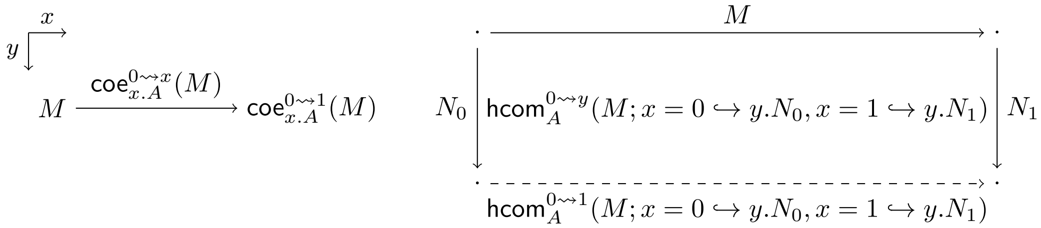
Once we have established that substitution-then-evaluation of M is functorial, it follows that the instances of M are equal to the instances of its value.

▶ **Lemma 11.** *If $M \in A$ $[\Psi]$, then $M \Downarrow V$ and $M \doteq V \in A$ $[\Psi]$.*

On the other hand, certain properties typical of intensional type theories are generally *not* expected to hold in our semantics. To check $M \in A$ $[\Psi]$, one must, at minimum, show that M terminates; this is clearly undecidable, because M can be an arbitrary untyped term. Moreover, terms do not have unique types, because the meanings of types need not be disjoint. In fact, modern Nuprl has a "Base" type containing every term [4].

4 Kan types

The judgmental apparatus described in Section 3 accounts for *non-fibrant* or *pre*types—whose paths are not necessarily composable or invertible. A pretype is *Kan fibrant*, or a Kan type, when equipped with two *Kan operations*: coercion (coe) and homogeneous composition (hcom). Coercion for a (Ψ, x)-dimensional type states that elements of $A\langle r/x\rangle$ can be coerced to $A\langle r'/x\rangle$ for any r, r', and this operation is the identity when $r = r'$. The coercion of M is written $\mathsf{coe}^{r\rightsquigarrow r'}_{x.A}(M)$. For example, if $M \in A\langle 0/x\rangle\ [\emptyset]$, then $\mathsf{coe}^{0\rightsquigarrow 1}_{x.A}(M) \in A\langle 1/x\rangle\ [\emptyset]$. Moreover, $\mathsf{coe}^{0\rightsquigarrow x}_{x.A}(M) \in A\ [x]$ is a line in A whose $\langle 0/x\rangle$ face is M (because $0 = x\langle 0/x\rangle$), and whose $\langle 1/x\rangle$ face is $\mathsf{coe}^{0\rightsquigarrow 1}_{x.A}(M)$.



Homogeneous composition is significantly more complicated, but essentially states that any open box in A (an n-cube without an interior or one of its faces) has a composite (the missing face). For example, given two lines in y, $N_0 \in A\langle 0/x\rangle\ [y]$ and $N_1 \in A\langle 1/x\rangle\ [y]$, and a line in x, $M \in A\ [x]$, that agrees with the y-lines when $y = 0$ ($M\langle 0/x\rangle \doteq N_\varepsilon \in A\langle \varepsilon/x\rangle\ [\emptyset]$ for $\varepsilon \in \{0, 1\}$), we can obtain an x-line that agrees with the y-lines when $y = 1$, written $\mathsf{hcom}^{0\rightsquigarrow 1}_{A}(M; x = 0 \hookrightarrow y.N_0, x = 1 \hookrightarrow y.N_1)$. Moreover, we can obtain the interior of that square, its *filler*, by composing to y rather than 1. The difficulty of homogeneous composition is that we must define arbitrary open boxes, at any dimension, in a manner that commutes with substitution. We introduce *dimension context restrictions* Ξ, or sets of pairs of dimension expressions (suggestively written as equations), to describe the spatial relationship between the faces of an open box.

▶ **Definition 12.** A context restriction $\overrightarrow{r_i = r'_i}$ is *valid* in Ψ when all r_i, r'_i are dimension expressions in Ψ, and either $r_i = r'_i$ for some i, or $r_i = r_j$, $r'_i = 0$, and $r'_j = 1$ for some i, j.

▶ **Definition 13.** A restricted judgment $\mathcal{J}\ [\Psi \mid \overrightarrow{r_i = r'_i}]$ holds when $\mathcal{J}\psi\ [\Psi']$ holds for every $\psi : \Psi' \to \Psi$ for which $r_i\psi = r'_i\psi$ for all i.

Restricted judgments behave as one might expect: $\mathcal{J}\ [\Psi \mid \emptyset]$ if and only if $\mathcal{J}\ [\Psi]$, $\mathcal{J}\ [\Psi, x \mid x = 0]$ if and only if $\mathcal{J}\langle 0/x\rangle\ [\Psi]$, and $\mathcal{J}\ [\Psi \mid 0 = 1]$ always. Crucially, they are closed under dimension substitution: if $\mathcal{J}\ [\Psi \mid \Xi]$ and $\psi : \Psi' \to \Psi$, then $\mathcal{J}\psi\ [\Psi' \mid \Xi\psi]$.

▶ **Definition 14.** $B\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]$, presupposing $B\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$, when for all $\psi : \Psi' \to \Psi$, the rules in Fig. 2 hold for $A := B\psi$. ($B \doteq B'\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]$, presupposing $B \doteq B'\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$, when B and B' are equipped with equal Kan operations.)

