Computational Semantics of Cartesian Cubical Type Theory

Carlo Angiuli
CMU-CS-TODO
(7/18/2019)

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Robert Harper, Chair
Jeremy Avigad
Karl Crary
Daniel R. Licata, Wesleyan University
Todd Wilson, California State University, Fresno

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2019 Carlo Angiuli

This research was sponsored by...
Keywords: dependent type theory, homotopy type theory, cubical type theory, two-level type theory, computational type theory, computational semantics
Dependent type theories are a family of logical systems that serve as expressive functional programming languages and as the basis of many proof assistants. In the past decade, type theories have also attracted the attention of mathematicians due to surprising connections with homotopy theory; the study of these connections, known as homotopy type theory, has in turn suggested novel extensions to type theory, including higher inductive types and Voevodsky’s univalence axiom. However, in their original axiomatic presentation, these extensions lack computational content, making them unusable as programming constructs and unergonomic in proof assistants.

In this dissertation, we present Cartesian cubical type theory, a univalent type theory that extends ordinary type theory with interval variables representing abstract hypercubes. We justify Cartesian cubical type theory by means of a computational semantics that generalizes Allen’s semantics of Nuprl [All87] to Cartesian cubical sets. Proofs in our type theory have computational content, as evidenced by the canonicity property that all closed terms of Boolean type evaluate to true or false. It is the second univalent type theory with canonicity, after the De Morgan cubical type theory of Cohen et al. [CCHM18], and affirmatively resolves an open question of whether Cartesian interval structure constructively models univalent universes.
## Contents

Acknowledgements

Contents \hfill vii

1 Introduction \hfill 1
   1.1 The Martin-Löf ethos \hfill 1
   1.2 The algebraic perspective \hfill 5
   1.3 Contributions \hfill 7

2 Idealized Nuprl \hfill 11
   2.1 Syntax and operational semantics \hfill 12
   2.2 Constructing type systems \hfill 15
   2.3 Closed and open judgments \hfill 23
   2.4 Rules of inference \hfill 28
   2.5 Semantics of types \hfill 32
   2.6 Modern Nuprl \hfill 41

3 Cubical methods \hfill 45
   3.1 Symmetric monoidal cubes \hfill 49
   3.2 Cartesian cubes \hfill 53
   3.3 De Morgan cubes \hfill 58
   3.4 Regularity \hfill 60
   3.5 Validity and other considerations \hfill 63

4 Cartesian cubical type theory \hfill 69
   4.1 Syntax and operational semantics \hfill 71
   4.2 Constructing cubical type systems \hfill 74
   4.3 Cubical judgments and Kan operations \hfill 88
   4.4 Semantics of types \hfill 101

5 Conclusion \hfill 143

A Computational-style rules \hfill 147

B Intensional-style rules \hfill 155

Bibliography \hfill 163
Introduction

One might come to believe that any decision to adopt a system of axioms about sets would be *arbitrary* in that no explanation could be given why the particular system adopted had any greater claim to describe what we conceive sets and the membership relation to be like than some other system, perhaps incompatible with the one chosen. One might think that no answer could be given to the question: why adopt this particular system rather than that or this other one?

—George Boolos, *The Iterative Conception of Set* [Boo71]

Dependent type theories are a family of logical systems employed by philosophers as foundations of constructive mathematics, and by computer scientists as expressive functional programming languages. In the past decade, type theories have also attracted the attention of mathematicians due to surprising connections with homotopy theory; the study of these connections, known as *homotopy type theory*, has in turn motivated powerful new logical principles in type theory. This dissertation extends traditional computer-scientific techniques in type theory to account for these novel principles.

1.1 The Martin-Löf ethos

To the great consternation of mathematicians, there is no *set of all type theories*. Instead, the study of type theory comprises techniques drawn from computer science, mathematics, and philosophy which, when applicable to an object, confer upon it the status of *type theory*. Many mainstream type theories, however—from the seminal work of Per Martin-Löf [ML75b; ML82; ML84] to Nuprl [Con+85] and the calculus of constructions [CH88]—share various hallmarks which we henceforth describe, with the caveat that these hallmarks are not all universal among type theories.

**Unity of collections and propositions** Set theory is typically presented as an axiomatic system that defines set membership as a proposition inside an ambient logic. A type theory, in contrast, is a self-contained foundation of mathematics in which types serve both as collections of objects and as logical propositions. Standard type theories comprise two core judgments—typehood (A type) and membership (\(M \in A\))\(^1\)—and the auxiliary judgments of

\(^1\)Once standard notation [ML75b; ML82; ML84; Con+85], \(M \in A\) is now more often written \(M : A\).
type and member equality under hypotheses. Regarding the type $A$ as a collection, $M \in A$ expresses that $M$ is an element of $A$; regarding $A$ instead as a proposition, $M \in A$ expresses that $M$ is a proof of $A$.

Given a type $A$ and a family of types $B(a)$ indexed by $a \in A$, the dependent function type $(a:A) \to B(a)$ (or $\prod_{a:A} B(a)$) classifies functions sending every $M \in A$ to an element of $B(M)$, and the dependent pair type $(a:A) \times B(a)$ type (or $\sum_{a:A} B(a)$) classifies pairs of $M \in A$ and an element of $B(M)$. If we interpret $B(\_)$ as a predicate over $A$, $(a:A) \to B(a)$ is the proposition that $B(a)$ holds for all $a \in A$, because its proofs specify a proof of $B(M)$ for every $M \in A$, and $(a:A) \times B(a)$ is the proposition that $B(a)$ holds for some $a \in A$, because its proofs specify an $M \in A$ and a proof of $B(M)$.

Now, because types are both collections and propositions, it is possible for a proposition to have many distinct proofs, and for subsequent proofs or constructions to be proof-relevant, that is, dependent on a choice of proof. For instance, the type of natural numbers is a proposition with infinitely many distinct proofs, each of which has a distinct successor. Some authors therefore only regard as propositions (or mere propositions [UF13, Definition 3.3.1]) types with at most one element; another possibility is to reject the collection–proposition dichotomy as intrinsically non-type-theoretic.

Readers unfamiliar with standard type formers or the style of mathematics developed inside type theories may wish to consult Nordström, Petersson, and Smith [NPS90] or the Homotopy Type Theory book [UF13] as necessary while reading this dissertation.

**Constructivity**  As logics, type theories are typically constructive, or intuitionistic, not classical. Intuitionism is a philosophical stance introduced by Brouwer, who held that mathematical objects exist only as constructions in one’s mind, that propositions are true exactly when mental constructions of their proofs exist, and that the sole purpose of mathematical language is to convey such constructions, which are themselves extralinguistic [Tv88, p. 4].

The Brouwer–Heyting–Kolmogorov (BHK) interpretation of the logical connectives therefore maps each connective to a construction—to prove $\exists n:\text{nat}.B(n)$ is to construct a natural number $n$ and a proof of $B(n)$; to prove a disjunction $A \lor B$ is to construct either a proof of $A$ or a proof of $B$ [Tv88, p. 9]: et cetera. It follows that the law of excluded middle ($A \lor \neg A$ for all $A$) is not intuitionistically valid: to prove $(\exists n:\text{nat}.B(n)) \lor \neg(\exists n:\text{nat}.B(n))$ for all $B$, one must produce either a concrete $n$ such that $B(n)$, or demonstrate none exists.

Constructivity is popularly characterized as the failure of excluded middle, but is more accurately captured by the existence property: if $\exists n:\text{nat}.B(n)$ is provable, one can construct a concrete numeral $\bar{n}$ for which $B(\bar{n})$ is provable [Tv88, p. 139]. From this perspective, it is fortunate that most principles of classical mathematics are constructively valid. On the other hand, there are constructive logics with principles false in classical mathematics, including Brouwer’s intuitionism and the Nuprl type theory, which adopt continuity
principles for choice sequences [RB16; Tv88, pp. 206–210], and Russian constructive recursive mathematics, which adopts Church’s thesis [Tv88, pp. 185–195].

Martin-Löf’s type theories, as he explains in *Constructive mathematics and computer programs* [ML82], are rooted in a modified form of intuitionism which considers mathematical objects to be constructions not of the mind but of a *programming language*. This conception of intuitionism is incompatible with Brouwer’s, as it is essentially linguistic. However, by equating construction and computation, it transforms intuitionism from merely a philosophical stance to a pragmatic one—intuitionistic mathematics is mathematics that computes, a fruitful endeavor regardless of one’s beliefs.

The intuitionistic readings of Martin-Löf’s type theories are provided in his *meaning explanations*, BHK interpretations which gloss each type former as a behavioral predicate over programs [ML82]. For instance, \( M \in (a:A) \times B(a) \) when \( M \) is a program that computes a pair \( \langle M_1, M_2 \rangle \) with \( M_1 \in A \) and \( M_2 \in B(M_1) \) (that is, \( M_1 \) computes in accordance with \( A \), and \( M_2 \) with \( B(M_1) \)). Hypotheses simply range over all programs with the specified behavior, so the assertion “assuming \( a \in A \), \( B(a) \) type” means that for every program \( M \in A \), the expression \( B(M) \) corresponds to a behavioral predicate. The *canonicity property*—if \( M \in \text{bool} \), then \( M \) computes to and is equated with a canonical Boolean, either true or false—is a type-theoretic analogue of the existence property, and an immediate consequence of the meaning explanations (because \( \text{bool} \) corresponds to precisely that predicate).

Meaning explanations are typically presented in an intuitive, premathematical style ambiguous about the programming language and rules of inference purportedly being explained. These choices are deliberate—appeals to intuition ensure the explanations take on a foundational character (that is, without appealing to an ambient logic), while ambiguity over computation and deductive systems ensures the explanations remain open-ended with respect to the choice of programs or predicates. Further discussion of the role of intuitionism in type theories is beyond the scope of this dissertation; we direct the interested reader to the writings of Martin-Löf [ML82; ML13] and Granström [Gra09].

As type theories grow in complexity, it is increasingly difficult to convey their intended meanings through intuition alone, or be confident that such intuitions are sensible. (Such is the case in this dissertation.) One can instead transform the meaning explanations into precise mathematical constructions, which we call *computational semantics*. The variants of computational semantics used in Chapters 2 and 4 of this dissertation serve as constructive proofs of consistency and canonicity, and explicit Curry–Howard or propositions-as-types\(^2\) correspondences [How80] that construct type theories as program logics for functional programming languages.

\(^2\)By *propositions-as-types* we mean that the logical propositions (that is, the *types*) of a type theory can equally well be viewed as the *type system* of a programming language, not merely the observation that the types of a type theory serve also as its logical propositions.
Suitability for computer implementation Of course, the study of type theories intersects computer science not only through computational semantics, but also through the development of proof assistants—software for developing computer-checked mathematical proofs—many of which are based on dependent type theories, including Agda [Agda], Coq [Coq], Lean [Mou+15], and Nuprl [Con+85].

In recent years, computer scientists have begun to coordinate large-scale uses of proof assistants in formally specifying and verifying the behavior of realistic software programs and protocols [App+17]. And while proof assistants remain on the fringe of mathematical practice, expert users have formalized increasingly sophisticated results, culminating famously in the Coq proofs of the four-color [Gon08] and odd-order theorems [Gon+13].

Proof assistants serve to bridge the gap between high-level mathematical arguments and low-level formal proofs composed of primitive inferences, thereby reducing the correctness of the former to the correctness of the latter. In practice, proof assistants require vastly more detail than is customary in paper proofs, but do perform humanly-impossible feats of logical bureaucracy and even automated combinatorial reasoning [Gon+13, p. 173].

The primitive inferences of a proof assistant, being the linchpin of its correctness, are often isolated in a trusted kernel; complex features are not trusted, but instead emit proofs checked by the kernel. For type theories, these primitive inferences are rules specifying when a term is a type or element of a type; in contrast to the open-ended character of meaning explanations, the kernel necessarily fixes a collection of rules, and hence, of types and elements.

The hallmarks of type theories, we argue, contribute in part to the success of type-theoretic proof assistants. First, types directly provide many basic mathematical objects (functions, products, natural numbers, et cetera) that are derived notions in, say, set theory. Ordinary mathematical proofs should in theory reduce to applications of the axioms of set theory; a proof assistant based on set theory must in fact perform such a reduction (or else define these objects primitively), significantly widening the gap between informal and formal proof. In contrast, Agda’s interface is usable despite being quite close to its underlying type theory—users prove \( A \) by providing (most of) an \( M \) such that \( M \in A \).

Secondly, the constructive nature of type theories enables proof assistants to seamlessly incorporate computation in many beneficial ways. The judgments of type theory are closed under computation, allowing users to simplify proof goals by evaluating them; canonicity ensures that elements of types constructed without hypotheses evaluate to canonical forms. Proof by reflection allows users to implement and even verify decision procedures written in the very programming language underlying the type theory itself [Gon+13]. Program extraction converts proofs into standalone, correct-by-construction functional programs, and has been used to develop a verified C compiler [Ler09]. Implementation of such features is beyond the scope of this dissertation; we discuss their relationship to computational semantics in Section 2.4.
1.2 The algebraic perspective

The aforementioned philosophical and computational desiderata constitute one perspective on type theories. Another perspective, particularly common amongst homotopy type theorists, is to regard as type theories all collections of rules bearing a superficial resemblance to type theories in the former sense. A type theorist of this genus regards a type theory as an algebraic structure defined by its collection of rules, and asks: what counts as a model (resp., morphism of models) of this type theory? Does this type theory have interesting models? What properties are shared by all of its models?

One purpose of such questions is to establish metatheorems about the rules implemented in a proof assistant. Logical consistency follows from the existence of non-trivial models. Proof assistants often employ only rules whose premises are decidable, a property commonly established by normalization-by-evaluation models [ACD07]. Independence results follow from the existence of models both validating and invalidating certain principles. In a classic paper, Smith [Smi88] proved the independence of Peano’s fourth axiom \((\forall n:\text{nat}.0 \neq n + 1)\) from a type theory without universe types, by constructing a model in which the natural numbers are a one-element set. (The standard computational semantics validate Peano’s axiom.) More recently, Coquand and Mannaa [CM16] proved the independence of Markov’s principle from type theory by a forcing argument.

A second purpose is to establish the mathematical significance of theorems proven in a proof assistant. We expect, for example, that the Coq proof of the four-color theorem implies the truth of the four-color theorem as ordinarily construed. Such an implication rests not only on Coq’s consistency but also the ability to interpret Coq’s logical connectives and rules as classical set-theoretic connectives and tautologies. If classical truth is one’s only goal, one may even choose to disrupt the constructive nature of a type theory by adding to it classical axioms for real numbers [BLM15].

Conversely, theorems in a proof assistant carry more content than their classical counterparts. Standard rules of type theories are valid in all locally Cartesian closed categories [Cur93; See84]—not only in sets—thus licensing type-theoretic language in presheaf categories. Orton and Pitts [OP16] have recently employed this principle to carry out cubical set constructions of Cohen et al. [CCHM18] in Agda extended with an axiomatic interval type, thereby both formalizing and generalizing the original constructions.

**Homotopy type theory** The intensional rules of type theory [ML75b], which form the basis of Agda and Coq, define an identity type \(\text{Id}_A(M, N)\) whose elements are proofs that \(M, N \in A\) are equal. Standard computational and set-theoretic semantics model identity types as collections with one element when true and zero elements when false. Surprisingly,

---

3 The precise class of such theories, which Voevodsky [Voe15] calls "dependent type theory of Martin-Löf ‘genus,’” currently lacks a satisfactory definition.
Hofmann and Streicher [HS98] showed that uniqueness of identity proofs—the principle that any two proofs of $\text{id}_A(M, N)$ are equal—is independent of intensional type theory, by constructing a groupoid model in which some identity types have multiple elements. The Hofmann–Streicher groupoid model nevertheless validates uniqueness of identity proofs of identity proofs, that is, any two proofs of $\text{id}_{\text{id}_A(M, N)}(P, Q)$ are equal.

The field of homotopy type theory originates with the observations of Awodey and Warren [AW09] and Voevodsky [KL16] that $n$-fold iterated identity types admit complex structure closely related to that of $n$-dimensional paths in topological spaces, by constructing models of intensional type theory in, respectively, Quillen model categories and Kan simplicial sets. Voevodsky [Voe10a] further observed that the simplicial model validates a univalence axiom stating that identity of types is homotopy equivalent to homotopy equivalence of types (or informally, that isomorphic types can be regarded as equal). Finally, Lumsdaine and Shulman [LS17] and others observed that many familiar topological spaces can be defined axiomatically in type theory as higher inductive types.

Voevodsky subsequently championed Univalent Foundations—intensional type theory with the univalence axiom—as a new foundation for formalized, structuralist mathematics [Voe10b]. Parallel community efforts over 2012–2013 at the Institute for Advanced Study produced the Homotopy Type Theory book [UF13], which uses intensional type theory extended with axioms for univalence and higher inductive types as an axiom system in which to develop the homotopy theory of spaces. (In a confusing instance of synecdoche, that type theory is often called homotopy type theory; we instead call it Book HoTT.)

Are Univalent Foundations and Book HoTT—as emulated in Coq [VAG+; Bau+17], Agda [Bru+18], and Lean [DRB17]—the most perspicuous axiomatizations of univalence and higher inductive types? On what basis should we judge type theories qua algebraic structures? Or, as Boolos [Boo71] asks, “Why adopt this particular system rather than that or this other one?”

Consistency is paramount, and follows from the simplicial set semantics of Voevodsky [KL16] and Lumsdaine and Shulman [LS17]. Those semantics moreover establish that theorems of Book HoTT hold also in standard homotopy theory, because the homotopy theory of simplicial sets is famously the same as that of topological spaces.

Importantly, Book HoTT concerns an abstract notion of space, defined without reference to, say, topologies or real numbers; as a result, Book HoTT has already led to novel generalizations of theorems [Ane+17]. Alternatives to Book HoTT, including those proposed in this dissertation, are thus judged in part on the generality of their models.

---

4 This is a woefully incomplete summary of early ideas in homotopy type theory; other early contributors to the homotopy-theoretic semantics of type theory include Gambino and Garner [GG08], Lumsdaine [Lum09], van den Berg and Garner [vG11], and Streicher [Str06].

5 Excepting closure of type universes under parametrized higher inductive types, which was later addressed in cubical set models of Coquand, Huber, and Mörtberg [CHM18] and Cavallo and Harper [CH19a].
Moreover, homotopy type theorists conjecture that type theories with univalence are in a precise sense sound and complete for mathematical domains known as elementary \(\infty\)-toposes. However, the community is very far from proving this conjecture—the state of the art includes only dependent pairs and identity types [KS17]—and in fact, lacks consensus on the very definition of elementary \(\infty\)-topos.

As foundations of mathematics, Univalent Foundations and Book HoTT have several advantages over set theory: they are more amenable to computer formalization, and more elegantly express many ideas from homotopy theory and higher category theory. However, univalence is not as straightforward as the slogan isomorphic types are equal: a minor change in its statement renders it inconsistent [UF13, Exercise 4.6(iii)], and its standard model in simplicial sets requires powerful mathematical tools. Nor are these theories obviously constructive in any sense: Univalent Foundations and Book HoTT lack the canonicity and existence properties, and thus even the failure of excluded middle in these theories is subtle [UF13, Corollary 3.2.7]. Such subtleties, in the author’s opinion, jeopardize the ontological primacy of these systems.

The failure of canonicity in Book HoTT is caused by a dearth of equations regarding univalence and higher inductive types. For instance, higher inductive types are generated by element and path constructors, and functions out of such types specify where to send those generators; in Book HoTT, a function so defined only weakly sends path constructors to their specified images (that is, up to a path), requiring complex bookkeeping in otherwise straightforward proofs [Soj16].

Stranger yet, because Univalent Foundations and Book HoTT co-opt type-theoretic identity to mediate path structure, they lack any internal notion of traditional mathematical equality. That is, one cannot speak of two types being exactly the same, nor define relations finer than homotopy, nor argue that a function can be silently replaced by another whose pointwise behavior is identical, despite nearly all models validating such notions. The inability to impose strict coherence conditions has in fact frustrated the definition of semisimplicial types in these systems, prompting Voevodsky to propose extending Univalent Foundations with strict equality in his Homotopy Type System [Voe13].

1.3 Contributions

This dissertation aims to understand the higher-dimensional concepts of univalence and higher inductive types through computational semantics. We describe Cartesian cubical type theory and its computational semantics, which reconstruct these concepts by means of a novel notion of untyped, higher-dimensional computation, supporting the thesis that higher-dimensional types classify higher-dimensional programs extensionally according to their behaviors. Our type theory satisfies the canonicity property, and provides a constructive, elementary (although admittedly complex) semantics of univalence.
We have three main contributions. First and foremost, ours is the second type theory admitting both univalence and canonicity, after the cubical type theory of Cohen et al. [CCHM18]. In both cases, cubical refers to presentations of higher-dimensional structure using interval variables, products of which represent abstract $n$-dimensional hypercubes. We equip our interval with Cartesian structure and our types with a generalized Kan filling operation, while Cohen et al. [CCHM18] equip their interval with a stronger De Morgan structure and their types with a weaker Kan composition operation. We therefore affirmatively resolve the open question of whether Cartesian interval structure constructively models univalent universes [Coq14; LB14].

Secondly, our semantics consist of programs and thus hew closely to the syntax of type theory, illuminating which rules and even which evaluation strategies to consider. In Appendix A we present a type theory in the style of the Nuprl computational type theory [Con+85], which admits a strict equality type with the equality reflection principle. Ours is the first type theory with canonicity that, like Voevodsky’s Homotopy Type System, has both path types and strict equality types. Such type theories are called two-level [ACK16], because they must stratify types into those that respect paths and those that do not (notably, certain equality types). We moreover present a finer stratification of types than previously described, allowing us to deduce that some equality types do respect paths.

Thirdly, we place a novel validity restriction on the shapes of Kan filling scenarios to obtain a strong canonicity result for higher inductive types: all closed 0-dimensional elements of higher inductive types compute to constructors. In contrast, Cohen et al. [CCHM18] also admit 0-dimensional Kan compositions as canonical forms. This result simplifies many proofs involving higher inductive types, and we believe it is critical for higher-dimensional programming—for instance, all elements of a type quotient are guaranteed to compute to constructors of the underlying type.

Cartesian cubical type theory has already been implemented in two experimental proof assistants, RedPRL [Red16] and redtt [Red18], with which the author and his collaborators have considered applications to synthetic homotopy theory, constructive mathematics, and programming. Notably, we have formalized the definition of semisimplicial types in RedPRL. Cavallo and Harper [CH19a] have extended the present work with a schema for indexed higher inductive types, and en passant complete the Cartesian cubical semantics of Book HoTT by defining Book HoTT–style identity types.

Outline In Chapter 2 we introduce standard computational semantics in the setting of Idealized Nuprl, a core type theory modeled after Nuprl. We discuss the rationale
behind different systems of inference rules implemented by proof assistants, including the computational type theory of Nuprl and the intensional type theory of Coq, Agda, and Lean.

In Chapter 3 we describe the history of cubical type theories, compare various cube categories and Kan operations, and connect the path types of Cartesian cubical type theory with the identity types of Book HoTT. We also describe regularity, a technical problem which arises in cubical models of type theory, and validity, a refinement of our Kan operations which strengthens our canonicity theorem.

Chapter 4 presents the main technical results of this dissertation, namely, the computational semantics of Cartesian cubical type theory. These semantics directly justify the two-level, computational-style rules listed in Appendix A (roughly what is implemented in RedPRL), and also model the intensional-style rules of Appendix B (roughly redtt).

TODO: In Chapter 5 we discuss future avenues of research.

Publications

This dissertation’s main results have been published:

- Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities [AFH18] details the main results of this dissertation. The associated preprint Computational Higher Type Theory III: Univalent Universes and Exact Equality [AFH17] is superseded by Chapter 4 of this dissertation.


- Meaning explanations at higher dimension [AH18] informally describes our semantics as a generalization of Martin-Löf’s meaning explanations of type theory.

- Computational Higher-Dimensional Type Theory [AHW17] explains an earlier version of our semantics that did not yet account for univalent universes. The associated preprint Computational Higher Type Theory II: Dependent Cubical Realizability [AH17] is superseded by Chapter 4.

In addition, the preprint Syntax and Models of Cartesian Cubical Type Theory [Ang+19] outlines an alternative presentation of Cartesian cubical type theory without exact equality, along with an Agda formalization of its denotational semantics.

Readers may also wish to consult the author's paper Homotopical patch theory [Ang+16] (an extended version of an earlier conference paper [Ang+14]), which predates constructive models of univalence, and describes a conjectural application of Book HoTT to dependently-typed programming: modeling version control systems and their implementations as higher inductive types and functions from such types into univalent universes.
One paper not coauthored by the author deserves particular mention. *Higher Inductive Types in Cubical Computational Type Theory* [CH19a], with associated preprint *Computational Higher Type Theory IV: Inductive Types* [CH18], proposes a schema for indexed higher inductive types (including Book HoTT–style identity types) in Cartesian cubical computational type theory, and extends our computational semantics accordingly.
Idealized Nuprl

Types of theories:

1. Theories with one intended model, such as number theory and (for the Platonist) analysis and classical set theory.

2. Theories with many intended models, such as group theory.

In case (1), the primary aims of axiomatization are clarity and rigor. On the contrary, in case (2), the prime goal is generality, and rigor and clarity are by-products.

—Michael J. Beeson, Foundations of Constructive Mathematics [Bee85, p. 83]

This chapter introduces Idealized Nuprl, a (non-univalent, non-cubical) constructive type theory based on Nuprl [Con+85], with dependent functions, dependent pairs, natural numbers, booleans, and two parallel cumulative hierarchies of universes. Like Nuprl, Idealized Nuprl defines types as behavioral predicates over computer programs, using Allen-style computational semantics [All87] that mathematize Martin-Löf’s meaning explanations of type theory [ML82].

We intend this chapter to serve as a primer on computational semantics and the Nuprl tradition, divorced from cubical complications that arise in Chapter 4. We therefore intentionally omit many features of modern Nuprl that we have not yet developed in the cubical setting, including untyped computational equivalence [How89]. Unlike Nuprl, we include a predicative hierarchy Prop_i of universes of subsingleton types that are subtypes of the standard universe hierarchy Type_i, foreshadowing the Kan and pretype universes in Chapter 4. Parallel universes were previously considered by Krishnaswami, Pradic, and Benton [KPB15]; as far as we know, we are the first to consider subtyping between them.

To be clear, the main results of this chapter are not novel, and were previously described by (among others) Allen [All87], Harper [Har92], and Anand and Rahli [AR14]—the last of whom provide a Coq formalization! However, the author is unsatisfied with existing presentations for various reasons, and has strived for detail and clarity in this chapter.

Why computational semantics? There are many styles of type-theoretic semantics, often based on category theory. Computational semantics are essential to Nuprl as they comprise its only intended semantics—unlike intensional [ML75b] and extensional [ML82]
type theory which, as discussed in Section 1.2, are celebrated in part for their range of models. As a result, the rules of Nuprl permit structurally ill-typed terms, terms with many distinct types, untyped reduction, non-termination, and many other features sensible in the $\lambda$-calculus but foreign to categorical semantics of type theories. Categorically, computational semantics are a model of type theory in modest sets, sets whose elements are tracked by collections of elements of a partial combinatory algebra [Oos08]. However, while universes of modest sets model an impredicative type-theoretic universe, the universes of computational semantics do not, as the latter are inductive-recursively defined modest sets of named collections of types.

More generally, our freestyle approach has the advantage of providing self-contained, elementary, constructive semantics directly justifying program extraction: indeed, we define $M \in A$ to mean that $M$ is a program with behavior $A$. Computational semantics of intensional type theory, for instance, validate erasure of type annotations and identity proofs during extraction, as they are unneeded at runtime. In Cartesian cubical type theory, computational semantics allow us to consider the efficiency of Kan operations, suggesting faster algorithms valid only for closed terms, or containing structurally ill-typed subterms.

Being self-contained and constructive, computational semantics are quite portable across metatheories. We have written Chapters 2 and 4 in an agnostic style compatible with both (constructive) set theory and extensional type theory. Sterling and Harper [SH18] develop the computational semantics of guarded computational type theory in the internal language of a particular presheaf topos that they simulate by Coq with axioms.

Computational semantics are an instance of the technique of logical relations, or Tait’s method [Tai67], which is commonly used in type theory to establish completeness of normalization by evaluation [Abe13], and in programming language theory to prove properties of type systems, such as termination, parametricity, and safety. Our computational semantics constitute a logical relations proof that the rules of Idealized Nuprl enforce behavioral properties of extracts. (The Nuprl perspective, however, is typically the reverse—viewing not the rules but the semantic relations as definitive.)

## 2.1 Syntax and operational semantics

Idealized Nuprl is founded on an untyped functional programming language whose syntax is presented in Figure 2.1. This language extends the $\lambda$-calculus in a standard fashion and is an instance of both Martin-Löf’s system of arities [NPS90, Chapter 3] and Harper’s abstract binding trees [Har16, Chapter 1]. We include type constructors because dependency induces computation in types; types and terms cannot have distinct syntactic sorts because typehood in Idealized Nuprl is posterior to operational semantics.

We follow Harper’s syntax chart notation in Figure 2.1, writing every term first as an abstract binding tree, then in a friendlier concrete notation, and finally in English. Capital
$$M := \Pi(A, a.B) \quad (a:A) \to B$$

dependent function type

$$\lambda(a.M) \quad \lambda a.M$$

lambda abstraction

$$\text{app}(M, N) \quad MN$$

function application

$$\Sigma(A, a.B) \quad (a:A) \times B$$

dependent pair type

$$\text{pair}(M, N) \quad \langle M, N \rangle$$

pairing

$$\text{fst}(M) \quad \text{fst}(M)$$

first projection

$$\text{snd}(M) \quad \text{snd}(M)$$

second projection

$$\text{Eq}(A, M, N) \quad \text{Eq}_A(M, N)$$

equality type

$$\text{refl} \quad \star$$

equality proof

$$\text{natt} \quad \text{natt}$$

natural number type

$$z \quad z$$

zero

$$\text{s}(M) \quad \text{s}(M)$$

successor

$$\text{natrec}(M, N_1, n.a.N_2) \quad \text{natrec}(M; N_1, n.a.N_2)$$

natural number recursion

$$\text{bool} \quad \text{bool}$$

boolean type

$$\text{true} \quad \text{true}$$

true

$$\text{false} \quad \text{false}$$

false

$$\text{if}(M, N_1, N_2) \quad \text{if}(M; N_1, N_2)$$

boolean recursion

$$\text{Prop}[i] \quad \text{Prop}_i$$

ith subsingleton universe

$$\text{Type}[i] \quad \text{Type}_i$$

ith type universe

Figure 2.1: Syntax of Idealized Nuprl.

letters \((M, N, A, \ldots)\) represent terms, and lowercase letters \((a, n)\) represent variables and binders. (We deliberately avoid \(x, y, z\) as Chapter 4 uses these for interval variables.) Note that in \((a:A) \to B\) and \((a:A) \times B\), the variable \(a\) is bound in \(B\); we write \(A \to B\) and \(A \times B\) respectively when \(a\) does not occur in \(B\).

As is customary, we equate \(\alpha\)-equivalent terms\(^1\) (quotienting terms by renaming of bound variables, or choosing a nameless representation of variable binders), and equip terms with a capture-avoiding substitution \(M[N/a]\) that replaces free occurrences of \(a\) with \(N\) in \(M\). Readers unfamiliar with these concepts may wish to consult Harper [Har16, Chapter 1]. We do not quotient terms by \(\beta\)- or \(\eta\)-equivalence at this stage. (Modern Nuprl quotients untyped terms by applicative bisimulation [How89], as we discuss in Section 2.6.)

We give operational meaning to closed terms (those without free variables) using a structural operational semantics [Plo81] defined in Figure 2.2. The judgment \(M_0\) val specifies when \(M_0\) is a value, and \(M \rightsquigarrow M'\) specifies when \(M\) takes one step of weak head evaluation to \(M'\). We will write \(M \rightsquigarrow^* M'\) when \(M\) steps to \(M'\) in zero or more steps, and \(M \Downarrow M_0\), or \(M\) evaluates to \(M_0\), when \(M \rightsquigarrow^* M_0\) and \(M_0\) val.

\(^1\)Surprisingly, Allen does not equate \(\alpha\)-equivalent terms [All87, p. 66]!
Figure 2.2: Operational semantics of Idealized Nuprl.

\[
\begin{array}{lllll}
(a:A) \rightarrow B & \lambda a.M & MN & (\lambda a.M)N & M \rightarrow M' \\
(a:A) \times B & (M, N) & \text{fst}(M) & \text{fst}(M') & M \rightarrow M' \\
& & \text{snd}(M) & \text{snd}(M') & \\
\text{nat} & z & s(M) & \text{natrec}(M; Z, n.a.S) & \text{natrec}(M'; Z, n.a.S) \\
\text{natrec}(z; Z, n.a.S) & \rightarrow Z & \text{natrec}(s(M); Z, n.a.S) & \rightarrow S[M/n][\text{natrec}(M; Z, n.a.S)/a] & \\
\text{bool} & \text{true} & \text{false} & M \rightarrow M' & \\
\text{if}(M; T, F) & \rightarrow \text{if}(M'; T, F) & \text{if}(\text{true}; T, F) & \rightarrow T & \text{if}(\text{false}; T, F) & \rightarrow F \\
\text{Prop}_i & \text{Type}_i & \\
\end{array}
\]
These judgments satisfy two key properties easily proven by induction on their definitions. First, if $M$ val, then $M \rightarrow\rightarrow$. (The converse is not the case; counterexamples, such as $\text{fst}(\lambda a.a)$, are called stuck.) Second is determinacy of evaluation: if $M \rightarrow\rightarrow M'$ and $M \rightarrow\rightarrow M''$, then $M' = M''$.

2.2 Constructing type systems

Our programming language contains building blocks of mathematics—functions, pairs, natural numbers, et cetera—which we must assemble into the types and elements of Idealized Nuprl. Roughly speaking, we define a collection $V_0 \subset \text{Tm}$ of closed terms that name types, and assign to each $A \in V_0$ a collection $\text{El}(A) \subset \text{Tm}$ of its closed term elements.

We define $V_0$ inductively. To define the type of Booleans, we specify that $\text{bool} \in V_0$. Clearly true and false denote Booleans, as do all programs evaluating to true or false, such as $(\lambda a.a)$ true; thus $\text{El}($true$) = \{ M \mid M \Downarrow \text{true} \lor M \Downarrow \text{false} \}$. To define the type of equality proofs, we specify that $\text{Eq}(M, N) \in V_0$ when $A$ is a type $(A \in V_0)$ with $M$ and $N$ as elements $(M, N \in \text{El}(A))$—a strange condition, because the definition of $V_0$ now refers to $\text{El}$, a function out of $V_0$!

What are elements of function type? Intuitively, $M \in \text{El}(A \rightarrow B)$ when application of $M$ induces a mathematical function $M_\rightarrow \in \text{El}(A) \rightarrow \text{El}(B)$; for example, $\lambda a.a$ and $\lambda a.\text{if}(a, \text{true}, \text{false})$ are elements of $\text{El}(\text{bool} \rightarrow \text{bool})$ with equal extension, namely, the identity function on $\text{El}(\text{bool})$. A higher-order function behaving differently on these inputs would not denote a mathematical function $(\text{El}(\text{bool}) \rightarrow \text{El}(\text{bool})) \rightarrow \text{El}(A)$ and should therefore not be an element of $\text{El}(\text{bool} \rightarrow \text{bool} \rightarrow A)$.

We must therefore assign to each type not only $\text{El}(-)$ but also an equivalence relation on $\text{El}(-)$, and require that elements of $\text{El}(A \rightarrow B)$ send equal elements of $\text{El}(A)$ to equal elements of $\text{El}(B)$. These equivalence relations are inherently typed: we cannot equate $\lambda a.a$ and $\lambda a.\text{if}(a; \text{true}, \text{false})$ as elements of $\text{Tm}$, as the former is an element of $\text{El}(\text{nat}) \rightarrow \text{El}(\text{nat})$ while the latter is not! Similarly, we must define an equivalence relation on $V_0$ to propagate term equality into type dependencies (for instance, equating $\text{Eq}(\text{bool} \rightarrow \text{bool})((\lambda a.a, M)$ and $\text{Eq}(\text{bool} \rightarrow \text{bool})((\lambda a.\text{if}(a; \text{true}, \text{false}), M$) as types) and must check that $\text{El}$ respects type equality.

Finally, given $V_0$ and $\text{El}$, we construct a second, larger collection of types $V_1$ closed under the same type formers, with an additional base type $\text{Type}_0 \in V_1$ whose elements are types of the previous level $(\text{El}(\text{Type}_0) = V_0)$. We obtain a cumulative hierarchy of universes in $V_\omega$ by iterating this process $V_0 \subset V_1 \subset \cdots$ where $\text{Type}_0, \text{Type}_1, \ldots, \text{Type}_{n-1} \in V_n$.

Actualizing the vision  Defining simultaneously a set $V_i$, an equivalence relation on $V_i$, a map $\text{El} : V_i \rightarrow \text{Set}$, and an equivalence relation on each $\text{El}(A)$ is a process requiring careful justification. As a cautionary tale, recall that type theories with the rule $\text{Type}_i \in \text{Type}_i$ are inconsistent, essentially for the same reason that there is no set of all sets [Hur95]. Such a
principle is not obviously incompatible with the semantics outlined above—in particular, there is no size issue with a construction in which \( \text{Type}_i \in V_i \) and \( \text{El}($\text{Type}_i$) = V_i \). Instead, we will see that such a definition is subtly ill-founded.

Induction-recursion is a generalization of type-theoretic inductive definition in which an inductive type is defined simultaneously with a recursive function out of that type [Dyb00]. Our definition of \( V_i \) can be justified by induction-recursion, and was in fact its motivating example. In this dissertation we opt instead for a fixed point construction described by Allen [All87]: we find Allen’s construction more straightforward on paper and appreciate that it can be carried out in set theory or type theories without induction-recursion. Indeed, Allen’s construction is also used by Nuprl in Coq [AR14], as Coq lacks induction-recursion.

Rather than specifying first a subset of \( Tm \) and then an equivalence relation on that subset, we define both simultaneously as a partial equivalence relation (PER), or a symmetric and transitive binary relation. A partial equivalence relation \( R \) on \( Tm \) corresponds to the subset \( \{ M \in Tm \ | \ R(M, M) \} \) on which it is an equivalence relation; conversely, an equivalence relation \( R \) on a subset of \( Tm \) is a PER on the whole of \( Tm \). Because typehood and membership should respect evaluation, we define them as PERs on values and lift to terms by evaluation. We cannot actually quotient terms by semantic equality at any point during this process because evaluation can distinguish semantically equal terms.

Summing up, we must define a PER of value types, and for each value type, a PER of value elements. Harper [Har92] constructs these as the least fixed point of a monotone function on the pointed directed-complete partial order \((V \cdot \text{PER}(\text{Val})) \times (\text{Val}/V \rightarrow \text{PER}(\text{Val}))\). Unfortunately, the existence of such a fixed point requires a classical cardinality argument, and to make Harper’s construction precise, one must verify the monotone function sends PERs to PERs and respects type equality. Following Allen [All87], we instead achieve ontological reality with a least fixed point on the complete lattice of candidate (or possible) type systems \( \mathcal{P}(\text{Val} \times \text{Val} \times \mathcal{P}(\text{Val} \times \text{Val})) \), and observe after the fact that we have constructed a proper type system.

**Definition 2.1.** A candidate type system is a relation \( \tau(A_0, B_0, \varphi) \) over \( A_0 \text{ val} \), \( B_0 \text{ val} \), and binary relations \( \varphi(M_0, N_0) \) over \( M_0 \text{ val} \) and \( N_0 \text{ val} \).

**Definition 2.2.** A type system is a candidate type system \( \tau \) satisfying:

1. **Unicity:** If \( \tau(A_0, B_0, \varphi) \) and \( \tau(A_0, B_0, \varphi') \) then \( \varphi = \varphi' \).

2. **PER-valuation:** If \( \tau(A_0, B_0, \varphi) \) then \( \varphi \) is symmetric and transitive.

3. **Symmetry:** If \( \tau(A_0, B_0, \varphi) \) then \( \tau(B_0, A_0, \varphi) \).

4. **Transitivity:** If \( \tau(A_0, B_0, \varphi) \) and \( \tau(B_0, C_0, \varphi) \) then \( \tau(A_0, C_0, \varphi) \).
The relation \( \tau(A_0, B_0, \varphi) \) encodes that \( A_0 \) and \( B_0 \) are equal value types with element relation \( \varphi \). The conditions of Definition 2.2 ensure that types have unique element relations, and that type and element equality are PERs.

Our construction uses the Knaster–Tarski fixed point theorem, which states that any monotone (order-preserving) function \( F(x) \) on a complete lattice has a least fixed point \( \mu x.F(x) \) that is also its least pre-fixed point [DP02, 2.35]. We consider monotone functions on the complete lattice of candidate type systems ordered by inclusion, built using the following combinator that resurface as judgments in Definitions 2.12 and 2.20:

**Definition 2.3** (Candidate judgments). Given a candidate type system \( \tau \):

1. \( A \sim A' \downarrow \alpha \in \tau \) when \( A \triangleright A_0, A' \triangleright A'_0, \) and \( \tau(A_0, A'_0, \alpha) \).
2. \( M \sim M' \in \alpha \) when \( M \triangleright M_0, M' \triangleright M'_0, \) and \( \alpha(M_0, M'_0) \).
3. \( a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau \) when for all \( M \sim M' \in \alpha, B[M/a] \sim B'[M'/a] \downarrow \beta_{M,M'} \in \tau \).
4. \( a : \alpha \triangleright N \sim N' \in \beta \) when for all \( M \sim M' \in \alpha, N[M/a] \sim N'[M'/a] \in \beta_{M,M'} \).

In Figure 2.3, we define each type former as a monotone function on candidate type systems. For instance, \( \text{Pair}(\tau) \) is a candidate type system whose types are all possible dependent pairs of types and type families of \( \tau \), defined by formation and introduction rules. Base types \( \text{Nat} \) and \( \text{Bool} \) take no input because their meaning is constant in \( \tau \). Candidate type system \( \tau_0 = \mu \tau.\text{Types}(\varnothing, \tau) \) is the least fixed point of the union (disjoint by construction) of the type formers, and is therefore the least candidate type system closed under the non-universe type formers. As with \( \text{V}_0 \) earlier, \( \tau_0 \) will serve as universe \( \text{Type}_0 \) of the final type system.

Idealized Nuprl has a second universe hierarchy \( \text{Prop}_i \subset \text{Type}_i \) whose elements are *subsingletons*, types all of whose elements are equal. We construct \( \pi_0 = \mu \pi.\text{Props}(\pi, \tau_0) \) as the least fixed point of the type formers that preserve subsingletons: \( \text{Eq}_A(M, N) \) for any \( A \) (drawn from \( \tau_0 \)), \( (a:A) \times B(a) \) for subsingletons \( A \) and \( B(a) \) (both drawn from \( \pi_0 \)), and (\( (a:A) \rightarrow B \)) for \( A \) in \( \tau_0 \) and \( B(a) \) in \( \pi_0 \).

We define a third candidate type system, \( \nu_1 \), containing only two types—\( \text{Type}_0 \) whose elements are types of \( \tau_0 \), and \( \text{Prop}_0 \) whose elements are types of \( \pi_0 \)—and then a fourth candidate type system, \( \tau_1 \), the least fixed point of the type formers and the universes of \( \nu_1 \). We iterate this process for all \( i \in \{0, 1, \ldots, \omega\} \) to obtain three sequences of candidate type systems: \( \tau_i \) of all types with universes \( < i \), \( \pi_i \) of subsingleton types with universes \( < i \), and \( \nu_i \) of only universes \( < i \).

---

2 Type systems do not form a complete lattice, as unicity is not preserved by joins.
3 Standard logical relations define function and pair types not by introduction but rather elimination: one demands not that \( M \in (a:A) \times B(a) \) evaluate to a pair, but instead that \( \text{fst}(M) \in A \) and \( \text{snd}(M) \in B(\text{fst}(M)) \).

Either definition is workable in this chapter, but relying on introduction forms will be crucial in Chapter 4.
\[
\text{Fun}(\tau, \pi) = \{(a:A) \to B, (a:A') \to B', \varphi) | \\
\exists \alpha, \beta. (A \sim A' \downarrow \alpha \in \tau) \land (a : \alpha \triangleright B \sim B' \downarrow \beta \in \pi) \\
\land \varphi = \{((\lambda a.N, \lambda a.N') | a : \alpha \triangleright N \sim N' \in \beta}\}
\]

\[
\text{Pair}(\tau) = \{((a:A) \times B, (a:A') \times B', \varphi) | \\
\exists \alpha, \beta. (A \sim A' \downarrow \alpha \in \tau) \land (a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau) \\
\land \varphi = \{((M, N), (M', N')) | (M \sim M' \in \alpha) \land (N \sim N' \in \beta)\}\}
\]

\[
\text{Eq}(\tau) = \{((\text{Eq}_A(M, N), \text{Eq}_{A'}(M', N'), \varphi) | \\
\exists \alpha. (A \sim A' \downarrow \alpha \in \tau) \land (M \sim M' \in \alpha) \land (N \sim N' \in \alpha) \\
\land (\varphi = \{(\varnothing, \varnothing) | M \sim N \in \alpha)\}\}
\]

\[
\text{Nat} = \{(\text{nat}, \text{nat}, \varphi) | \varphi = \mu R. (\{(z, z)\} \cup \{(s(M), s(M')) | M \sim M' \in R)\})\}
\]

\[
\text{Bool} = \{(\text{bool}, \text{bool}, \varphi) | \varphi = \{(\text{true}, \text{true}), (\text{false}, \text{false})\}\}
\]

\[
\text{Prop}(\nu) = \{(\text{Prop}_i, \text{Prop}_i, \varphi) | \text{Prop}_i \sim \text{Prop}_i \downarrow \varphi \in \nu\}
\]

\[
\text{Type}(\nu) = \{(\text{Type}_i, \text{Type}_i, \varphi) | \text{Type}_i \sim \text{Type}_i \downarrow \varphi \in \nu\}
\]

\[
\text{Props}(\pi, \tau) = \text{Fun}(\tau, \pi) \cup \text{Pair}(\pi) \cup \text{Eq}(\tau)
\]

\[
\text{Types}(\nu, \tau) = \text{Fun}(\tau, \nu) \cup \text{Pair}(\tau) \cup \text{Eq}(\nu) \cup \text{Nat} \cup \text{Bool} \cup \text{Prop}(\nu) \cup \text{Type}(\nu)
\]

\[
\nu_n = \{\text{Prop}_i, \text{Prop}_i, \varphi) | (i < n) \land (\varphi = \{(A_0, B_0) | \exists \alpha.A_0 \sim B_0 \downarrow \alpha \in \pi_i)\}\}
\]

\[
\cup \{(\text{Type}_i, \text{Type}_i, \varphi) | (i < n) \land (\varphi = \{(A_0, B_0) | \exists \alpha.A_0 \sim B_0 \downarrow \alpha \in \tau_i)\}\}
\]

\[
\pi_n = \mu \pi. \text{Props}(\pi, \nu_n)
\]

\[
\tau_n = \mu \tau. \text{Types}(\nu_n, \tau)
\]

Figure 2.3: PER semantics of Idealized Nuprl.
Suppose we had instead considered a single universe $\text{Type} \in \text{Type}$ containing itself:

$$\text{Types}(\tau) = \text{Fun}(\tau, \tau) \cup \cdots \cup \{(\text{Type}, \text{Type}, \{(A_0, B_0) \mid \tau(A_0, B_0, \varphi)\})\}$$

Such a $\text{Types}$ function is not monotone, since adding types to $\tau$ does not preserve the elements of Type, so we cannot consider its least fixed point. Readers who prefer type theory as a metatheory can read Figure 2.3 quite directly as an inductive definition of $\tau_i, \pi_i : \text{Val} \times \text{Val} \times (\text{Val} \times \text{Val} \rightarrow \text{Prop}) \rightarrow \text{Prop}$ for an impredicative $\text{Prop}$, in which case the above $\text{Types}$ would be insufficiently positive. In the absence of impredicative $\text{Prop}$, one can define $n$ levels of Idealized Nuprl with $(n + 1)$ predicative universes [AR14].

The rest of this section is devoted to proofs that $\tau_i, \pi_i$ are type systems (Theorem 2.8) and that $\pi_i \subseteq \tau_i$ (Theorem 2.11). We hope some readers will find these proofs instructive, as the author was unable to locate any paper versions of these proofs longer than “By induction.” The uninterested reader may nevertheless safely skip to Section 2.3.

**Lemma 2.4.** Let $F(R) = \{(z, z)\} \cup \{(s(M), s(M')) \mid M \sim M' \in R\}$. Then $\mathbb{N} = \mu R.F(R)$ is symmetric and transitive.

**Proof.**

1. **Symmetry.**

   Let $\Phi = \{(M'_0, M_0) \mid \mathbb{N}(M_0, M'_0)\}$, and show that $\Phi$ is a pre-fixed point of $F$ (that is, $F(\Phi) \subseteq \Phi$); $\mathbb{N}$ is the least pre-fixed point of $F$, so $\mathbb{N} \subseteq \Phi$, and thus $\mathbb{N}$ is symmetric.

   There are two cases. If $F(\Phi)(z, z)$, then we must show $\Phi(z, z)$, which is trivial. If $F(\Phi)(s(M'), s(M))$, then $M' \sim M \in \Phi$ and hence $M \sim M' \in \mathbb{N}$; we must show $\mathbb{N}(s(M), s(M'))$, which follows by $\mathbb{N} = F(\mathbb{N})$.

2. **Transitivity.**

   Let $\Phi = \{(M_0, M'_0) \mid \forall M'_0.\mathbb{N}(M'_0, M''_0) \Rightarrow \mathbb{N}(M_0, M''_0)\}$, and show that $F(\Phi) \subseteq \Phi$; it follows that $\mathbb{N} \subseteq \Phi$, and therefore that $\mathbb{N}$ is transitive.

   Again, there are two cases. If $F(\Phi)(z, z)$, we must show $\Phi(z, z)$, that is, for all $M_0$ such that $\mathbb{N}(z, M_0)$ we have $\mathbb{N}(z, M_0)$, which is trivial. If $F(\Phi)(s(M), s(M'))$ (that is, $M \sim M' \in \Phi$), we must show $\Phi(s(M), s(M'))$; for all values $M''_0$ such that $\mathbb{N}(s(M'), M''_0)$, we must show $\mathbb{N}(s(M), M''_0)$. But then $M''_0 = s(M'')$ and $M' \sim M'' \in \mathbb{N}$ for some $M''$, so by $M \sim M' \in \Phi$, $M \sim M'' \in \mathbb{N}$. The result follows by $\mathbb{N} = F(\mathbb{N})$.  

Notice that $\mathbb{N}$ and $\Phi$ are defined only on values, but the successor case of $F$ forces us to confront non-values as well. In (1), $M' \sim M \in \Phi$ implies $M \sim M' \in \mathbb{N}$ not immediately by definition, but by observing that $M \nmid M_0, M' \nmid M'_0$, and $\Phi(M'_0, M_0)$, hence $\mathbb{N}(M_0, M'_0)$, hence $M \sim M' \in \mathbb{N}$. The next proofs are complicated further by $\text{Fun}$ and $\text{Pair}$, which lift inductive hypotheses to families of open terms; in the proof of Lemma 2.6, this lifting forces us to simultaneously prove transitivity on the left and right.
Lemma 2.5. If $v$ is a type system, then $\mu \text{Types}(v) = \mu \alpha. \text{Types}(v, \alpha)$ satisfies unicity.

Proof. Let $\Phi = \{(A_0, B_0, \varphi) \mid \forall \varphi'. \mu \text{Types}(v)(A_0, B_0, \varphi) \implies (\varphi = \varphi')\}$ and prove $\Phi$ is a pre-fixed point of $\text{Types}(v, \cdot)$ (that is, $\text{Types}(v, \Phi) \subseteq \Phi$). Because $\mu \text{Types}(v)$ is the least pre-fixed point of $\text{Types}(v, \cdot)$, it will follow that $\mu \text{Types}(v) \subseteq \Phi$, and therefore that $\mu \text{Types}(v)$ satisfies unicity. We can consider each type former separately because they are disjoint. We will write out the case for $\text{Fun}$ carefully; $\text{Pair}$ and $\text{Eq}$ are similar, $\text{Nat}$ and $\text{Bool}$ are immediate, and $\text{Prop}$ and $\text{Type}$ follow from the unicity of $v$.

Consider the $\text{Fun}$ clause of $\text{Types}(v, \Phi)$: suppose $\text{Fun}(\Phi, \Phi)((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$ and show $\Phi((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$, that is, if $\mu \text{Types}(v)((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi')$ then $\varphi = \varphi'$. Unrolling $\mu \text{Types}(v)$, we have $A \sim A' \downarrow \alpha' \in \mu \text{Types}(v)$ and $a : \alpha' \mapsto B \sim B' \downarrow \beta' \in \mu \text{Types}(v)$ for some $\alpha', \beta'$. Unrolling $\text{Fun}(\Phi, \Phi)$, we have $A \sim A' \downarrow \alpha \in \Phi$, or for any $\alpha''$ such that $A \sim A' \downarrow \alpha'' \in \mu \text{Types}(v)$, $\alpha = \alpha''$. Such an $\alpha''$ exists (namely, $\alpha'$), so it is unique. Similarly, by $a : \alpha \mapsto B \sim B' \downarrow \beta \in \Phi$, we know that for any $M, M'$ with $M \sim M' \in \alpha$, there is a unique $\beta'_{M,M'}$ such that $B[M/a] \sim B'[M'/a] \downarrow \beta'_{M,M'} \in \mu \text{Types}(v)$. Because $\varphi, \varphi'$ are determined by the choice of $\alpha'$ and those $\beta'_{M,M'}$ we conclude that $\varphi = \varphi'$.

Lemma 2.6. If $v$ is a type system, then $\mu \text{Types}(v) = \mu \alpha. \text{Types}(v, \alpha)$ is a type system.

Proof. We must prove PER-valuation, symmetry, and transitivity simultaneously. Let

$$
\Phi(A_0, B_0, \varphi) = \{\mu \text{Types}(v)(A_0, B_0, \varphi) \land \varphi \text{ is a PER} \land \mu \text{Types}(v)(B_0, A_0, \varphi) \\
\land (\forall C_0, \varphi'. \mu \text{Types}(v)(B_0, C_0, \varphi') \implies \mu \text{Types}(v)(A_0, C_0, \varphi) \land \varphi = \varphi') \\
\land (\forall C_0, \varphi'. \mu \text{Types}(v)(C_0, A_0, \varphi') \implies \mu \text{Types}(v)(A_0, C_0, \varphi) \land \varphi = \varphi')\}
$$

We prove $\text{Types}(v, \Phi) \subseteq \Phi$; it will follow that $\mu \text{Types}(v) \subseteq \Phi$, and by Lemma 2.5 that $\mu \text{Types}(v)$ is a type system. (Surprisingly, unicity figures nowhere else in this proof!) Suppose $\text{Fun}(\Phi, \Phi)((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$. Unfolding definitions,

1. $A \sim A' \downarrow \alpha \in \mu \text{Types}(v)$,
2. $\alpha$ is a PER,
3. $A' \sim A \downarrow \alpha \in \mu \text{Types}(v)$,
4. if $A' \sim A'' \downarrow \alpha' \in \mu \text{Types}(v)$ then $A \sim A'' \downarrow \alpha \in \mu \text{Types}(v)$ and $\alpha = \alpha'$,
5. if $A'' \sim A \downarrow \alpha' \in \mu \text{Types}(v)$ then $A'' \sim A \downarrow \alpha \in \mu \text{Types}(v)$ and $\alpha = \alpha'$,
6. for all $M \sim M' \in \alpha$, $B[M/a] \sim B'[M'/a] \downarrow \beta_{M,M'} \in \mu \text{Types}(v)$,
7. for all $M \sim M' \in \alpha$, $\beta_{M,M'}$ is a PER,
8. for all $M \sim M' \in \alpha$, $B'[M'/a] \sim B[M/a] \downarrow \beta_{M,M'} \in \mu \text{Types}(v)$,
9. for all $M \sim M' \in \alpha$, if $B'[M'/a] \sim C \downarrow \beta'_{M,M'} \in \mu_{\text{Types}}(v)$ then $B[M/a] \sim C \downarrow \beta_{M,M'} \in \mu_{\text{Types}}(v)$ and $\beta_{M,M'} = \beta'_{M,M'}$.

10. for all $M \sim M' \in \alpha$, if $C \sim B[M/a] \downarrow \beta_{M,M'} \in \mu_{\text{Types}}(v)$ then $C \sim B'[M'/a] \downarrow \beta_{M,M'} \in \mu_{\text{Types}}(v)$ and $\beta_{M,M'} = \beta'_{M,M'}$, and

11. $\varphi = \{(\lambda a.N, \lambda a.N') \mid a : \alpha \triangleright N \sim N' \in \beta\}$.

We prove each component of $\Phi((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$ separately:

1. We have $\mu_{\text{Types}}(v)((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$ by (1,6,11).

2. PER-valuation.

   To prove $\varphi$ is transitive, suppose that $a : \alpha \triangleright N \sim N' \in \beta$, $a : \alpha \triangleright N' \sim N'' \in \beta$, and $M \sim M' \in \alpha$, and show $N[M/a] \sim N''[M'/a] \in \beta_{M,M'}$. Because $\alpha$ is a PER, $M \sim M \in \alpha$, so $N[M/a] \sim N''[M'/a] \in \beta_{M,M}$ and $N'[M/a] \sim N''[M'/a] \in \beta_{M,M'}$. We know $\beta_{M,M'}$ is transitive, so it suffices to show that $\beta_{M,M} = \beta_{M,M'}$:

   \[
   \begin{align*}
   B'[M'/a] \sim B[M/a] \downarrow \beta_{M,M} & \in \mu_{\text{Types}}(v) & \text{by (8)} \\
   B[M/a] \sim B'[M'/a] \downarrow \beta_{M,M'} & \in \mu_{\text{Types}}(v) & \text{by (6)} \\
   \beta_{M,M} = \beta_{M,M'} & \text{by (10)}
   \end{align*}
   \]

   To prove $\varphi$ is symmetric, suppose that $a : \alpha \triangleright N \sim N' \in \beta$ and $M \sim M' \in \alpha$, and show $N'[M/a] \sim N[M'/a] \in \beta_{M,M'}$. Because $\alpha$ is a PER, $M' \sim M \in \alpha$, so $N'[M/a] \sim N[M'/a] \in \beta_{M,M}$. We know $\beta_{M,M'}$ is symmetric, so it suffices to show that $\beta_{M,M} = \beta_{M,M'}$:

   \[
   \begin{align*}
   B[M/a] \sim B'[M'/a] \downarrow \beta_{M,M} & \in \mu_{\text{Types}}(v) & \text{by (6)} \\
   B'[M'/a] \sim B[M/a] \downarrow \beta'_{M,M} & \in \mu_{\text{Types}}(v) & \text{by (8)} \\
   \beta'_{M',M} = \beta_{M,M} = \beta_{M,M'} & \text{by (9), } \beta_{M,M} = \beta_{M,M'}
   \end{align*}
   \]


   We must show $\mu_{\text{Types}}(v)((a:A) \rightarrow B', (a:A) \rightarrow B, \varphi)$. By (3) we have $A' \sim A \downarrow \alpha \in \mu_{\text{Types}}(v)$. It suffices to show $a : \alpha \triangleright B' \sim B' \downarrow \beta \in \mu_{\text{Types}}(v)$. Suppose $M \sim M' \in \alpha$. By (2,8) we have $B'[M'/a] \sim B[M'/a] \downarrow \beta_{M',M} \in \mu_{\text{Types}}(v)$, and by the above argument, $\beta_{M',M} = \beta_{M,M'}$.

4. Right transitivity.

   Suppose $\mu_{\text{Types}}(v)((a:A') \rightarrow B', C_0, \varphi')$, and show $\mu_{\text{Types}}(v)((a:A) \rightarrow B, C_0, \varphi)$ and $\varphi = \varphi'$. By inspecting the definition of Types, we conclude $C_0 = \mu_{\text{Types}}(v)((a:A'')) \rightarrow B''$, if $A'' \sim A \downarrow \alpha \in \mu_{\text{Types}}(v)$.
\( A' \sim A'' \downarrow \alpha' \in \mu_{\text{types}}(v) \), and \( a : \alpha' \triangleright B' \sim B'' \downarrow \beta' \in \mu_{\text{types}}(v) \). By (4), \( A \sim A'' \downarrow \alpha \in \mu_{\text{types}}(v) \) and \( \alpha = \alpha' \). It suffices to show \( a : \alpha \triangleright B \sim B'' \downarrow \beta \in \mu_{\text{types}}(v) \). Suppose \( M \sim M' \in \alpha \).

\[
\begin{align*}
B[M/a] & \sim B'[M/a] \downarrow \beta_{M,M} \in \mu_{\text{types}}(v) & \text{by (6)} \\
B'[M/a] & \sim B''[M'/a] \downarrow \beta'_{M,M'} \in \mu_{\text{types}}(v) & \text{by hypothesis} \\
B[M/a] & \sim B''[M'/a] \downarrow \beta_{M,M'} \in \mu_{\text{types}}(v) & \text{by (9), } \beta_{M,M} = \beta_{M,M'}
\end{align*}
\]

5. Left transitivity.

Suppose \( \mu_{\text{types}}(v)(C_0, (a:A) \rightarrow B, \phi') \), and show \( \mu_{\text{types}}(v)(C_0, (a:A') \rightarrow B', \varphi) \) and \( \varphi = \phi' \). By the definition of \( \text{types}, C_0 = (a:A'') \rightarrow B'', A'' \sim A \downarrow \alpha' \in \mu_{\text{types}}(v) \), and \( a : \alpha' \triangleright B'' \sim B \downarrow \beta' \in \mu_{\text{types}}(v) \). By (5), \( A'' \sim A' \downarrow \alpha \in \mu_{\text{types}}(v) \) and \( \alpha = \alpha' \). To show \( a : \alpha \triangleright B'' \sim B' \downarrow \beta \in \mu_{\text{types}}(v) \), suppose \( M \sim M' \in \alpha \).

\[
\begin{align*}
B''[M/a] & \sim B[M'/a] \downarrow \beta'_{M,M'} \in \mu_{\text{types}}(v) & \text{by hypothesis} \\
B[M'/a] & \sim B'[M'/a] \downarrow \beta_{M',M'} \in \mu_{\text{types}}(v) & \text{by (6)} \\
B''[M/a] & \sim B'[M'/a] \downarrow \beta_{M',M'} \in \mu_{\text{types}}(v) & \text{by (10)}
\end{align*}
\]

It remains only to show \( \beta_{M',M'} = \beta_{M,M'} \):

\[
\begin{align*}
B[M/a] & \sim B'[M'/a] \downarrow \beta_{M,M} \in \mu_{\text{types}}(v) & \text{by (6)} \\
B'[M'/a] & \sim B[M'/a] \downarrow \beta_{M',M'} \in \mu_{\text{types}}(v) & \text{by (8)} \\
\beta_{M',M'} & = \beta_{M,M'} & \text{by (9)}
\end{align*}
\]

The cases for \( \text{Pair} \) and \( \text{Eq} \) are very similar. \( \text{PER-valuation} \) follows from Lemma 2.4 for \( \text{Nat} \), trivially for \( \text{Bool} \), and from \( \text{PER-valuation of } v \) for \( \text{Prop} \) and \( \text{Type} \). Symmetry, left transitivity, and right transitivity for \( \text{Nat} \), \( \text{Bool} \), \( \text{Prop} \), and \( \text{Type} \) are immediate. \( \square \)

**Lemma 2.7.** If \( \tau \) is a type system, then \( \mu_{\text{props}}(\tau) = \mu.\text{props}(\pi, \tau) \) is a type system.

**Proof.** The proof follows those of Lemmas 2.5 and 2.6. The \( \text{Pair} \) case is identical. In the \( \text{Eq} \) case, the index type is drawn from \( \tau \) and not \( \pi \), so we have no inductive hypotheses and instead observe that the desired properties follow from those of \( \tau \). In the \( \text{Fun} \) case, the domain is drawn from \( \tau \) while the codomain type is drawn from \( \pi \), so we only have inductive hypotheses arising from \( a : \alpha \triangleright B \sim B' \downarrow \beta \in \pi \) and \( a : \alpha \triangleright N \sim N' \in \beta \). \( \square \)

**Theorem 2.8.** For all \( i \in \{0, 1, \ldots, \omega\} \), \( v_i \), \( \tau_i \), and \( \pi_i \) are type systems.

**Proof.** By strong induction on \( i \). In the base case, \( v_0 \) is empty and thus a type system; \( \tau_0 = \mu_{\text{types}}(v_0) \) is a type system by Lemma 2.6; and \( \pi_0 = \mu_{\text{props}}(\tau_0) \) is a type system by
Lemma 2.7. In the inductive case, suppose that $\nu_i, \tau_i, \pi_i$ are type systems for all $i < n$. Then $\nu_n$ is a type system: unicity, symmetry, and transitivity are immediate, and PEr-valuation follows from symmetry and transitivity of $\tau_i$ and $\pi_i$ for $i < n$. Once again, $\tau_n$ and $\pi_n$ are type systems by Lemmas 2.6 and 2.7. (The $\omega$ case is identical.) □

Finally, to prove our universes are cumulative ($\text{Type}_i \subseteq \text{Type}_j$ and $\text{Prop}_i \subseteq \text{Prop}_j$) we must observe that the sequences of type systems by Lemmas 2.6 and 2.7. (The case is identical.)

Lemma 2.9. In any complete lattice, if $F(x)$ and $G(x)$ are monotone and $F(x) \subseteq G(x)$ for all $x$, then $\mu x.F(x) \subseteq \mu x.G(x)$.

Proof. $\mu x.G(x)$ is a pre-fixed point of $F$ because $F(\mu x.G(x)) \subseteq G(\mu x.G(x)) = \mu x.G(x)$. But $\mu x. F(x)$ is the least such, so $\mu x. F(x) \subseteq \mu x.G(x)$. □

Lemma 2.10. In any complete lattice, if $F(x, y)$ and $G(x, y)$ are monotone and $F(x, y) \subseteq G(x, y)$ whenever $x \leq y$, then $\mu F \leq \mu G$ where $(\mu F, \mu G) = \mu(x, y). (F(x, y), G(x, y))$.

Proof. Let $\mu(, ) = \mu F \cap \mu G$. $(\mu(, ))$ is a pre-fixed point of $(x, y) \mapsto (F(x, y), G(x, y))$ because, by assumption and $(\mu F, \mu G)$ being a fixed point, $F(\mu(, )) \subseteq F(\mu F, \mu G) = \mu F$ and $G(\mu(, )) \subseteq G(\mu F, \mu G) = \mu G$. This implies $(\mu F, \mu G) \subseteq (\mu(, ))$ and thus $\mu F \subseteq \mu G$. □

Theorem 2.11. For $i, j \in \{0, 1, \ldots, \omega\}$ where $i \leq j$, we have $\tau_i \subseteq \tau_j$, $\pi_i \subseteq \pi_j$, and $\pi_i \subseteq \pi_j$.

Proof. The functions $\text{Types}$ and $\text{Props}$ are monotone in both arguments, so by Lemma 2.9, $\mu\text{Types}$ and $\mu\text{Props}$ are also monotone. When $i \leq j$ we have $\nu_i \subseteq \nu_j$ by construction, so $\tau_i = \mu\text{Types}(\nu_i) \subseteq \mu\text{Types}(\nu_j) = \tau_j$ and $\pi_i = \mu\text{Props}(\tau_i) \subseteq \mu\text{Props}(\tau_j) = \pi_j$.

By a theorem of Bekić [Bek84, p. 39] on simultaneous fixed points, for all $\nu$,

$$\mu(\pi, \tau). (\text{Props}(\pi, \tau), \text{Types}(\nu, \tau)) = (\mu\text{Props}(\mu\text{Types}(\nu)), \mu\text{Types}(\nu))$$

When $\pi \subseteq \tau$, $\text{Props}(\pi, \tau) \subseteq \text{Types}(\nu, \tau)$; thus $\pi_i \subseteq \tau_i$ follows by Lemma 2.10. □

2.3 Closed and open judgments

Unlike the candidate judgments of Section 2.2, the judgments of Idealized Nuprl contain only terms (not also relations $\alpha, \beta$) and define typehood and membership under arbitrarily many hypotheses (not only zero or one). We first define the closed judgments of type and element equality under no hypotheses.

Definition 2.12 (Closed judgments). Given a type system $\tau$:

1. $A \vdash B$ type when $A \sim B \downarrow \alpha \in \tau$.\(^4\)

\(^4\)We adopt the $\vdash$ notation because homotopy type theorists commonly use $=$ for homotopy, not equality.
2. \( M \doteq N \in A \), presupposing \( A \doteq A \) type, when \( A \sim A \downarrow \alpha \in \tau \) and \( M \sim N \in \alpha \).

A presupposition is an assumption required to make sense of a judgment. The statement “\( M \) and \( N \) are equal elements of \( A \)” is only well-formed if \( A \) is a type (and hence equipped with a notion of equal element). Whenever we assert \( M \doteq N \in A \), we implicitly assert \( A \) is a type; whenever \( A \) is a type, we will write \( \llbracket A \rrbracket \) for its PER of elements (\( \alpha \) in Definition 2.12).

The closed judgments are symmetric and transitive by the corresponding properties of type systems. As a corollary, \( A \doteq A \) type whenever \( A \doteq B \) type; by unicity of type systems, both judgments give rise to the same \( \llbracket A \rrbracket \). We henceforth abbreviate \( A \doteq A \) type by \( A \) type, and \( M \doteq M \in A \) by \( M \in A \).

Lemma 2.13. If \( A \doteq B \) type and \( M \doteq N \in A \) then \( M \doteq N \in B \).

Proof. By unicity of type systems, \( \llbracket A \rrbracket = \llbracket B \rrbracket \); \( M \sim N \in \llbracket A \rrbracket \) implies \( M \sim N \in \llbracket B \rrbracket \). □

Definition 2.12 is parametrized by a choice of type system serving as an ambient universe of types. The type system \( \tau_\omega \) constructed in Section 2.2 is the ambient universe of Idealized Nuprl, but we must also consider \( \tau_i, \pi_i \) to prove properties of its internal universes Type\(_i\), Prop\(_i\). We write \( \tau \models J \) for any judgment \( J \) to make the choice of \( \tau \) explicit; when unspecified, judgments are relative to \( \tau_\omega \).

Lemma 2.14. If \( A \doteq B \in \text{Prop}_i \) or \( A \doteq B \in \text{Type}_i \) then \( A \doteq B \) type.

Proof. Suppose \( \tau_\omega \models (A \doteq B \in \text{Prop}_i) \); then \( A \sim B \downarrow \alpha \in \pi_i \). By Theorem 2.11, \( \pi_i \subseteq \tau_\omega \), so \( A \sim B \downarrow \alpha \in \tau_\omega \), and thus \( A \doteq B \) type. Similarly, when \( \tau_\omega \models (A \doteq B \in \text{Type}_i) \), we have \( A \sim B \downarrow \alpha \in \tau_i \) and \( \tau_i \subseteq \tau_\omega \). □

Types whose elements are types (as in Lemma 2.14) are known as universes à la Russell, in contrast to universes à la Tarski, whose elements are the names of types (\( \hat{\text{nat}} \in \text{Type}_i \) and \( \text{El}(\hat{\text{nat}}) \doteq \text{nat} \) type) [ML84]. The distinction is blurred in computational semantics, because elements of universes are the names of types—\( \llbracket \text{Type}_i \rrbracket \) contains nat, not \( \llbracket \text{nat} \rrbracket \)—but the typehood judgment also concerns the names of types!

The signature feature of Nuprl is that its judgments range over untyped terms, allowing users to establish typehood and membership judgments by untyped evaluation—for instance, \( A \) type by \( A \downarrow A_0 \) and \( A_0 \) type. In contrast, the judgments of most type theories range over structurally well-typed terms, but are closed under evaluation—if \( A \) type (hence \( A \doteq A \) type) and \( A \downarrow A_0 \), then \( A \doteq A_0 \) type.

Lemma 2.15 (Head expansion).

1. If \( A' \) type and \( A \mapsto^* A' \), then \( A \doteq A' \) type.

2. If \( M' \in A \) and \( M \mapsto^* M' \), then \( M \doteq M' \in A \).
Proof. For part (1), $A'$ type implies $A' \Downarrow A_0$ and $r_o(A_0, A_0, \alpha)$. By determinacy of evaluation, $A \Downarrow A_0$ as well, and therefore $A \equiv A'$ type by definition. Part (2) follows similarly. □

**Lemma 2.16 (Evaluation).** If $M \in A$ then $M \equiv M_0 \in A$ where $M \Downarrow M_0$.

Proof. By definition, $M \Downarrow M_0$ and $\llbracket A \rrbracket(M_0, M_0)$, and therefore $M \equiv M_0 \in A$. □

**Open judgments** The open judgments of type and element equality under hypotheses express that an equality holds as a closed judgment for all instantiations by elements (proofs) of the hypotheses. We begin by defining lists of hypotheses and their instantiations.

**Definition 2.17.** A **telescope** is either

1. nil, written $\cdot$, or
2. $\text{cons}(A, a, \Gamma)$, written $(a : A, \Gamma)$, for a term $A$ and telescope $\Gamma$.

**Definition 2.18.** For $\gamma, \gamma'$ lists and $\Gamma$ a telescope, $\gamma \sim \gamma' \in \Gamma$ when

1. $\cdot \sim \cdot \in \cdot$, or
2. $(M, \gamma) \sim (M', \gamma') \in (a : A, \Gamma)$ when $M \equiv M' \in A$ and $\gamma \sim \gamma' \in \Gamma[M/a]$.