Operationally, both hcom and coe evaluate their type argument and behave according to the outermost type former. For each type former, we will first show that the formation, introduction, elimination, computation, and eta rules hold; then, using those rules, we show that if its component types are Kan, then it is Kan (for example, if $A\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]$ and $a : A \gg B\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]$, then $(a{:}A) \to B\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]$). The only exceptions are exact equality types $\mathsf{Eq}_A(M, N)$ (Section 5.5), which are not generally Kan even when A is Kan.

$$\frac{\begin{array}{ll} & \overrightarrow{r_i = r_i'}\ \mathsf{valid}\ [\Psi] \\ & A\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi] \\ & M \in A\ [\Psi] \\ (\forall i, j) & N_i \doteq N_j \in A\ [\Psi, y \mid r_i = r_i', r_j = r_j'] \\ (\forall i) & N_i\langle r/y\rangle \doteq M \in A\ [\Psi \mid r_i = r_i'] \end{array}}{\begin{array}{l} \mathsf{hcom}_A^{r \rightsquigarrow r'}(M; \overrightarrow{r_i = r_i' \hookrightarrow y.N_i}) \in A\ [\Psi] \\ \doteq \begin{cases} M & \text{when } r = r' \\ N_i\langle r'/y\rangle & \text{when } r_i = r_i' \end{cases} \end{array}}$$

$$\frac{A\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi, x] \qquad M \in A\langle r/x\rangle\ [\Psi]}{\begin{array}{c} \mathsf{coe}_{x.A}^{r \rightsquigarrow r'}(M) \in A\langle r'/x\rangle\ [\Psi] \\ \mathsf{coe}_{x.A}^{r \rightsquigarrow r}(M) \doteq M \in A\langle r/x\rangle\ [\Psi] \end{array}}$$

Figure 2 Kan operations.

These Kan operations are variants of the uniform Kan conditions first proposed by Bezem et al. [12]. In unpublished work in 2014, Licata and Brunerie [22] and Coquand [18] considered uniform Kan operations in Cartesian cubical sets, but did not succeed in defining univalent type theories based on those operations. Our Kan operations introduce two important innovations. First, we allow open boxes with sides attached along *diagonals* $x = z$, in addition to faces; this is essential to construct univalent universes (Sections 5.6 and 6). Second, the validity condition requires that every box must contain at least one opposing pair of sides $x = 0$ and $x = 1$; this sharpens our canonicity results for higher inductive types (Section 5.2). We defer further comparison of Kan operations to Section 7.

5 Type formers

We proceed to construct a cubical type system with booleans and the circle (as a representative higher inductive type), and closed under dependent function and pair types, path types, exact equality types, and univalent universes. (Our preprint [7] also includes an empty type and natural numbers.) Each of these type formers is given meaning as a value-coherent Ψ-PER on values, and shown to validate the appropriate rules of inference. (We focus on closed-term rules, from which the open rules follow.) In this section we analyze each type former separately, excepting pretype and Kan universes, which we defer to Section 6.

5.1 Booleans

There are two boolean values at every dimension: $[\![\mathsf{bool}]\!]_\Psi = \{(\mathsf{true}, \mathsf{true}), (\mathsf{false}, \mathsf{false})\}$. This context-indexed PER is clearly value-coherent, as the constructors are unaffected by dimension substitution. The canonicity property follows directly from this definition:

▶ **Theorem 15** (Canonicity). *If $M \in \mathsf{bool}\ [\Psi]$ then $M \Downarrow V$ and $M \doteq V \in \mathsf{bool}\ [\Psi]$, for $V = \mathsf{true}$ or $V = \mathsf{false}$.*

Proof. Then $\mathsf{Tm}([\![\mathsf{bool}]\!])(M, M)$, so $M \Downarrow V$ and $[\![\mathsf{bool}]\!](V, V)$. By Lemma 11, $M \doteq V \in \mathsf{bool}\ [\Psi]$, and by the definition of $[\![\mathsf{bool}]\!]$, $V = \mathsf{true}$ or $V = \mathsf{false}$. ◀

Consistency is similar: $\mathsf{true} \doteq \mathsf{false} \in \mathsf{bool}\ [\Psi]$ implies $[\![\mathsf{bool}]\!](\mathsf{true}, \mathsf{false})$, which is impossible.

The rules in Fig. 3 all hold: true and false are elements, the elimination rule holds essentially by Theorem 15, and the computation rules hold by Lemma 9. The Kan operations of bool are identity functions, because every line in bool is degenerate.

$$\frac{}{\mathsf{bool}\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]} \qquad \frac{}{\mathsf{true} \in \mathsf{bool}\ [\Psi]} \qquad \frac{}{\mathsf{false} \in \mathsf{bool}\ [\Psi]}$$

$$\frac{b : \mathsf{bool} \gg A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi] \quad M \in \mathsf{bool}\ [\Psi] \quad T \in A[\mathsf{true}/b]\ [\Psi] \quad F \in A[\mathsf{false}/b]\ [\Psi]}{\mathsf{if}(M; T, F) \in A[M/b]\ [\Psi]}$$

$$\frac{T \in A\ [\Psi]}{\mathsf{if}(\mathsf{true}; T, F) \doteq T \in A\ [\Psi]} \qquad \frac{F \in A\ [\Psi]}{\mathsf{if}(\mathsf{false}; T, F) \doteq F \in A\ [\Psi]}$$

$$\frac{}{\mathbb{S}^1\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]} \qquad \frac{}{\mathsf{base} \in \mathbb{S}^1\ [\Psi]} \qquad \frac{}{\mathsf{loop}_r \in \mathbb{S}^1\ [\Psi]} \qquad \frac{}{\mathsf{loop}_\varepsilon \doteq \mathsf{base} \in \mathbb{S}^1\ [\Psi]}$$

$$\frac{\begin{array}{ccc} & c : \mathbb{S}^1 \gg A\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi] & M \in \mathbb{S}^1\ [\Psi] \\ P \in A[\mathsf{base}/c]\ [\Psi] & L \in A[\mathsf{loop}_x/c]\ [\Psi, x] & (\forall \varepsilon)\ L\langle \varepsilon/x \rangle \doteq P \in A[\mathsf{base}/c]\ [\Psi] \end{array}}{\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(M; P, x.L) \in A[M/c]\ [\Psi]}$$

$$\frac{P \in B\ [\Psi]}{\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(\mathsf{base}; P, x.L) \doteq P \in B\ [\Psi]} \qquad \frac{L \in B\ [\Psi, x] \quad (\forall \varepsilon)\ L\langle \varepsilon/x \rangle \doteq P \in B\langle \varepsilon/x \rangle\ [\Psi]}{\mathbb{S}^1\text{-}\mathsf{elim}_{c.A}(\mathsf{loop}_r; P, x.L) \doteq L\langle r/x \rangle \in B\langle r/x \rangle\ [\Psi]}$$