A telescope is a cons list whose tail has a bound variable, allowing later hypotheses to be parametrized over earlier ones. Telescopes are abstract binding trees and inherit notions of $\alpha$-equivalence and substitution $\Gamma[M/a]$. The relation $\gamma \sim \gamma' \in \Gamma$, which Crary [Cra98, p. 56] calls **assignment similarity**, expresses that $\gamma, \gamma'$ are pointwise equal in the types of $\Gamma$; however, because $\Gamma[M/a]$ and $\Gamma[M'/a]$ need not have equal elements when $M \equiv M' \in A$, assignment similarity is neither symmetric nor transitive! Following Martin-Löf [ML82], we restrict attention to **contexts** $(a : A, \Gamma)$ in which the telescope $\Gamma(a)$ respects equality in $A$. (Allen [All87] has a rather different definition of context, as we discuss in Section 2.6.)

**Definition 2.19 (Contexts).** For $\Gamma, \Gamma'$ telescopes, $\Gamma \equiv \Gamma'$ **ctx** when

1. $\cdot \equiv \cdot \in \cdot$, or
2. $(a : A, \Gamma) \equiv (a : A', \Gamma') \text{ ctx}$ when $A \equiv A'$ type and for all $M \equiv M' \in A$, $\Gamma[M/a] \equiv \Gamma'[M'/a] \text{ ctx}$.

Thus, as in the categorical semantics of type theory, a context $\Gamma$ is simply a nested dependent pair type written as a telescope, an assignment $\gamma \sim \gamma \in \Gamma$ is an element of that type written as a list, and an open judgment $\Gamma \Rightarrow J$ is a dependent function $(\gamma:\Gamma) \rightarrow J(\gamma)$.

**Definition 2.20 (Open judgments).**
Lemma 2.21 (Context extension). If \( \Gamma \vdash \Gamma' \) ctx and \( \Gamma \Rightarrow A \equiv A' \) type then \( (\Gamma, a : A) \Rightarrow (\Gamma', a : A') \) ctx.

Proof. The telescope \((\Gamma, a : A)\) is obtained by concatenating \(\Gamma\) and \(\text{cons}(A, a.\text{nil})\), and can be defined by recursion on \(\Gamma\). We prove the lemma by induction on \(\Gamma \vdash \Gamma' \) ctx. If \(\Gamma = \Gamma' = \cdot\), we must show that if \(A \equiv A'\) type then \((a : A) \equiv (a : A')\) ctx, which is immediate.

Now suppose \((b : B, \Gamma) \vdash (b : B', \Gamma')\) ctx and \(b : B, \Gamma \Rightarrow A \equiv A'\) type, and show \((b : B, \Gamma), a : A) \equiv ((b : B', \Gamma'), a : A')\) ctx. By definition, \((b : B, \Gamma), a : A) = (b : B, (\Gamma, a : A))\). We know \(B \equiv B'\) type, so it suffices to show for any \(N \equiv N' \in B\) that \((\Gamma[N/b], a : A[N/b]) \equiv (\Gamma'[N'/b], a : A'[N'/b])\) ctx. The result follows by the inductive hypothesis, \(\Gamma[N/b] = \Gamma'[N'/b]\) ctx, and \(\Gamma[N/b] \Rightarrow A[N/b] \equiv A'[N'/b]\) type (by unrolling the definition of \(b : B, \Gamma \Rightarrow A \equiv A'\) type).

Judgments satisfy the structural rules of hypothesis, weakening, and substitution.

Lemma 2.22 (Hypothesis). If \((\Gamma, a : A, \Delta) \Rightarrow \) then \(\Gamma, a : A, \Delta \Rightarrow a \in A\).

Proof. We establish the presupposition \((\Gamma, a : A, \Delta) \Rightarrow A\) type by induction on \(\Gamma\), unrolling \((\Gamma, a : A, \Delta) \Rightarrow \) ctx to obtain \(A \equiv A'\) type for all \(\gamma \sim \gamma' \in \Gamma\), and observing that \(a\) and the variables of \(\Delta\) cannot occur in \(A\). Now suppose \((\gamma, M, \delta) \sim (\gamma', M', \delta') \in (\Gamma, a : A, \Delta)\) (hence \(\gamma \sim \gamma' \in \Gamma\) and \(M \equiv M' \in A\gamma\)) and show \(M \equiv M' \in A\gamma\), which is immediate.
Lemma 2.23 (Weakening). Supposing \( \Gamma \Rightarrow A \) type:

1. If \( \Gamma, \Delta \Rightarrow B \triangleq B' \) type then \( \Gamma, a : A, \Delta \Rightarrow B \triangleq B' \) type.

2. If \( \Gamma, \Delta \Rightarrow N \triangleq N' \in B \) then \( \Gamma, a : A, \Delta \Rightarrow N \triangleq N' \in B \).

Proof. Suppose \( \gamma \sim \gamma' \in \Gamma \). We establish the presupposition \((\Gamma, a : A, \Delta) \text{ ctx}\) by induction on \(\Gamma\), obtaining \(A\gamma \triangleq A\gamma' \) type from \(\Gamma \Rightarrow A \text{ type}\) and \(\Delta\gamma \triangleq \Delta\gamma' \text{ ctx}\) from \((\Gamma, \Delta) \text{ ctx}\). (The latter suffices because \(a\) cannot occur in \(\Delta\).) For part (1), suppose \((\gamma, M, \delta) \sim (\gamma', M', \delta') \in (\Gamma, a : A, \Delta)\) and show \(By[M/a]\delta \triangleq B'\gamma'[M'/a]\delta' \) type. Because \(a\) cannot occur in \(B\), this is exactly \(B\gamma\delta \triangleq B'\gamma'\delta' \) type, which holds by assumption. Part (2) follows similarly. \(\square\)

Lemma 2.24 (Substitution). Supposing \( \Gamma \Rightarrow M \triangleq M' \in A \):

1. If \( \Gamma, a : A, \Delta \Rightarrow B \triangleq B' \) type then \( \Gamma, \Delta[M/a] \Rightarrow B[M/a] \triangleq B'[M'/a] \) type.

2. If \( \Gamma, a : A, \Delta \Rightarrow N \triangleq N' \in B \) then \( \Gamma, \Delta[M/a] \Rightarrow N[M/a] \triangleq N'[M'/a] \in B[M/a] \).

Proof. As in the previous lemmas, the presupposition \((\Gamma, \Delta[M/a]) \text{ ctx}\) is established by induction on \(\Gamma\), noting that for any \(\gamma \sim \gamma' \in \Gamma\), we have \(M\gamma \in A\gamma\) and \(\Delta[M/a]\gamma \triangleq \Delta\gamma[M/a]\). For part (1), suppose \((\gamma, \delta) \sim (\gamma', \delta') \in (\Gamma, \Delta[M/a])\), where \(\gamma \sim \gamma' \in \Gamma\) and \(\delta \sim \delta' \in \Delta[M/a]\). Then \((\gamma, M\gamma, \delta) \sim (\gamma', M'\gamma', \delta') \in (\Gamma, a : A, \Delta)\) by \(M\gamma \triangleq M'\gamma' \in A\gamma\) and \(\Delta[M/a]\gamma \triangleq \Delta\gamma[M/a]\), so by hypothesis we have \(By[M\gamma/a]\delta \triangleq B'\gamma'[M'/a]\delta' \) type and hence \(B[M/a]\gamma\delta \triangleq B'[M'/a]\gamma'\delta' \) type as required. Part (2) follows similarly. \(\square\)

Surprisingly, Lemma 2.24 does not hold in ordinary Nuprl—in Allen’s nonstandard semantics of open judgments, the cut principle holds only if \(a\) does not occur in \(\Delta\)—but it does hold in Idealized Nuprl and Chapter 4 of this dissertation.

It remains to show that \(\tau_0\) admits the formation, introduction, elimination, and computation principles of dependent function types, universe types, and et cetera. We will prove only the closed, binary forms of these principles, as the open forms follow directly by commuting substitutions past constructors. By way of illustration, consider the open form of Lemma 2.13:

Lemma 2.25. If \( \Gamma \Rightarrow A \triangleq B \) type and \( \Gamma \Rightarrow M \triangleq N \in A \) then \( \Gamma \Rightarrow M \triangleq N \in B \).

Proof. Suppose \( \gamma \sim \gamma' \in \Gamma \) and show \(M\gamma \triangleq N\gamma' \in By\). By our second assumption, \(M\gamma \triangleq N\gamma' \in A\gamma\). By our first assumption and \(\gamma \sim \gamma \in \Gamma\) (by \(\Gamma \text{ ctx}\)), \(A\gamma \triangleq B\gamma\) type. The result follows by Lemma 2.13. \(\square\)
2.4 Rules of inference

Sections 2.2 and 2.3 define Idealized Nuprl as a mathematical object, but these definitions alone are neither useful in practice (how does one establish a judgment?) nor obviously correct (is \((a:A) \rightarrow B\) a dependent function type in the usual sense?). We address both concerns by proving that Idealized Nuprl admits standard type-theoretic rules of inference, listed in Figure 2.4. Each rule is annotated with the lemma containing its proof: structural rules were proven in Section 2.3, while rules governing type formers are proven in Section 2.5. We suppress ambient hypotheses in these rules for clarity; every judgment implicitly contains additional hypotheses \(\Gamma\).

These natural deduction–style rules characterize type formers by rules of formation, establishing when an instance is a type; introduction, establishing how to construct an element of that type; and elimination, establishing how to use an element of that type [Pra65]. Some type formers also have a uniqueness, or \(\eta\), rule establishing that all elements are equal to introduction forms.\(^5\) In standard natural deduction, type formers have computation, or \(\beta\), rules establishing how elimination acts on introduction:

\[
\frac{a : A \triangleright M \in B \quad N \in A}{(\lambda a. M) N \equiv M[N/a] \in B[N/a]}
\]

Computation rules are true but superfluous in Idealized Nuprl, because they are instances of head expansion (Lemma 2.15).

Choosing rules As the computation rules have already illustrated, there are many possible collections of inference rules for any type theory. Their relative merits are a topic of frequent debate amongst type theorists, and while we do not enter the fray in this dissertation, we will nevertheless sketch various considerations. First, the very meaning of a type theory qua algebraic structure (Section 1.2), by definition, depends on its collection of rules; the biggest open problem in homotopy type theory is to characterize all models of Book HoTT and various cubical type theories.

Secondly, type theories are most commonly used through proof assistants, which are based on collections of primitive inference steps (Section 1.1). Such inferences cannot depend on notational conventions or semantic presuppositions, however sensible. If a reader were to conclude by shadowing variable names that the rules of Figure 2.4 are inconsistent, they would be admonished to recall standard notational conventions; a proof assistant, on the other hand, must actually implement variable binding.

Unlike variable freshness, which is fully automated, our judgments’ presuppositions can induce subgoals requiring user input. For instance, to maintain the invariant that

\(^5\)Many type theories have few or no uniqueness rules. In Idealized Nuprl, every type has a uniqueness principle; those of nat and bool can be derived from their elimination rules.
Figure 2.4: Idealized Nuprl: structural rules, functions, pairs.
Figure 2.4: Idealized Nuprl: equality, natural numbers, Booleans, universes.
\( \Gamma \) \text{ctx} \text{ holds whenever } \Gamma \gg B \text{ type is provable in our proof assistant, we might demand that users prove new hypotheses are types before extending contexts. Typehood cannot be established automatically, as it is undecidable (if } \text{Eq}_A(M, M) \text{ type then } M \text{ halts) and commonly intractable, often requiring proofs that two functions are extensionally equal.}

\text{Computational type theories—implemented in the Nuprl [Con+85] and RedPRL [Red16] proof assistants—alleviate the burden by allowing users to employ computation in their proofs. Computational type theories are defined by two characteristics: having a computational semantics, and using inference rules that irrevocably commit to that semantics. The latter is evident in the head expansion and evaluation rules of Figure 2.4, which expose the operational semantics, but crucially, allow users to prove theorems by first computing them. On top of that, Nuprl’s developers frequently extend Nuprl with useful rules subject only to validation by their computational semantics.}

\text{Figure 2.4 differs from the rules used by Nuprl and RedPRL in a few important ways. Both proof assistants adopt sequent calculi, not natural deduction rules, in which elimination rules are replaced by left rules explaining how to use hypotheses of each type. In practice, one needs head expansion for open terms, despite our operational semantics being defined only for closed terms. Fortunately, } \mapsto-\mapsto \text{ is stable under substitution when lifted to open terms—that is, if a rule of } \mapsto-\mapsto \text{ applies to open terms } M \text{ and } M', \text{ then } M\gamma \mapsto-\mapsto M'\gamma \text{ for all substitutions } \gamma—\text{allowing us to reduce subgoals like } \Gamma \gg \text{fst}(\langle a, b \rangle) \in A \text{ to } \Gamma \gg a \in A. \text{(We will not be as fortunate in Chapter 4.)}

\text{Intensional type theories—implemented in the Agda [Agda] and Coq [Coq] proof assistants—axiomatize and automate a decidable underapproximation of the judgments, reducing users’ proof burden while restricting what is provable. These type theories notably lack an equality reflection rule converting proofs of } \text{Eq}_A(M, N) \text{ to } M = N \in A; \text{ instead, silent equations are restricted to those provable by computation, while any appeals to non-trivial equations (those requiring inductive proofs) must be annotated.}

\text{Those silent equations, known as definitionial equalities [ML75a], do not constitute a mathematically natural notion of equality, unlike the equality judgment considered in this dissertation—many types lack definitionial uniqueness principles, and therefore lack strict universal properties in the categorical sense.}^6 \text{ Definitionial equality is also brittle, as extensions require significant effort both devising algorithms and proving them correct. However, the benefit of being fully automated cannot be understated.}

\text{Computational type theory is often considered strange for having undecidable judgments. On the contrary, it is perfectly natural for a judgment to be sensible without being obviously sensible; in ordinary mathematical practice, a theorem’s statement may require explanation even before its proof. Definitionial equality is decidable not because decidability is most natural, but because it allows proof assistants to fully automate all}

^6\text{Nor, in the author’s opinion, does definitionial equality constitute a philosophically natural notion of intension, as the name of intensional type theory might suggest.}
appeals to computation.

2.5 Semantics of types

We now prove the formation, introduction, elimination, and uniqueness rules that govern each type former. We will also establish that Idealized Nuprl is consistent (Theorem 2.47), and that it enjoys the canonicity (Theorem 2.46) and existence properties (Theorem 2.42) characteristic of constructive type theories.

Recall that the judgments of Idealized Nuprl are defined relative to the type system $\tau_\omega$ constructed in Section 2.2. Formation and introduction rules for each type hold essentially by construction, after expanding judgments into the candidate judgments of Figure 2.3; elimination rules hold by observing that elimination forms compute on introduction forms; and uniqueness rules hold by induction (all elements are equal to introduction forms) coupled with introduction rules (to see that two introduction forms are equal).

2.5.1 Dependent functions

Recall from Figure 2.3 that for $i \in \{0, 1, \ldots, \omega\}$ and $R \in \{\tau_i, \pi_i\}$,

$$R((a:A) \to B, (a:A') \to B', \{\lambda a.M, \lambda a.N' | a : \alpha \triangleright N \sim N' \in \beta\})$$

if and only if $A \sim A' \downarrow \alpha \in \tau_i$ and $a : \alpha \triangleright B \sim B' \downarrow \beta \in R$.

**Rule 2.26** (Formation).

1. If $A \triangleq A' \in Type_i$ and $a : A \Rightarrow B \triangleq B' \in Type_i$ then $(a:A) \to B \triangleq (a:A') \to B' \in Type_i$.

2. If $A \triangleq A' \in Type_i$ and $a : A \Rightarrow B \triangleq B' \in Prop_i$ then $(a:A) \to B \triangleq (a:A') \to B' \in Prop_i$.

3. If $A \triangleq A'$ type and $a : A \Rightarrow B \triangleq B'$ type then $(a:A) \to B \triangleq (a:A') \to B'$ type.

**Proof.** For part (1), $A \triangleq A'$ implies $A \sim A' \downarrow \alpha \in \tau_i$ (by the definition of $\llbracket Type_i \rrbracket$), $A \triangleq A'$ type (by Lemma 2.14), and $\alpha = \llbracket A \rrbracket$ (by Theorem 2.11). Furthermore, $a : A \Rightarrow B \triangleq B' \in Type_i$ states that for any $M \triangleq M' \in A$ (equivalently, $M \sim M' \in \alpha$), $B[M/a] \triangleq B'[M'/a] \in Type_i$, and thus $a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau_i$. We conclude that $(a:A) \to B \sim (a:A') \to B' \downarrow \llbracket (a:A) \to B \rrbracket \in \tau_i$, and therefore $(a:A) \to B \triangleq (a:A') \to B' \in Type_i$, where $\llbracket (a:A) \to B \rrbracket(\lambda a.N, \lambda a.N')$ holds if and only if $a : \alpha \triangleright N \sim N' \in \beta$. Parts (2) and (3) are similar, appealing to the definitions of $\pi_i$ and $\tau_\omega$, respectively, instead of $\tau_i$.  

**Rule 2.27** (Introduction). If $a : A \Rightarrow M \triangleq M' \in B$ then $\lambda a.M \triangleq \lambda a.M' \in (a:A) \to B$. 


Proof. The presupposition \((a:A) \rightarrow B\) type holds by the formation rule applied to the premise’s presupposition \(a : A \Rightarrow B\) type. We know for any \(N \simeq N' \in A\) that \(M[N/a] \simeq M'[N'/a] \in B[N/a];\) thus \(a : [A] \Rightarrow M \simeq M' \in \beta\) where \(a : [A] \Rightarrow B \sim B \downarrow \beta \in \tau_\omega\) (because unicity implies \(\beta_{N,N'} = \llbracket B[N/a] \rrbracket\)), and hence \(\llbracket (a:A) \rightarrow B \rrbracket(\lambda a.M, \lambda a.M')\). \(\square\)

Observe that \(a : A \Rightarrow B \simeq B'\) type and \(a : A \Rightarrow M \simeq M' \in B\) are equivalent to \(a : [A] \Rightarrow B \sim B' \downarrow \beta \in \tau_\omega\) and \(a : [A] \Rightarrow M \sim M' \in \beta\) respectively. We henceforth freely use this equivalence when unfolding definitions of type formers.

**Rule 2.28 (Elimination).** If \(M \simeq M' \in (a:A) \rightarrow B\) and \(N \simeq N' \in A\) then \(M N \simeq M' N' \in B[N/a]\).

**Proof.** By the definition of \(\llbracket (a:A) \rightarrow B \rrbracket\), we know \(M \llbracket \lambda a.O, M \llbracket \lambda a.O', a : A \Rightarrow O \simeq O' \in B\). By the latter, \(O[N/a] \simeq O'[N'/a] \in B[N/a]\). By Lemma 2.15 and \(M N \leftrightarrow^* (\lambda a.O) N \leftrightarrow O[N/a]\), we know \(M N \simeq O[N/a] \in B[N/a]\), and analogously, \(M' N' \simeq O'[N'/a] \in B[N/a]\). The result follows by symmetry and transitivity of equality. \(\square\)

As discussed in Section 2.4, we omit computation rules from Figure 2.4. We will prove Rule 2.29 for illustrative purposes, and leave further computation rules as an exercise.

**Rule 2.29 (Computation).** If \(a : A \Rightarrow M \in B\) and \(N \in A\) then \((\lambda a.M) N \simeq M[N/a] \in B[N/a]\).

**Proof.** Our hypotheses imply \(M[N/a] \in B[N/a]\). The result follows by Lemma 2.15 and \((\lambda a.M) N \leftrightarrow^* M[N/a]\). \(\square\)

**Rule 2.30 (Uniqueness).** If \(M \in (a:A) \rightarrow B\) then \(M \simeq \lambda a.M a \in (a:A) \rightarrow B\).

**Proof.** By \(M \in (a:A) \rightarrow B\), we know \(M \llbracket \lambda a.O, a : A \Rightarrow O \in B\). By the definition of \(\llbracket (a:A) \rightarrow B \rrbracket\), it suffices to show \(a : A \Rightarrow O \simeq M a \in B\). Suppose \(N \simeq N' \in A\). Then by \(a : A \Rightarrow O \in B\) we have \(O[N/a] \simeq O[N'/a] \in B[N/a]\); by Lemma 2.15 and \(M N' \leftrightarrow^* (\lambda a.O) N' \leftrightarrow O[N'/a]\) we have \(O[N'/a] \simeq M N' \in B[N/a]\); and thus \(O[N/a] \simeq M N' \in B[N/a]\) as required. \(\square\)

### 2.5.2 Dependent pairs

For \(i \in \{0, 1, \ldots, \omega\}\) and \(R \in \{\tau_i, \pi_i\}\),

\[
R((a:A) \times B, (a':A') \times B', \{((M, N), (M', N')) \mid (M \sim M' \in \alpha) \land (N \sim N' \in \beta_{M,M'})\})
\]

if and only if \(A \sim A' \downarrow \alpha \in R\) and \(a : \alpha \Rightarrow B \sim B' \downarrow \beta \in R\).

**Rule 2.31 (Formation).**

1. If \(A \simeq A' \in \text{Type}_i\) and \(a : A \Rightarrow B \simeq B' \in \text{Type}_i\) then \((a:A) \times B \simeq (a:A') \times B' \in \text{Type}_i\).
2. If $A \triangleq A' \in \text{Prop}_i$ and $a : A \Rightarrow B \triangleq B' \in \text{Prop}_i$ then $(a:A) \times B \triangleq (a:A') \times B' \in \text{Prop}_i$.

3. If $A \triangleq A'$ type and $a : A \Rightarrow B \triangleq B'$ type then $(a:A) \times B \triangleq (a:A') \times B'$ type.

Proof. The proof proceeds identically to that of Rule 2.26. Note that the Pair clause of $\pi_i$ requires $A \sim A' \downarrow \alpha \in \pi_i$ and not $A \sim A' \downarrow \alpha \in \tau_i$ as in the Fun clause. □

Rule 2.32 (Introduction). If $a : A \Rightarrow B \in \text{Type}_i$, $M \triangleq M' \in A$, and $N \triangleq N' \in B[M/a]$, then $(M, N) \triangleq (M', N') \in (a:A) \times B$.

Proof. The first premise and the formation rule ensure the presupposition $(a:A) \times B$ type. It remains to show $\llbracket (a:A) \times B \rrbracket ((M, N), (M', N'))$, which follows from $M \sim M' \in \llbracket A \rrbracket$ and $N \sim N' \in \beta_{M,M'}$ where $a : \llbracket A \rrbracket \Rightarrow B \downarrow \beta \in \tau_\omega$ (by $\beta_{M,M'} = \llbracket B[M/a] \rrbracket$). □

Rule 2.33 (Elimination). If $P \triangleq P' \in (a:A) \times B$ then

1. $\text{fst}(P) \triangleq \text{fst}(P') \in A$ and

2. $\text{snd}(P) \triangleq \text{snd}(P') \in B[\text{fst}(P)/a]$.

Proof. By the definition of $\llbracket (a:A) \times B \rrbracket$, we know $P \parallel (M, N), P' \parallel (M', N')$, $M \triangleq M' \in A$, and $N \triangleq N' \in B[M/a]$. For part (1), by Lemma 2.15 and $\text{fst}(P) \longmapsto \ast \text{fst}((M, N)) \longmapsto M$, we know $\text{fst}(P) \triangleq M \in A$ and analogously $M' \triangleq \text{fst}(P') \in A$; the result follows by transitivity.

For part (2), by Lemma 2.15 and $\text{snd}(P) \longmapsto \ast \text{snd}((M, N)) \longmapsto N$, we know $\text{snd}(P) \triangleq N \in B[M/a]$ and analogously $N' \triangleq \text{snd}(P') \in B[M/a]$, hence $\text{snd}(P) \triangleq \text{snd}(P') \in B[M/a]$. We have $a : A \Rightarrow B$ type by $(a:A) \times B$ type and inversion on the definition of $\tau_\omega$; thus $B[M/a] \triangleq B[\text{fst}(P)/a]$ type, and the result follows by Lemma 2.13. □

Rule 2.34 (Uniqueness). If $P \in (a:A) \times B$ then $P \triangleq (\text{fst}(P), \text{snd}(P)) \in (a:A) \times B$.

Proof. By $P \in (a:A) \times B$, we know $P \parallel (M, N), M \in A$, and $N \in B[M/a]$. By the definition of $\llbracket (a:A) \times B \rrbracket$, it suffices to show $M \triangleq \text{fst}(P) \in A$ and $N \triangleq \text{snd}(P) \in B[M/a]$; both are immediate by Lemma 2.15. □

2.5.3 Equalities

For $i \in \{0, 1, \ldots, \omega\}$ and $R \in \{\tau_i, \pi_i\}$,

$$R(\text{Eq}_A(M, N), \text{Eq}_{A'}(M', N'), \{\ast, \ast\} | M \sim N \in \alpha)$$

if and only if $A \sim A' \downarrow \alpha \in \tau_i, M \sim M' \in \alpha$, and $N \sim N' \in \alpha$.

Rule 2.35 (Formation).

1. If $A \triangleq A' \in \text{Type}_i, M \triangleq M' \in A$, and $N \triangleq N' \in A$, then $\text{Eq}_A(M, N) \triangleq \text{Eq}_{A'}(M', N') \in \text{Type}_i$. 

2. If $A \equiv A' \in \text{Type}_i$, $M \equiv M' \in A$, and $N \equiv N' \in A$, then $\text{Eq}_A(M, N) \equiv \text{Eq}_{A'}(M', N') \in \text{Prop}_i$. 

3. If $A \equiv A'$ type, $M \equiv M' \in A$, and $N \equiv N' \in A$, then $\text{Eq}_A(M, N) \equiv \text{Eq}_{A'}(M', N')$ type.

Proof. For part (1), $A \equiv A' \in \text{Type}_i$ implies $A \sim A' \downarrow \llbracket A \rrbracket \in \tau_i$. From $M \sim M' \in \llbracket A \rrbracket$ and $N \sim N' \in \llbracket A \rrbracket$ we conclude $\text{Eq}_A(M, N) \sim \text{Eq}_{A'}(M', N') \downarrow \llbracket \text{Eq}_A(M, N) \rrbracket \in \tau_i$ and therefore $\text{Eq}_A(M, N) \equiv \text{Eq}_{A'}(M', N') \in \text{Type}_i$, where $\llbracket \text{Eq}_A(M, N) \rrbracket(\star, \star)$ holds if and only if $M \sim N \in \llbracket A \rrbracket$. Parts (2) and (3) are similar, appealing to the definitions of $\pi_i$ and $\tau_{\omega}$, respectively, instead of $\tau_i$. \hfill $\square$

Rule 2.36 (Introduction). If $M \equiv N \in A$ then $\ast \in \text{Eq}_A(M, N)$.

Proof. The premise implies $A$ type, $M \in A$, and $N \in A$, so by the formation rule, the presupposition $\text{Eq}_A(M, N)$ type holds. It remains to show $\llbracket \text{Eq}_A(M, N) \rrbracket(\ast, \ast)$, which is immediate by $M \sim N \in \llbracket A \rrbracket$. \hfill $\square$

Rule 2.37 (Elimination). If $E \in \text{Eq}_A(M, N)$ then $M \equiv N \in A$.

Proof. By the definition of $\llbracket \text{Eq}_A(M, N) \rrbracket$, $E \downarrow \ast$ and $M \equiv N \in A$. \hfill $\square$

Rule 2.37 is commonly known as equality reflection. As discussed in Section 2.4, intensional type theories omit such a rule despite it holding in their computational semantics (in the non-univalent case).

Rule 2.38 (Uniqueness). If $E \in \text{Eq}_A(M, N)$ then $E \equiv \ast \in \text{Eq}_A(M, N)$.

Proof. By $E \in \text{Eq}_A(M, N)$ we know $E \downarrow \ast$ and $M \equiv N \in A$, which by the definition of $\llbracket \text{Eq}_A(M, N) \rrbracket$ is exactly what we must show. \hfill $\square$

2.5.4 Natural numbers

For $i \in \{0, 1, \ldots, \omega\}$, $\tau_i(\text{nat}, \text{nat}, \mu R. F(R))$ holds, where

$$F(R) = \{(z, z) \cup \{(s(M), s(M')) : M \sim M' \in R\}.$$

Rule 2.39 (Formation). nat $\in \text{Type}_i$.

Proof. Immediate by the definition of $\llbracket \text{Type}_i \rrbracket$. \hfill $\square$

Rule 2.40 (Introduction).

1. $z \in \text{nat}$.

2. If $M \equiv M' \in \text{nat}$ then $s(M) \equiv s(M') \in \text{nat}$. 

Proof. Immediate by the definition of $\llbracket \text{Type}_i \rrbracket$. \hfill $\square$
Proof. The presupposition nat type holds by the formation rule and Lemma 2.14. Part (1) is immediate by $F(\langle \text{nat} \rangle)(z, z)$, because $\text{nat} = F(\langle \text{nat} \rangle)$. Part (2) is similarly immediate by $F(\langle \text{nat} \rangle)(s(M), s(M'))$ whenever $M \sim M' \in \text{nat}$. □

Rule 2.41 (Elimination). If $n : \text{nat} \Rightarrow A \in \text{Type}_i$, $M \equiv M' \in \text{nat}$, $Z \equiv Z' \in A[z/n]$, and $n : \text{nat}$, $a : A \Rightarrow S \equiv S' \in A(s(n)/n)$, then $\text{natrec}(M; Z, n.a.S) \equiv \text{natrec}(M'; Z', n.a.S') \in A[M/n]$.

Proof. Let $\Phi = \{ (N_0, N'_0) \mid \text{nat}(N_0, N'_0) \wedge \text{natrec}(N_0; Z, n.a.S) \equiv \text{natrec}(N'_0; Z', n.a.S') \in A[N_0/n] \}$. First, show $N \sim N' \in \Phi$ implies $\text{natrec}(N; Z, n.a.S) \equiv \text{natrec}(N'; Z', n.a.S') \in A[N/n]$. Since $\Phi \subseteq \text{nat}$, we have $N \in \text{nat}$ and thus the presupposition $A[N/n]$ type. By definition, $N \Downarrow N_0, N' \Downarrow N'_0$, and $\Phi(N_0, N'_0)$. Therefore, by Lemma 2.15, it suffices to show $\text{natrec}(N_0; Z, n.a.S) \equiv \text{natrec}(N'_0; Z', n.a.S') \in A[N/n]$. These terms are equal in the type $A[N_0/n]$ by $\Phi(N_0, N'_0)$; the result follows by $A[N_0/n] \equiv A[N/n]$ type (by $N \equiv N_0 \in \text{nat}$) and Lemma 2.13.

Now let us show $F(\Phi) \subseteq \Phi$; because $\text{nat}$ is the least such relation, it will follow that $\text{nat} \subseteq \Phi$. Suppose $F(\Phi)(N_0, N'_0)$ and show $\Phi(N_0, N'_0)$. There are two possibilities. If $N_0 = N'_0 = z$, then $\text{nat}(z, z)$. By Lemma 2.15 and $\text{natrec}(z; Z, n.a.S) \mapsto Z$, it suffices to show $Z \equiv Z' \in A[z/n]$, which we have assumed. Otherwise, $N_0 = s(N), N'_0 = s(N')$, and $N \sim N' \in \Phi$. Since $\Phi \subseteq \text{nat}$ we have $\text{nat}(s(N), s(N'))$. By Lemma 2.15 and $\text{natrec}(s(N); Z, n.a.S) \mapsto s(N)[n.a.S]/a$, it suffices to show

$$S[N/n][\text{natrec}(N; Z, n.a.S)/a] = S'[N'/n][\text{natrec}(N'; Z', n.a.S')/a] \in A[s(n)/n]$$

which follows from $N \equiv N' \in \text{nat}$, $\text{natrec}(N; Z, n.a.S) \equiv \text{natrec}(N'; Z', n.a.S') \in A[N/n]$ (by the observation in the previous paragraph), and our assumption on $S$ and $S'$.

By $M \equiv M' \in \text{nat}$ and $\text{nat} \subseteq \Phi$, $M \sim M' \in \Phi$, from which the result is immediate. □

Idealized Nuprl, as formulated in Figure 2.4, is constructive in the sense of satisfying the existence property [Tv88, p. 139]. We demand not only that our computational semantics have existential witnesses but in fact that our chosen rules determine these witnesses—in the absence of the latter, users of Idealized Nuprl would not reap the benefits of the former.

Theorem 2.42 (Existence). If $n : \text{nat} \Rightarrow B \in \text{Type}_i$ and $M \in (n : \text{nat}) \times B$, there exist terms $\bar{n} = s \cdots s(z)$ and $P$ such that $P \in B[\bar{n}/n]$. Furthermore, the rules of Figure 2.4 suffice to derive $\bar{n}$ and $P$.

Proof. By Lemma 2.16 (and the definition of $\langle (n : \text{nat}) \times B \rangle$), $M \Downarrow \langle N, P \rangle$ and $M \equiv \langle N, P \rangle \in (n : \text{nat}) \times B$. By Rule 2.33 and Lemma 2.16, $\text{fst}(M) \equiv N \in \text{nat}$ and $\text{snd}(M) \equiv P \in B[\text{fst}(M)/n]$. To determine $\bar{n} = s \cdots s(z)$ such that $N \equiv \bar{n} \in \text{nat}$, apply Lemma 2.16 recursively to $N$—either $N \Downarrow z$ or $N \Downarrow s(N')$ such that $N' \equiv \bar{n}' \in \text{nat}$, and in the latter case, $s(N') \equiv s(\bar{n}') \in \text{nat}$ by Rule 2.40. (To project $N'$ from $s(N')$, subtract $s(z)$ and evaluate.) Then $B[\text{fst}(M)/n] \equiv B[\bar{n}/n] \in \text{Type}_i$ by Lemma 2.24, and $P \in B[\bar{n}/n]$ by Lemma 2.13.
2.5.5 Booleans

Theorem 2.47

To see why the recursive procedure succeeds, let \( \Phi = \{ (N_0, N'_0) \mid \exists \bar{n}. N_0 \equiv \bar{n} \in \text{nat} \} \). We show \( F(\Phi) \subseteq \Phi \), and therefore \( \llbracket \text{nat} \rrbracket \subseteq \Phi \); it follows that whenever \( \bar{N} \in \text{nat} \), \( N \llbracket N_0 \rrbracket \) and \( N_0 \equiv \bar{n} \in \text{nat} \). Suppose \( F(\Phi)(N_0, N'_0) \). If \( F(\Phi)(z, z) \) then \( \bar{n} \equiv z \) and \( z \equiv z \in \text{nat} \). Otherwise \( F(\Phi)(s(M), s(M')) \) and \( \Phi(M, M') \), and therefore \( M \llbracket M_0 \rrbracket \) and \( M_0 \equiv \bar{n}' \in \text{nat} \); then \( \bar{n} \equiv s(\bar{n}') \) and \( s(M) \equiv \bar{n} \in \text{nat} \) by Lemma 2.16 and Rule 2.40. 

2.5.5 Booleans

For \( i \in \{0, 1, \ldots, \omega\} \), \( \tau_i(\text{bool}, \text{bool}, \{(\text{true}, \text{true}), (\text{false}, \text{false})\}) \).

Rule 2.43 (Formation). \( \text{bool} \in \text{Type}_i \).

Proof. Immediate by the definition of \( \llbracket \text{Type}_i \rrbracket \).

Rule 2.44 (Introduction).

1. \( \text{true} \in \text{bool} \).
2. \( \text{false} \in \text{bool} \).

Proof. The presupposition \( \text{bool} \) type holds by the formation rule and Lemma 2.14. Both parts are immediate by the definition of \( \llbracket \text{bool} \rrbracket \).

Rule 2.45 (Elimination). If \( b : \text{bool} \Rightarrow A \in \text{Type}_i \), \( M \equiv M' \in \text{bool} \), \( T \equiv T' \in A[\text{true}/b] \), and \( F \equiv F' \in A[\text{false}/b] \), then \( \text{if}(M; T, F) \equiv \text{if}(M'; T', F') \in A[M/b] \).

Proof. There are two possibilities for \( M \equiv M' \in \text{bool} \). If \( M \llbracket \text{true} \rrbracket \) and \( M' \llbracket \text{true} \rrbracket \), then by Lemma 2.15 and \( \text{if}(M; T, F) \rightarrow^* \text{if}(\text{true}; T, F) \rightarrow T \), it suffices to show \( T \equiv T' \in A[M/b] \). The result follows by \( T \equiv T' \in A[\text{true}/b] \), \( A[\text{true}/b] \equiv A[M/b] \) type (by \( \text{true} \equiv M \in \text{bool} \)), and Lemma 2.13. The false case is analogous.

By the definition of \( \llbracket \text{bool} \rrbracket \), every closed Boolean equals either true or false. Canonicity, like the existence property, states that the rules of Idealized Nuprl derive such equations.

Theorem 2.46 (Canonicity). If \( M \in \text{bool} \) then either \( M \llbracket \text{true} \rrbracket \) and \( M \equiv \text{true} \in \text{bool} \) or \( M \llbracket \text{false} \rrbracket \) and \( M \equiv \text{false} \in \text{bool} \); moreover, the rules of Figure 2.4 derive these equations.

Proof. By the definition of \( \llbracket \text{bool} \rrbracket \), either \( M \llbracket \text{true} \rrbracket \) or \( M \llbracket \text{false} \rrbracket \); the result follows by Lemma 2.16, which is a rule in Figure 2.4.

Of course, Idealized Nuprl is also consistent, as it has empty types.

Theorem 2.47 (Consistency). There is no \( E \) such that \( E \in \text{Eq}_{\text{bool}}(\text{true}, \text{false}) \).

Proof. Suppose such an \( E \) existed. Then \( \text{true} \equiv \text{false} \in \text{bool} \) by Rule 2.37, which contradicts the definition of \( \llbracket \text{bool} \rrbracket \).
2.5.6 Universes

For $i, j \in \{0, 1, \ldots, \omega\}$ with $i < j$:

\[
\tau_i(\text{Prop}_i, \text{Prop}_i, \{(A_0, B_0) \mid \exists \alpha. \pi_i(A_0, B_0, \alpha)\}) \\
\tau_j(\text{Type}_i, \text{Type}_i, \{(A_0, B_0) \mid \exists \alpha. \pi_i(A_0, B_0, \alpha)\})
\]

**Rule 2.48** (Formation).

1. $\text{Type}_i \in \text{Type}_{i+1}$.
2. $\text{Prop}_i \in \text{Type}_{i+1}$.

**Proof.** The presupposition $\text{Type}_{i+1}$ type is immediate by the definition of $\tau_{\omega}$; parts (1) and (2) are immediate by the definition of $[\text{Type}_{i+1}]$. $\square$

We have already stated and proven the introduction rules of $\text{Prop}_i$ and $\text{Type}_i$ as formation rules of other types. The elimination rules of $\text{Prop}_i$ and $\text{Type}_i$ are simply that their elements are types, which we established in Lemma 2.14.

**Rule 2.49** (Subtyping). If $A \bowtie A' \in \text{Prop}_i$ then $A \bowtie A' \in \text{Type}_i$.

**Proof.** By the definition of $[\text{Prop}_i]$, $A \sim A' \downarrow \alpha \in \pi_i$. By Theorem 2.11 and monotonicity of candidate judgments, $A \sim A' \downarrow \alpha \in \pi_i$; the result follows by the definition of $[\text{Type}_i]$. $\square$

**Rule 2.50** (Cumulativity).

1. If $A \bowtie A' \in \text{Type}_i$ then $A \bowtie A' \in \text{Type}_{i+1}$.
2. If $A \bowtie A' \in \text{Prop}_i$ then $A \bowtie A' \in \text{Prop}_{i+1}$.

**Proof.** For part (1), $A \sim A' \downarrow \alpha \in \pi_i$ implies $A \sim A' \downarrow \alpha \in \pi_{i+1}$ by Theorem 2.11, and the result follows by the definition of $[\text{Type}_{i+1}]$. Part (2) is analogous. $\square$

**Rule 2.51** (Subsingleton). If $M \in A$, $M' \in A$, and $A \in \text{Prop}_i$, then $M \bowtie M' \in A$.

**Proof.** Let $\Phi = \{(A_0, A'_0, \varphi) \mid \pi_i(A_0, A'_0, \varphi) \land \forall M, M' (M, M' \in A_0 \implies M \bowtie M' \in A_0)\}$. (Note that $\pi_i(A_0, A'_0, \varphi)$ implies $A_0$ type because $\pi_i \subseteq \tau_{\omega}$ by Theorem 2.11.) We begin by showing $\text{PROPS}(\Phi, \tau_i) \subseteq \Phi$; consulting Figure 2.3, there are three cases.

1. $\text{FUN}(\tau_i, \Phi)((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$.

By construction, $\Phi \subseteq \pi_i$, and therefore $\pi_i((a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)$ by $\text{FUN}(\tau_i, \Phi) \subseteq \text{FUN}(\tau_i, \pi_i) \subseteq \pi_i$. Suppose $M, M' \in (a:A) \rightarrow B$. To show $M \bowtie M'$ are equal, it suffices by Rules 2.27 and 2.30 to show $a : A \Rightarrow M a \bowtie M' a \in B$, and hence that $M N \bowtie M' N' \in B[N/a]$ for any $N \bowtie N' \in A$. Unrolling the definition of $\text{FUN}(\tau_i, \Phi)$,
We sketch the operational semantics of typecase and Computational semantics are open-ended but in Chapter 4 we cannot internalize the statement that a type is Kan. T/y.sc/p.sc/e.sc/s.sc

In this chapter, of course, one can simply define Prop ≔ (A:Type) × ((a:A) → (a’:A) → Eq_A(a, a’)), but in Chapter 4 we cannot internalize the statement that a type is Kan.

On open-endedness Computational semantics are open-ended in the sense of being modular and, like the BHK interpretation, easily extended with additional type formers, thereby reflecting the intuitionistic perspective that mathematics evolves over time [Dum77]. However, Idealized NuPrl’s universes are not open-ended; on the contrary, they are fixed, inductively-defined collections of types! In this regard, computational semantics deviate significantly from categorical semantics, in which universes typically classify all collections below some size cutoff.

The inductive nature of our universes has some surprising consequences. First, not all subsingleton types are elements of Prop_i; for instance, (n:nat) × Eq_{nat}(n, z) ∉ Prop_i because nat is not subsingleton. Secondly, functions out of Type_i can be defined by cases on type formers if the underlying programming language has such a facility, often called typecase. We sketch the operational semantics of typecase and the resulting Type_i elimination rule in Figure 2.5.

The situation can be remedied in several ways. Allen suggests a variant of computational semantics parametric in extensions of the Types operator (Figure 2.3) that generates

\[
\begin{align*}
a : [A] & \to B \sim B' \downarrow \beta \in \Phi \text{ and therefore } B[N/a] \downarrow B_0 \text{ where } B_0 \in \text{Prop}_i \text{ and all elements of } B_0 \text{ are equal. But then } B[N/a] \equiv B_0 \text{ type, so } M N = M' N' \in B[N/a] \text{ by Lemma 2.13.} \\
2. \ \text{PAIR}(\Phi)((a:A) \times B, (a:A') \times B', \varphi). \\
\text{Again, } \pi_i((a:A) \times B, (a:A') \times B', \varphi) \text{ by PAIR}(\Phi) \subseteq \text{PAIR}(\pi_i) \subseteq \pi_i. \text{ Suppose } M, M' \in (a:A) \times B. \text{ By Rules 2.32 and 2.34, it suffices to show } \text{fst}(M) = \text{fst}(M') \in A \text{ and } \text{snd}(M) = \text{snd}(M') \in B[\text{fst}(M)/a]. \text{ Unrolling the definition of PAIR}(\Phi), A \sim A' \downarrow [A] \in \Phi \text{ and } a : [A] \to B \sim B' \downarrow \beta \in \Phi. \text{ By the former, } A \downarrow A_0, A_0 \in \text{Prop}_i, \text{ and all elements of } A_0 \text{ are equal; thus by Lemma 2.13, } \text{fst}(M) = \text{fst}(M') \in A. \text{ By the latter, } B[\text{fst}(M)/a] \downarrow B_0, B_0 \in \text{Prop}_i, \text{ and all elements of } B_0 \text{ are equal; again, by Lemma 2.13, } \text{snd}(M) = \text{snd}(M') \in B[\text{fst}(M)/a]. \\
3. \ \text{Eq}(\pi_i)(\text{Eq}_A(M, N), \text{Eq}_A(M', N'), \varphi). \\
\text{We have } \pi_i(\text{Eq}_A(M, N), \text{Eq}_A(M', N'), \varphi) \text{ by Eq}(\pi_i) \subseteq \pi_i. \text{ Suppose } E, E' \in \text{Eq}_A(M, N). \text{ By Rule 2.38, } E \equiv \star \in \text{Eq}_A(M, N) \text{ and } E' \equiv \star \in \text{Eq}_A(M, N), \text{ and therefore } E \equiv E' \in \text{Eq}_A(M, N).
\end{align*}
\]

Because \( \pi_i \) is the least pre-fixed point of Props(−, \( \tau_i \)), \( \pi_i \subseteq \Phi \). Suppose \( A \in \text{Prop}_i \). Then \( A \downarrow A_0, A \equiv A_0 \in \text{Prop}_i \), and by \( [\text{Prop}_i] = \pi_i \subseteq \Phi, \Phi(A_0, A_0) \); therefore all elements of \( A_0 \) (and \( A \)) are equal, completing the proof. \( \square \)
the universe hierarchy [All87, Chapter 6]. In Chapter 4 of this dissertation, universes are inductively defined but include a univalence operator requiring maps out of $U^\text{Kan}_i$ to respect type isomorphism. Cavallo and Harper [CH19b] extend cubical type theory with a relativity principle that refutes excluded middle by ensuring maps out of the universe are in some sense parametric. Conversely, inductively-defined universes are in some cases desirable; Dagand and McBride have proposed closed universes of datatypes as an implementation strategy for datatype-generic dependent programming [Dag13].

Although its universes are not open-ended, Idealized Nuprl is nevertheless open-ended in two significant respects. First, because its judgments are monotone in the choice of type system, the ambient universe $\tau_\omega$ is easily extended to a larger type system $\tau'_\omega$. For instance, to add an empty type to Idealized Nuprl, define

$$
\tau'_\omega = \mu \tau . \left( \text{Types}(\nu_\omega, \tau) \cup \{\text{void}, \text{void}, \{\}\} \right).
$$

The containment $\tau_\omega \subseteq \tau'_\omega$ follows from Lemma 2.9, $\tau'_\omega$ is a type system by a trivial modification to Lemma 2.6, and thus, by monotonicity of judgments, the rules of Figure 2.4 hold relative to $\tau'_\omega$ (as does void type, but importantly, not void $\in \text{Type}$).

Secondly, Idealized Nuprl is stable under extensions of the syntax and operational semantics of its underlying programming language, subject to certain weak restrictions (notably, determinacy of evaluation) [How91]. Howe calls this property computational open-endedness, and, with Stoller, has defined a variant of the Nuprl type theory that includes all classical set-theoretic functions [HS94].
2.6 Modern Nuprl

The PRL group at Cornell has used the Nuprl theorem prover [Con+85] continuously since the mid-1980s, when Allen [All87] first described its computational semantics. They have extended the system and its semantics many times in the intervening decades, both increasing its expressivity and refining their methodology of computational type theory (described in Section 2.4). We close our discussion of Idealized Nuprl by summarizing some of modern Nuprl’s most significant innovations.

**Continuity principles**  In Nuprl, any \( F \in (\text{nat} \to \text{nat}) \to \text{nat} \) provably depends on only finitely many values of its input functions [RB16]; intuitively, a terminating program cannot check infinitely many values. Brouwer used this and other continuity principles to prove that, intuitionistically, all functions on the real numbers are continuous [Tv88, pp. 206–210]. (Both of these statements are classically false, of course, so modern Nuprl is not compatible with classical logic.) Nuprl validates this principle by means of exceptions [Har16, Chapter 29] in its programming language that allow users to intercept \( F \)'s calls to its input and compute an explicit bound.

**Non-termination**  Nuprl’s programming language extends the untyped \( \lambda \)-calculus, which famously contains divergent terms [Har16, Chapter 21]. Building on earlier work of Smith [Smi89], Crary [Cra98] extends Nuprl with partial types \( \bar{A} \) whose elements are either divergent or elements of \( A \). Unlike Idealized Nuprl, in which all elements terminate, Crary defines types as evaluation-respecting PERs on \( \text{Tm} \) (if \( \llbracket A \rrbracket(M, M) \) and \( M \mapsto M' \) then \( \llbracket A \rrbracket(M, M') \)). Finally, Crary establishes a broad class of admissible types \( A \) whose partial types contain all fixed points of functions \( \bar{A} \to \bar{A} \):

\[
\begin{align*}
M \in \bar{A} & \rightarrow \bar{A} \quad A \text{ admiss} \\
\text{fix}(M) & \in \bar{A}
\end{align*}
\]

**Computational equivalence**  As discussed in Section 2.4, Idealized Nuprl respects open computation at top level in both terms and types. In practice, one wishes also to compute inside larger terms to prove, for instance, that \( \Gamma \gg M[\text{fst}(\langle a, b \rangle)/c] = M[a/c] \in A \). Such a principle holds definitionally in intensional type theory, but is defeasible in computational semantics—in theory, \( M \) could detect the head constructor of an unevaluated subterm.

Howe [How89] proves that Nuprl’s type system respects applicative bisimulation (essentially, untyped equivalence of evaluation behavior), and moreover, that applicative bisimulation is a congruence if every operator depends only on the evaluation behavior of its arguments, unlike our hypothetical \( M \)—a condition satisfied by Nuprl’s programming language. Nuprl’s users can therefore replace arbitrary subterms of proof goals by
equivalent terms, including but not limited to $\beta$-reductions; Rahli, Bickford, and Anand [RBA13] have applied non-trivial optimizations to extracted Paxos code. Modern Nuprl defines types as applicative bisimulation–respecting PERs, and includes a base type whose elements are all closed terms modulo applicative bisimulation [AR14].

**Membership and PER types**  The base type is an essential component of modern Nuprl’s expressive power. In Idealized Nuprl, the equality type on $A$ ranges over elements of $A$, so the proposition $\text{Eq}_A(M, M)$ carries no information—whenever it is a type, it is also true. In Nuprl, equality types range instead over base, allowing them to internalize not only equality but also membership in the form of $\text{Eq}_A(M, M)$ [AR14].

Anand et al. [Ana+14] recently extended Nuprl with a pertype $(R)$ type former that, for any symmetric and transitive $R \in \text{base} \rightarrow \text{base} \rightarrow \text{Type}$, has $R$ as its underlying PER of elements. Many standard type formers are instances of PER types and can therefore be defined internally in Nuprl, including dependent function types, inductive types, partial types, and quotient types.

**Pointwise functionality**  Idealized Nuprl’s open judgments range over contexts, telescopes of types that respect the equality of, or are functional in, all earlier types. Nuprl is based instead on pointwise functionality, a notion introduced by Allen [All87, Chapter 8] and defined by Anand and Rahli [AR14] as follows:

**Definition 2.52** (Pointwise functionality). For $\gamma$ a list and $\Gamma$ a telescope, $\Gamma \text{ ctx } \gamma$ when $\Gamma@\gamma \sim \Gamma@\gamma'$ type for all $\gamma'$ such that $\gamma \sim \gamma' \in \Gamma$, where

1. $\cdot@\cdot \sim \cdot@\cdot$ type, and
2. $(a : A, \Gamma)@(M, \gamma) \sim (a : A', \Gamma')@(M', \gamma')$ type when $A \equiv A'$ type and $\Gamma[M/\gamma]@\gamma \sim \Gamma'[M'/\gamma']@\gamma'$ type.

**Definition 2.53** (Pointwise open judgments).

1. $\Gamma \gg A \equiv A'$ type when for all $\gamma \sim \gamma' \in \Gamma$, if $\Gamma \text{ ctx } \gamma$ then $A\gamma \equiv A'\gamma'$ type.
2. $\Gamma \gg M \equiv M'$ is $A$ when for all $\gamma \sim \gamma' \in \Gamma$, if $\Gamma \text{ ctx } \gamma$ then $A\gamma \equiv A\gamma'$ type and $M\gamma \equiv M'\gamma' \in A\gamma$.

Unlike our Definition 2.19, which requires each type of $\Gamma$ to respect all equalities in its indices, pointwise functionality requires only that each type respect equality at total assignments $\gamma$ of $\Gamma$. As a consequence, Nuprl admits several nonstandard rules, notably:
The first, called pointwise functionality, allows users to rewrite a hypothesis \( z : A \) as a hypothesis \( z : \text{base} \) such that \( z \in A \), so long as the conclusion is functional in equality in \( A \). The second is a very strong elimination rule for \( \text{nat} \) that omits the usual premise \( n : \text{nat} \gg A \in \text{Type}_i \). In Idealized Nuprl, that premise is required for the third premise to be well-formed, because \((n : \text{nat}, a : A)\) must be a context; in Nuprl, the pointwise functionality of \( A \) in \( \text{nat} \) is established inductively by earlier instances of \( S \). This rule nearly halves the number of inductions on \( \text{nat} \) one performs, because establishing \( n : \text{nat} \gg A \in \text{Type}_i \) itself often requires induction.

On the other hand, Nuprl’s open judgments do not validate the dependent cut principle\(^8\) (Lemma 2.24), but instead the weaker rule

\[
\Gamma, a : A, \Delta \gg M \in B \\
\Gamma \gg N \in A \\
\Gamma, \Delta \gg M[N/a] \in B[N/a]
\]

where \( \Delta \) does not depend on \( a \). Why does dependent cut fail? In both Nuprl and Idealized Nuprl, \( \Delta[M/a] \) may respect equality in \( \Gamma \) while \( \Delta \) does not respect equality in \((\Gamma, a : A)\). In Nuprl, this causes \((\Gamma, \Delta[M/a])\) \( \text{ctx} \) \( @ (\gamma, \delta) \) to hold while \((\Gamma, a : A, \Delta)\) \( \text{ctx} \) \( @ (\gamma, M, \delta) \) fails, and because these occur in antecedents of implications in Definition 2.53, the conclusion of dependent cut fails while its premises hold.

---

\(^8\)Dependent cut is only admissible in intensional type theory, allowing one in principle to consider models not closed under it; however, usual notions of categorical model of type theory [Dyb96] are closely related to explicit substitution calculi [Gra09, Chapter V] in which dependent cut is derivable.
3

Cubical methods

In this note it will be indicated how a homotopy theory may be developed for all abstract cubical complexes which satisfy only a certain extension axiom; homotopy groups will be introduced for all such complexes.

—Daniel M. Kan, Abstract Homotopy. I [Kan55, p. 1092]

This dissertation extends the computational semantics of type theory to account for the homotopy-type-theoretic features of univalence and higher inductive types. We begin this chapter by describing these features’ original formulations in Book HoTT [UF13]. Next, we motivate and compare various cubical approaches to homotopy type theory, from the symmetric monoidal cubical model of Bezem, Coquand, and Huber [BCH14; BCH18] to our Cartesian cubical type theory and the De Morgan cubical type theory of Cohen et al. [CCHM18]. Finally, we discuss possible variations on existing cubical techniques.

Book HoTT The equality type of Idealized Nuprl, like those of Nuprl [Con+85] and extensional type theory [ML82], satisfies straightforward rules that capture a mathematical notion of equality—proofs of equality are unique, equality is extensional at every type, and any element can be silently replaced by an equal element. In contrast, as discussed in Section 2.4, intensional type theory [ML75b] restricts silent definitional equality to α-, β-, and certain η-equivalences, mediating all other equations through an identity type whose rules are listed in Figure 3.1 [ML75b; UF13, Section A.2.10]. (We write \( \Gamma \vdash M = N : A \) for the judgments of intensional type theories, to emphasize that they have different intended properties than judgments of computational semantics \( \Gamma \gg M \equiv N \in A \).)

To understand the effects of definitional equality, consider addition of natural numbers, defined by recursion on the left argument (that is, \( z + m = m \) and \( s(n) + m = s(n + m) \)):

\[ n + m := \text{natrec}(n; m, \_, a.s(a)) \]

Here the equation \( z + m = m \) holds definitionally, whereas \( \text{Id}_{\text{nat}}(n + z, n) \) only holds propositionally, or up to the identity type, because the latter equation must be established by induction on \( n \). Given a nat-indexed family of types \( C(n) \), an element of \( C(z + m) \) is also an element of \( C(m) \), because these types are definitionally equal, but an element of \( C(n + z) \) must be transported over a proof of \( \text{Id}_{\text{nat}}(n + z, n) \) to obtain an element of \( C(n) \).
The requirement that proofs explicitly annotate uses of non-trivial equations allows validating erasing all uses of transport—because all closed identity proofs are reflexive—elements of not only $\text{Id}$ singleton contractibility in the following weaker sense, often called function extensionality, others fail completely, including allowing extracted programs to avoid any overhead induced by these annotations. Computational semantics nevertheless implement intensional type theory to decide equality of elements and thence typing judgments, at the cost of added bureaucracy. Cubical methods nevertheless validate erasing all uses of transport—because all closed identity proofs are reflexive—allowing extracted programs to avoid any overhead induced by these annotations.

Many semantically-natural equations hold only propositionally in intensional type theory, including uniqueness principles for many types [UF13, Corollary 2.7.3]. Many others fail completely, including function extensionality [Str93, p. 106], the principle that $(a:A) \to \text{Id}_{B}(f(a), g(a))$ implies $\text{Id}_{A \to B}(f, g)$, and uniqueness of identity proofs [HS98], the principle that $\text{Id}_{\text{Id}_{A}(a, a')}(p, p')$ for all $p, p': \text{Id}_{A}(a, a')$. However, identity proofs are unique in the following weaker sense, often called singleton contractibility:

$$\text{transport} : (C:B \to \text{Type}) \to (p:\text{Id}_{B}(b, b')) \to C b \to C b'$$

$$\text{transport } C \ p := J_{a,a',c(C(a) \to C(a'))}(p: \_\_c.c)$$

The requirement that proofs explicitly annotate uses of non-trivial equations allows implementations of intensional type theory to decide equality of elements and thence typing judgments, at the cost of added bureaucracy. Computational semantics nevertheless validate erasing all uses of transport—because all closed identity proofs are reflexive—allowing extracted programs to avoid any overhead induced by these annotations.

Transport is a crucial but straightforward consequence of identity elimination:

$$\text{transport} : (C:B \to \text{Type}) \to (p:\text{Id}_{B}(b, b')) \to C b \to C b'$$

$$\text{transport } C \ p := J_{a,a',c(C(a) \to C(a'))}(p: \_\_c.c)$$

As discussed in Section 1.2, the intensional identity type is compatible with non-trivial elements of not only $\text{Id}_{A}(a, b)$ but also $\text{Id}_{\text{Id}_{A}(a, b)}(\alpha, \beta).$ One such
element is postulated by Voevodsky’s univalence axiom [Voe10a], originally formulated:

\[ \text{idequiv} : (A : \text{Type}) \rightarrow \text{IsEquiv}(\lambda a. a) \]
\[ \text{idtoequiv} : \text{Id}_{\text{Type}_i}(A, B) \rightarrow ((f : A \rightarrow B) \times \text{IsEquiv}(f)) \]
\[ \text{idtoequiv} p := \lambda a. b. z. (\lambda a. a, \text{idequiv}(z)) \]
\[ \text{univalence} : \text{IsEquiv}(\text{idtoequiv}) \]

Here, IsEquiv\( (f) \) is a propositionally-subsingleton notion of isomorphism up to the identity type, and idequiv\( (A) \) is a proof that the identity function \( A \rightarrow A \) is such an equivalence. (Consult the Homotopy Type Theory book for many possible definitions of equivalence [UF13, Chapter 4].) For the purposes of this dissertation, we consider an equivalent but simpler presentation of univalence suggested by Licata [Lic16]:

\[ \text{ua} : (f : A \rightarrow B) \rightarrow \text{IsEquiv}(f) \rightarrow \text{Id}_{\text{Type}_i}(A, B) \]
\[ \text{ua} \beta : (f : A \rightarrow B) \rightarrow (e : \text{IsEquiv}(f)) \rightarrow (a : A) \rightarrow \text{Id}_{\text{B}}(\text{transport} (\lambda A.A) (\text{ua} f e) a, f a) \]

To see that univalence refutes uniqueness of identity proofs, consider the two proofs of \( \text{Id}_{\text{Type}_i}(\text{bool}, \text{bool}) \) obtained by applying \( \text{ua} \) to the equivalences \( \lambda b. b \) and \( \lambda b. \text{if} (b; \text{false}, \text{true}) \). If these were equal, (transport \( (\lambda A.A) \rightarrow \text{true} \)) across each would be equal, but \( \text{ua} \beta \) propositionally equates these transports to true and false respectively. Moreover, a hierarchy of \( n \) univalent universes refutes uniqueness of \( n \)-fold iterated identity proofs [KS15].

Models of univalent type theory must therefore account for arbitrarily-iterated identity structure; Voevodsky’s model [KL16] interprets types as simplicial sets—a combinatorial representation of topological spaces using triangles and tetrahedra of arbitrary dimension—and \( n \)-fold iterated identity types as (approximately) the collection of \( n \)-dimensional simplices of a type.

**Computational content**  Univalence essentially adds new introduction rules (\( \text{ua} \) and \( \text{ua} \beta \)) to Book HoTT’s identity type without adding computation rules describing how \( J_{a. b. p. c. (\_ ; z.Q)} \) acts on these identity proofs. Book HoTT therefore lacks the canonicity and existence properties characteristic of type theories; for instance, transporting true across univalence yields a closed Boolean definitionally equal to neither true nor false:

\[ \text{transport} (\lambda A.A) (\text{ua} (\lambda b. b) (\text{idequiv bool})) \text{ true} : \text{bool} \]

As a result, proof assistants based on Book HoTT cannot fully simplify proof goals by open computation. Users of Book HoTT instead “compute propositionally” by manually invoking lemmas to simplify transports according to type family or identity proof.\(^1\) The \( \text{ua} \beta \) axiom is one such lemma, simplifying the above transport to true; another propositionally

\(^1\)Readers can consult Angiuli et al. [Ang+16] for many examples of “propositional computation.”
equates the transport of \((M, N)\) in a family of product types to the pair of transports of \(M\) and \(N\) \([\text{UF13, Theorem 2.6.4}]\).

In fact, Sattler and Kapulkin have recently shown\(^2\) that “propositional computation” enjoys homotopy canonicity \([\text{Voe10b}]\)—every closed Boolean can be propositionally simplified to true or false. Such simplifications are nevertheless challenging in practice, as famously demonstrated by Brunerie’s number, a topological invariant of type \(\text{nat}\) defined in Book HoTT. In ordinary type theories, any such definition must compute to a concrete numeral. In Book HoTT, however, Brunerie could not prove it propositionally equal to any numeral; he later gave a different construction in which the invariant is \(2\) \([\text{Bru16}]\).

These and other difficulties led researchers to seek univalent type theories with definitional canonicity, hoping that such type theories would be more usable (as their proof goals would simplify further), would enable the use of univalence in dependently-typed programming (as explored, for instance, by Angiuli et al. \([\text{Ang+16}]\), and would produce a constructive model of univalence simpler than Voevodsky’s model, which is essentially classical \([\text{BC15}]\).

To this end, Licata and Harper \([\text{LH12}]\) established canonicity for a “truncated” type theory with non-trivial proofs of \(\text{Id}_{\text{A}}(M, N)\), but definitional uniqueness of \(\text{Id}_{\text{id}(M,N)}(P, Q)\) proofs. This type theory has an auxiliary judgment \(\Gamma \vdash P : M \simeq_{\text{A}} N\) expressing that \(P\) is a “term equivalence” between \(M\) and \(N\) in \(\text{A}\); any such term equivalence gives rise to an element of \(\text{Id}_{\text{A}}(M, N)\), and vice versa. Using their auxiliary judgment, Licata and Harper explicitly axiomatize all operations on term equivalences (transport, symmetry and transitivity of identity, \textit{et cetera}), providing enough computation rules (transport in a family of product types, transport across univalence, \textit{et cetera}) to achieve canonicity.

Licata and Harper’s term equivalence judgment reconciles non-trivial identity proofs with the type-theoretic aphorism\(^3\) that \textit{type formers internalize judgmental structure}. Indeed, dependent function types internalize membership-under-hypotheses \((\lambda a.M : (a:A) \to B\) whenever \(a:A \vdash M : B\)), identity types originally internalized definitional equality \((\text{refl}(M) : \text{Id}_{\text{A}}(M, N)\) whenever \(M = N : A\)), and the identity types of Licata and Harper \([\text{LH12}]\) internalize term equivalence. The failure of this aphorism for Book HoTT’s identity type—which is generated by \(\text{refl}(M)\), univalence, and even identity elimination—syntactically obscures the principle that \(\text{Id}_{\text{A}}(M, N)\) mirrors the structure of \(A\) (for instance, that transport in a family of product types is a pair of transports, or that propositional equality in a product type is a pair of propositional equalities).

However, Licata and Harper’s approach does not scale to the infinitely-iterated identity structure found in Book HoTT, which would require not only infinitely many term equivalence judgments, but also infinitely many operations and computation rules.

\(^2\)See the TYPES ’19 abstract at \url{http://www.ii.uib.no/~bezem/abstracts/TYPES_2019_paper_110}.

\(^3\)Also known as Martin-Löf’s judo move.
3.1 Symmetric monoidal cubes

Bezem, Coquand, and Huber [BCH14; BCH18] sparked a cubical revolution by presenting a constructive model of univalent type theory in symmetric monoidal cubical sets, a combinatorial presentation of infinite-dimensional structure analogous to simplicial sets. Cubical sets and simplicial sets generalize directed graphs ($n = 1$) in two different ways, as illustrated in Figure 3.2—an $n$-dimensional simplex has $n + 1$ faces of dimension $n - 1$, while an $n$-dimensional cube has $2n$ faces of dimension $n - 1$. Symmetric monoidal cubical sets are the first of three variations on cubical sets discussed in this dissertation, each with more operations than the last.

**Definition 3.1.** The symmetric monoidal cube category $\Box_\otimes$ has as objects finite sets, and as morphisms $I \to J$ set-theoretic functions $f : J \to I + \{0, 1\}$ for which $f|_{f^{-1}(I)}$ is injective (equivalently, for which $f(x) = f(y) \in I$ implies $x = y$) [BCH14; Pit15]. Composition of morphisms is the evident Kleisli composition, namely, $(g \circ f)(x) = f(x)$ when $f(x) \in \{0, 1\}$ and $g(f(x))$ otherwise.

Notable morphisms of $\Box_\otimes$ include face maps

$$(x_i \coloneqq 0), (x_i \coloneqq 1) : (\{x_1, \ldots, x_n\} \setminus \{x_i\}) \to \{x_1, \ldots, x_n\}$$

that are identity functions except for sending $x_i$ to 0 or 1, and degeneracy maps

$$\hat{y} : \{x_1, \ldots, x_n, y\} \to \{x_1, \ldots, x_n\}$$

Bernardy, Coquand, and Moulin [BCM15] consider a very similar category whose morphisms $I \to J$ are functions $f : J \to I + \{0\}$ satisfying the same injectivity condition.
that miss $y$ and are otherwise identity functions.

A symmetric monoidal cubical set $F$ is a presheaf on $\square_\otimes$, that is, a functor $\square_\otimes^{op} \to \text{Set}$, or a family of sets $F(I)$ for each $I \in \square_\otimes$ equipped with functions $F(f) : F(I) \to F(J)$ for each $f : J \to I$ in $\square_\otimes$. Geometrically, one can imagine $F$ as a space and $F(\{x_1, \ldots, x_n\})$ as the set of continuous functions $[0, 1]^n \to F$, or $n$-cubes of $F$; then face maps $F(x_i := \epsilon)$ restrict $n$-cubes to $(n - 1)$-cubes by setting the $x_i$ coordinate to $\epsilon$, while degeneracy maps $F(\hat{y})$ regard $n$-cubes as $(n + 1)$-cubes constant in the new $y$ coordinate. (See Figure 3.3.) Faces and degeneracies satisfy the expected geometric identities—the “top” face of the “left” face of a square is also the “left” face of its “top” face (because $(x := 0) \circ (y := 0) = (y := 0) \circ (x := 0)$), and the “top” and “bottom” faces of a vertically-degenerate square are equal (because $\hat{y} \circ (y := 0) = \hat{y} \circ (y := 1) = \text{id}$).

Cubical sets are a convenient representation of infinitely-iterated data, but lack enough structure needed to model spaces or identity types. In a space $X$, every path $p : [0, 1] \to X$ has an inverse path $p^{-1}$ with $p^{-1}(0) = p(1)$ and $p^{-1}(1) = p(0)$; similarly, in intensional type

\[
\begin{array}{c}
\text{Faces} \\
(\square_\otimes, \square, \square_{\text{DM}}) \\
\begin{array}{c}
p \downarrow \\
\rightarrow \\
q \downarrow \\
\rightarrow \\
r \downarrow \\
\rightarrow \\
s \downarrow \\
\rightarrow \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Degeneracies} \\
(\square_\otimes, \square, \square_{\text{DM}}) \\
\begin{array}{c}
a \longrightarrow b \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Diagonals} \\
(\square, \square_{\text{DM}}) \\
\begin{array}{c}
a \longrightarrow b \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Connections} \\
(\square_{\text{DM}}) \\
\begin{array}{c}
a \longrightarrow b \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Reversals} \\
(\square_{\text{DM}}) \\
\begin{array}{c}
a \longrightarrow b \\
\end{array} \\
\end{array}
\]

Figure 3.3: Operations in cubical sets.
theory, identity is symmetric:

\[ \lambda p. \text{transport} (\lambda c. \text{Id}_A(c, a)) \; p \; \text{refl}(a) : \text{Id}_A(a, b) \to \text{Id}_A(b, a) \]

In symmetric monoidal (or Cartesian) cubical sets, however, \( n \)-cubes need not be invertible. Mathematicians describe spacelike cubical sets as those satisfying the Kan condition [Kan55], namely, for any configuration of \( (n-1) \)-cubes forming all but one face of an \( n \)-cube, there exists an \( n \)-cube with that boundary. For \( n = 2 \), the Kan condition asserts that for any three lines forming the left, top, and right sides of a square (evocatively called an open box), there exists a square (a filler) whose left, top, and right faces are those lines.

In their model of type theory, Bezem, Coquand, and Huber [BCH14] introduce uniform Kan operations, which refine the Kan condition in three critical ways. First, rather than stipulating the mere existence of fillers, the uniform Kan operations determine functions sending each open box to a chosen filler. Secondly, open boxes are allowed to omit more than one face from a cube, and in fact can specify as few as one face. Thirdly, the uniform Kan operations must commute with face and degeneracy maps—degenerating the filler of an open box must coincide with the filler of the degenerated open box:

Bezem, Coquand, and Huber [BCH14; BCH18] therefore model types as symmetric monoidal cubical sets equipped with a uniform Kan operation; their model includes dependent function types, dependent pair types, identity types, and univalent universes.

Towards cubical type theory The model of Bezem, Coquand, and Huber [BCH14] brought us closer to a univalent type theory with canonicity—it allows one to compute the values of well-typed terms, but not to reflect those computations as definitional (or even propositional) equalities of the theory. However, various researchers immediately set out to develop a type theory based on uniform Kan cubical sets, using the insight that cubical sets have a natural syntactic representation.

In Section 3.2 we will define uniform Kan operations more precisely in the Cartesian setting. Readers specifically interested in the setting of Bezem, Coquand, and Huber [BCH14] may wish to consult the detailed exposition of Harper and Hou (Favonia) [HF15].

In December 2013, Bezem, Coquand, and Huber released cubical (https://github.com/simhu/cubical), a prototype evaluator with precisely this purpose.
By the Yoneda lemma, the \( n \)-cubes of a cubical set \( F \) are in bijection with the natural transformations \( \text{hom}_{\Delta^n}(-, \{x_1, \ldots, x_n\}) \rightarrow F \). The Yoneda embedding preserves limits, including the symmetric monoidal product \( \otimes \) of \( \Delta^n \), for which \( \{x_1, \ldots, x_n\} = \{x_1\} \otimes \cdots \otimes \{x_n\} \). Therefore, writing \( I = \text{hom}_{\Delta^n}(-, \{\star\}) \) for the representable \( 1 \)-cube, we obtain a bijection between the \( n \)-cubes of \( F \) and natural transformations \( \mathbb{I}^n \rightarrow F \).

These calculations justify our earlier intuition that \( n \)-cubes of \( F \) are continuous functions \( [0, 1]^n \rightarrow F \). (Unlike \([0, 1], I \) is an abstract interval with only two elements, 0 and 1.) Critically, \( F \)'s structure is determined by morphisms out of a product of cubical sets, just as the judgments of type theory describe morphisms out of products of types. We therefore consider cubical judgments expressing that \( M \) is an \( n \)-cube of \( A \):

\[
x_1 : I, \ldots, x_n : I \vdash M : A
\]

Cubical judgments involve a new form of interval variable \( x_i \), subject to interval substitutions \( \langle 0/x_i \rangle \) and \( \langle 1/x_i \rangle \) which implement the action of face maps \((x_i := 0)\) and \((x_i := 1)\), and weakenings, which implement the action of degeneracy maps. Cubical type theory therefore scales Licata and Harper’s type theory \([LH12]\) to arbitrary dimension: interval variables uniformly define \( n \)-dimensional judgments, and uniform Kan operations, which commute with interval substitution and can therefore be represented syntactically, compactly axiomatize all operations on \( n \)-cubes.

A frequent question from mathematicians is: why cubical type theory and not simplicial type theory? Unlike in cubical sets, the representable \( n \)-simplex is not a product of representable \( 1 \)-simplices. Mathematicians often prefer simplicial sets because geometric realization commutes with products of simplicial sets up to homotopy, while the same is not true of symmetric monoidal cubical sets \([BM17]\); however, as our spacelike structure is induced explicitly through the uniform Kan operation, rather than abstractly through equivalence of categories \([Cis06]\), these concerns appear irrelevant to type theory.