Figure 3 Boolean and circle type.

5.2 Circle

It is tempting to define the circle as the least context-indexed PER generated by a base point and a loop: $[\![\mathbb{S}^1]\!]_\Psi(\mathsf{base}, \mathsf{base})$ and $[\![\mathbb{S}^1]\!]_{(\Psi,x)}(\mathsf{loop}_x, \mathsf{loop}_x)$. Unlike bool, $\mathbb{S}^1$ has non-degenerate lines, so we must explicitly add composites of open boxes to $\mathbb{S}^1$ if we want it to be Kan. We therefore equip $\mathbb{S}^1$ with the following *free* Kan structure (writing ξ_i to abbreviate $r_i = r_i'$):

$$\begin{array}{rll} \mathsf{coe}^{r \rightsquigarrow r'}_{x.\mathbb{S}^1}(M) & \longmapsto_{\square} M & \\ \mathsf{hcom}^{r \rightsquigarrow r'}_{\mathbb{S}^1}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i}) & \longmapsto_{\square} M & \text{if } r = r' \\ \mathsf{hcom}^{r \rightsquigarrow r'}_{\mathbb{S}^1}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i}) & \longmapsto N_j\langle r'/y \rangle & \text{if } r \neq r',\ r_j = r_j',\ r_i \neq r_i' \text{ for } i < j \\ \mathsf{hcom}^{r \rightsquigarrow r'}_{\mathbb{S}^1}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i})\ \ \mathsf{val} & & \text{if } r \neq r',\ r_i \neq r_i' \end{array}$$

These operational semantics satisfy the equations in Fig. 2: when $r = r'$ in hcom, line (2) applies; when $r_i = r_i'$, line (3) applies; and for every hcom, one of lines (2–4) applies. Disequalities are needed in lines (3–4) to maintain determinacy. To account for value hcoms, we add a clause that $[\![\mathbb{S}^1]\!]_\Psi(\mathsf{hcom}^{r \rightsquigarrow r'}_{\mathbb{S}^1}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i}), \mathsf{hcom}^{r \rightsquigarrow r'}_{\mathbb{S}^1}(M'; \overrightarrow{\xi_i \hookrightarrow y.N_i'}))$ whenever these are values and satisfy the premises of the hcom rule in Fig. 2. Value-coherence of $[\![\mathbb{S}^1]\!]$ follows from the operational semantics of $\mathsf{hcom}_{\mathbb{S}^1}$ and the premises of the hcom typing rule. By limiting the Kan operations to valid context restrictions, we ensure that $[\![\mathbb{S}^1]\!]_\emptyset$ contains no hcoms—there are no valid restrictions at dimension $\emptyset$ in which $r_i \neq r_i'$ for all i.

The rules for the circle can be found in Fig. 3, including the eliminator mapping from $\mathbb{S}^1$ into any Kan type with a point P and line $x.L$ satisfying $L\langle 0/x \rangle \doteq L\langle 1/x \rangle \doteq P$. The eliminator sends base to P, loop_y to $L\langle y/x \rangle$, and $\mathsf{hcom}_{\mathbb{S}^1}$ to a Kan composition in the target type. (See our preprint [7] for the latter operational semantics step, which requires a derived notion of *heterogeneous* composition in which the type varies across the open box.) It is therefore essential that the target type is Kan.

5.3 Dependent function and pair types

When A $\mathsf{type_{pre}}$ $[\Psi]$ and $a : A \gg B$ $\mathsf{type_{pre}}$ $[\Psi]$,

$$\llbracket (a{:}A) \to B \rrbracket_\psi = \{(\lambda a.N, \lambda a.N') \mid a : A\psi \gg N \doteq N' \in B\psi\ [\Psi']\}$$
$$\llbracket (a{:}A) \times B \rrbracket_\psi = \{(\langle M, N\rangle, \langle M', N'\rangle) \mid M \doteq M' \in A\psi\ [\Psi'] \wedge N \doteq N' \in B\psi[M/a]\ [\Psi']\}$$

Rules for dependent function and dependent pair types are listed in Fig. 4, including judgmental η principles. The Kan operations for dependent function types are:

$$\mathsf{hcom}^{r \rightsquigarrow r'}_{(a:A)\to B}(M; \overrightarrow{\xi_i \hookrightarrow y.N_i}) \longmapsto_{\square} \lambda a.\mathsf{hcom}^{r \rightsquigarrow r'}_{B}(\mathsf{app}(M, a); \overrightarrow{\xi_i \hookrightarrow y.\mathsf{app}(N_i, a)})$$
$$\mathsf{coe}^{r \rightsquigarrow r'}_{x.(a:A)\to B}(M) \longmapsto_{\square} \lambda a.\mathsf{coe}^{r \rightsquigarrow r'}_{x.B[\mathsf{coe}^{r' \rightsquigarrow x}_{x.A}(a)/a]}(\mathsf{app}(M, \mathsf{coe}^{r' \rightsquigarrow r}_{x.A}(a)))$$

If A $\mathsf{type_{Kan}}$ $[\Psi]$ and $a : A \gg B$ $\mathsf{type_{Kan}}$ $[\Psi]$, then by the above steps and the introduction, elimination, and eta rules, $(a{:}A) \to B$ $\mathsf{type_{Kan}}$ $[\Psi]$ (and similarly [7], $(a{:}A) \times B$ $\mathsf{type_{Kan}}$ $[\Psi]$).