Symmetric monoidal cubes nevertheless pose two difficulties to type theorists. First, their interval variables are substructural—by the injectivity condition of Definition 3.1, there are no contraction substitutions joining two variables—and many subtleties arise when unifying dependency and substructurality \([KPB15]\). More importantly, the elimination principles of higher inductive types appear to require contraction. Suppose we have a higher inductive type \( T \) with \( 1 \)-cube constructor \( x : I \vdash \text{line}(x) \). Functions \( (t : T) \rightarrow A \) are therefore determined by \( 1 \)-cubes \( x : I \vdash L : A[\text{line}(x)/t] \), and send constructors \( \text{line}(y) \) to \( L(y/x) \), which contracts \( x \) and \( y \) in \( L \):

\[
\begin{align*}
\Gamma, t : T \vdash A : \text{Type}_i \\
\Gamma \vdash M : T \\
\Gamma, x : I \vdash L : A[\text{line}(x)/t] \\
\Gamma \vdash \text{elim}_{I,A}(M;x.L) : A[M/t]
\end{align*} \quad \begin{align*}
\Gamma, t : T \vdash A : \text{Type}_i \\
\Gamma, x : I \vdash L : A[\text{line}(x)/t] \\
\Gamma, t \vdash \text{elim}_{I,A}(\text{line}(y);x.L) = L(y/x) : A[\text{line}(y)/t]
\end{align*}
\]
3.2 Cartesian cubes

Cartesian cubical sets augment symmetric monoidal cubical sets with contraction substitutions \((y/x)\), whose actions compute the diagonals of \((n+2)\)-cubes. (See Figure 3.3.) Cartesian interval variables are structural (satisfy exchange, weakening, and contraction) and, unlike De Morgan interval variables, are subject to no equations.

**Definition 3.2.** The **Cartesian cube category** \(\Box\) has as objects finite sets, and as morphisms \(I \rightarrow J\) set-theoretic functions \(J \rightarrow I + \{0, 1\}\), with the evident Kleisli composition [AFH18; Ang+19; Awo18; Par15].

Work on Cartesian cubes dates to around 2014, when Coquand\(^7\) and Licata and Brunerie [LB14] presented fragments of Cartesian cubical type theories. In 2016, Awodey [Awo18] presented a model of identity types in Cartesian cubical sets. In 2017, the author and collaborators succeeded in defining a univalent Cartesian cubical type theory with canonicity, presented both as a computational type theory (Appendix A) with a computational semantics (Chapter 4) [AFH18], and as an intensional type theory (Appendix B) with a denotational semantics in Cartesian cubical sets [Ang+19].

In Chapter 4 we will decompose the uniform Kan operation into **homogeneous composition**, which fills boxes in cubically-constant types, and **coercion**, a form of transport for cubically-dependent types. This decomposition is essential when defining higher inductive types [CH19a]; De Morgan cubical type theory adopts a similar decomposition but only at higher inductive types [CHM18]. Figure 3.4 contains rules for our Kan operations, in which \(r\) and \(r'\) are interval terms, or elements of \(I\) (concretely, either 0, 1, or interval variables), and \(\xi_i\) are interval equations (concretely, \(r = r'\)) that restrict judgments to particular subcubes.

\(^7\)See the note at [http://www.cse.chalmers.se/~coquand/diag.pdf](http://www.cse.chalmers.se/~coquand/diag.pdf), dated April 9, 2014.
The reflective path on \( x \) vary in.

Identity elimination

To familiarize ourselves with these Kan operations, we will consider path types (Figure 3.5), a cubical analogue of identity types\(^8\) that propositionally—but not definitionally—model the computation rule in Figure 3.1. Elements of \( \text{Path}_{x:A}(P_0, P_1) \) are dependent functions \((x:\mathbb{I}) \to A\) that send 0 and 1 to \( P_0 \) and \( P_1 \) respectively; like ordinary functions, paths are introduced by abstraction \((x)M\) and eliminated by application \((M@r)\). The reflexive path on \( M : A \) is a constant function \(_xM\), which has type \( \text{Path}_{x:A}(M, M) \).

We define the “identity eliminator” of path types using the folkloric observation that identity elimination is interderivable with transport and uniqueness. Transport for path types is a straightforward instance of coercion:

\[
\text{transport}_{\mathbb{I}} : (C:B \to \text{Type}) \to (P:\text{Path}_{x:B}(b, b')) \to C b \to C b'
\]

\[
\text{transport}_{\mathbb{I}} C p c := \text{coe}^{\text{Type} \to \text{Type}}_{x:C} (p@x)(c)
\]

observing that \( (C (p@x))(0/x) = C b \) and \( (C (p@x))(1/x) = C b' \).

Singleton contractibility is best explained graphically:

\[
\text{Path}_A(a) := (b:A) \times \text{Path}_{x:A}(a, b)
\]

\[
\text{uniqueness}_{\mathbb{I}} : (a:A) \to (\hat{P}:\text{Path}_A(a)) \to \text{Path}_{x: \text{Path}_A(a)}((a, (_x a), \hat{P})
\]

\[
\text{uniqueness}_{\mathbb{I}} a (b, p) := \langle x \rangle \left(a \begin{array}{c} \text{\_} \\ \text{\_} \\ \end{array} \right) \left( b, \langle y \rangle a \begin{array}{c} \text{\_} \\ \text{\_} \\ \end{array} p \right)
\]

\(^8\)In fact, they correspond to Book HoTT’s dependent paths [UF13, Section 2.3], because the type \( A \) can vary in \( x : \mathbb{I} \).
To construct an element of type $\text{Path}_{\text{Path}_A(a)}(\langle a, \bot \rangle, \hat{p})$, we abstract $x : \top$ and construct an element of type $(b:A) \times \text{Path}_{\text{Path}_A(a, b)}$ equal to $\langle a, \bot \rangle$ when $x = 0$ (on the left) and $\langle b, p \rangle$ when $x = 1$ (on the right). Such an element is itself a pair; for its first component we could choose $p@x$, as $p$ is a path from $a$ to $b$, but let us momentarily postpone our choice.

By the ordinary rules of dependent pair types, the second component must have type $\text{Path}_{\text{Path}_A(a, q)}$, where $q$ is the first component. We exhibit such a path by abstracting $y : \top$ and constructing an element of $A$ equal to $a$ when $y = 0$ (on the top) and $q$ when $y = 1$ (on the bottom). However, these are not our only constraints—as we noted when abstracting $x : \top$, the path must also equal $\langle \bot \rangle a$ when $x = 0$ and $p$ when $x = 1$.

In sum, we require a square equal to $a$ when $x = 0$ or $y = 0$, to $p$ when $x = 1$, and to $q$ when $y = 1$. We select the following filler, depicted in our definition of uniqueness as a hatched square:

$$F := \text{hcom}^0_{\top} \{a; x = 0 \leftrightarrow \bot, a, x = 1 \leftrightarrow y, p@y\}$$

The premises of the hcom rule require (1) $A$ is a type, (2) $a : A$, (3) $a : A$ when $x = 0$ and $p@y : A$ when $x = 1$, (4) $a = p@y : A$ whenever both $x = 0$ and $x = 1$, which is vacuous, and (5) $a@0/y = a : A$ when $x = 0$ and $(p@y)@0/y) = a : A$ when $x = 1$. The conclusions assert (1) $F : A$, (2) $F@0/y = a : A$, and (3) $F = a : A$ when $x = 0$ and $F = p@y : A$ when $x = 1$. These conclusions discharge all of our obligations except $F@1/y = q : A$, which we discharge by defining $q := F@1/y$, the missing bottom face of the filler.

The hcom rule in Figure 3.4 is quite technical because it systematically accounts for all possible filling scenarios; particular instances, such as $F$, are often much simpler. For example, it is essential that $\xi$, not depend on $y : \top$; otherwise, one could fill closed boxes and obtain uniqueness of identity proofs. We defer most technical details to Section 4.3.2, whose definition of hcom differs from Figure 3.4 in primarily cosmetic ways: the computational semantics divide interval variables, interval equations, and ordinary variables into three separate contexts, and explicitly close hcom under interval substitution by admitting composition to any $r'$, not only $y : \top$. (A more significant difference is the validity condition, discussed in Section 3.5.)

We are finally prepared to derive an uncurried variant of identity elimination:

$$J^\top: (a:A) \to (C: \text{Path}_A(a) \to \text{Type}) \to C \langle a, \bot \rangle \to (\hat{p}: \text{Path}_A(a)) \to C \hat{p}$$

$$J^\top a C c \hat{p} := \text{transport}^\top C (\text{uniqueness}^\top a \hat{p}) c$$

We do not obtain $J^\top a C c \langle a, \bot \rangle = c : C \langle a, \bot \rangle$, which seems to require both $\text{uniqueness}^\top a \langle a, \bot \rangle$ to be a constant path, and $\text{transport}^\top C p c$ to be the identity function when $p$ is constant. These conditions, jointly named regularity, are $a$ priori sensible but have proven quite elusive; we discuss regularity further in Section 3.4.

Instead, we exhibit a path between $J^\top a C c \langle a, \bot \rangle$ and $c$, or an element of:

$$(a:A) \to (C: \text{Path}_A(a) \to \text{Type}) \to (c:C \langle a, \bot \rangle) \to \text{Path}_{\text{Path}_A(a, c)}(\langle a, \bot \rangle, c)$$
Because \( J \) is an instance of transport, which is in turn an instance of coercion, we obtain a dependent path between \( J a C c \langle a, \langle \_ \rangle a \rangle \) and \( c \) by coercing to an interval variable:

\[
\lambda C. \lambda p. \lambda c. (x) \text{coe}_{x, C (p@x)}^0 (c) : (C : B \rightarrow \text{Type}) \rightarrow (p : \text{Path}_\_ (b, b')) \rightarrow (c : C b) \rightarrow \text{Path}_{x, C (p@x)} (c, \text{transport}_{\_} C p c)
\]

observing that \( \text{coe}_{x, C (p@x)}^0 (c)) (0/x) = c : C b \) and \( \text{coe}_{x, C (p@x)}^0 (c)) (1/x) \) is by definition transport \( C p c \).

When specialized to \( J \), we obtain an element of

\[
\text{Path}_{x, C ((\text{uniqueness}_{\_} a \langle a, \langle \_ \rangle a \rangle)@x)} (c, J a C c \langle a, \langle \_ \rangle a \rangle)
\]

a path with the desired endpoints but in the wrong type. To correct its type, we exhibit a path between \( \text{uniqueness}_{\_} a \langle a, \langle \_ \rangle a \rangle \) and \( \langle \_ \rangle \langle a, \langle \_ \rangle a \rangle \), which amounts to a path between \( \langle \_ \rangle a \) and the composition of \( a, a, \) and \( a \), defined as the front face of a filler whose back, right, top, and bottom faces are \( a \), and whose left face is the filler of \( a, a, \) and \( a \):

![Diagram](image)

We complete the proof by transporting our dependent path over the propositional equality between \( \text{uniqueness}_{\_} a \langle a, \langle \_ \rangle a \rangle \) and \( \langle \_ \rangle \langle a, \langle \_ \rangle a \rangle \), then inverting the result.

Path types are incredibly useful in cubical type theory, notwithstanding their failure to strictly model the computation rule of identity types. However, there are also constructions that \textit{do} strictly model the rules of identity types (but not the rules of path types). The first, due to Swan [Swa18a], was originally conceived in the symmetric monoidal and De Morgan settings but has later been ported to the Cartesian setting by Angiuli et al. [Ang+19]. The second, due to Cavallo and Harper [CH19a], obtains identity types as an instance of indexed inductive types.

**Closure under type formers** To achieve canonicity, all hcom and coe terms must compute. In Chapter 4, we include computation steps defining the Kan operations at compound types in terms of the Kan operations at constituent types. The same principle applies in denotational semantics, in which one must check that, for example, the product of
Kan cubical sets is again Kan. Closure of Kan operations under the type formers, and particularly univalent universes, is one of the most technical aspects of cubical type theory.

Our construction of $J_{\mathcal{E}}$ has already required coercions $0 \rightsquigarrow r'$ and homogeneous compositions $0 \rightsquigarrow r'$ with equations $(x = 0), (x = 1)$. We now argue that closure under dependent function and path types requires coercion and composition from arbitrary interval terms $r$ as well.

Suppose $x : \mathbb{I} \vdash (a : A) \rightarrow B : Type_1$ and $M : ((a : A) \rightarrow B)(0/x), and define a function $((a : A) \rightarrow B)(r'/x)$ by abstracting $a : A(r'/x)$. We cannot apply $M$ to $a$ directly, but we can apply $M$ to $\text{coe}_{x,A}^{r' \rightsquigarrow 0}(a)$, obtaining an element of $B(0/x)[\text{coe}_{x,A}^{r' \rightsquigarrow 0}(a)/a]$. If $B$ were not dependent, we would have an element of $B(0/x)$ and could simply coerce $0 \rightsquigarrow r'$ in $B$. Instead, we must coerce in $B[\text{coe}_{x,A}^{r' \rightsquigarrow x}(a)/a]$, a type whose $\langle 0/x \rangle$ face is $B(0/x)[\text{coe}_{x,A}^{r' \rightsquigarrow 0}(a)/a]$, and whose $\langle r'/x \rangle$ aspect is $B(\langle r'/x \rangle)$ (by $\text{coe}_{x,A}^{r' \rightsquigarrow r'}(a) = a : A(\langle r'/x \rangle)$). In sum:

$$\text{coe}_{x,(a : A) \rightarrow B}(M) := \lambda a.\text{coe}_{\langle 0/x \rangle \rightarrow \langle a \rangle}^{r' \rightsquigarrow 0}x,B[\text{coe}_{x,A}^{r' \rightsquigarrow 0}(a)/a](M \text{coe}_{x,A}^{r' \rightsquigarrow 0}(a))$$

Because we need coercion $0 \rightsquigarrow r'$ in all types to define $J_{\mathcal{E}}$, and we need coercion $r' \rightsquigarrow x$ in all types to obtain coercion $0 \rightsquigarrow r'$ in all dependent function types, we conclude that all coercions $r \rightsquigarrow r'$ are required.

Now suppose $y : \mathbb{I} \vdash \text{Path}_{x.A}(P_0, P_1) : Type_i$ and $M : (\text{Path}_{x.A}(P_0, P_1))(r/y)$ and define a path $(\text{Path}_{x.A}(P_0, P_1))(r/y)$ by abstracting $x : \mathbb{I}$. The coercion $\text{coe}_{y,A}^{r' \rightsquigarrow x}(M@x) : A(\langle r'/y \rangle)$ has the correct type, but its $\langle \epsilon/x \rangle$ faces are $\text{coe}_{y,A(\epsilon/x)}^{r' \rightsquigarrow x}(P_{\epsilon}(r/y))$ and not $P_\epsilon(\langle r'/y \rangle)$ as required. However, under $x = \epsilon$, $\text{coe}_{y,A}^{r' \rightsquigarrow x}(P_{\epsilon}) : A(\langle r'/y \rangle)$ is a $y$-cube whose $\langle r/y \rangle$ aspect agrees with our coercion of $M@x$, and whose $\langle r'/y \rangle$ aspect agrees with $P_\epsilon(\langle r'/y \rangle)$. We obtain the result by composing the three aforementioned elements of $A(\langle r'/y \rangle)$:

$$\text{coe}_{y,\text{Path}_{x.A}(P_0,P_1)}^{r' \rightsquigarrow x}(M) := (\langle x \rangle)\text{hcom}_{A(\langle r'/y \rangle)}^{r' \rightsquigarrow x}(\text{coe}_{y,A}^{r' \rightsquigarrow x}(M@x); x = \epsilon \leftrightarrow y.\text{coe}_{y,A}^{r' \rightsquigarrow x}(P_{\epsilon}))$$

To define coercion $r \rightsquigarrow r'$ in path types, we thus require composition $r \rightsquigarrow r'$ in all types.

Finally, consider homogeneous composition in $\text{Path}_{x.A}(P_0, P_1)$:

$$\text{hcom}_{\text{Path}_{x.A}(P_0,P_1)}^{r' \rightsquigarrow x}(M; \xi_i \leftrightarrow y.N_i) := (\langle x \rangle)\text{hcom}_{A(\langle r'/y \rangle)}^{r' \rightsquigarrow x}(M@x; x = \epsilon \leftrightarrow y.\text{coe}_{y,A}^{r' \rightsquigarrow x}(P_{\epsilon}))$$

If we omitted the equations $(x = 0), (x = 1)$ in the definiens, its $\langle \epsilon/x \rangle$ faces would be $\text{hcom}_{A(\langle r'/x \rangle)}^{r' \rightsquigarrow x}(P_{\epsilon}, \xi_i \leftrightarrow _{-}P_{\epsilon})$ and not $P_{\epsilon}$ as required. Therefore, to define composition of shape $\xi_1, \ldots, \xi_n$ in path types, we require composition of shape $\xi_1, \ldots, \xi_n, (x = 0), (x = 1)$.

Coercion $r \rightsquigarrow r'$ and composition $r \rightsquigarrow r'$ of shape $(x_1 = 0), (x_1 = 1), \ldots, (x_n = 0), (x_n = 1)$ suffice to define dependent function, dependent pair, and path types [AHW17].

---

9One must also verify $\text{coe}_{x,(a : A) \rightarrow B}(M) = M$. Simplifying our definition, $\text{coe}_{x,(a : A) \rightarrow B}(M) = \lambda a.M a$, so the result follows by the (definitional) uniqueness rule for dependent function types.
As we will see in Chapter 4, univalent universes require also (1) equations corresponding to diagonals \(x = z\), introduced by Angiuli, Hou (Favonia), and Harper [AFH17], and (2) closure of composition shapes under the deletion of equations that depend on \(x : \mathbb{I}\), an operation named “\(\forall x\)” by Cohen et al. [CCHM18].

At the time of writing, two prototype proof assistants implement Cartesian cubical type theory. The first, RedPRL [Red16], implements a computational type theory—in the sense of Section 2.4—with both equality types (as in Chapter 2) and path types. The second, redtt [Red18], implements an intensional type theory—in the sense of Section 2.4—and currently lacks equality types. (With Anders Mörberg, the author has also implemented yaccctt (https://github.com/mortberg/yacctt), a defunct experimental type-checker for intensional Cartesian cubical type theory.)

### 3.3 De Morgan cubes

In 2016, while investigations into Cartesian cubical type theory were ongoing, Cohen et al. [CCHM18] discovered that one can substantially simplify the Cartesian uniform Kan operation at the cost of adding further operations to the cube category, leading them to define a univalent De Morgan cubical type theory with canonicity [Hub18] and a denotational semantics in De Morgan cubical sets [CCHM18; OP16; Lic+18].

**Definition 3.3.** The De Morgan cube category \(\square_{\text{DM}}\) has as objects finite sets, and as morphisms \(I \to J\) set-theoretic functions \(J \to \text{DM}(I)\), where \(\text{DM}(I)\) is the free De Morgan algebra on \(I\), with the evident Kleisli composition [CCHM18]. (A De Morgan algebra has binary operations \(\cap, \lor\) and constants 0, 1 forming a bounded distributive lattice, and an involution \(\neg\) satisfying \(\neg(r \lor r') = \neg r \land \neg r'\) and \(\neg(r \land r') = \neg r \lor \neg r'\).

De Morgan cubical type theory augments Cartesian cubical type theory with three interval term formers: **connections** \(r \land r'\) and \(r \lor r'\) which compute the minimum and maximum of \(r, r'\) respectively, and **reversals** \(\neg r\) which compute \(1 - r\). Geometrically, connections provide an alternative form of degeneracy that is constant on two non-opposing faces, while reversals exchange the endpoints of one interval. (See Figure 3.3.)

These connections and reversals allow Cohen et al. [CCHM18] to consider a uniform Kan operation limited to \(0 \rightsquigarrow 1\) and shapes generated by conjunctions and disjunctions of \((x_i = 0)\) and \((x_i = 1)\). Figure 3.6 describes their heterogeneous composition operation for cubically-dependent types, which encompasses both homogeneous composition (when \(A\) is constant in \(y : \mathbb{I}\)) and coercion (when there are no \(\xi_i\)). It is often easier to construct instances of comp than of the Cartesian hcom and coe—unlike hcom and coe, the source and target \(r \rightsquigarrow r'\) of comp can never become equal, so comp is subject only to one equation (“under \(\xi_i\)” and not two (“under \(\langle r/y \rangle\)”)).
\[\Gamma, y : \mathbb{I} \vdash A : \text{Type}_i\]
\[\Gamma \vdash M : A\langle 0/y \rangle\]
\[(\forall i)\quad \Gamma, \xi_i, y : \mathbb{I} \vdash N_i : A\]
\[(\forall i, j)\quad \Gamma, \xi_i, \xi_j, y : \mathbb{I} \vdash N_i = N_j : A\]
\[(\forall i)\quad \Gamma, \xi_i \vdash N_i\langle 0/y \rangle = M : A\langle 0/y \rangle\]
\[\Gamma \vdash \text{comp}_{y, A}(M; \xi_i \hookrightarrow y.N_i) : A\langle 1/y \rangle\]
\[= N_i\langle 1/y \rangle\quad \text{under} \quad \xi_i\]

Figure 3.6: De Morgan cubical Kan operation.

Surprisingly, many of the constructions described in Section 3.2 can be recovered in the De Morgan setting. For instance, forward coercion \(0 \Rightarrow 1\) in \(x : I \vdash (a : A) \rightarrow B : \text{Type}_i\) previously required backward coercion \(1 \Rightarrow 0\) in \(x : I \vdash A : \text{Type}_i\), but here we can simply coerce (forward) in \(x : I \vdash A\langle \neg x/x \rangle : \text{Type}_i\). As before, \(J_{\Rightarrow}\) (with propositional computation rule) follows from transport_{\Rightarrow} and uniqueness_{\Rightarrow}, but the latter is now a direct consequence of connections: uniqueness_{\Rightarrow} \(a \langle b, p \rangle := \langle x \rangle \langle p@x, \langle y \rangle p@(x \land y) \rangle\). Finally, while composition \(0 \Rightarrow 1\) appears to provide only the “missing face” of a filler and not its interior, one can recover filling from composition and connections in an ingenious fashion:

That is, we can define the filler of \(p, q,\) and \(r\) as the missing bottom face of an open box whose top and back faces are \(p\) degenerated, and whose left and right faces are connections of \(r\) and \(q\), that is, squares whose top and back faces are degenerate and whose front and bottom faces are \(r\) and \(q\) respectively.

Conversely, in the Cartesian setting, one can define weak connections and reversals that have the correct faces and diagonals but do not satisfy the equations of a De Morgan algebra; for example, reversing a path twice is only propositionally an involution.

De Morgan cubical type theory is the basis of a recent cubical extension to the Agda proof assistant [VMA19]; earlier, Cohen, Coquand, Huber, and Mörtberg implemented it in an experimental type-checker cubicaltt (https://github.com/mortberg/cubicaltt).

\(^{10}\)In fact, Cohen et al. [CCHM18] need reversals only for backward composition, and implicitly also define a distributive lattice cubical type theory that replaces reversals by \(\text{comp}_{y, A}^{0 \Rightarrow 1}\) and \(\text{comp}_{y, A}^{1 \Rightarrow 0}\).
3.4 Regularity

In cubical type theories, paths in $A$ correspond to open elements of $A$. Path types are therefore quite well-behaved: they are automatically dependent (in the sense of Book HoTT [UF13, Section 2.3]), and commute appropriately with other type formers, including dependent function types:

$$\text{funext} : ((a : A) \to \text{Path}_{\sim B}(f, g) \to \text{Path}_{\sim (a : A) \to B}(f, g))$$

$$\text{funext } h := \lambda a. (h a) @ x$$

One of their few drawbacks is that path types do not strictly satisfy the rules of identity types, which would allow cubical type theory to import proofs from Book HoTT in a natural way, sending identity types to path types.

Recall our cubical definition of $J$ in Section 3.2. To validate the definitional computation rule of identity types, we seem to require $\text{transport}_{\sim C} (\langle \_ \rangle b) c$ to be the identity function on $C b$, and $\text{uniqueness}_{\sim a} (\langle a, \langle \_ \rangle a \rangle)$ to be a degenerate path at $\langle a, \langle \_ \rangle a \rangle$. In Cartesian cubical type theory, the former holds if coercion in degenerate types ($x : I \vdash C : \text{Type}_i$ such that $C = C(0/x)$ : Type) is constant, and the latter holds if any open box whose specified $\xi_i$ sides are degenerate has a degenerate filler:

(In De Morgan cubical type theory, $\text{uniqueness}_{\sim a} (\langle a, \langle \_ \rangle a \rangle)$ is reflexive when defined with connections, but $\text{transport}_{\sim C} (\langle \_ \rangle b) c$ is not the identity.)

These conditions on coercion and composition, jointly known as regularity, were discovered by several researchers, including Coquand and Awodey [Awo18]. Early investigations into cubical type theory focused on defining regular uniform Kan operations closed under the type formers of Book HoTT. In April 2015, Cohen, Coquand, Huber, and Mörtberg circulated an early version of De Morgan cubical type theory apparently satisfying regularity. In May 2015, Licata, Harper, Morehouse, and the author discovered an error in their treatment of universes, which Cohen et al. corrected in July 2015 at the cost of abandoning regularity.

**Regularity for universes** All open boxes in a universe must have fillers. Because those fillers are elements of a universe, they must themselves be types. Consider the bottom

---

face of a filler whose other faces are types $A$, $B$, and $C$ (below, right). Coercion in the filler’s $y$-coordinate requires this type to be equivalent to $A$. However, this type is not equal to $A$, because its $\langle 0/x \rangle$ and $\langle 1/x \rangle$ faces are $C(1/y)$ and $B(1/y)$. Instead, we regard it as a new type whose elements are triples of $a : A$, $b : B(1/y)$, and $c : C(1/y)$ such that $a(0/x) = \text{coe}_{y,y}^{1\mapsto 0}(c) : A(0/x)$ and $a(1/x) = \text{coe}_{y,y}^{1\mapsto 0}(b) : A(1/x)$:

\[
\begin{array}{cc}
\text{y} & \text{x} \\
\downarrow & \text{c} \\
\text{y} & \text{b} \\
\end{array}
\]

\[
\begin{array}{c}
\text{y} \\
\downarrow \\
\text{y} \\
\end{array}
\]

This type must moreover admit a Kan operation, which failed to be regular in the April 2015 draft of De Morgan cubical type theory. It remains unknown how to equip this type with a regular Kan operation, and a semantic analysis by Swan [Swa18b] proves regularity is in fact impossible to achieve in certain classes of models. A representative case of the difficulty with regularity follows.

Suppose $A$, $B$, and $C$ vary also in $z : I$; for simplicity, let $C$ be degenerate in $y : I$. Coercion $0 \mapsto 1$ in the composite type’s $z$-coordinate takes as input $x : I \vdash a : A(0/z)$ and $b : B(0/z)(1/y)$ such that $a(1/x) = \text{coe}_{y,y}^{1\mapsto 0}(b) : B(0/z)(0/y)$, and must produce $x : I \vdash a' : A(1/z)$ and $b' : B(1/z)(1/y)$ such that $a'(1/x) = \text{coe}_{y,y}^{1\mapsto 0}(b') : B(1/z)(0/y)$:

\[
\begin{array}{c}
\text{z} \\
\downarrow \\
\text{y} \\
\end{array}
\]

By definition, $b'$ is the $\langle 1/x \rangle$ face of our coercion. On the other hand, the $\langle 1/x \rangle$ face of the type in which we coerce equals $B(1/y)$, so the $\langle 1/x \rangle$ face of our coercion must equal $\text{coe}_{z,y}^{0\mapsto 1}(b)$ by commuting coercion and substitution. We therefore define $b' := \text{coe}_{z,y}^{0\mapsto 1}(b)$.
It is tempting to define \( x : \bar{1} \vdash a' : A(1/z) \) as \( \text{coe}^{0 \dashv 1}_{z} (a) \), but this definition would not satisfy \( a' (1/x) = \text{coe}^{1 \dashv 0}_{y,z,A(1/x)} (b') : B(1/z)(0/y) \) as required:

\[
\begin{align*}
\text{coe}^{0 \dashv 1}_{z} (\text{coe}^{1 \dashv 0}_{y,z,A(1/x)} (b)) & = \text{coe}^{0 \dashv 1}_{z} (\text{coe}^{1 \dashv 0}_{y,B(0/z)} (b)) \\
& \neq \text{coe}^{1 \dashv 0}_{y,B(0/z)} (b')
\end{align*}
\]

That is, coercing \( b \) up then forward is not the same as coercing it forward then up. There is, however, a path between those coercions, which we obtain by coercing \( b \) up to \( B(0/z) \), then coercing forward along all of \( B \):

\[
\text{coe}^{0 \dashv 1}_{z} (\text{coe}^{1 \dashv 0}_{y,z,B(0/z)} (b)) \xrightarrow{\text{coe}^{1 \dashv 0}_{y,B(0/z)} (\text{coe}^{0 \dashv 1}_{y,z,B(1/y)} (b))} \text{coe}^{0 \dashv 1}_{z} (\text{coe}^{1 \dashv 0}_{y,z,B(1/y)} (b))
\]

Define \( a' \) as the composite of \( \text{coe}^{0 \dashv 1}_{A} (a) \) and the above adjustment path.

We have just defined coercion in the composite of \( A \) and \( B \), but our definition does not satisfy regularity. Suppose the composite type (and hence also \( A \) and \( B \)) is degenerate in \( z : \bar{1} \); then regularity requires our coercion to be the identity function (that is, \( a' = a \) and \( b' = b \)). Assuming regularity holds for coercion in \( A \) and \( B \), we have \( b' = b \) and \( \text{coe}^{0 \dashv 1}_{A} (a) = a \), but the adjustment path above requires coercion in \( y : \bar{1} \) and thus only simplifies to \( \text{coe}^{0 \dashv 0}_{y,B} (\text{coe}^{1 \dashv 0}_{y,B} (b)) \). Therefore \( a' \neq a \), and regularity fails for our definition.

**Regularity for other types** On the other hand, regular uniform Kan operations are straightforwardly closed under dependent function, dependent pair, and path types, and in fact, regularity simplifies the definition of composition in \( \text{Path}_{x,A}(P_{0}, P_{1}) \) that we presented in Section 3.2. Unlike in ordinary cubical type theory, we can define:

\[
\text{hcom}_{\text{Path}_{x,A}(P_{0}, P_{1})}^{r \dashv r'} (M; \xi_{i} \mapsto y.N_{i}) := \langle x \rangle \text{hcom}_{A}^{r \dashv r'} (M@x; \xi_{i} \mapsto y.N_{i}@x)
\]

The \( \langle e/x \rangle \) faces of the definiens are \( \text{hcom}_{A(\varepsilon/x)}^{r \dashv r'} (P_{\varepsilon}; \xi_{i} \mapsto P_{\varepsilon}) \), which by regularity equal \( P_{\varepsilon} \) (because \( P_{0}, P_{1} \) are constant in \( y : \bar{1} \)).

To obtain a cubical type theory closed under dependent function, dependent pair, and path types that strictly model identity types, it suffices to equip each type with regular coercion \( r \dashv r' \) and regular composition \( r \dashv r' \) of shape \( (x = 0), (x = 1) \) only.\(^{12}\) (We still need compositions of this shape to define \( J_{mod} \) and coercion in path types, but composition in path types no longer alters composition shapes.) Awodey [Awo18] models identity types in Cartesian cubical sets by a similar observation.

\(^{12}\)Sterling, Angiuli, and Gratzer [SAG19] use exactly these Kan operations in a cubical type theory with non-univalent universes and path types satisfying uniqueness of identity proofs.
3.5 **Validity and other considerations**

Regularity is one of many subtleties in defining a uniform Kan operation. Another is: do we permit 0-dimensional filling problems? Cohen et al. [CCHM18] answer in the affirmative, allowing compositions of empty shape \( \cdot \vdash \text{comp}_{y, A}(M; \cdot) : A(1/y) \) for \( \cdot \vdash M : A(0/y) \). Such compositions simplify only according to type-specific principles analogous to those in Section 3.2, because the sole type-generic equation governing \( \text{comp} \) (Figure 3.6) does not apply to empty shapes. (If \( \text{comp} \) were regular, then \( \text{comp}_{\cdot, A}(M; \cdot) = M \) would hold generically.) For a higher inductive type \( T \), \( \text{comp}_{\cdot, T}(M; \cdot) \) never simplifies, because \( T \)'s elimination principle stipulates that \( T \) is the least uniform Kan type with given constructors.

Perhaps surprisingly, empty compositions in higher inductive types pose major difficulties in cubicaltt, where closed terms of higher inductive type often compute to values with hundreds of nested empty compositions \( \text{comp}_{\cdot, T}(\cdots \cdot; \cdot) \). Such empty compositions partially negate the benefits of computation. Imagine, for instance, computing a closed element of a set quotient \( A/R \) [UF13, Section 6.10]. In cubicaltt, such an element’s value may contain arbitrarily many empty compositions; if we could disallow empty compositions, its value would always be the injection \( \text{in}(M) \) of an element of \( A \).

We contribute a method for disallowing empty compositions, allowing us to obtain a strong canonicity theorem for 0-cubes of higher inductive type (Theorem 4.77); recently, Vezzosi, Mörtberg, and Abel [VMA19] have developed such a method in the De Morgan setting in cubical Agda. To understand our approach, recall that uniform Kan operations must commute with interval substitutions. One must therefore prohibit not only empty compositions but also all compositions with empty substitution instances. For example, while \( \text{hcom}^0_{A}(M; x = 1 \hookrightarrow y.A) \) is not empty, its \( 0/x \) face has shape \( 0 = 1 \), which is either empty or equally problematic, because it is unsatisfiable.

We thus restrict \( \text{hcom} \) to shapes \( \xi_1, \ldots, \xi_n \) all of whose closed instantiations contain at least one true clause—that is, for which \( (\forall x_1, \ldots, x_m \in \{0, 1\}. \xi_1 \lor \cdots \lor \xi_n) \) holds. Geometrically, such open boxes cover all 0-cells of their fillers (below, left but not right):

\[
\begin{align*}
\text{hcom}^0_{A}(M; x = 1 \hookrightarrow y.A, x = 0 \hookrightarrow y.A') & \quad \text{hcom}^0_{A}(M; x = 1 \hookrightarrow y.A)
\end{align*}
\]

In Chapter 4 we underapproximate the above condition, requiring simply that *valid* composition shapes contain either a true clause or a pair of clauses \( (x = 0), (x = 1) \) for some \( x \) (Definition 4.28).
Following the observations of Section 3.2, valid homogeneous composition \( r \rightsimeq r' \) and coercion \( r \rightsimeq r' \) suffice to define dependent function, dependent pair, and path types. Univalent universes are more challenging, as the aforementioned \( \forall x \) operation [CCHM18] produces invalid shapes from valid ones, for example, when deleting all \( x \) clauses from the shape \((x = 0), (x = 1), (z = 1)\). In such cases we use a derived operation called generalized homogeneous composition, which fills boxes of possibly-invalid shape by iterated valid fillers (Theorem 4.34). Our generalized compositions compute not according to their type but rather the length of \( \xi_1, \ldots, \xi_n \); moreover, while generalized compositions satisfy the rules of Figure 3.4, they do not equal the corresponding hcoms if applied to valid shapes.

**Equality of compositions** Our discussion of validity raises another question: when are two compositions equal? Many syntactically distinct filling problems have equal extensions as open boxes, including:

\[
\begin{align*}
\text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N, 0 = 1 \mapsto y.N') &= \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N) \\
\text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N, \xi' \mapsto y.N') &= \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi' \mapsto y.N', \xi \mapsto y.N) \\
\text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N, \xi \mapsto y.N) &= \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N) \\
\text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; s = s' \mapsto y.N) &= \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; s' = s \mapsto y.N)
\end{align*}
\]

The first three are definitional equalities of De Morgan cubical type theory, where they are used in the definition of identity types [CCHM18, Section 9.1]. (The fourth is moot because Cohen et al. [CCHM18] consider only easily-oriented equations \((x = 0), (x = 1)\).) We adopt none of the above equations in this dissertation; Cavallo and Harper [CH19a] provide an alternative definition of identity types not reliant on these equations. Aside from identity types, such equations seem expensive in practice, and complicate both implementations and our definition of generalized composition.

Readers often ask whether coercions satisfy the equation:

\[
\text{coe}^\rightsimeq_{r \rightsimeq r'}(\text{coe}^\rightsimeq_{r \rightsimeq r'}(M)) = \text{coe}^\rightsimeq_{r \rightsimeq r'}(M)
\]

While plausible, this equation implies that every path in \( \text{Type}_i \) induces a strict isomorphism between its endpoints (by \( \lambda a.\text{coe}^{0 \mapsto 1}_{x,p@x}(a) \) and \( \lambda a.\text{coe}^{1 \mapsto 0}_{x,p@x}(a) \)) rather than an equivalence, ruling out many instances of univalence. These coercions are nevertheless related by \( \text{coe}^\rightsimeq_{x.A} (\text{coe}^\rightsimeq_{x.A}(M)) \), which equals the left one under \( \langle r' / x \rangle \) and the right under \( \langle r / x \rangle \).