5.4 Path types

Whenever A $\mathsf{type_{pre}}$ $[\Psi, x]$ and $P_\varepsilon \doteq P'_\varepsilon \in A\langle\varepsilon/x\rangle$ $[\Psi]$ for $\varepsilon \in \{0, 1\}$, $\llbracket \mathsf{Path}_{x.A}(P_0, P_1) \rrbracket_\psi = \{(\langle x\rangle M, \langle x\rangle M') \mid M \doteq M' \in A\psi\ [\Psi', x] \wedge \forall\varepsilon.(M\langle\varepsilon/x\rangle \doteq P_\varepsilon\psi \in A\psi\langle\varepsilon/x\rangle\ [\Psi'])\}$. That is, paths are abstracted lines with specified endpoints, and dimension abstraction ($\langle x\rangle M$) and application ($M@r$) pack and unpack them. Rules for path types are listed in Fig. 4; once again, Kan operations (see [7]) ensure that $\mathsf{Path}_{x.A}(P_0, P_1)$ $\mathsf{type_{Kan}}$ $[\Psi]$ when A $\mathsf{type_{Kan}}$ $[\Psi, x]$.

Notably, while homotopy type theory relies on the identity type to generate path structure, in this setting the path type merely internalizes a preexisting judgmental notion of paths. The homotopy-type-theoretic identity elimination principle is definable for $\mathsf{Path}_{\_.A}(M, N)$ when A is Kan, but as in Cohen et al. [17], its computation rule holds only up to a path.

5.5 Exact equality types

Whenever A $\mathsf{type_{pre}}$ $[\Psi]$, $M \in A$ $[\Psi]$, and $N \in A$ $[\Psi]$, we have $\llbracket \mathsf{Eq}_A(M, N) \rrbracket_\psi = \{(\star, \star) \mid M\psi \doteq N\psi \in A\psi\ [\Psi']\}$. That is, $\mathsf{Eq}_A(M, N)$ is (uniquely) inhabited if and only if $M \doteq N \in A$ $[\Psi]$, and therefore equality reflection holds. Rules for equality types are listed in Fig. 4.

Unlike the previous cases, $\mathsf{Eq}_A(M, N)$ is not necessarily Kan when A is Kan, because coercion in $\mathsf{Eq}_A(M, N)$ implies uniqueness of identity proofs in A. We allow $\mathsf{Eq}_A(M, N)$ $\mathsf{type_{Kan}}$ $[\Psi]$ when A is *discrete Kan* [7], roughly, contains only degenerate paths (for example, $A = \mathsf{bool}$).

5.6 Univalence

Voevodsky's univalence axiom [31] concerns a notion of *type equivalence* $\mathsf{Equiv}(A, B)$:

$$\mathsf{isContr}(C) := C \times ((c{:}C) \to (c'{:}C) \to \mathsf{Path}_{\_.C}(c, c'))$$
$$\mathsf{Equiv}(A, B) := (f{:}A \to B) \times ((b{:}B) \to \mathsf{isContr}((a{:}A) \times \mathsf{Path}_{\_.B}(\mathsf{app}(f, a), b)))$$

Essentially, $\mathsf{Equiv}(A, B)$ if there is a map $A \to B$ such that the (homotopy) preimage in A of any point in B is contractible (has exactly one point up to homotopy). In homotopy type theory, univalence states that $\mathsf{idtoequiv} : \mathsf{Id}_{\mathcal{U}}(A, B) \to \mathsf{Equiv}(A, B)$ (definable in intensional type theory) is itself an equivalence. By a theorem of Licata [21], univalence in the present setting is equivalent to the existence of a map $\mathsf{ua} : \mathsf{Equiv}(A, B) \to \mathsf{Path}_{\_.\mathcal{U}^{\mathsf{Kan}}_j}(A, B)$ and a homotopy $\mathsf{ua}_\beta(E)$ between the functions underlying the equivalences E and $\mathsf{idtoequiv}(\mathsf{ua}(E))$.

$$\frac{A\ \mathsf{type}_\kappa\ [\Psi] \qquad a : A \gg B\ \mathsf{type}_\kappa\ [\Psi]}{(a{:}A) \to B\ \mathsf{type}_\kappa\ [\Psi]} \qquad \frac{a : A \gg M \in B\ [\Psi]}{\lambda a.M \in (a{:}A) \to B\ [\Psi]}$$

$$\frac{M \in (a{:}A) \to B\ [\Psi] \qquad N \in A\ [\Psi]}{\mathsf{app}(M,N) \in B[N/a]\ [\Psi]} \qquad \frac{a : A \gg M \in B\ [\Psi] \qquad N \in A\ [\Psi]}{\mathsf{app}(\lambda a.M, N) \doteq M[N/a] \in B[N/a]\ [\Psi]}$$

$$\frac{M \in (a{:}A) \to B\ [\Psi]}{M \doteq \lambda a.\mathsf{app}(M,a) \in (a{:}A) \to B\ [\Psi]}$$

$$\frac{A\ \mathsf{type}_\kappa\ [\Psi] \qquad a : A \gg B\ \mathsf{type}_\kappa\ [\Psi]}{(a{:}A) \times B\ \mathsf{type}_\kappa\ [\Psi]} \qquad \frac{M \in A\ [\Psi] \qquad N \in B[M/a]\ [\Psi]}{\langle M,N \rangle \in (a{:}A) \times B\ [\Psi]}$$

$$\frac{P \in (a{:}A) \times B\ [\Psi]}{\mathsf{fst}(P) \in A\ [\Psi]} \qquad \frac{P \in (a{:}A) \times B\ [\Psi]}{\mathsf{snd}(P) \in B[\mathsf{fst}(P)/a]\ [\Psi]} \qquad \frac{M \in A\ [\Psi]}{\mathsf{fst}(\langle M,N \rangle) \doteq M \in A\ [\Psi]}$$

$$\frac{N \in B\ [\Psi]}{\mathsf{snd}(\langle M,N \rangle) \doteq N \in B\ [\Psi]} \qquad \frac{P \in (a{:}A) \times B\ [\Psi]}{P \doteq \langle \mathsf{fst}(P), \mathsf{snd}(P) \rangle \in (a{:}A) \times B\ [\Psi]}$$

$$\frac{\begin{array}{c} A\ \mathsf{type}_\kappa\ [\Psi, x] \\ (\forall \varepsilon) \quad P_\varepsilon \in A\langle \varepsilon/x \rangle\ [\Psi] \end{array}}{\mathsf{Path}_{x.A}(P_0,P_1)\ \mathsf{type}_\kappa\ [\Psi]} \qquad \frac{\begin{array}{c} M \in A\ [\Psi, x] \\ (\forall \varepsilon) \quad M\langle \varepsilon/x \rangle \doteq P_\varepsilon \in A\langle \varepsilon/x \rangle\ [\Psi] \end{array}}{\langle x \rangle M \in \mathsf{Path}_{x.A}(P_0,P_1)\ [\Psi]}$$

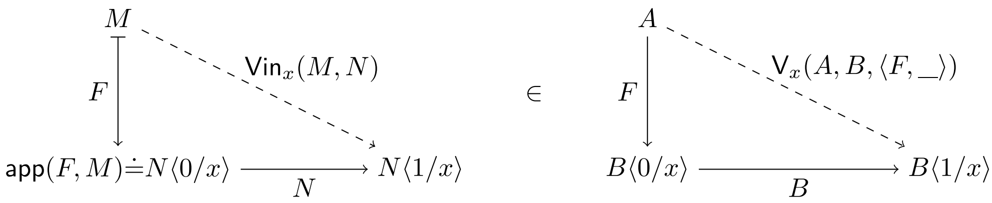
$$\frac{M \in \mathsf{Path}_{x.A}(P_0,P_1)\ [\Psi]}{M@r \in A\langle r/x \rangle\ [\Psi]} \qquad \frac{M \in \mathsf{Path}_{x.A}(P_0,P_1)\ [\Psi]}{M@\varepsilon \doteq P_\varepsilon \in A\langle \varepsilon/x \rangle\ [\Psi]}$$

$$\frac{M \in A\ [\Psi, x]}{(\langle x \rangle M)@r \doteq M\langle r/x \rangle \in A\langle r/x \rangle\ [\Psi]} \qquad \frac{M \in \mathsf{Path}_{x.A}(P_0,P_1)\ [\Psi]}{M \doteq \langle x \rangle (M@x) \in \mathsf{Path}_{x.A}(P_0,P_1)\ [\Psi]}$$

$$\frac{A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi] \qquad M \in A\ [\Psi] \qquad N \in A\ [\Psi]}{\mathsf{Eq}_A(M,N)\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]} \qquad \frac{M \doteq N \in A\ [\Psi]}{\star \in \mathsf{Eq}_A(M,N)\ [\Psi]}$$

$$\frac{E \in \mathsf{Eq}_A(M,N)\ [\Psi]}{M \doteq N \in A\ [\Psi]} \qquad \frac{E \in \mathsf{Eq}_A(M,N)\ [\Psi]}{E \doteq \star \in \mathsf{Eq}_A(M,N)\ [\Psi]}$$

Figure 4 Dependent functions, dependent pairs, paths, and exact equalities.



We achieve both conditions by defining a new type former "V", such that whenever $A\ \mathsf{type}_{\mathsf{pre}}\ [\Psi, x \mid x = 0]$, $B\ \mathsf{type}_{\mathsf{pre}}\ [\Psi, x]$, and $E \in \mathsf{Equiv}(A, B)\ [\Psi, x \mid x = 0]$, $\mathsf{V}_x(A, B, E)$ is a type with faces $A\langle 0/x\rangle$ and $B\langle 1/x\rangle$, whose elements are pairs of $N \in B\ [\Psi, x]$ and $M \in A\langle 0/x\rangle\ [\Psi]$ such that E sends M to exactly $N\langle 0/x\rangle$. (Bezem et al. [13] employ the same approach in their "G" types.) We then define:

$$\mathsf{idtoequiv} := \lambda p.\mathsf{coe}^{0\rightsquigarrow 1}_{x.\mathsf{Equiv}(A,p@x)}(\langle \lambda a.a, \mathsf{idisequiv}\rangle)$$
$$\mathsf{ua} := \lambda e.\langle x\rangle \mathsf{V}_x(A, B, e)$$
$$\mathsf{ua}_\beta := \lambda e.\lambda a.\langle x\rangle \mathsf{coe}^{x\rightsquigarrow 1}_{\_.B}(\mathsf{app}(\mathsf{fst}(e), a))$$

where $\mathsf{idisequiv}$ is a proof that the identity function is an equivalence, and ua_β relies on coercion across an equivalence: $\mathsf{coe}^{0\rightsquigarrow r'}_{x.\mathsf{V}_x(A,B,E)}(M) \longmapsto_{\square} \mathsf{Vin}_{r'}(M, \mathsf{coe}^{0\rightsquigarrow r'}_{x.B}(\mathsf{app}(\mathsf{fst}(E\langle 0/x\rangle), M)))$.