**Compositions of types** As suggested in Section 3.4, we regard homogeneous composition of types as a type former in its own right, and must therefore specify its introduction, elimination, computation, and uniqueness principles. For example, an element of

\[\text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N, 0 = 1 \mapsto y.N') \quad \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N) \quad \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N, \xi' \mapsto y.N') \quad \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi' \mapsto y.N', \xi \mapsto y.N) \quad \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; \xi \mapsto y.N, \xi \mapsto y.N) \quad \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; s = s' \mapsto y.N) \quad \text{hcom}^\rightsimeq_{r \rightsimeq r'}(M; s' = s \mapsto y.N)
\]

Our construction of univalent universes is only simplified by omitting validity (and thus, generalized composition); see Appendix B or, for an alternate construction, Angiuli et al. [Ang+19].
hcom\text{_{\text{Type}}}_{1}^{0 \to 1}(A; x = 0 \leftrightarrow y.C, x = 1 \leftrightarrow y.B)$, written box\text{_{\text{Type}}}_{1}^{0 \to 1}(a; x = 0 \leftrightarrow c, x = 1 \leftrightarrow b), consists of an element $a : A$ and two partial elements $(x = 0) \vdash c : C(1/y)$ and $(x = 1) \vdash b : B(1/y)$ satisfying $(x = 0) \vdash a = \text{coe}_{y.C}^{1 \to 0}(c) : A$ and $(x = 1) \vdash a = \text{coe}_{y.B}^{1 \to 0}(b) : A$.

\[
\begin{array}{c}
\text{y} \\
\downarrow \quad \cdot \\
\text{x} \\
\Downarrow \quad \cdot \\
\text{c} \quad \cdots \quad \text{b} \\
\end{array}
\quad
\begin{array}{c}
\text{coe}_{y.C}^{1 \to 0}(c) \\
\downarrow \quad \cdot \\
\text{a} \quad \cdots \quad \cdot \\
\Downarrow \quad \cdot \\
\text{coe}_{y.B}^{1 \to 0}(b) \\
\end{array}
\quad
\begin{array}{c}
\text{A} \\
\downarrow \quad \cdot \\
\text{C} \\
\Downarrow \quad \cdot \\
\text{B} \\
\end{array}
\]

This introduction principle is far from arbitrary. If $x = 1$, the composite type must equal $B(1/y)$; accordingly, if $x = 1$, the above data specifies precisely an element of $B(1/y)$—the partial element $c$ is vacuous because $x = 0$ is impossible\textsuperscript{14} and $a$ must be exactly $\text{coe}_{y.B}^{1 \to 0}(b)$ because $x = 1$ is tautologous. Coercion in $y.\text{hcom}_{\text{Type}}_{1}^{0 \to 1}(A; x = 0 \leftrightarrow y.C, x = 1 \leftrightarrow y.B)$ implies the composite type is equivalent to $A$, which is why boxes include $a : A$ (as well as its equations with $b$ and $c$, lest the type be equivalent instead to $A \times C(1/y) \times B(1/y)$). Conversely, if composite types do not admit coercion, they need not be equivalent to $A$; in Section 4.4.10 we define compositions of non-Kan types to be empty.

Finally, to compose and coerce in composite types, we must access the components of boxes. We easily obtain $b$ and $c$ as the $\langle 1/x \rangle$ and $\langle 0/x \rangle$ faces of the box above, while we compute $a$ using cap, the elimination form of composite types. In general, eliminating an element of $\text{hcom}_{\text{Type}}_{1}^{0 \to 1}(A; x = 0 \leftrightarrow y.C, x = 1 \leftrightarrow y.B)$ produces an element of $A$. The cap of a box is simply its $a : A$ component; if $x = 1$, the input to cap is not a box but an element of $B(1/y)$, and cap instead applies $\text{coe}_{y.B}^{1 \to 0}(-)$. (These computations agree by $(x = 1) \vdash a = \text{coe}_{y.B}^{1 \to 0}(b) : A$.) See Section 4.4.11 for details.

**V-types** If $\text{Type}_{1}$ is univalent, we have a function:

$$\text{ua}_{\square} : (A:B:\text{Type}_{1}) \to ((f:A \to B) \times \text{IsEquiv}(f)) \to \text{Path}_{\square.\text{Type}_{1}}(A, B)$$

Applying elimination rules, $(\text{ua}_{\square} A B \langle f, e \rangle)@x$ must therefore be a type with extent in $x : \mathbb{I}$. In this dissertation, we regard this type as a new type former written $V_x(A, B, \langle f, e \rangle)$, equal to $A$ when $x = 0$ and $B$ when $x = 1$.

What is the formation principle of $V_x(A, B, \langle f, e \rangle)$? In particular, what do occurrences of $x$ mean in $A, B, f, e$? (No such occurrences exist in our motivating example above.) We cannot insist that $x$ does not occur, because formation would not be closed under diagonals—typehood of $V_x(A, B, \langle f, e \rangle)$ would not imply typehood of $(V_x(A, B, \langle f, e \rangle))(y/x)$ when $y$

\textsuperscript{14}We discuss judgments under interval equations in Section 4.3.2.
occurs in $A, B, f, e$.\footnote{This definition, known as the G-type, is possible in the symmetric monoidal setting [BCH18].} Nor can we ask that $A$ and $B$ be disjoint partial types $(x = 0) \vdash A : \text{Type}_i$ and $(x = 1) \vdash B : \text{Type}_i$, because we would not be able to form $A \to B$.

Instead, we require types $(x = 0) \vdash A : \text{Type}_i$ and $B : \text{Type}_i$ and an equivalence $(x = 0) \vdash \langle f, e \rangle : (f : A \to B) \times \text{IsEquiv}(f)$. That is, given a type $B$ and a partial type $A$ equivalent to $B(0/x)$ under $x = 0$, we obtain a line $V_x(A, B, \langle f, e \rangle)$ between $A$ and $B(1/x)$, effectively composing the equivalence $f$ and the type $B$:

\[
\begin{array}{ccc}
\xrightarrow{x} & f & a \\
\downarrow & B & \downarrow f \\
a & B & A
\end{array}
\]

By analogy with composite types, an element of $V_x(A, B, \langle f, e \rangle)$, written $\text{Vin}_x(a, b)$, consists of $b : B$ and a partial element $(x = 0) \vdash a : A$ satisfying $(x = 0) \vdash f a = b : B$; $\text{Vin}_x(a, b)$ equals $a$ when $x = 0$ and $b(1/x)$ when $x = 1$. The elimination form for $V_x(A, B, \langle f, e \rangle)$, written $\text{Vproj}_x(-, f)$, produces an element of $B$. Applied to $\text{Vin}_x(a, b)$, $\text{Vproj}$ returns $b$; if $x = 0$, the argument of $\text{Vproj}$ is an element of $A$ instead, and $\text{Vproj}$ applies $f$. See Section 4.4.9 for details.

$V$-types and composite types arise as special cases of the Glue types of Cohen et al. [CCHM18], in which a single type is composed with any configuration of partial types equivalent to it. (One obtains composite types by using coercion to transform each $y.B_i$ into an equivalence.) Angiuli et al. [Ang+19] develop Glue types in the Cartesian setting; we do not discuss them further in this dissertation.

**Extension types** We consider several variations on path types in RedPRL and redtt. Elements of line types are dependent functions from $\mathbb{I}$ with arbitrary endpoints:

\[
\begin{align*}
\Gamma & , x : \mathbb{I} \vdash A : \text{Type}_i & \Gamma & , x : \mathbb{I} \vdash M : A & \Gamma & \vdash M : (x:\mathbb{I}) \to A \\
\Gamma & \vdash (x:\mathbb{I}) \to A : \text{Type}_i & \Gamma & \vdash (x:\mathbb{I}) \to A : \text{Type}_i & \Gamma & \vdash M \circ r : A(r/x)
\end{align*}
\]

Elements of a restriction type $A[\xi_i \leftarrow N_i]$ are elements of $A$ that equal $N_i$ when $\xi_i$:

\[
\begin{align*}
\Gamma & \vdash A : \text{Type}_k & \Gamma & \vdash M : A & \Gamma & \vdash M : A[\xi_i \leftarrow N_i] \\
(\forall i) & \Gamma, \xi_i \vdash N_i : A & (\forall i) & \Gamma, \xi_i \vdash M = N_i : A & \Gamma & \vdash \text{out}(M) : A[\xi_i \leftarrow N_i] \\
(\forall i, j) & \Gamma, \xi_i, \xi_j \vdash N_i = N_j : A & (\forall i) & \Gamma, \xi_i \vdash M = N_i : A & \Gamma & \vdash \text{in}(M) : A[\xi_i \leftarrow N_i]
\end{align*}
\]
Agda currently supports line and restriction types (the latter via “partial cubical types”) but
similarly, one can consider Path_{x:A}(M, N) a subtype of (x:ι) → A.)

We can recover path types as a combination of line and restriction types:

\[
\text{Path}_{x:A}(a, a') := (x:ι) \to A \ [x = 0 \leftrightarrow a, x = 1 \leftrightarrow a']
\]

However, the flexibility of line and restriction types makes them more ergonomic than
path types. Consider binary path composition, which takes three points \(a, b, c : A\) and
two paths \(\text{Path}_{x:A}(a, b)\) and \(\text{Path}_{x:A}(b, c)\). Using line types, we can eliminate the
first three arguments \(a, b, c\), instead stating that composition takes a line \(p : ι \to A\) and a line
\(q : (x:ι) \to A \ [x = 0 \leftrightarrow p@1]\) whose left endpoint equals the right endpoint of \(p\):

\[
\text{composition} : (a \ b \ c:A) \to \text{Path}_{x:A}(a, b) \to \text{Path}_{x:A}(b, c) \to \text{Path}_{x:A}(a, c)
\]

\[
\text{composition'} : (p:ι \to A) \to (q;(x:ι) \to A \ [x = 0 \leftrightarrow p@1]) \to \text{Path}_{x:A}(p@0, q@1)
\]

Another example is the type of squares bounded by four paths \(p : \text{Path}_{x:A}(a, b), q : \text{Path}_{x:A}(a, c), r : \text{Path}_{x:A}(b, d),\) and \(s : \text{Path}_{x:A}(c, d)\), typically
defined as the iterated path type \(\text{Path}_{x:A}(p@x, s@x)(q, r).\)

\[
(x \ y:ι) \to A \ [x = 0 \leftrightarrow q@y, x = 1 \leftrightarrow r@y, y = 0 \leftrightarrow p@x, y = 1 \leftrightarrow s@x]
\]

If \(x : ι \vdash A : \text{Type}\) is Kan then \((x:ι) \to A\) is Kan. The same is not true for restriction
types, as that would imply that all paths exist:

\[
\langle x \rangle_{\text{out}(\text{coe}_{x:A}^{0\to x}[x=0\to \text{true}, x=1\to \text{false}])}(\text{in}(\text{true})) : \text{Path}_{A\cdot \text{bool}}(\text{true}, \text{false})
\]

Fortunately, \(\langle x:ι \vdash A[\xi_j \leftrightarrow N_j]\) is Kan whenever \(\xi_j\) mentions only interval variables
in \(\vec{x}_i\)—that is, whenever line types bind all the variables involved in a restriction type—a
condition satisfied by path types, composition', square types, and many other examples.

For that reason, we consider \((x:ι) \to A[\xi_j \leftrightarrow N_j]\) to be a primitive type former known
as 
extension types, following Riehl and Shulman [RS17], who first introduced them. The
Kan operations for extension types are very similar to those of path types (Section 3.2),
adjusting homogeneous compositions by \(\xi_j \leftrightarrow \_\cdot N_j\) and coercions by \(\xi_j \leftrightarrow y.\text{coe}_{y:A'}^{y\to y'}(N_j)\).

The reddtt proof assistant currently supports extension types, and uses them to reify
boundary constraints on subgoals. For example, if a user writes the partial term:

\[
h\text{com}_{A}^{2\to 1}(M; x = 0 \leftrightarrow y.\_\cdot x = 1 \leftrightarrow ?)
\]

reddtt displays \((y:ι) \to A \ [y = 0 \leftrightarrow M, 1 = 0 \leftrightarrow \_\cdot\) for the type of the hole \(?\). Cubical
Agda currently supports line and restriction types (the latter via “partial cubical types”) but
not Kan extension types, nor does it formulate subgoals using extension types [VMA19,
Section 3.3]. We anticipate these features will be implemented in the future.
Now, it is the contention of the intuitionists (or constructivists, I shall use these terms synonymously) that the basic mathematical notions, above all the notion of function, ought to be interpreted in such a way that the cleavage between mathematics (classical mathematics, that is) and programming that we are witnessing at present disappears.

—Per Martin-Löf, *Constructive mathematics and computer programming* [ML82]

This chapter describes the computational semantics of Cartesian cubical type theory, which extends ordinary constructive type theory with path, circle, and univalent universe types. Like De Morgan cubical type theory [CCHM18], Cartesian cubical type theory uses interval variables $x : \mathbb{I}$ and uniform Kan operations to mediate the higher-dimensional structure induced by univalence and higher inductive types.

Our computational semantics directly justify *Cartesian cubical computational type theory* (Appendix A), a Nuprl-style type theory implemented in the RedPRL proof assistant [Red16]. Cartesian cubical computational type theory is a two-level type theory, like Voevodsky’s Homotopy Type System [Voe13], containing both Kan types equipped with homogeneous composition and coercion operations (as described in Section 3.2), and pretypes possibly lacking such operations. As in Chapter 2, we establish consistency (Theorem 4.58) and canonicity (Theorems 4.63 and 4.77) for our computational type theory.

Cartesian cubical computational type theory contains two cumulative hierarchies of universes: $\mathcal{U}_i^{\text{Kan}}$ whose elements are Kan types, and $\mathcal{U}_i^{\text{pre}} \supset \mathcal{U}_i^{\text{Kan}}$ whose elements are pretypes. Both $\mathcal{U}_i^{\text{Kan}}$ and $\mathcal{U}_i^{\text{pre}}$ are Kan types, and each $\mathcal{U}_i^{\text{Kan}}$ is univalent. Our pretypes include strict equality $\text{Eq}_A(M, N)$ with equality reflection; strict equality is not generally Kan, as that would imply that all paths are trivial:

$$\lambda p. \text{transport}_{\mathcal{S}} (\lambda b. \text{Eq}_A(a, b)) p \star : \text{Path}_{\mathcal{A}}(a, a') \to \text{Eq}_A(a, a')$$

This chapter derives heavily from the author’s *Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities* [AFH18] and its associated preprint [AFH17], but incorporates additional exposition and various technical improvements, notably: simpler definitions of coercion in V-types and homogeneous composition in $\text{hcom}_{\mathcal{U}_i^{\text{Kan}}}$ types suggested by Anders Mörtberg (Figure 4.2); a construction of $\text{hcom}_{\mathcal{U}_i^{\text{pre}}}$
types suggested by Andrew Pitts enabling $U^\text{pre}_i$ to be Kan (Section 4.4.10); and new notation for candidate judgments after Chapter 2 (Definition 4.5).

Cavallo and Harper [CH19a] extend our computational semantics with a schema for indexed higher inductive types accounting for many types considered in Book HoTT, including spheres, homotopy pushouts, truncations, and the identity type (with strict computation rule). In a separate line of work, Cavallo and Harper [CH19b] extend our computational semantics with bridge variables expressing internal parametricity primitives in the style of Bernardy, Coquand, and Moulin [BCM15], which refute the law of excluded middle for homotopy subsingleton types [UF13, Section 3.4]. We do not present these extensions in this dissertation.

**Denotational and intensional formulations** Although we analyze Cartesian cubical type theory from the computational perspective, our uniform Kan operation can be formulated for arbitrary Cartesian cubical sets. Angiuli et al. [Ang+19] show that Cartesian cubical sets with such an operation are closed under dependent functions, dependent pairs, univalent universes, et cetera, accompanied by an Agda formalization in the axiomatic style developed by Orton and Pitts [OP16] and Licata et al. [Lic+18].

Moreover, just as the computational semantics of Idealized Nuprl also model intensional type theory (in the sense of Section 2.4), we can define a Cartesian cubical intensional type theory and regard this chapter as a proof of its consistency. Angiuli et al. [Ang+19] propose one such type theory; we propose another in Appendix B closer to what is implemented in the redtt [Red18] proof assistant and described by our computational semantics (namely, with $V$ and $\text{hcom}_{U^\text{Kan}}$ type formers instead of Glue types [Ang+19, Section 2.11]).

Neither Cartesian cubical intensional type theory contains strict equality types—while equality reflection is semantically justified and natural in computational type theory, it is undesirable for definitional equality as reflection disrupts its decidability. Altenkirch, Capriotti, and Kraus [ACK16] and Boulier and Tabareau [BT17] have proposed two-level intensional type theories (without canonicity) whose strict equality types are identity types augmented by axioms for function extensionality and uniqueness of identity proofs. More recently, Sterling, Angiuli, and Gratzer [SAG19] proposed XTT, a cubical intensional type theory with canonicity, whose path type satisfies function extensionality and uniqueness of identity proofs. We hope to integrate XTT into redtt to obtain strict equality without reflection in an intensional, two-level cubical type theory with canonicity.

**Outline** Before diving into technical details, we briefly outline the difficulties that arise in this chapter. Cartesian cubical type theory is based on a cubical programming language whose programs contain interval variables that can be instantiated with 0 and 1. As in

---

1 Canonicity does not follow immediately, as intensional type theories contain only typed equations, but we expect that this chapter and Huber’s canonicity proof [Hub18] contain all the necessary ingredients.
Chapter 2, our operational semantics range only over programs with no free term variables, and open judgments express the truth of their conclusion for all closed elements of their hypotheses. However, interval variables cannot stand only for their endpoints 0 and 1, because such an interpretation would equate all lines with equal boundaries and thus imply uniqueness of identity proofs.

We therefore define our cubical operational semantics for programs with free interval variables (but no free term variables), and define types by their PERs of closed value elements at every dimension, that is, for every set of interval variables. We ensure that the PERs of elements of each closed type determine a Cartesian cubical set whose restriction maps are given by interval substitution followed by evaluation. (Evaluation is needed because the endpoints of values are not necessarily values.)

We equip every Kan type with coercion and homogeneous Kan composition operations sufficient for defining transport and singleton contractibility (Section 3.2). To define homogeneous Kan composition (Definition 4.29), we must specify when a collection of $n$-cubes forms an open box of a given shape. We implement the Kan operations of each type former using operational semantics rules which rely on the Kan operations of constituent types. As outlined in Section 3.4, homogeneous compositions of Kan types are themselves types whose elements are formal boxes of elements of the constituent types. To ensure our universes are univalent, given any types $A$, $B$ and equivalences $E$ between them, we define a type $V_x(A, B, E)$ whose endpoints are $A$ and $B$. These “V-types” are a special case of the “Glue types” of Cohen et al. [CCHM18], and closely related to the “G-types” of Bezem, Coquand, and Huber [BCH18].

Finally, as in Section 2.5, we prove the formation, introduction, elimination, and uniqueness rules for each type former. In the cubical setting, we must also verify that each type is Kan, that is, that its operational semantics rules properly implement its Kan operations. In Section 2.5, we proved most rules using head expansion (Lemma 2.15), by which it suffices to consider a term’s evaluation behavior; in Section 4.4, we must often rely instead on coherent expansion (Lemma 4.18), which requires us to consider the evaluation behavior of all interval substitution instances of a term.

4.1 Syntax and operational semantics

Cartesian cubical type theory is based on a untyped functional programming language with two sorts: interval terms and ordinary terms. The only interval terms are 0, 1, and interval variables $x$, $y$, . . .; ordinary terms include standard type and term constructors previously seen in Idealized Nuprl, as well as new constructs (for Kan operations, higher inductive types, et cetera) with interval term arguments.

We define the syntax of ordinary terms in Figure 4.1. As before, capital letters $M, N, A, . . .$ represent ordinary terms and $a, b, . . .$ represent ordinary term variables and
**Cartesian cubical type theory**

M := \( \Pi(A, a.B) \) (dependent function type) \\
\( \lambda(a.M) \) (lambda abstraction) \\
app(M, N) (function application) \\
\( \Sigma(A, a.B) \) (dependent pair type) \\
pair(M, N) (pairing) \\
fst(M) (first projection) \\
snd(M) (second projection) \\
Eq(A, M, N) (equality pretype) \\
refl (equality proof) \\
nat (natural number type) \\
z (zero) \\
s(M) (successor) \\
natrec(M, N_1, n.a.N_2) (natural number recursion) \\
bool (boolean type) \\
true (true) \\
false (false) \\
if(M, N_1, N_2) (boolean recursion) \\
void (empty type)

<table>
<thead>
<tr>
<th>Path(x.A, M, N)</th>
<th>Path_{x.A}(M, N)</th>
<th>path type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ilam(x.M)</td>
<td>( \langle x \rangle M )</td>
<td>interval abstraction</td>
</tr>
<tr>
<td>iapp(M, r)</td>
<td>M@r</td>
<td>interval application</td>
</tr>
<tr>
<td>V(r, A, B, E)</td>
<td>V_r(A, B, E)</td>
<td>weak univalence axiom</td>
</tr>
<tr>
<td>Vin(r, M, N)</td>
<td>Vin_r(M, N)</td>
<td>V-type injection</td>
</tr>
<tr>
<td>Vproj(r, M, F)</td>
<td>Vproj_r(M, F)</td>
<td>V-type projection</td>
</tr>
<tr>
<td>circle</td>
<td>( \mathbb{S}^1 )</td>
<td>circle type</td>
</tr>
<tr>
<td>base</td>
<td>base</td>
<td>base point</td>
</tr>
<tr>
<td>loop(r)</td>
<td>loop_r</td>
<td>loop</td>
</tr>
<tr>
<td>celim(c.A, M, N_1, x.N_2)</td>
<td>( \mathbb{S}^1 )elim_{c.A}(M; N_1, x.N_2)</td>
<td>circle induction</td>
</tr>
<tr>
<td>Pre[i]</td>
<td>( U_i^{pre} )</td>
<td>ith pretype universe</td>
</tr>
<tr>
<td>Kan[i]</td>
<td>( U_i^{kan} )</td>
<td>ith Kan type universe</td>
</tr>
</tbody>
</table>

Figure 4.1: Syntax of Cartesian cubical type theory: basic constructs.

binders. We additionally write \( r, s, \ldots \) for interval terms, \( x, y, \ldots \) for interval variables and binders, \( \epsilon \) for interval constants 0 and 1, and FI(M) for the set of interval variables free in M. One can substitute ordinary terms into ordinary terms \( M[N/a] \), and interval terms into either sort: \( M\langle r/x \rangle \) and \( s\langle r/x \rangle \).
\[
\begin{align*}
\text{coe}(x.A, r, r', M) & \quad \text{coe}_{x.A}^{r \rightarrow r'}(M) & \quad \text{coercion} \\
\text{hcom}[n](A, r, r', M, r_i, r'_i, y.N_i) & \quad \text{hcom}_{y.A}^{r \rightarrow r'}(M; r_i = r'_i \iff y.N_i) & \quad \text{homog. comp.} \\
\text{com}[n](y.A, r, r', M, r_i, r'_i, y.N_i) & \quad \text{com}_{y.A}^{r \rightarrow r'}(M; r_i = r'_i \iff y.N_i) & \quad \text{heterog. comp.} \\
\text{ghcom}[n](A, r, r', M, r_i, r'_i, y.N_i) & \quad \text{ghcom}_{y.A}^{r \rightarrow r'}(M; r_i = r'_i \iff y.N_i) & \quad \text{generalized hcom} \\
\text{gcom}[n](y.A, r, r', M, r_i, r'_i, y.N_i) & \quad \text{gcom}_{y.A}^{r \rightarrow r'}(M; r_i = r'_i \iff y.N_i) & \quad \text{generalized com} \\
\text{box}[n](r, r', M, r_i, r'_i, y.N_i) & \quad \text{box}_{y.A}^{r \rightarrow r'}(M; r_i = r'_i \iff y.N_i) & \quad \text{box formation} \\
\text{cap}[n](r, r', M, r_i, r'_i, y.B_i) & \quad \text{cap}_{y.A}^{r \rightarrow r'}(M; r_i = r'_i \iff y.B_i) & \quad \text{cap projection}
\end{align*}
\]

Figure 4.1: Syntax of Cartesian cubical type theory: Kan operations.

The Kan composition operators are indexed by external natural numbers. For instance, \( \text{hcom}[n] \) represents homogeneous composition along \( n \) equations, and has \( 3n + 4 \) arguments, the last \( 3n \) of which are grouped in triples \( r_i, r'_i, y.N_i \) specifying \( N_i \) as the \( r_i = r'_i \) component of an open box. We write \( \xi_i \) for equations \( r_i = r'_i \) (formally, pairs of interval terms), and pun \( \xi_i \) and \( \neg \xi_i \) for the propositions that such an equation holds or fails to hold.

In Figure 4.2, we define a structural operational semantics [Plo81] over terms with free interval variables but no free term variables. The judgment \( M_0 \) val specifies when \( M_0 \) is a value, and \( M \rightsquigarrow M' \) specifies when \( M \) takes one step of evaluation to \( M' \). We write \( M \rightsquigarrow^* M' \) when \( M \) steps to \( M' \) in zero or more steps, and \( \downarrow M_0 \) when \( M \rightsquigarrow^* M_0 \) and \( M_0 \) val. Curiously, coercion is the only place where evaluation descends under interval binders, as \( \text{coe} \) must evaluate its type argument \( x.A \) and dispatch on \( A \)'s head constructor.

These judgments satisfy three key properties easily proven by induction on their definitions. First, if \( M \) val, then \( M \rightsquigarrow \). (As before, the converse is not the case.) Secondly, evaluation is deterministic: if \( M \rightsquigarrow M' \) and \( M \rightsquigarrow M'' \), then \( M' = M'' \). Finally, evaluation never introduces additional free interval variables: if \( M \rightsquigarrow M' \), then \( \text{Fl}(M') \subseteq \text{Fl}(M) \).

Finite sets of interval variables and total interval substitutions present the Cartesian cube category described in Definition 3.2.

**Definition 4.1.** Given finite sets \( \Psi, \Psi' \) of interval variables, a total interval substitution \( \psi : \Psi' \rightarrow \Psi \) assigns to each element of \( \Psi \) either 0, 1, or an element of \( \Psi' \).

Our computational semantics associate to each closed type a Cartesian cubical set whose \( \Psi \)-cubes are values \( M_0 \) with \( \text{Fl}(M_0) \subseteq \Psi \), and whose restriction maps are given by interval substitution followed by evaluation. Evaluation is necessary because, crucially, our operational semantics are not cubically stable, or stable under interval substitution. (Recall that, in contrast, evaluation in Idealized Nuprl is stable under substitution when lifted to open terms.) They do, however, respect permutations of interval variables, which form a “nominal” [Pit13; Pit15] wide subcategory of the Cartesian cube category.
To see why the operational semantics do not respect interval substitution, consider the circle $S^1$, a higher inductive type generated by a point base and a line loop. The constructors base and loop are values; we ensure the faces of loop are base by the operational step $(\text{loop}_x)(0/x) = \text{loop}_0 \mapsto \text{base}$. (Thus $M \text{ val} \not\Rightarrow M\psi \text{ val}.)$ 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). The operational step $S^1\text{-elim}_{c.A}(\text{loop}_x; P, x.L) \mapsto L$ is not stable under interval substitution, because:

\[
(S^1\text{-elim}_{c.A}(\text{loop}_x; P, x.L))(0/x) = S^1\text{-elim}_{c.A}(0/x)(\text{loop}_0; P(0/x), x.L) \mapsto S^1\text{-elim}_{c.A}(0/x)(\text{base}; P(0/x), x.L) \mapsto P(0/x)
\]

where $L$ and $P(0/x)$ are arbitrary terms. (Thus $M \mapsto M'$ does not imply $M\psi \mapsto M'\psi$.)

Fortunately, many operational semantics rules are in fact cubically stable, including all the rules shared with Idealized Nuprl. In Figure 4.2, many rules are annotated with $\triangleright$, defining an additional pair of judgments $M \text{ val}_{\triangleright}$ and $M \mapsto_{\triangleright} M'$ by replacing every occurrence of val (resp., $\mapsto$) in those rules with $\text{val}_{\triangleright}$ (resp., $\mapsto_{\triangleright}$). The judgments $M \text{ val}_{\triangleright}$ and $M \mapsto_{\triangleright} M'$ are cubically-stable subrelations of the operational semantics.

**Lemma 4.2 (Cubically-stable evaluation).** For all total interval substitutions $\psi$:

1. If $M \text{ val}_{\triangleright}$ then $M\psi \text{ val}$.
2. If $M \mapsto_{\triangleright} M'$ then $M\psi \mapsto M'\psi$.

Cubically-stable evaluation figures prominently in this chapter’s proofs, as head expansion (Lemma 4.32) holds only for cubically-stable steps. In RedPRL, we define a more sophisticated $\mapsto_{\triangleright}$ relation which accounts also for steps that are stable by virtue of occurring under interval binders [Ang+18]. For instance:

\[
\coe_{x,S^1\text{-elim}_{c.A}(\text{loop}_x; P, x.L)}^{r\mapsto r'}(M) \mapsto_{\triangleright} \coe_{x,L}^{r\mapsto r'}(M)
\]

is stable because the $x$ of loop is bound and thus unaffected by interval substitutions.

### 4.2 Constructing cubical type systems

We now assemble the cubical programs of Section 4.1 into the types and elements of our Cartesian cubical type theory. As in Section 2.2, we use Allen’s fixed point construction [All87] to define a PER of value types, and for each value type, a PER of value elements. In the cubical setting, however, we must define such PERs at every dimension (that is, for each finite set of interval variables).
Figure 4.2: Operational semantics: generic Kan operations.
Figure 4.2: Operational semantics: functions, pairs, paths.
\[
\begin{align*}
\text{Eq}_A(M, N) &\rightarrow \star\val \rightarrow \\
\text{hcom}_{\text{Eq}_A(E_0, E_1)}(M; \xi_i \leftrightarrow y.N_i) &\rightarrow \star
\end{align*}
\]

\[
\begin{align*}
\text{void} &\rightarrow \\
\text{nat} &\rightarrow \text{z} &\rightarrow \text{s}(M)
\end{align*}
\]

\[
\begin{align*}
M &\rightarrow M' \\
\text{natrec}(M; Z, n.a.S) &\rightarrow \text{natrec}(M'; Z, n.a.S) \\
\text{natrec}(z; Z, n.a.S) &\rightarrow Z
\end{align*}
\]

\[
\begin{align*}
\text{natrec}(s(M); Z, n.a.S) &\rightarrow S[M/n][\text{natrec}(M; Z, n.a.S)/a] \\
\text{hcom}_{\text{nat}}(M; \xi_i \leftrightarrow y.N_i) &\rightarrow M \\
\text{coe}_{x.\text{nat}}(M) &\rightarrow M
\end{align*}
\]

\[
\begin{align*}
\text{bool} &\rightarrow \text{true} &\rightarrow \text{false} \\
\text{if}(M; T, F) &\rightarrow \text{if}(M'; T, F) \\
\text{if}(\text{true}; T, F) &\rightarrow T \\
\text{if}(\text{false}; T, F) &\rightarrow F \\
\text{hcom}_{\text{bool}}(M; \xi_i \leftrightarrow y.N_i) &\rightarrow M \\
\text{coe}_{x.\text{bool}}(M) &\rightarrow M
\end{align*}
\]

Figure 4.2: Operational semantics: equality, void, natural numbers, Booleans.
\[
\begin{align*}
&\text{Figure 4.2: Operational semantics: circle, universes.}
\end{align*}
\]
\[ \mathcal{U}_j \]  

\[ \mathcal{U}_j^{\text{val}} \]  

\[ \mathcal{U}_j^{\text{Kan}} \]  

\[ \text{coe}^r_{x,u_j}(M) \rightarrow M \]

\[ \text{box}^{r \rightarrow r'}(M; \xi_i \leftarrow N_i) \rightarrow M \]

\[ r = r' \]

\[ \neg \xi_i (\forall i < j) \]

\[ \xi_j \]

\[ \text{cap}^{r \rightarrow r'}(M; \xi_i \leftarrow y.B_i) \rightarrow M \]

\[ r \neq r' \]

\[ \neg \xi_i (\forall i) \]

\[ M \rightarrow M' \]

\[ \text{cap}^{r \rightarrow r'}(M; \xi_i \leftarrow y.B_i) \rightarrow \text{cap}^{r' \rightarrow r'}(M'; \xi_i \leftarrow y.B_i) \]

\[ r \neq r' \]

\[ r \neq r' \]

\[ r_i \neq r'_i (\forall i) \]

\[ \text{cap}^{r' \rightarrow r'}(\text{box}^{r \rightarrow r'}(M; \xi_i \leftarrow N_i); r_i = r'_i \leftarrow y.B_i) \rightarrow M \]

\[ \text{P}_j = \text{hcom}^{r \rightarrow r'}_{A_B(z/s')}((M; r_i = r'_i \leftarrow y.N_i)) \]

\[ O_i = \text{cap}^{s \rightarrow s'}(N_i; s_j \leftarrow s'_j \leftarrow z.B_j) \]

\[ Q = \text{hcom}^{r \rightarrow r'}_{A_B(z/s')}((M; s_j = s'_j \leftarrow z.B_j); T) \]

\[ \overrightarrow{T} = r_i \leftarrow y.O_i; s_j \leftarrow s'_j \leftarrow y.\text{coe}^{s \rightarrow s'}_{z.B_j}(P_j), s = s' \leftarrow y.\text{hcom}^{r \rightarrow r'}_{A_B(z/s')}((M; r_i = r'_i \leftarrow y.N_i)) \]

\[ \text{hcom}^{r \rightarrow r'}_{A_B(z/s')}((M; r_i = r'_i \leftarrow y.N_i) \rightarrow \text{box}^{s \rightarrow s'}_{y}(Q; s_j = s'_j \leftarrow P_j(r'/y))) \]

---

Figure 4.2: Operational semantics: universes (continued).
Figure 4.2: Operational semantics: univalence.
**Definition 4.3.** A candidate cubical type system is a relation \( \tau(\Psi, A_0, B_0, \varphi) \) over a finite set \( \Psi \) of interval variables, \( A_0 \) val, \( B_0 \) val, and binary relations \( \varphi(M_0, N_0) \) over \( M_0 \) val and \( N_0 \) val, where \( \text{Fl}(A_0, B_0, M_0, N_0) \subseteq \Psi \).

The relation \( \tau(\{x_1, \ldots, x_n\}, A_0, B_0, \varphi) \) encodes that \( A_0 \) and \( B_0 \) are equal \( n \)-dimensional value types \( (x_1: \mathbb{I}, \ldots, x_n: \mathbb{I} \vdash A_0 = B_0 : \text{Type}_1 \) in Chapter 3’s notation) whose \( n \)-dimensional value elements are specified by \( \varphi \).

In Idealized Nuprl, we define the closed judgments (Definition 2.12) as the evaluation lifting of \( \tau \), thereby ensuring they are closed under evaluation. In cubical type theory, judgments must be closed under both evaluation and interval substitution—if \( x_1: \mathbb{I}, \ldots, x_n: \mathbb{I} \vdash M : A \) and \( \psi : \{x_1, \ldots, x_m\} \rightarrow \{x_1, \ldots, x_n\} \) then \( x_1: \mathbb{I}, \ldots, x_m: \mathbb{I} \vdash M\psi : A\psi \). The latter condition is complicated by the fact that types contain interval variables, as types depend on terms, and terms contain interval variables. Therefore, not only must a type itself evaluate to a value type, but also all of its interval substitution instances must as well. We track this data by assigning to each type over \( \Psi \) a \( \Psi \)-relation of elements.

**Definition 4.4.** A \( \Psi \)-relation is a family of binary relations \( \alpha_{\varphi}(M, N) \) indexed by interval substitutions \( \psi : \Psi' \rightarrow \Psi \), over terms \( \text{Fl}(M, N) \subseteq \Psi' \). We write \( \alpha(M, N) \) for \( \alpha_{\text{id}_{\Psi}}(M, N) \). A \( \Psi \)-relation that depends only on \( \Psi' \) (and not \( \psi \)) is context-indexed, and we write \( \alpha_{\Psi}(M, N) \).

A \( \Psi \)-PER captures the data of a functor \( (\square/\Psi)^{\text{op}} \rightarrow \text{Set} \). Intuitively, such functors are dependent Cartesian cubical sets, because a dependent type over the Yoneda embedding of \( \Psi \) is an object of \( [(\square/\Psi)^{\text{op}}, \text{Set}]^{\text{op}} \). Like such a functor, we can precompose a \( \Psi \)-relation \( \alpha \) by an interval substitution \( \psi : \Psi' \rightarrow \Psi \), obtaining a \( \Psi' \)-relation \( (\alpha_{\psi})_{\psi'}(M, N) := \alpha_{\psi\psi'}(M, N) \). A context-indexed PER captures the data of an ordinary Cartesian cubical set, or a functor \( \square_{\Psi}^{\text{op}} \rightarrow \text{Set} \).

As in Section 2.2, we define candidate cubical judgments for use in our fixed point construction. Every type \( A \) over \( \Psi \) must give rise to a \( \Psi \)-PER \( [A] \) of value elements, whose \( \psi_1 : \Psi_1 \rightarrow \Psi \) component we extract from \( \tau \) as the relation \( \varphi \) for which \( \tau(\Psi_1, A_1, A_1, \varphi) \) where \( A\psi_1 \Downarrow A_1 \). Moreover, meanings of types must be preserved by both evaluation \( ([A] = [A_0]) \) when \( A \Downarrow A_0 \) and interval substitution \( ([A]\psi_1 = [A\psi_1]) \) when \( \psi_1 : \Psi_1 \rightarrow \Psi \). Thus \( [A]\psi_1 = [A_1] \) when \( A\psi_1 \Downarrow A_1 \); unrolling definitions, we must require that for all \( \psi_2 : \Psi_2 \rightarrow \Psi_1 \), \( \tau \) assigns the same relation to the values of \( A\psi_1\psi_2 \) and \( A_1\psi_2 \) at \( \Psi_2 \).