When implementing $\mathsf{coe}^{y\rightsquigarrow r'}_{x.\mathsf{V}_x(A,B,E)}(M)$, we make essential use of an open box with a diagonal $y = r'$ side, to ensure coercion $y \rightsquigarrow y$ is the identity. (See our preprint [7] for this and the other Kan operations.) We have formalized the full proof of univalence for our system in RedPRL (see `https://git.io/vFjUQ`).

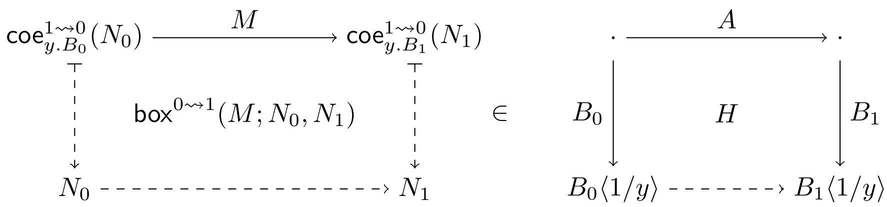
6 Universes

Finally, we define two cumulative hierarchies of universes, $\mathcal{U}^{\mathsf{pre}}_j$ and $\mathcal{U}^{\mathsf{Kan}}_j$, classifying pretypes and Kan types respectively, each closed under the appropriate type formers, and satisfying:

$$\frac{}{\mathcal{U}^{\kappa}_j\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]} \qquad \frac{A \in \mathcal{U}^{\kappa}_j\ [\Psi]}{A\ \mathsf{type}_{\kappa}\ [\Psi]} \qquad \frac{A \in \mathcal{U}^{\kappa}_j\ [\Psi]}{A \in \mathcal{U}^{\kappa}_{j+1}\ [\Psi]} \qquad \frac{A \in \mathcal{U}^{\mathsf{Kan}}_j\ [\Psi]}{A \in \mathcal{U}^{\mathsf{pre}}_j\ [\Psi]}$$

In order for our type theory to be a suitable setting for synthetic homotopy theory, it is essential that $\mathcal{U}^{\mathsf{Kan}}_j$ is Kan; this is needed, for example, to define maps $\mathbb{S}^1 \to \mathcal{U}^{\mathsf{Kan}}_j$ used in the calculation of the fundamental group of the circle [29]. As with $\mathbb{S}^1$, universes are not automatically Kan, so we equip both with free Kan structure analogous to $\mathsf{hcom}_{\mathbb{S}^1}$.

Because elements of $\mathcal{U}^{\mathsf{pre}}_j$ are pretypes, we must ensure $\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathcal{U}^{\mathsf{pre}}_j}(A; \overrightarrow{\xi_i \hookrightarrow y.B_i})\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$ for pretypes $A, \overrightarrow{B_i}$ satisfying the appropriate equations. We define these types to be empty. Similarly, we require $\mathsf{hcom}^{r\rightsquigarrow r'}_{\mathcal{U}^{\mathsf{Kan}}_j}(A; \overrightarrow{\xi_i \hookrightarrow y.B_i})\ \mathsf{type}_{\mathsf{Kan}}\ [\Psi]$ for Kan types $A, \overrightarrow{B_i}$ satisfying the appropriate equations. In order to equip $\mathsf{hcom}_{\mathcal{U}^{\mathsf{Kan}}_j}$ with Kan operations, we define its elements to be open boxes consisting of an element $M \in A\ [\Psi]$, and a family of elements $N_i \in B_i\langle r'/y\rangle\ [\Psi \mid \xi_i]$ such that $\mathsf{coe}^{r'\rightsquigarrow r}_{y.B_i}(N_i) \doteq M \in A\ [\Psi \mid \xi_i]$. The diagram below illustrates an element of $H := \mathsf{hcom}^{0\rightsquigarrow 1}_{\mathcal{U}^{\mathsf{Kan}}_j}(A; x = 0 \hookrightarrow y.B_0, x = 1 \hookrightarrow y.B_1)$.



When $r = r'$, $H \doteq A$ and the $\mathsf{box} \doteq M$. When ξ_i holds, $H \doteq B_i\langle r'/y\rangle$ and the $\mathsf{box} \doteq N_i$. These agree when both $r = r'$ and ξ_i hold: $A \doteq B_i\langle r/y\rangle = B_i\langle r'/y\rangle$ and $M \doteq \mathsf{coe}^{r' \rightsquigarrow r}_{y.B_i}(N_i) \doteq N_i$.

For the complete definition of $\mathsf{hcom}_{\mathcal{U}^{\mathsf{Kan}}_j}$ and its Kan operations, see our preprint [7]. Coercion requires heterogeneous compositions that may not be valid in the sense of Definition 12, but which are nevertheless definable in our setting. (Such compositions are closely related to the $\forall i.\varphi$ operation of Cohen et al. [17].) Finally, to ensure these Kan operations agree with those of A when $r = r'$, we once again make essential use of open boxes with diagonal sides.

Intuitively, each universe $[\![\mathcal{U}^\kappa_j]\!]$ is defined as the least context-indexed PER closed under all type formers yielding κ-types, that are present in a type theory with j universes. Of course, typehood and membership are mutually defined ($\mathsf{Eq}_A(M, N)\ \mathsf{type}_{\mathsf{pre}}\ [\Psi]$ when $M, N \in A\ [\Psi]$), so the values of each universe depend on both the names *and* semantics of types.

Following Allen [1], we make this construction precise by introducing *candidate cubical type systems*, relations $\tau(\Psi, A_0, B_0, \varphi)$ as in Definition 4 without any conditions of functionality, symmetry, and so forth. Candidate cubical type systems form a complete lattice when ordered by inclusion, so we define each universe as the least fixed point of a monotone operator (guaranteed to exist by the Knaster–Tarski fixed point theorem).