**Definition 4.5** (Candidate cubical judgments). Given a candidate cubical type system \( \tau \), and writing \( \tau^{\downarrow}(\Psi, A, B, \varphi) \) for \( A \Downarrow A_0, B \Downarrow B_0 \), and \( \tau(\Psi, A_0, B_0, \varphi) \):

1. \( A \sim A' \Downarrow \alpha \in \tau[\Psi] \) when for all \( \psi_1 : \Psi_1 \rightarrow \Psi \) and \( \psi_2 : \Psi_2 \rightarrow \Psi_1 \), we have \( A\psi_1 \Downarrow A_1 \) and \( A'\psi_1 \Downarrow A'_1 \); there exists \( \varphi \) such that \( \tau^{\downarrow}(\Psi_2, -,-, \varphi) \) relates the pairs \( (A_1\psi_2, A\psi_1) \), \( (A'\psi_1, A'\psi_1) \), \( (A_1\psi_2, A_1\psi_1) \), \( (A'\psi_1, A'\psi_1) \), and \( (A_1\psi_2, A'_1\psi_2) \); and \( \alpha \) is a \( \Psi \)-relation on values satisfying \( \tau^{\downarrow}(\Psi', A\psi, A'\psi, \alpha_{\psi}) \) for all \( \psi : \Psi' \rightarrow \Psi \).
2. $M \sim M' \in \alpha [\Psi]$ when for all $\psi_1 : \Psi_1 \rightarrow \Psi$ and $\psi_2 : \Psi_2 \rightarrow \Psi_1$, we have $M\psi_1 \downarrow M_1$, $M'\psi_1 \downarrow M'_1$, and $(\alpha_\psi_1 \psi_2)$ relates the pairs $(M_1\psi_2, M_1\psi_1\psi_2)$, $(M_1\psi_2, M_1\psi_1\psi_2)$, $(M'_1\psi_2, M'_1\psi_1\psi_2)$, and $(M_1\psi_2, M'_1\psi_2)$.

3. $a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau [\Psi]$ when for all $\psi : \Psi' \rightarrow \Psi$ and $M \sim M' \in \alpha\psi [\Psi']$, $B\psi[M/a] \sim B'\psi[M'/a] \downarrow \beta^{\psi,M,M'} \in \tau [\Psi']$.

4. $a : \alpha \triangleright N \sim N' \in \beta [\Psi]$ when for all $\psi : \Psi' \rightarrow \Psi$ and $M \sim M' \in \alpha\psi [\Psi']$, $N\psi[M/a] \sim N'\psi[M'/a] \in \beta^{\psi,M,M'} [\Psi']$.

One can easily check that the candidate judgments are monotone in $\tau$ and closed under interval substitution. Monotonicity is essential, as we define the computational semantics of Cartesian cubical type theory as a cubical type system given by the least fixed point of a monotone function on the complete lattice of candidate cubical type systems.

**Definition 4.6.** A cubical type system is a candidate cubical type system $\tau$ satisfying:

1. **Unicity:** If $\tau(\Psi, A_0, B_0, \phi)$ and $\tau(\Psi, A_0, B_0, \phi')$ then $\phi = \phi'$.

2. **PER-valuation:** If $\tau(\Psi, A_0, B_0, \phi)$ then $\phi$ is symmetric and transitive.

3. **Symmetry:** If $\tau(\Psi, A_0, B_0, \phi)$ then $\tau(\Psi, B_0, A_0, \phi)$.

4. **Transitivity:** If $\tau(\Psi, A_0, B_0, \phi)$ and $\tau(\Psi, B_0, C_0, \phi)$ then $\tau(\Psi, A_0, C_0, \phi)$.

5. **Value-coherence:** If $\tau(\Psi, A_0, B_0, \phi)$ then $A_0 \sim B_0 \downarrow \alpha \in \tau [\Psi]$ for some $\alpha$.

If $\tau$ is a cubical type system, then $A \sim A' \downarrow \alpha \in \tau [\Psi]$ enjoys unicity, symmetry, transitivity, and is $\Psi$-PER-valued. If $\alpha$ is a $\Psi$-PER, then $\sim \sim \sim \in \alpha [\Psi]$ is a PER. Value-coherence ensures that the value types of $\tau$ are types in the sense of the candidate judgments, which was automatic in Section 2.2; it also ensures that value types form a Cartesian cubical set whose restriction maps are given by interval substitution followed by evaluation. (Without it, one might have $\tau(\Psi, A_0, A_0, \phi)$ but not $\tau^{\uparrow}(\Psi', A_0\psi, A_0\psi, \phi')$. ) In Definition 4.11, we will impose an analogous condition on the meaning of each type.

**Definition 4.7.** A $\Psi$-relation on values $\alpha$ is **value-coherent**, written $\text{Coh}(\alpha)$, when for all $\psi : \Psi' \rightarrow \Psi$, if $\alpha\psi(M_0, N_0)$ then $M_0 \sim N_0 \in \alpha\psi [\Psi']$. Given $a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau [\Psi]$, we write $\text{CohFam}(\beta)$ when $\text{Coh}(\beta^{\psi,M,M'})$ for all $\psi : \Psi' \rightarrow \Psi$ and $M \sim M' \in \alpha\psi [\Psi']$.

Despite their complexity, the above definitions in fact coincide with Definitions 2.2 and 2.3 if our cubical type theory has no cubical structure (that is, if programs cannot contain interval variables and $\tau$ is constant in $\Psi$). Thinking in terms of categories, one can regard a set-theoretic model as a cubical set model restricted to constant presheaves.
In Figure 4.3, we define each type former by its formation and introduction rules, as a monotone function on candidate cubical type systems, using the abbreviations:

\[
\text{valid}(r_i = r'_j) := \exists j, k. (r_j = r_k) \land (r'_j = 0) \land (r'_k = 1) \\
isConstr(C) := (c:C) \times ((c':C) \rightarrow \text{Path}_{-c}(c', c)) \\
\text{Equiv}(A, B) := (\bar{f}:A \rightarrow B) \times ((\bar{b}:B) \rightarrow \text{isConstr}((a:A) \times \text{Path}_{-b}(f \ a, b)) )
\]

In the cubical setting, it is essential that we define types by their introduction (rather than elimination) rules, because our proofs of value-coherence in Section 4.4 rely on knowing the head constructors of value elements in order to characterize their evaluation behavior under interval substitutions.

In Figure 4.3, the candidate cubical type system $K(v, \sigma)$ generates Kan types, drawing universes from $v$ and constituent Kan types from $\sigma$; $P(v, \sigma, \tau)$ generates pretypes (arbitrary types), drawing universes from $v$, constituent Kan types from $\sigma$, and constituent pretypes from $\tau$. Pretypes depend on Kan types because of homogeneous compositions of Kan types (HKAN); the only non-Kan type formers are Eq and HPRE. The value elements of certain base types (NAT and CIRC) are mutually defined for all $\Psi$ and are therefore defined as fixed points (\N and \C).

As in Idealized Nuprl, we have two universe hierarchies $\mathcal{U}_i^{\text{Kan}}$ and $\mathcal{U}_i^{\text{pre}}$ whose elements are Kan types and pretypes respectively. For all $i \in \{0, 1, \ldots, \omega\}$, we define $v_i$ as the candidate cubical type system consisting of universes $< i$, $\tau_i^{\text{Kan}} := \mu \sigma. K(v_i, \sigma)$ as the $i$th Kan type universe, and $\tau_i^{\text{pre}} := \mu \tau. P(v_i, \tau_i^{\text{Kan}}, \tau)$ as the $i$th pretype universe. The rest of this section is devoted to rather technical proofs that $\tau_i^{\text{pre}}$ and $\tau_i^{\text{Kan}}$ are cubical type systems (Theorem 4.9), and that $\tau_i^{\text{Kan}} \subset \tau_i^{\text{pre}}$ (Theorem 4.10).

**Lemma 4.8.** If $v$ and $\sigma$ are cubical type systems, then $\mu^{\text{Kan}}(v) := \mu \sigma. K(v, \sigma)$ and $\mu^{\text{pre}}(v, \sigma) := \mu \tau. P(v, \sigma, \tau)$ are cubical type systems.

**Proof.** We can consider each type former Fun, Pair, . . . separately because they are disjoint. We describe the proof for $\mu^{\text{pre}}(v, \sigma)$; the proof for $\mu^{\text{Kan}}(v)$ is analogous. Unlike in Lemma 2.6, we can prove each property separately, as the candidate cubical judgments inadvertently contain “cross cases” needed for symmetry and transitivity of type families.

1. **Unicity.**

Let $\Phi = \{ (\Psi, A_0, B_0, \psi) \mid \forall \psi'. \mu^{\text{pre}}(v, \sigma)(\Psi, A_0, B_0, \psi') \implies (\psi = \psi') \}$, and show that $\Phi$ is a pre-fixed point of $P(v, \sigma, \_)$ (that is, $P(v, \sigma, \Phi) \subseteq \Phi$). Because $\mu^{\text{pre}}(v, \sigma)$ is the least pre-fixed point, it will follow that $\mu^{\text{pre}}(v, \sigma) \subseteq \Phi$, and that $\mu^{\text{pre}}(v, \sigma)$ has unicity.

Assume that Fun($\Phi$)($\Psi$, $(a:A) \rightarrow B$, $(a:A') \rightarrow B'$, $\phi$). Thus $A \sim A' \upharpoonright \alpha \in \Phi [\Psi]$, and in particular, for all $\psi : \Psi' \rightarrow \Psi$, $\mu^{\text{pre}}(v, \sigma)(\Psi', A\psi, A'\psi, \phi')$ implies $\alpha_{\psi} = \phi'$, so $\alpha$ is unique in $\mu^{\text{pre}}(v, \sigma)$ when it exists. Similarly, each $\beta^{(-(-\_))}$ is unique in $\mu^{\text{pre}}(v, \sigma)$ when
\[ \text{Fun}(\tau) = \{ (\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi) \mid \\
\forall \alpha, \beta^{\sim\sim\sim}.(A \sim A' \downarrow \alpha \in \tau[\Psi]) \land \text{Coh}(\alpha) \\
\land (a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau[\Psi]) \land \text{CohFam}(\beta) \\
\land (\varphi = \{((\lambda a.N, \lambda a.N') \mid a : \alpha \triangleright N \sim N' \in \beta[\Psi])\}) \} \]

\[ \text{Pair}(\tau) = \{ (\Psi, (a:A) \times B, (a:A') \times B', \varphi) \mid \\
\forall \alpha, \beta^{\sim\sim\sim}.(A \sim A' \downarrow \alpha \in \tau[\Psi]) \land \text{Coh}(\alpha) \\
\land (a : \alpha \triangleright B \sim B' \downarrow \beta \in \tau[\Psi]) \land \text{CohFam}(\beta) \\
\land (\varphi = \{((\langle M, N \rangle, \langle M', N' \rangle) \mid (M \sim M' \in \alpha[\Psi]) \land (N \sim N' \in \beta^{\text{idv},M'M'}[\Psi])\}) \} \]

\[ \text{Path}(\tau) = \{ (\Psi, \text{Path}_{x.A}(P_0, P_1), \text{Path}_{x.A'}(P_0', P_1'), \varphi) \mid \\
\forall \alpha. (A \sim A' \downarrow \alpha \in \tau[\Psi]) \land \text{Coh}(\alpha) \land (\forall \varepsilon.P_\varepsilon \sim P_\varepsilon' \in \alpha(\varepsilon/x)[\Psi]) \\
\land (\varphi = \{((\langle x \rangle M, \langle x \rangle M') \mid \\
(M \sim M' \in \alpha[\Psi, x]) \land (\forall \varepsilon.M(\varepsilon/x) \sim P_\varepsilon \in \alpha(\varepsilon/x)[\Psi])\}) \} \]

\[ \text{Eq}(\tau) = \{ (\Psi, \text{Eq}_{x.A}(M, N), \text{Eq}_{x'A'}(M', N'), \varphi) \mid \\
\forall \alpha. (A \sim A' \downarrow \alpha \in \tau[\Psi]) \land \text{Coh}(\alpha) \land (M \sim M' \in \alpha[\Psi]) \land (N \sim N' \in \alpha[\Psi]) \\
\land (\varphi = \{((\bullet, \bullet) \mid M \sim N \in \alpha[\Psi])\}) \} \]

\[ \text{V}(\tau) = \{ ((\Psi, x), \text{V}_x(A, B, E), \text{V}_x(A', B', E'), \varphi) \mid \\
\forall \beta, \alpha^{\sim\sim\sim}.(B \sim B' \downarrow \beta \in \tau[\Psi, x]) \land \text{Coh}(\beta) \\
\land (\forall \psi.(x\psi = 0) \implies (A\psi \sim A'\psi \downarrow \alpha^{\psi} \in \tau[\Psi']) \land \text{Coh}(\alpha^{\psi}) \\
\land (\text{Equiv}(A\psi, B\psi) \sim \text{Equiv}(A\psi, B\psi) \downarrow \eta^{\psi} \in \tau[\Psi'] \land (E\psi \sim E'\psi \in \eta^{\psi}[\Psi'])) \\
\land (\varphi = \{(\text{Vin}_x(M, N), \text{Vin}_x(M', N')) \mid (N \sim N' \in \beta[\Psi, x]) \land (\forall \psi.(x\psi = 0) \implies \\
(M\psi \sim M'\psi \in \alpha^{\psi}[\Psi']) \land (\text{fst}(E\psi) M\psi \sim N\psi \in \beta^{\psi}[\Psi'])\}) \} \}

\[ \text{HPre}(\tau) = \{ (\Psi, \text{hcom}_{\text{hcom}_{x.A}}(\xi_i \mapsto y.B_i), \text{hcom}_{\text{hcom}_{x'A'}}(\xi_i \mapsto y'.B'_i), \{} \} \mid \\
\forall \alpha, \beta^{\sim\sim\sim}.(r \neq r') \land (\forall i, \xi_i) \land \text{valid}(\xi_i) \land (A \sim A' \downarrow \alpha \in \tau[\Psi]) \land \text{Coh}(\alpha) \\
\land (\forall i, j, \psi.(\xi_i \psi \land \xi_j \psi) \implies (B_i \psi \sim B'_j \psi \downarrow \beta^{\psi,i,j} \in \tau[\Psi']) \land \text{Coh}(\beta^{\psi,i,j}) \\
\land (\forall i, \psi.\xi_i \psi \implies B_i(r/y) \psi \sim A\psi \downarrow _{\_} \in \tau[\Psi']) \} \}

Figure 4.3: PER semantics.
\[
\begin{align*}
\text{Void} &= \{(\Psi, \text{void}, \text{void}, \{\})\} \\
\text{NAT} &= \{(\Psi, \text{nat}, \text{nat}, \varphi) \mid \varphi = \{(M_0, M'_0) \mid \mathbb{N}(\Psi, M_0, M'_0)\}\} \\
\text{BOOL} &= \{(\Psi, \text{bool}, \text{bool}, \{(\text{true}, \text{true}), (\text{false}, \text{false})\})\} \\
\text{CIRC} &= \{(\Psi, S^1, S^1, \varphi) \mid \varphi = \{(M_0, M'_0) \mid \mathbb{C}(\Psi, M_0, M'_0)\}\} \\
\text{UPRE}(v) &= \{(\Psi, \mathcal{U}^\text{pre}_j, \mathcal{U}^\text{pre}_j, \varphi) \mid v(\Psi, \mathcal{U}^\text{pre}_j, \mathcal{U}^\text{pre}_j, \varphi)\} \\
\text{UKAN}(v) &= \{(\Psi, \mathcal{U}^\text{Kan}_j, \mathcal{U}^\text{Kan}_j, \varphi) \mid v(\Psi, \mathcal{U}^\text{Kan}_j, \mathcal{U}^\text{Kan}_j, \varphi)\} \\
\text{HKAN}(\tau) &= \{(\Psi, \text{hcom}^\text{Kan}_{\mathcal{U}_\beta^{\text{box}}} (A; \xi_i \hookrightarrow y.B_i), \text{hcom}^\text{Kan}_{\mathcal{U}_\beta^{\text{box}}} (A'; \xi_i \hookrightarrow y.B'_i), \varphi) \mid \\
&\quad \exists \alpha, \beta^{(-, -, -)}(r \neq r') \wedge (\forall i. -\xi_i) \wedge \text{valid}(\overrightarrow{\xi_i}) \wedge (A \sim A' \downarrow \alpha \in \tau [\Psi]) \wedge \text{Coh}(\alpha) \\
&\quad \wedge (\forall i, j, \psi : \Psi' \rightarrow (\Psi, y).\langle \xi_i \psi \wedge \xi_j \psi \rangle) \implies \\
&\quad (B_i \psi \sim B'_j \psi \downarrow \beta^{\psi,i,j} \in \tau [\Psi']) \wedge \text{Coh}(\beta^{\psi,i,j}) \\
&\quad \wedge (\forall i, j, \psi : \Psi' \rightarrow (\Psi, y).\langle \xi_i \psi \wedge \xi_j \psi \rangle) \implies B_i(r/y) \psi \sim A \psi \downarrow \in \tau [\Psi'] \\
&\quad \wedge (\forall i, j, \psi : \Psi' \rightarrow (\Psi, y).\langle \xi_i \psi \wedge \xi_j \psi \rangle) \implies N_i \psi \sim N'_j \psi \in \beta^{\psi,i,j}(r'/y) [\Psi'] \\
&\quad \wedge (\forall i, j, \psi : \Psi' \rightarrow (\Psi, y).\langle \xi_i \psi \wedge \xi_j \psi \rangle) \implies M \psi \sim \text{coe}^{r'/y, \psi}(N_i \psi \in \alpha \psi [\Psi'])\}} \\
\mathbb{N} &= \mu R.((\Psi, z, z) \cup \{(\Psi, s(M), s(M')) \mid M \sim M' \in R [\Psi]) \\
\mathbb{C} &= \mu R.((\Psi, \text{base}, \text{base}), ((\Psi, x), \text{loop}_x, \text{loop}_x)) \cup \\
&\quad \{(\Psi, \text{hcom}^\text{box}_{\mathcal{U}_\beta^{\text{box}}} (M; \xi_i \hookrightarrow y.N_i), \text{hcom}^\text{box}_{\mathcal{U}_\beta^{\text{box}}} (M'; \xi_i \hookrightarrow y.N'_i)) \mid \\
&\quad (r \neq r') \wedge (\forall i. -\xi_i) \wedge \text{valid}(\overrightarrow{\xi_i}) \wedge (M \sim M' \in R [\Psi]) \\
&\quad \wedge (\forall i, j, \psi : \Psi' \rightarrow (\Psi, y).\langle \xi_i \psi \wedge \xi_j \psi \rangle) \implies N_i \psi \sim N'_j \psi \in R [\Psi'] \\
&\quad \wedge (\forall i, j, \psi : \Psi' \rightarrow (\Psi, y).\langle \xi_i \psi \wedge \xi_j \psi \rangle) \implies M \psi \sim \text{coe}^{r'/y, \psi}(N_i \psi \in \alpha \psi [\Psi'])?\}} \\
\end{align*}
\]

Figure 4.3: PER semantics.
\[ P(v, \sigma, \tau) = \text{Fun}(\tau) \cup \text{Pair}(\tau) \cup \text{Path}(\tau) \cup \text{Eq}(\tau) \cup \text{V}(\tau) \cup \text{HKan}(\sigma) \cup \text{HPre}(\tau) \]
\[ \cup \text{Void} \cup \text{Nat} \cup \text{Bool} \cup \text{Circ} \cup \text{UPre}(v) \cup \text{UKan}(v) \]
\[ K(v, \sigma) = \text{Fun}(\sigma) \cup \text{Pair}(\sigma) \cup \text{Path}(\sigma) \cup \text{V}(\sigma) \cup \text{HKan}(\sigma) \]
\[ \cup \text{Void} \cup \text{Nat} \cup \text{Bool} \cup \text{Circ} \cup \text{UPre}(v) \cup \text{UKan}(v) \]

\[ v_n = \{(\Psi, \mathcal{U}^\kappa_i, \mathcal{U}^\kappa_i, \phi) \mid (i < n) \land (\phi = \{(A_0, B_0) \mid \tau_i^\kappa(\Psi, A_0, B_0, -))\} \}
\]
\[ \tau_n^{\text{Kan}} = \mu\sigma.K(v_n, \sigma) \]
\[ \tau_n^{\text{pre}} = \mu\tau.P(v_n, \tau_n^{\text{Kan}}, \tau) \]

Figure 4.3: PER semantics.

it exists. The relation \( \varphi \) is determined uniquely by \( \alpha \) and \( \beta^{(\sim,\sim,\sim)} \). Now let us show \( \Phi(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi) \), that is, assume \( \mu^{\text{pre}}(v, \sigma)(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi') \) and show \( \varphi = \varphi' \). It follows that \( A \sim A' \downarrow \alpha' \in \mu^{\text{pre}}(v, \sigma) [\Psi] \) for some \( \alpha' \), and similarly for some family \( \beta' \), but \( \alpha = \alpha' \) and \( \beta^{(\sim,\sim,\sim)} = \beta'^{(\sim,\sim,\sim)} \). Because \( \varphi' \) is defined using the same \( \alpha \) and \( \beta^{(\sim,\sim,\sim)} \) as \( \varphi \), we conclude \( \varphi = \varphi' \). Other cases are similar; for HKAN, UPRE, UKAN we use that \( v, \sigma \) have unicity.

2. PER-valuation.

Let \( \Phi = \{(\Psi, A_0, B_0, \varphi) \mid \varphi \) is a PER\}, and show that \( \Phi \) is a pre-fixed point of \( P(v, \sigma, -) \).

It follows that \( \mu^{\text{pre}}(v, \sigma) \) is PER-valued, by \( \mu^{\text{pre}}(v, \sigma) \subseteq \Phi \).

Assume that \( \text{Fun}(\Phi)(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi) \). Then \( A \sim A' \downarrow \alpha \in \Phi [\Psi] \), and in particular, for all \( \psi : \Psi' \rightarrow \Psi, \Phi^\psi(\Psi', A\psi, A'\psi, \alpha\psi) \), so each \( \alpha\psi \) is a PER. Similarly, each \( \beta^\psi_{\psi'}^{M,M'} \) is a PER. Now we must show \( \Phi(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi) \).

The relation \( \varphi \) is a PER because \( \alpha \) and \( \beta \) are PER-valued so their candidate equality judgments are PERs. Most cases proceed in this fashion. For \( \text{Nat} \) and \( \text{Circ} \) we show that \( \mathbb{N} \) and \( \mathbb{C} \) are symmetric and transitive at all \( \Psi \) (employing the same strategy as in parts (3–4)); for HKAN, UPRE, UKAN we use that \( \sigma, v \) are PER-valued.


Let \( \Phi = \{(\Psi, A_0, B_0, \varphi) \mid \mu^{\text{pre}}(v, \sigma)(\Psi, B_0, A_0, \varphi)\} \), and show that \( \Phi \) is a pre-fixed point of \( P(v, \sigma, -) \). It will follow that \( \mu^{\text{pre}}(v, \sigma) \) is symmetric, by \( \mu^{\text{pre}}(v, \sigma) \subseteq \Phi \).

Assume that \( \text{Fun}(\Phi)(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi) \). Then \( A \sim A' \downarrow \alpha \in \Phi [\Psi] \) and \( \text{Coh}(\alpha) \), and therefore \( \mu^{\text{pre}}(v, \sigma)^\downarrow(\Psi', A'\psi, A\psi, \alpha\psi), A\psi_1 \downarrow A_1, A'\psi_1 \downarrow A'_1 \), and \( \mu^{\text{pre}}(v, \sigma)^\downarrow(\Psi_2, -,-, \varphi) \) relates the pairs \((A\psi_1\psi_2, A_1\psi_2), (A_1\psi_2, A'\psi_1\psi_2), (A'\psi_1\psi_2, A'_1\psi_2)\).
(\(A'_1\psi_2, A'\psi_1\psi_2\)), and (\(A'_1\psi_2, A_1\psi_2\)). Similar facts hold for instances of \(B, B', \beta\). We must show \(\Phi(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)\), that is, \(\mu^{pre}(v, \sigma)(\Psi, (a:A) \rightarrow B', (a:A) \rightarrow B, \varphi)\). This requires \(A' \sim A \downarrow \alpha \in \mu^{pre}(v, \sigma) [\Psi] \) and \(\Coh(\alpha)\), which follows from the above facts; and also \(B'\psi[M/a] \sim B\psi[M'/a] \downarrow \beta^{\psi,M,M'} \in \mu^{pre}(v, \sigma) [\Psi] \) and \(\Coh(\beta^{\psi,M,M'})\) whenever \(M \sim M' \in \alpha\psi [\Psi']\), which follows from the symmetry of \(- \sim - \in \alpha\psi [\Psi']\) (because each \(\alpha\psi\) is a PER, by (2)), and the above facts. Other cases are similar; for HKEN we use that \(\sigma\) is symmetric.

4. Transitivity.

Let \(\Phi = \{ (\Psi, A_0, B_0, \varphi) \mid \forall C_0. \mu^{pre}(v, \sigma)(\Psi, B_0, C_0, \varphi) \implies \mu^{pre}(v, \sigma)(\Psi, A_0, C_0, \varphi) \}\), and show that \(\Phi\) is a pre-fixed point of \(P(v, \sigma, -)\). It will follow that \(\mu^{pre}(v, \sigma)\) is transitive, by \(\mu^{pre}(v, \sigma) \subseteq \Phi\).

Assume that \(\Fun(\Phi)(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)\). Then \(A \sim A' \downarrow \alpha \in \Phi [\Psi]\), and thus if \(\mu^{pre}(v, \sigma)(\Psi, A', \alpha\psi, C_0, \alpha\varphi)\) then \(\mu^{pre}(v, \sigma)(\Psi, A', \alpha\psi, C_0, \alpha\varphi)\). Furthermore, \(A\psi_1 \downarrow A', A'\psi_1 \downarrow A'_1\), and for any \(C_0, \mu^{pre}(v, \sigma)(\Psi, 0, \beta\psi, \beta\varphi)\) relates \((A\psi_1\psi_2, C_0)\) if and only if \((A\psi_1\psi_2, C_0)\) if and only if \((A\psi_1\psi_2, C_0)\); and if \((A\psi_1\psi_2, C_0)\) then \((A\psi_1\psi_2, C_0)\). Similar facts hold for \(B, B', \beta\) by \(\beta : \alpha \rightarrow B \sim B' \downarrow \beta \in \Phi [\Psi]\).

Now we must show \(\Phi(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)\), that is, if \(\mu^{pre}(v, \sigma)(\Psi, (a:A) \rightarrow B', C_0, \varphi)\) then \(\mu^{pre}(v, \sigma)(\Psi, (a:A) \rightarrow B, C_0, \varphi)\). Inspecting \(P, C_0 = (a:A') \rightarrow B''\); thus \(A' \sim A'' \downarrow \alpha' \in \mu^{pre}(v, \sigma) [\Psi] \) and \(\Coh(\alpha')\), so \(\mu^{pre}(v, \sigma)(\Psi, A', A''\psi, A''\varphi)\), and by hypothesis, \(\mu^{pre}(v, \sigma)(\Psi, A', A''\psi, A''\varphi)\) and \(\Coh(\alpha)\). We already know \(A\psi_1 \downarrow A, A''\psi_1 \downarrow A''\), and that \(\mu^{pre}(v, \sigma)(\Psi, 0, \beta\psi, \beta\varphi)\) relates \((A''\psi_1\psi_2, A''\psi_2)\) and vice versa. By \((A''\psi_1\psi_2, A''\psi_2)\) and the above, we have \((A\psi_1\psi_2, A\psi_1\psi_2)\). Finally, by \((A\psi_1\psi_2, A\psi_1\psi_2)\) and transitivity we have \((A\psi_1\psi_2, A\psi_1\psi_2)\), hence by transitivity and symmetry \((A\psi_1\psi_2, A\psi_1\psi_2)\), and again by transitivity \((A\psi_1\psi_2, A\psi_1\psi_2)\); as needed, \((A\psi_1\psi_2, A\psi_1\psi_2)\) and vice versa follow by transitivity. Similarly, using transitivity of each \(\alpha\psi\) (by (2)), whenever \(M \sim M' \in \alpha\psi [\Psi]\), \(B\psi[M/a] \sim B'\psi[M'/a] \downarrow \beta^{\psi,M,M'} \in \mu^{pre}(v, \sigma) [\Psi']\) and \(\Coh(\beta^{\psi,M,M'})\). Other cases are similar; for HKEN we use that \(\sigma\) is transitive.

5. Value-coherence.

Let \(\Phi = \{ (\Psi, A_0, B_0, \varphi) \mid A_0 \sim B_0 \downarrow \alpha \in \mu^{pre}(v, \sigma) [\Psi] \}\), and show that \(\Phi\) is a pre-fixed point of \(P(v, \sigma, -)\). It will follow that \(\mu^{pre}(v, \sigma)\) is value-coherent, by \(\mu^{pre}(v, \sigma) \subseteq \Phi\).

We will check \(\Fun(\text{Pair}, \text{Path}, \text{and Eq are similar})\) and \(V\) (HKEN and HPre are similar); the property \(P(v, \sigma, \Phi) \subseteq \Phi\) is trivial for base types (\(\text{Void}, \text{Nat}, \ldots\) ) and universes (\(\text{UPre}, \text{UKan}\)).

Assume \(\Fun(\Phi)(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \varphi)\). Then by \(A \sim A' \downarrow \alpha \in \Phi [\Psi]\) and \(\Coh(\alpha)\), we have \((\Psi, A\psi_1, A'\psi_1, \alpha\psi_1), A\psi_1 \downarrow A_1, A'\psi_1 \downarrow A'_1, \Phi(\Psi, A_1\psi_2, A_1\psi_2, \varphi')\), and so forth. Note that on values, \(A_0 \sim B_0 \downarrow \alpha \in \tau [\Psi]\) implies \(\tau(\Psi, A_0, B_0, \alpha_{id\psi})\).
Therefore $\mu_{\text{pre}}(v, \sigma) = (\Psi, A\psi, A'\psi, \alpha_{\psi})$, and so forth. Similar facts hold for $B, B', \beta$ by $a : \alpha \rightarrow B \sim B' \downarrow \beta \in \Phi[\Psi]$ and CohFam($\beta$). We must show $\Phi(\Psi, (a:A) \rightarrow B, (a:A') \rightarrow B', \phi')$, that is, $(a:A) \rightarrow B \sim (a:A') \rightarrow B' \downarrow \gamma \in \mu_{\text{pre}}(v, \sigma) [\Psi]$. We know $(a:A) \rightarrow B$ val$_{\text{P}}$, and by the above, $A \sim A' \downarrow \alpha \in \mu_{\text{pre}}(v, \sigma) [\Psi]$, Coh($\alpha$), and for all $M \sim M' \in \alpha_{\psi} [\Psi']$, $\psi[M/a] \sim B'\psi[M'/a] \downarrow \beta_{\psi,M,M'} \in \mu_{\text{pre}}(v, \sigma) [\Psi']$ and Coh($\beta_{\psi,M,M'}$). The result holds because value-coherence and the candidate cubical judgments are closed under interval substitution.

The V case is more complicated because not all instances of $V_x(A, B, E)$ have the same head constructor. As before, by $V(\Phi)(\Psi, V_x(A, B, E), V_x(A', B', E'), \phi)$ we have $B \sim B' \downarrow \beta \in \mu_{\text{pre}}(v, \sigma) [\Psi]$ and for all $\psi$ with $x\psi = 0$, $A\psi \sim A'\psi \downarrow \alpha_{\psi} \in \mu_{\text{pre}}(v, \sigma) [\Psi']$. However, to see that $V_x(A, B, E) \sim V_x(A', B', E') \downarrow \gamma \in \mu_{\text{pre}}(v, \sigma) [\Psi']$, we must consider the cases in which $V_x(\Psi, B, E)$ steps to $\Psi$ or $B\psi$, and show that for every $\psi_1, \psi_2$, the appropriate relations hold in $\mu_{\text{pre}}(v, \sigma)$. See Rule 4.78 for full details, and Lemma 4.88 and Rule 4.87 for corresponding results about HKAN and HPRE. □

**Theorem 4.9.** For all $i \in \{0, 1, \ldots, \omega\}$, $v_i$, $\tau_i^{\text{pre}}$, and $\tau_i^{\text{Kan}}$ are cubical type systems.

**Proof.** By strong induction on $i$. In the base case, $v_0$ is empty and thus a cubical type system; therefore, by Lemma 4.8, so are $\tau_0^{\text{Kan}} = \mu^{\text{Kan}}(v_0)$ and $\tau_0^{\text{pre}} = \mu_{\text{pre}}(v_0, \mu^{\text{Kan}}(v_0))$. In the inductive case, suppose $\tau_i^{\text{Kan}}$ are cubical type systems for $i < n$. Then $v_n$ is a cubical type system: unicity, symmetry, transitivity, and value-coherence are immediate; PER-valuation follows from symmetry and transitivity of $\tau_i^{\text{Kan}}$ for $i < n$. By Lemma 4.8, $\tau_n^{\text{Kan}}$ and thence $\tau_n^{\text{pre}}$ are cubical type systems. (The $\omega$ case is identical.) □

**Theorem 4.10.** For $i, j \in \{0, 1, \ldots, \omega\}$ where $i \leq j$, we have $\tau_i^{\text{Kan}} \subseteq \tau_j^{\text{Kan}}$ and $\tau_i^{\text{pre}} \subseteq \tau_j^{\text{pre}}$.

**Proof.** The functions $P$ and $K$ are monotone in all arguments, so by Lemma 2.9, $\mu^{\text{pre}}$ and $\mu^{\text{Kan}}$ are as well. When $i \leq j$, $v_i \subseteq v_j$ by construction, so $\tau_i^{\text{Kan}} = \mu^{\text{Kan}}(v_i) \subseteq \mu^{\text{Kan}}(v_j) = \tau_j^{\text{Kan}}$, and similarly, $\tau_i^{\text{pre}} = \mu_{\text{pre}}(v_i, \tau_i^{\text{Kan}}) \subseteq \mu_{\text{pre}}(v_j, \tau_j^{\text{Kan}}) = \tau_j^{\text{pre}}$.

By a theorem of Bekić [Bek84] on simultaneous fixed points, for all $v$,

$$ (\mu^{\text{Kan}}(v), \mu_{\text{pre}}(v, \mu^{\text{Kan}}(v))) = (\mu, \tau).K(v, \sigma), P(v, \sigma, \tau) $$

When $\sigma \subseteq \tau$, $K(v, \sigma) \subseteq P(v, \sigma, \tau)$; thus $\tau_i^{\text{Kan}} \subseteq \tau_i^{\text{pre}}$ follows by Lemma 2.10. □

### 4.3 Cubical judgments and Kan operations

The typehood and membership judgments of Cartesian cubical type theory mirror those of Idealized Nuprl, parametrized by finite sets of interval variables $\Psi$. This section nevertheless differs from Section 2.3 in two key respects. First, the candidate cubical judgments
(Definition 4.5) are more complex than Idealized Nuprl’s candidate judgments (Definition 2.3), and must be established by more sophisticated means—one must consider the evaluation behavior of all interval substitution instances of a term, not only the term itself. Secondly, we must define the uniform Kan operations (Definition 4.29), introduced in Chapter 3, which serve an analogous role to Book HoTT’s identity elimination.

**Definition 4.11** (Closed judgments). Given a cubical type system $\tau$:

1. $A \simeq B$ type$_{\text{pre}} [\Psi]$ when $A \sim B \downarrow \alpha \in \tau [\Psi]$ and $\text{Coh}(\alpha)$.

2. $M \simeq N \in A [\Psi]$, presupposing $A \simeq A$ type$_{\text{pre}} [\Psi]$, when $A \sim A \downarrow \alpha \in \tau [\Psi]$ and $M \sim N \in \alpha [\Psi]$.

The closed judgments are symmetric and transitive, and therefore $A \simeq A$ type$_{\text{pre}} [\Psi]$ whenever $A \simeq B$ type$_{\text{pre}} [\Psi]$. As in Section 2.3, we abbreviate $A \simeq A$ type$_{\text{pre}} [\Psi]$ by $A$ type$_{\text{pre}} [\Psi]$ and $M \simeq M \in A [\Psi]$ by $M \in A [\Psi]$ (and similarly for all subsequent judgments). Every judgment $\mathcal{J} [\Psi]$ presupposes implicitly that $\text{Fl}(\mathcal{J}) \subseteq \Psi$. All judgments are relative to a choice of cubical type system $\tau$, which we can notate explicitly by $\tau \models \mathcal{J} [\Psi]$; when unspecified, judgments are relative to $\tau_0$. When $A \sim B \downarrow \alpha \in \tau [\Psi]$ and $\tau$ is a cubical type system, the $\Psi$-PER $\alpha$ is uniquely determined and independent of $B$, so we write $\llbracket A \rrbracket_\alpha$ for $\alpha$.

As before, equal pretypes have equal elements; moreover, the closed judgments are preserved by interval substitutions.