For each κ, we define an operator $F^\kappa(\tau^u, \tau^{\mathsf{pre}}, \tau^{\mathsf{Kan}})$ whose arguments are candidate cubical type systems defining (1) all smaller universes, (2) pretype formers, and (3) Kan type formers, following the meanings given in Section 5. These operators are monotone because $\mathsf{Tm}(-)$ is monotone, and hence the judgments defined in Section 3 are monotone in τ.

Then construct the simultaneous least fixed points $\tau^\kappa_i = F^\kappa(\tau^u_i, \tau^{\mathsf{pre}}_i, \tau^{\mathsf{Kan}}_i)$ for each $i \geq 0$, where τ^u_i defines each $[\![\mathcal{U}^\kappa_j]\!]$ (for $j < i$) as $\tau^u_i(\Psi, \mathcal{U}^\kappa_j, \mathcal{U}^\kappa_j, \{(A_0, B_0) \mid \tau^\kappa_j(\Psi, A_0, B_0, \_)\})$, that is, the typehood relation of τ^κ_j. We establish by induction that each τ^κ_i is in fact a cubical type system in the sense of Definition 4, and each is closed under the appropriate type formers. We take the "outermost" cubical type system $\tau^{\mathsf{pre}}_\omega$ (containing universes for all j) as our model, validating every rule presented in this paper. This construction requires no classical reasoning, and in fact Anand and Rahli [4] carry out Allen's original Nuprl semantics inside the Coq proof assistant using inductive types rather than fixed points.

7 Conclusion and Related Work

We have constructed a two-level type theory with fibrant, univalent universes closed under dependent function, dependent pair, and path types. The non-fibrant (pretype) level includes these type formers as well as exact (strict) equality types with equality reflection. Following the tradition of the Nuprl computational type theory [2] and Martin-Löf's meaning explanations, our types are relations over untyped programs equipped with an operational semantics, and thereby satisfy canonicity (Theorem 15) by construction. Full details and proofs are available in our associated preprint [7]. An early version of our cubical PER semantics appeared in Angiuli et al. [6], but for a type theory including neither univalence, nor universes, nor exact equality, and equipped with a variant of our Kan operations restricted to open boxes with sides $\overline{r_i = 0, r_i = 1}$ (and in particular, without $x = z$ sides critical for univalent universes).

We are currently implementing the RedPRL [28] proof assistant based on this type theory. RedPRL implements a proof refinement sequent calculus in the style of Nuprl, rather than the natural deduction rules presented in this paper; we view it as the extension of core Nuprl to a higher-dimensional notion of program.

Cavallo and Harper [16] define a schema of higher inductive types constructible in the semantic framework we describe. Their *fiber family* type validates the rules of the homotopy-type-theoretic identity type (strictly, unlike path types). Our type theory, extended with fiber families, constitutes a fully computational model of univalent intensional type theory.

7.1 Two-level type theories

Voevodsky's HTS [33] extends homotopy type theory with exact equality types satisfying equality reflection. Our semantics validate the rules of HTS, excepting resizing rules. More recently, Altenkirch et al. [3] have proposed a two-level type theory with two intensional identity types: one to internalize paths, and the other satisfying uniqueness of identity proofs and function extensionality, but not equality reflection. Both theories consider all strict equality types non-fibrant, and neither theory satisfies canonicity, because univalence (and in the latter, uniqueness of identity proofs and function extensionality) are added as axioms that do not compute.

Our contributions to two-level type theory are twofold: (1) we define the first two-level type theory satisfying canonicity, and (2) by introducing the notion of discrete Kan types (see our preprint [7]), we obtain a type theory in which some exact equality types are fibrant.

7.2 Cubical type theories

Our use of cubical structure and uniform Kan conditions traces back to the Bezem et al. [12] cubical set model of type theory, which has only face and degeneracy maps. The cubical type theory of Cohen et al. [17] uses a De Morgan algebra of cubes containing not only face, diagonal, and degeneracy maps, but also connection and reversal maps.

From a proof-theoretic perspective, our semantics can be seen as *cubical logical relations* suitable for proving canonicity (and consistency) for a set of inference rules. In fact, Huber's canonicity argument [19] for Cohen et al. [17] resembles our PER semantics in various ways, most notably his "expansion lemma," which is closely related to Lemma 10.

The fibrant fragment of our system constitutes the second univalent type theory with canonicity—after the cubical type theory of Cohen et al. [17]—and the first to employ Cartesian cubical structure. Licata and Brunerie [22] and Coquand [18] previously considered Cartesian cubes, but did not succeed in defining univalent universes. However, neither considered Kan operations with diagonal sides $x = z$, which figure prominently in our constructions of both univalence and fibrant universes. Diagonal sides also permit us to define connections in Kan types, although we remain unable to define an involutive reversal operation, as in Cohen et al. [17].

In ongoing work with Brunerie, Coquand, and Licata [5], we are investigating proof-theoretic and category-theoretic aspects of "diagonal" Kan composition. That project includes an Agda formalization of the Kan operations of various type formers, including a variant of the "Glue" types employed by Cohen et al. [17] to obtain both univalence and fibrancy of the universe. Here we decompose Glue types into V and $\mathsf{hcom}_{\mathcal{U}_j^{\mathsf{Kan}}}$, simplifying ua_β.

Unlike prior Kan conditions, we restrict to open boxes containing a pair of sides $x = 0, x = 1$ (Definition 12), in order to trivialize all Kan compositions at dimension zero. Thus we obtain a stronger canonicity result for the circle than Cohen et al. [17]: if $M \in \mathbb{S}^1\ [\emptyset]$ then $M \Downarrow \mathsf{base}$. We believe this property to be valuable for programming applications of cubical type theory, by allowing higher inductive types to function as observables at dimension zero. The tradeoff is that we must develop additional machinery to define coercion in $\mathsf{hcom}_{\mathcal{U}_j^{\mathsf{Kan}}}$, essentially because the $\forall i.\varphi$ operation of Cohen et al. [17] does not preserve box validity.

References

1. Stuart F. Allen. A Non-type-theoretic Definition of Martin-Löf's Types. In D. Gries, editor, *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science*, pages 215–224, Ithaca, NY, June 1987. IEEE Computer Society Press.
2. 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.
3. Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Extending homotopy type theory with strict equality. In *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, pages 21:1–21:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.CSL.2016.21`.
4. Abhishek Anand and Vincent Rahli. Towards a formally verified proof assistant. In *Interactive Theorem Proving*, pages 27–44, Cham, 2014. Springer International Publishing.
5. Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. Cartesian cubical type theory. Preprint, December 2017. URL: `https://github.com/dlicata335/cart-cube`.
6. Carlo Angiuli, Robert Harper, and Todd Wilson. Computational higher-dimensional type theory. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, pages 680–693, New York, NY, USA, 2017. ACM. `doi:10.1145/3009837.3009861`.
7. Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Computational higher type theory III: Univalent universes and exact equality. December 2017. `arXiv:1712.01800`.
8. Danil Annenkov, Paolo Capriotti, and Nicolai Kraus. Two-level type theory and applications. May 2017. `arXiv:1705.03307`.
9. Steve Awodey. A cubical model of homotopy type theory. June 2016. URL: `https://www.andrew.cmu.edu/user/awodey/preprints/stockholm.pdf`.
10. Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, January 2009. `doi:10.1017/S0305004108001783`.
11. Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Michael Shulman, Matthieu Sozeau, and Bas Spitters. The HoTT library: A formalization of homotopy type theory in Coq. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2017, pages 164–172, New York, NY, USA, 2017. ACM. `doi:10.1145/3018610.3018615`.
12. 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, Toulouse, France, 2014. Dagstuhl Publishing.
13. Marc Bezem, Thierry Coquand, and Simon Huber. The univalence axiom in cubical sets. October 2017. `arXiv:1710.10941`.
14. Guillaume Brunerie, Kuen-Bang Hou (Favonia), Evan Cavallo, Eric Finster, Jesper Cockx, Christian Sattler, Chris Jeris, Michael Shulman, et al. Homotopy type theory in Agda, 2018. URL: `https://github.com/HoTT/HoTT-Agda`.
15. Ulrik Buchholtz and Edward Morehouse. Varieties of cubical sets. In *Relational and Algebraic Methods in Computer Science: 16th International Conference, RAMiCS 2017, Lyon, France, May 15-18, 2017, Proceedings*, pages 77–92. Springer International Publishing, Cham, 2017. `doi:10.1007/978-3-319-57418-9_5`.
16. Evan Cavallo and Robert Harper. Computational higher type theory IV: Inductive types. January 2018. `arXiv:1801.01568`.
17. Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69, pages 5:1–5:34, Dagstuhl,

Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2015.5.

18 Thierry Coquand. Variations on cubical sets. 2014. URL: http://www.cse.chalmers.se/~coquand/diag.pdf.

19 Simon Huber. *Cubical Interpretations of Type Theory*. PhD thesis, University of Gothenburg, November 2016.

20 Chris Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after Voevodsky). June 2016. arXiv:1211.2851.

21 Daniel R. Licata. Weak univalence with "beta" implies full univalence. Email to the Homotopy Type Theory mailing list, September 2016. URL: https://groups.google.com/forum/#!topic/homotopytypetheory/j2KBIvDw53s.

22 Daniel R. Licata and Guillaume Brunerie. A cubical type theory, November 2014. Talk at Oxford Homotopy Type Theory Workshop. URL: http://dlicata.web.wesleyan.edu/pubs/lb14cubical/lb14cubes-oxford.pdf.

23 Peter LeFanu Lumsdaine and Mike Shulman. Semantics of higher inductive types. May 2017. arXiv:1705.07088.

24 P. Martin-Löf. Constructive mathematics and computer programming. *Philosophical Transactions of the Royal Society of London Series A*, 312:501–518, October 1984. doi:10.1098/rsta.1984.0073.

25 Per Martin-Löf. *Intuitionistic type theory*. Bibliopolis, Naples, Italy, 1984.

26 A. M. Pitts. Nominal presentation of cubical sets models of type theory. In *20th International Conference on Types for Proofs and Programs (TYPES 2014)*, pages 202–220, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2014.202.

27 W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967. doi:10.2307/2271658.

28 The RedPRL Development Team. RedPRL – the People's Refinement Logic, 2018. URL: http://www.redprl.org/.

29 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. http://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

30 Floris van Doorn, Jakob von Raumer, and Ulrik Buchholtz. Homotopy type theory in Lean. In *Interactive Theorem Proving*, pages 479–495, Cham, 2017. Springer. doi:10.1007/978-3-319-66107-0_30.

31 Vladimir Voevodsky. The equivalence axiom and univalent models of type theory, 2010. Notes from a talk at Carnegie Mellon University. URL: http://www.math.ias.edu/vladimir/files/CMU_talk.pdf.

32 Vladimir Voevodsky. Univalent foundations project. Modified version of an NSF grant application, October 2010. URL: http://www.math.ias.edu/vladimir/files/univalent_foundations_project.pdf.

33 Vladimir Voevodsky. A type system with two kinds of identity types. Slides available at https://uf-ias-2012.wikispaces.com/file/view/HTS_slides.pdf/410105196/HTS_slides.pdf, February 2013. URL: https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/HTS.pdf.