**Lemma 4.12.** If $A \simeq B$ type$_{\text{pre}} [\Psi]$ and $M \simeq N \in A [\Psi]$ then $M \simeq N \in B [\Psi]$.

**Proof.** By the unicity and symmetry properties of $\tau$, $\llbracket A \rrbracket = \llbracket B \rrbracket$; thus $M \sim N \in \llbracket A \rrbracket [\Psi]$ implies $M \sim N \in \llbracket B \rrbracket [\Psi]$. $\square$

**Lemma 4.13.** Supposing $\psi : \Psi' \rightarrow \Psi$:

1. If $A \simeq B$ type$_{\text{pre}} [\Psi]$ then $A\psi \simeq B\psi$ type$_{\text{pre}} [\Psi']$.

2. If $M \simeq N \in A [\Psi]$ then $M\psi \simeq N\psi \in A\psi [\Psi']$.

**Proof.** For part (1), $A \sim B \downarrow \llbracket A \rrbracket \in \tau [\Psi]$ implies $A\psi \sim B\psi \downarrow \llbracket A \rrbracket \psi \in \tau [\Psi']$ by precomposing $\psi_1$ with $\psi$ in Definition 4.5, and by the unicity property of $\tau$. Similarly, $\text{Coh}(\llbracket A \rrbracket \psi)$ follows from $\text{Coh}(\llbracket A \rrbracket)$ by precomposition with $\psi$ in Definition 4.7. For part (2), the presupposition $A\psi$ type$_{\text{pre}} [\Psi']$ is resolved by part (1); the result follows by $\llbracket A\psi \rrbracket = \llbracket A \rrbracket \psi$ and precomposing $\psi_1$ with $\psi$ in Definition 4.5. $\square$

The judgments of Idealized Nuprl are closed under both forward (Lemma 2.16) and backward (Lemma 2.15) evaluation, essentially by definition. Cubical typehood and membership are also closed under forward evaluation to a value, but the arguments are more subtle and require value-coherence (of the cubical type system and of pretypes, respectively).
Lemma 4.14 (Evaluation I). If $A \text{ type}_{\text{pre}} [\Psi]$ then $A \downarrow A_0$ where $A \equiv A_0 \text{ type}_{\text{pre}} [\Psi]$.

Proof. We obtain $A \downarrow A_0$ by instantiating $A \sim A \downarrow [A] \in \tau [\Psi]$ at substitutions $\text{id}_\Psi$ and $\text{id}_\Psi$. It remains only to show $A \sim A_0 \downarrow [A] \in \tau [\Psi]$; $\text{Coh}([A])$ follows from $A \text{ type}_{\text{pre}} [\Psi]$.

Suppose $\psi_1 : \Psi_1 \rightarrow \Psi$ and $\psi_2 : \Psi_2 \rightarrow \Psi_1$. We must show that $\tau^\downarrow(\psi_2, A\psi_1\psi_2, A_1\psi_2, \varphi)$, $\tau^\downarrow(\psi_2, A\psi_1\psi_2, A_0\psi_1\psi_2, \varphi')$, and $\tau^\downarrow(\psi_2, A_1\psi_2, A_1'\psi_2, \varphi'')$ where $A\psi_1 \downarrow A_1$ and $A_0\psi_1 \downarrow A_1'$; the remaining conditions of Definition 4.5 follow from unicity, symmetry, and transitivity of $\tau$.

The first holds by $A \sim A \downarrow [A] \in \tau [\Psi]$ at the substitutions $\psi_1$ and $\psi_2$. The second holds by $A \sim A \downarrow [A] \in \tau [\Psi]$ at $\text{id}_\Psi$ and $\psi_1\psi_2$. For the third, $A \sim A \downarrow [A] \in \tau [\Psi]$ at $\text{id}_\Psi$ implies $\tau^\downarrow(\psi_1, A_1, A\psi_1, \_)$ and at $\text{id}_\Psi, \psi_1$ implies $\tau^\downarrow(\psi_1, A_0\psi_1, A\psi_1, \_)$ by transitivity, $\tau(\psi_1, A_1, A_1', \_)$; by value-coherence of $\tau$, $A_1 \sim A_1' \downarrow [A_1] \in \tau [\Psi]$. Therefore $\tau^\downarrow(\psi_2, A_1\psi_2, A_1'\psi_2, \varphi'')$ as required. $\square$

Lemma 4.15. If $M \in A [\Psi]$, $N \in A [\Psi]$, and $[A]^\downarrow[M, N]$, then $M \equiv N \in A [\Psi]$.

Proof. For all $\psi_1 : \Psi_1 \rightarrow \Psi$ and $\psi_2 : \Psi_2 \rightarrow \Psi_1$, we must show $[A]^\downarrow_{\psi_1\psi_2}(M\psi_1\psi_2, N\psi_1\psi_2)$; the remaining conditions of Definition 4.5 follow from $M \in A [\Psi]$ and $N \in A [\Psi]$. By presupposition we have $\text{Coh}([A])$, so $[A]^\downarrow[M, N]$ implies $M_0 \equiv N_0 \in [A] [\Psi]$ where $M \equiv M_0$ and $N \equiv N_0$. We conclude that $[A]^\downarrow_{\psi_1\psi_2}(M_0\psi_1\psi_2, N_0\psi_1\psi_2)$. From $M \in A [\Psi]$ we know $[A]^\downarrow_{\psi_1\psi_2}(M_0\psi_1\psi_2, M_1\psi_1\psi_2)$ and similarly for $N$; the result follows by transitivity. $\square$

Lemma 4.16 (Evaluation II). If $M \in A [\Psi]$ then $M \equiv M_0 \in A [\Psi]$ where $M \equiv M_0$.

Proof. By $M \in A [\Psi]$ at $\text{id}_\Psi, \text{id}_\Psi$ we have $M \equiv M_0$ and $[A](M_0, M_0)$; by $\text{Coh}([A])$ this implies $M_0 \in A [\Psi]$. The result follows by Lemma 4.15. $\square$

Lemma 4.15 is much weaker than its counterpart in Idealized Nuprl, in which $M \equiv N \in A$ follows directly from $[A]^\downarrow[M, N]$! That statement is easy to refute in cubical type theory, as the term $\text{S}_1\text{-elim}_{\text{bool}}(\text{loop}_{\psi_1}, z, \_\_\_\_\_\text{true})$ evaluates to true but cannot be a Boolean because its $\langle 0/x \rangle$ face evaluates to $z$. Nor does it suffice to check that $[A]^\downarrow_{\psi_1}(M\psi_1, N\psi_1)$ for all $\psi$—while all instances of $\text{S}_1\text{-elim}_{\text{bool}}(\text{loop}_{\psi_1}; \text{false}, \_\_\_\_\_\text{true})$ evaluate to Booleans, evaluation under $\text{id}_\Psi$ then $\langle 0/x \rangle$ yields true, but evaluation under $\langle 0/x \rangle$ then $\text{id}_\Psi$ yields false.

In Idealized Nuprl, we commonly use head expansion (or closure under backward evaluation) to establish judgments. Its analogue in this chapter is coherent expansion,\(^2\) which states that a term is a pretype (resp., element) if all of its interval substitution instances eventually step to pretypes (resp., elements) in a suitably coherent way. More precisely, to prove $A \text{ type}_{\text{pre}} [\Psi]$, it suffices to exhibit a family of reducts $\{A_{\psi'} \mid A\psi \overset{!}{\rightarrow}^* A_{\psi'}\}_{\psi : \psi' \rightarrow \Psi}$ for which $A_{\psi'} \text{ type}_{\text{pre}} [\Psi']$ and $\langle A_{\psi'} (\psi) \rangle_{\psi'} \equiv (A^\downarrow_{\psi'})_{\psi'}$ type$_{\text{pre}} [\Psi'']$ (or equivalently, $A_{\psi'} \equiv (A_{\psi'})_{\text{id}_\Psi}$ type$_{\text{pre}} [\Psi']$).

\(^2\)A similar lemma appears independently in Huber’s canonicity proof [Hub18, Lemma 9].
Lemma 4.17 (Coherent expansion I). Suppose $A$ is a term and $\{A_\psi^\Psi\}_{\psi: \Psi \to \Psi}$ is a family of terms such that $A^\Psi_\psi \equiv (A^\Psi_{\psi_1})_{\psi_1: \Psi_1 \to \Psi}$ and $A_\psi \iff A^\Psi_\psi$ for all $\psi : \Psi \to \Psi$. Then $A \equiv A^\Psi_{\psi_1}$ type $\pre_{\Psi}$. 

Proof. It suffices to check for all $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$ that $A_\psi_1 \downarrow A_1$, $(A^\Psi_{\psi_1})_\psi_1 \downarrow A_1'$, and $\tau^\Psi_\psi(\Psi_2, -, - , - )$ relates $A_1, A_1', A_\psi_1, (A^\Psi_{\psi_1})_\psi_2, (A^\Psi_{\psi_1})_\psi_1, A_1, A_1'$. 

1. $\psi_1 \downarrow A_1$ and $\tau^\Psi_\psi(\Psi_2, A_1, A_1', A_\psi_1, A_\psi_2, \Phi)$. 

By $A^\Psi_{\psi_1}$ type $\pre_{\Psi_1}$ at $\id_{\psi_1}, A_\psi_1$ and $A_\psi_1 \iff A^\Psi_{\psi_1}$, we know $\tau^\Psi(\Psi_2, A_1, A_1', A_\psi_1, A_\psi_2, \Phi)$ where $A_\psi_1 \downarrow A_1$. By $A^\Psi_{\psi_1} \equiv (A^\Psi_{\psi_1})_\psi_1$ type $\pre_{\Psi_1}$, we have $(A^\Psi_{\psi_1})_\psi_1 \equiv A^\Psi_{\psi_1}$ type $\pre_{\Psi_1}$ and thus $\tau^\Psi(\Psi_2, A_\psi_1, A_\psi_2, \Phi)$. The result follows by transitivity of $\tau$ and $A_\psi_1 \iff A^\Psi_{\psi_1}$. 

2. $\tau^\Psi_\psi(\Psi_2, A_1, A_1', (A^\Psi_{\psi_2})_\psi_1, A_\psi_2, \Phi')$. 

By $A^\Psi_{\psi_2} \equiv (A^\Psi_{\psi_1})_\psi_2$ type $\pre_{\Psi_2}$ we have $\tau^\Psi(\Psi_2, A_\psi_2, (A^\Psi_{\psi_2})_\psi_2, \Phi')$; the result follows by $A_\psi_2 \downarrow A_1'$. 

3. $(A^\Psi_{\psi_1})_\psi_1 \downarrow A_1'$ and $\tau^\Psi_\psi(\Psi_2, A_1, A_1', A_\psi_1, A_\psi_2, \Phi'')$. 

By $A^\Psi_{\psi_1}$ type $\pre_{\Psi_1}$. 

Lemma 4.18 (Coherent expansion II). Suppose $A$ is a term, and $\{A^\Psi_{\psi_1}\}_{\psi_1: \Psi_1 \to \Psi}$ is a family of terms such that $M^\Psi_{\psi_1} \equiv (M^\Psi_{\psi_1})_{\psi_1: \Psi_1 \to \Psi}$ and $M_\psi \iff M^\Psi_{\psi_1}$ for all $\psi : \Psi \to \Psi$. Then $M \equiv M^\Psi_{\psi_1} \in A \[\Psi\]$. 

Proof. It suffices to check for all $\psi_1 : \Psi_1 \to \Psi$ and $\psi_2 : \Psi_2 \to \Psi_1$ that $M_\psi_1 \downarrow M_1$, $(M^\Psi_{\psi_1})_\psi_1 \downarrow M_1'$, and $[\Psi]$ relates $M_\psi_1, M_\psi_2, (M^\Psi_{\psi_1})_\psi_1, M_\psi_2$ and $M_\psi_1$. 

1. $M_\psi_1 \downarrow M_1$ and $[\Psi]$ relates $M_\psi_1, M_\psi_2$.

By $M^\Psi_{\psi_1} \equiv (M^\Psi_{\psi_1})_\psi_1$ at $\id_{\psi_1}, A_\psi_1$ and $M_\psi \iff M^\Psi_{\psi_1}$, we know $[\Psi]$ relates $M^\Psi_{\psi_1}, M^\Psi_{\psi_2}$. Where $M_\psi_1 \downarrow M_1$. By $M^\Psi_{\psi_1} \equiv (M^\Psi_{\psi_1})_\psi_1$ at $\Psi_1$ and $(M^\Psi_{\psi_1})_\psi_1 \equiv M^\Psi_{\psi_2} \in A_\psi_1$. We have $(M^\Psi_{\psi_1})_\psi_1 \equiv M^\Psi_{\psi_2} \in A_\psi_1$ and thus $[\Psi]$ relates $M^\Psi_{\psi_1}, M^\Psi_{\psi_2}$. The result follows by transitivity of $\tau$ and $M_\psi_1 \iff M^\Psi_{\psi_1}$. 

2. $[\Psi]$ relates $M^\Psi_{\psi_1}, M^\Psi_{\psi_2}$.

By $M^\Psi_{\psi_1} \equiv (M^\Psi_{\psi_1})_\psi_1$ at $\Psi_2$ we have $[\Psi]$ relates $M^\Psi_{\psi_1}, M^\Psi_{\psi_2}$; the result follows by $M_\psi_1 \iff M^\Psi_{\psi_1}$. 

(7/18 Cubical judgments and Kan operations 91)
3. \((M'_{\text{id}_{\psi}})_{\psi_1} \downarrow M'_{\psi_1}^I\) and \(\llbracket A \rrbracket_{\psi_1, \psi_2}^{\psi_1, \psi_2} ((M'_{\text{id}_{\psi}})_{\psi_1} \psi_2, M'_{\psi_1} \psi_2)\).

By \(M'_{\text{id}_{\psi}} \in A[\Psi]\).

In the special case that \(M \mapsto M' \) and \(M' \in A[\Psi]\), we can invoke Lemma 4.18 with \(M\) and the family \(\{M'\psi\}_{\psi: \Psi} \rightarrow \Psi\), because \(M\psi \in A[\Psi']\) (where \(M\psi = (M' \text{id}_{\psi})\psi\) on the nose) and \(M\psi \mapsto M'\psi\) for all \(\psi: \Psi' \rightarrow \Psi\). We therefore obtain head expansion with respect to \textit{cubically-stable} evaluation, as we observe in Lemma 4.32.

### 4.3.1 Open judgments

As in Idealized Nuprl, open judgments express that an equality holds as a closed judgment \(M\) and the family open judgments satisfy the structural rules. It remains to prove that contexts can be extended on the right by open types, and that the De/f_inition open judgments are symmetric, transitive, and closed under interval substitution.\(\Box\)

\(\Box\) for all instantiations by elements of the hypotheses \(\Gamma\). Our notation \(\Gamma \Rightarrow J[\Psi]\) distinguishes the contexts \(\Gamma\) and \(\Psi\) in accordance with their differing semantics. One can instead interleave term \((a : A)\) and interval \((x : I)\) variables in a single context; in fact, all current cubical proof assistants do so, including RedPRL, redtt, and cubical Agda.

**Definition 4.19.** For \(\gamma, \gamma'\) lists and \(\Gamma\) a telescope (Definition 2.17), \(\gamma \sim \gamma' \in \Gamma[\Psi]\) when

1. \(\cdot \sim \cdot \in [\Psi]\), or
2. \((M, \gamma) \sim (M', \gamma') \in (a : A, \Gamma)[\Psi]\) when \(M \equiv M' \in A[\Psi]\) and \(\gamma \sim \gamma' \in \Gamma[M/a][\Psi]\).

**Definition 4.20** (Contexts). For \(\Gamma, \Gamma'\) telescopes, \(\Gamma \equiv \Gamma'\) \(\text{ctx}[\Psi]\) when

1. \(\cdot \equiv \cdot \in \text{ctx}[\Psi]\), or
2. \((a : A, \Gamma) \equiv (a : A', \Gamma') \text{ ctx}[\Psi]\) when \(A \equiv A' \text{ type}_{\text{pre}}[\Psi]\) and for all \(\psi: \Psi' \rightarrow \Psi\) and \(M \equiv M' \in A[\Psi'][\Gamma], \Gamma[\Psi[M/a]] \equiv \Gamma'[\psi[M'/a]] \text{ ctx}[\Psi']\).

The relation \(\gamma \sim \gamma' \in \Gamma[\Psi]\) is symmetric and transitive when \(\Gamma\) \(\text{ctx}[\Psi]\); the judgment \(\Gamma \equiv \Gamma'\) \(\text{ctx}[\Psi]\) is always symmetric and transitive. Both are closed under interval substitution as a straightforward consequence of Lemma 4.13.

**Definition 4.21** (Open judgments).

1. \(\Gamma \Rightarrow A \equiv A' \text{ type}_{\text{pre}}[\Psi]\), presupposing \(\Gamma \text{ ctx}[\Psi]\), when for all \(\psi: \Psi' \rightarrow \Psi\) and \(\gamma \sim \gamma' \in \Gamma[\psi][\Psi'], A[\psi] \gamma \equiv A'[\psi] \gamma' \text{ type}_{\text{pre}}[\Psi']\).
2. \(\Gamma \Rightarrow M \equiv M' \in A[\Psi]\), presupposing \(\Gamma \Rightarrow A \text{ type}_{\text{pre}}[\Psi]\), when for all \(\psi: \Psi' \rightarrow \Psi\) and \(\gamma \sim \gamma' \in \Gamma[\psi][\Psi'], M[\psi] \gamma \equiv M'[\psi] \gamma' \in A[\psi][\Psi']\).

The open judgments are symmetric, transitive, and closed under interval substitution. It remains to prove that contexts can be extended on the right by open types, and that the open judgments satisfy the \textit{structural rules} of hypothesis, weakening, and substitution.
Lemma 4.22 (Context extension). If $\Gamma \vdash \Gamma'$ ct $\Psi$ and $\Gamma \Rightarrow A \vdash A'$ type $\Psi$ then $(\Gamma, a : A) \vdash (\Gamma', a : A')$ ct $\Psi$.

Proof. The telescope $(\Gamma, a : A)$ is defined by recursion on $\Gamma$, so we prove the lemma by induction on $\Gamma \vdash \Gamma'$ ct $\Psi$. If $\Gamma = \Gamma' = \cdot$, we must show that if $A\psi \vdash A'\psi$ type $\Psi$ for all $\psi : \Psi' \rightarrow \Psi$ then $A \vdash A'$ type $\Psi$, which is immediate.

Now suppose $(b : B, \Gamma) \vdash (b : B', \Gamma')$ ct $\Psi$ and $b : B, \Gamma \Rightarrow A \vdash A'$ type $\Psi$, and show $((b : B, \Gamma), a : A) \vdash ((b : B', \Gamma'), a : A')$ ct $\Psi$. By definition, $((b : B, \Gamma), a : A) = (b : B, (\Gamma, a : A))$. We know $B \vdash B'$ type $\Psi$, and must show for any $\psi : \Psi' \rightarrow \Psi$ and $N \doteq N' \in B\psi[\Psi']$ that $(\Gamma\psi[N/b], a : A\psi[N/b]) \vdash (\Gamma'\psi[N'/b], a : A'\psi[N'/b])$ ct $\Psi$.

It suffices to establish $\Gamma\psi[N/b] \Rightarrow A\psi[N/b] \doteq A'\psi[N'/b]$ type $\Psi$; the result then follows from the inductive hypothesis and $\Gamma\psi[N/b] \doteq \Gamma'\psi[N'/b]$ ct $\Psi$. We must show for all $\psi' : \Psi'' \rightarrow \Psi'$ and $\gamma \approx \gamma' \in \Gamma\psi[N/b]\psi'[\Psi'']$ that $A\psi[N/b]\psi'[\Psi']$ is a type $\Psi''$. This follows from $b : B, \Gamma \Rightarrow A \vdash A'$ type $\Psi$ instantiated at $\psi' \rightarrow \Psi'$ and $(N\psi', \gamma) \sim (N'\psi', \gamma') \in (b : B, \Gamma)\psi'[\Psi'']$.

Lemma 4.23 (Hypothesis). If $(\Gamma, a : A, \Delta) \text{ct} \Psi$ then $\Gamma, a : A, \Delta \Rightarrow a \in A \Psi$.

Proof. First, we establish the presupposition $(\Gamma, a : A, \Delta) \text{ct} \Psi$ by induction on $\Gamma$, unrolling the definition of $(\Gamma, a : A, \Delta) \text{ct} \Psi$, gathering iterated interval substitutions (as in the previous proof), and observing that $a$ and the variables of $\Delta$ cannot occur in $A$. To establish the judgment itself, suppose $\psi : \Psi' \rightarrow \Psi$ and $(\gamma, M, \delta) \sim (\gamma', M', \delta') \in (\Gamma, a : A, \Delta)\psi \Psi'$ (hence $\gamma \sim \gamma' \in \Gamma\psi \Psi'$ and $M \doteq M' \in A\psi[\Psi']$); we must show $M \doteq M' \in A\psi \Psi'$, which is immediate.

Lemma 4.24 (Weakening). Supposing $\Gamma \Rightarrow A$ type $\Psi$:

1. If $\Gamma, \Delta \Rightarrow B \doteq B'$ type $\Psi$ then $\Gamma, a : A, \Delta \Rightarrow B \doteq B'$ type $\Psi$.

2. If $\Gamma, \Delta \Rightarrow M \doteq M' \in B \Psi$ then $\Gamma, a : A, \Delta \Rightarrow M \doteq M' \in B \Psi$.

Proof. For part (1), as previously, we establish the presupposition $(\Gamma, a : A, \Delta) \text{ct} \Psi$ by induction on $\Gamma$, observing for all $\psi : \Psi' \rightarrow \Psi$ and $\gamma \sim \gamma' \in \Gamma\psi \Psi'$ that $A\psi \gamma \doteq A\psi \gamma'$ type $\Psi$ and $\Delta\psi \gamma \doteq \Delta\psi \gamma'$ ct $\Psi$. To establish the judgment, suppose $(\gamma, M, \delta) \sim (\gamma', M', \delta') \in (\Gamma, a : A, \Delta)\psi \Psi'$ and show $B\psi \gamma[M/a]\delta \doteq B\psi \gamma'[M'/a]\delta'$ type $\Psi$. Because $a$ cannot occur in $B$, this is exactly $B\psi \gamma \delta \doteq B\psi \gamma' \delta'$ type $\Psi$, which is immediate. Part (2) follows similarly.

Lemma 4.25 (Substitution). Supposing $\Gamma \Rightarrow M \doteq M' \in A \Psi$:

1. If $\Gamma, a : A, \Delta \Rightarrow B \doteq B'$ type $\Psi$ then $\Gamma, \Delta[M/a] \Rightarrow B[M/a] \doteq B'[M'/a]$ type $\Psi$.

2. If $\Gamma, a : A, \Delta \Rightarrow N \doteq N' \in B \Psi$ then $\Gamma, \Delta[M/a] \Rightarrow N[M/a] \doteq N'[M'/a] \in B[M/a] \Psi$. 
4.3.2 Kan operations

As we explained in Chapter 3, cubical type theories are motivated by two key insights. First, interval variables allow us to generalize the typehood and membership judgments to arbitrary dimension, and thereby represent the arbitrary-dimensional data induced by univalence and higher inductive types. Secondly, uniform Kan operations equip these types with structure that in Book HoTT is induced by identity elimination: composition and inversion of paths, transport, *et cetera*.

We therefore define a new judgment expressing that a type is Kan, or equipped with the uniform Kan operations of coercion and homogeneous composition described in Section 3.2. This judgment has similar force to the typehood judgment of Cohen et al. [CCHM18], in whose type theory all types are Kan. In contrast, we explicitly consider pretypes that are not Kan, including many equality types.

Like Cohen et al. [CCHM18], we express open boxes using *restricted judgments*, which limit the force of a cubical judgment to a subcube of $\Psi$ [Ang+19].

**Definition 4.26.** An interval substitution $\psi : \Psi' \to \Psi$ satisfies a set of unordered equational constraints $\Xi = \{r_1 = r'_1, \ldots, r_n = r'_n\}$ for which $\text{Fl}(\Xi) \subseteq \Psi$ when for all $i$, $r_i \psi = r'_i \psi$.

**Definition 4.27 (Restricted judgments).** For every judgment form $\mathcal{F}$, we say $\mathcal{F} [\Psi | \Xi]$, presupposing $\text{Fl}(\Xi) \subseteq \Psi$, when $\mathcal{F}[\psi][\Psi']$ for all $\psi : \Psi' \to \Psi$ satisfying $\Xi$.

Restricted judgments maintain the presuppositions of their unrestricted forms; for instance, $M \in A [\Psi | \Xi]$ presupposes $A \text{type}_{\text{pre}}[\Psi | \Xi]$. (This presupposition is needed to ensure the presupposition of $M\psi \in A\psi [\Psi']$ for all $\psi : \Psi' \to \Psi$ satisfying $\Xi$.) One can translate any $\mathcal{F} [\Psi | \Xi]$ into an unrestricted judgment by case analysis on $\Xi$:

1. $\mathcal{F} [\Psi | \cdot]$ if and only if $\mathcal{F} [\Psi]$.
   
   Both are equivalent to $\mathcal{F}[\psi][\Psi']$ for all $\psi$, because all interval substitutions satisfy $\cdot$, and all judgment forms are closed under interval substitution.

2. $\mathcal{F} [\Psi | \Xi, r = r]$ if and only if $\mathcal{F} [\Psi | \Xi]$.
   
   An interval substitution satisfies $\{\Xi, r = r\}$ if and only if it satisfies $\Xi$.

**Proof.** For part (1), once again we establish the presupposition $(\Gamma, \Delta[M/a]) \text{ctx} [\Psi]$ by induction on $\Gamma$, observing for all $\psi : \Psi' \to \Psi$ and $\gamma \sim \gamma' \in \Gamma\psi [\Psi']$ that $(\gamma, M\psi\gamma) \sim (\gamma', M\psi\gamma') \in (\Gamma, a : A)\psi [\Psi']$ and thus $(\Delta[M/a])\psi\gamma = (\Delta[M/a])\psi\gamma' \text{ ctx} [\Psi']$ (because $(\Delta[M/a])\psi\gamma = \Delta\psi\gamma[M\psi\gamma/a]$). To establish the judgment, show if $(\gamma, \delta) \sim (\gamma', \delta') \in (\Gamma, \Delta[M/a])\psi [\Psi']$ then $B[M/a]\psi\gamma\delta = B'[M'/a]\psi\gamma'\delta' \text{ type}_{\text{pre}}[\Psi']$, which follows by $B[M/a]\psi\gamma\delta = B\psi\gamma[M\psi\gamma/a]\delta$ and $(\gamma, M\psi\gamma, \delta) \sim (\gamma', M\psi\gamma', \delta') \in (\Gamma, a : A, \Delta)\psi [\Psi']$. Part (2) follows similarly. □

### 4.3.2.3 Kan operations

As we explained in Chapter 3, cubical type theories are motivated by two key insights. First, interval variables allow us to generalize the typehood and membership judgments to arbitrary dimension, and thereby represent the arbitrary-dimensional data induced by univalence and higher inductive types. Secondly, uniform Kan operations equip these types with structure that in Book HoTT is induced by identity elimination: composition and inversion of paths, transport, *et cetera*.

We therefore define a new judgment expressing that a type is Kan, or equipped with the uniform Kan operations of coercion and homogeneous composition described in Section 3.2. This judgment has similar force to the typehood judgment of Cohen et al. [CCHM18], in whose type theory all types are Kan. In contrast, we explicitly consider pretypes that are not Kan, including many equality types.

Like Cohen et al. [CCHM18], we express open boxes using *restricted judgments*, which limit the force of a cubical judgment to a subcube of $\Psi$ [Ang+19].

**Definition 4.26.** An interval substitution $\psi : \Psi' \to \Psi$ satisfies a set of unordered equational constraints $\Xi = \{r_1 = r'_1, \ldots, r_n = r'_n\}$ for which $\text{Fl}(\Xi) \subseteq \Psi$ when for all $i$, $r_i \psi = r'_i \psi$.

**Definition 4.27 (Restricted judgments).** For every judgment form $\mathcal{F}$, we say $\mathcal{F} [\Psi | \Xi]$, presupposing $\text{Fl}(\Xi) \subseteq \Psi$, when $\mathcal{F}[\psi][\Psi']$ for all $\psi : \Psi' \to \Psi$ satisfying $\Xi$.

Restricted judgments maintain the presuppositions of their unrestricted forms; for instance, $M \in A [\Psi | \Xi]$ presupposes $A \text{type}_{\text{pre}}[\Psi | \Xi]$. (This presupposition is needed to ensure the presupposition of $M\psi \in A\psi [\Psi']$ for all $\psi : \Psi' \to \Psi$ satisfying $\Xi$.) One can translate any $\mathcal{F} [\Psi | \Xi]$ into an unrestricted judgment by case analysis on $\Xi$:

1. $\mathcal{F} [\Psi | \cdot]$ if and only if $\mathcal{F} [\Psi]$.
   
   Both are equivalent to $\mathcal{F}[\psi][\Psi']$ for all $\psi$, because all interval substitutions satisfy $\cdot$, and all judgment forms are closed under interval substitution.

2. $\mathcal{F} [\Psi | \Xi, r = r]$ if and only if $\mathcal{F} [\Psi | \Xi]$.
   
   An interval substitution satisfies $\{\Xi, r = r\}$ if and only if it satisfies $\Xi$. 

**Proof.** For part (1), once again we establish the presupposition $(\Gamma, \Delta[M/a]) \text{ctx} [\Psi]$ by induction on $\Gamma$, observing for all $\psi : \Psi' \to \Psi$ and $\gamma \sim \gamma' \in \Gamma\psi [\Psi']$ that $(\gamma, M\psi\gamma) \sim (\gamma', M\psi\gamma') \in (\Gamma, a : A)\psi [\Psi']$ and thus $(\Delta[M/a])\psi\gamma = (\Delta[M/a])\psi\gamma' \text{ ctx} [\Psi']$ (because $(\Delta[M/a])\psi\gamma = \Delta\psi\gamma[M\psi\gamma/a]$). To establish the judgment, show if $(\gamma, \delta) \sim (\gamma', \delta') \in (\Gamma, \Delta[M/a])\psi [\Psi']$ then $B[M/a]\psi\gamma\delta = B'[M'/a]\psi\gamma'\delta' \text{ type}_{\text{pre}}[\Psi']$, which follows by $B[M/a]\psi\gamma\delta = B\psi\gamma[M\psi\gamma/a]\delta$ and $(\gamma, M\psi\gamma, \delta) \sim (\gamma', M\psi\gamma', \delta') \in (\Gamma, a : A, \Delta)\psi [\Psi']$. Part (2) follows similarly. □
3. \( \mathcal{J} \left[ \Psi \mid \Xi, 0 = 1 \right] \) always.
No interval substitutions satisfy \( \{ \Xi, 0 = 1 \} \).

4. \( \mathcal{J} \left[ \Psi, x \mid \Xi, x = r \right] \) if and only if \( \mathcal{J} \langle r/x \rangle \left[ \Psi \mid \Xi \langle r/x \rangle \right] \).

An interval substitution \( \psi : \Psi' \to (\Psi, x) \) satisfies \( \{ \Xi, x = r \} \) if and only if \( \psi = \langle r/x \rangle \psi' \) and \( \psi' : \Psi' \to \Psi \) satisfies \( \Xi \langle r/x \rangle \).

Restrictions are nevertheless critical because they specify subcubes in a manner that is both closed under interval substitution (because \( \psi' \) satisfies \( \Xi \psi \) if and only if \( \psi \psi' \) satisfies \( \Xi \)) and amenable to intersection. To illustrate the latter point, consider three lines in the \( x = 0, x = 1, \) and \( x = y \) instances of \( A \) type_{\text{pre}} [x, y] that agree on their intersections. Using restricted judgments, we can express this scenario with three singly-restricted elements and three doubly-restricted equations (below, left). Using interval substitutions, it is easy to express the three lines themselves, but their intersections are not computed uniformly—the intersection of \( x = 0 \) and \( x = 1 \) is empty, while the intersection of \( x = \varepsilon \) and \( x = y \) corresponds to the substitution \( \langle \varepsilon/x \rangle \langle \varepsilon/y \rangle \) (below, right):

\[
\begin{array}{c|c|c}
M \in A [x, y \mid x = 0] & M \in A \langle 0/x \rangle [y] \\
N \in A [x, y \mid x = 1] & N \in A \langle 1/x \rangle [y] \\
O \in A [x, y \mid x = y] & O \in A \langle y/x \rangle [y] \\
M \upmodels N \in A [x, y \mid x = 0, x = 1] & M(0/y) \upmodels O(0/y) \in A \langle 0/x \rangle (0/y) [\cdot] \\
M \upmodels O \in A [x, y \mid x = 0, x = y] & N(1/y) \upmodels O(1/y) \in A \langle 1/x \rangle (1/y) [\cdot] \\
N \upmodels O \in A [x, y \mid x = 1, x = y] & -
\end{array}
\]

We write \( A \) type_{\text{Kan}} [\Psi] when \( A \) is a pretype whose interval substitution instances admit the Cartesian cubical Kan operations (Figure 3.4), restricted to valid open box shapes.\(^3\)

As discussed in Section 3.5, the validity condition is defeasible (Angiuli et al. [Ang+19] omit it) and engenders additional complications (Theorem 4.34), but allows us to obtain a sharper canonicity result in which 0-dimensional elements of higher inductive types evaluate to constructors (Theorem 4.77).

**Definition 4.28.** A set of equational constraints \( r_i = r'_i \) is valid when 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 4.29 (Kan types).** \( A \upmodels B \) type_{\text{Kan}} [\Psi], presupposing \( A \upmodels B \) type_{\text{pre}} [\Psi], when for any \( \psi : \Psi' \to \Psi \):

1. If

\(^3\)More precisely, we require not only that \( A \psi \) admit Kan operations but moreover that the particular terms \( \text{hcom}_{A \psi} \) and \( \text{coe}_{x.A \psi} \) define such operations.
Lemma 4.30. Suppose $A \trianglerighteq A' \text{ type}_{\text{kan}} [\Psi]$, $B \trianglerighteq B' \text{ type}_{\text{kan}} [\Psi]$, and for all $\psi : \Psi' \rightarrow \Psi$ we have $A^\psi \trianglerighteq A'^\psi \text{ type}_{\text{kan}} [\Psi']$ where $A^\psi \trianglerighteq A'$ and $B^\psi \trianglerighteq B'$. Then $A \trianglerighteq B \text{ type}_{\text{kan}} [\Psi]$.

Proof. Let $\psi : \Psi' \rightarrow \Psi$. By Lemma 4.14, $A^\psi \trianglerighteq A'^\psi \text{ type}_{\text{pre}} [\Psi']$ and $B^\psi \trianglerighteq B'^\psi \text{ type}_{\text{pre}} [\Psi']$; therefore $A^\psi \trianglerighteq B^\psi \text{ type}_{\text{pre}} [\Psi']$, and in particular, the presupposition $A \trianglerighteq B \text{ type}_{\text{pre}} [\Psi]$ holds. Now, suppose

1. $\longleftrightarrow \xi_i$ is valid,
2. $M \trianglerighteq M' \in A^\psi [\Psi']$,
3. $N_i \trianglerighteq N'_j \in A^\psi [\Psi', y | \xi_i, \xi_j]$ for any $i, j$, and
4. $N_i(r/y) \trianglerighteq M \in A^\psi [\Psi' | \xi_i]$ for any $i,$

then

a) $\trianglerighteq \xi_i$ is valid,

b) $M \trianglerighteq M' \in A^\psi [\Psi'],$

c) $N_i \trianglerighteq N'_j \in A^\psi [\Psi', y | \xi_i, \xi_j]$ for any $i, j,$ and

d) $N_i(r/y) \trianglerighteq M \in A^\psi [\Psi' | \xi_i]$ for any $i,$

2. If $\Psi' = (\Psi'', x)$ and $M \trianglerighteq M' \in A^\psi \langle r/x \rangle [\Psi''],$ then

a) $\trianglerighteq \xi_i$ is valid,

b) if $r = r'$ then $\trianglerighteq \xi_i$ is valid,

c) if $\trianglerighteq \xi_i$ is valid,

then

a) $\trianglerighteq \xi_i$ is valid,

b) if $r = r'$ then $\trianglerighteq \xi_i$ is valid,

c) if $\trianglerighteq \xi_i$ is valid,
and show $hcom_{A_{\psi}^r}^{\bowtie r} (M; \xi_i \leftrightarrow y.N_i) \equiv hcom_{B_{\psi}^r}^{\bowtie r} (M'; \xi_i \leftrightarrow y.N_i') \in A_{\psi} \[\Psi\]$. Both sides of this equation are elements of $A_{\psi}$ (by Definition 4.29 and $A_{\psi} \equiv B_{\psi}^r$ type$_{\text{pre}} [\Psi']$), so by Lemma 4.15 it suffices to check that these terms are related by $\llbracket A_{\psi} \rrbracket$ or equivalently $\llbracket A_{\psi} \rrbracket$. This holds by $hcom_{A_{\psi}^r} \equiv hcom_{B_{\psi}^r} \in A_{\psi} \[\Psi\]$ (using $A_{\psi} \equiv B_{\psi}^r$ type$_{\text{Kan}} [\Psi]$) and, for the premises, $A_{\psi} \equiv A_{\psi}^r$ type$_{\text{pre}} [\Psi']$, $hcom_{A_{\psi}^r} \equiv hcom_{A_{\psi}^r}$, and $hcom_{B_{\psi}^r} \equiv hcom_{B_{\psi}^r}$.

The remaining hcom equations of Definition 4.29 follow by transitivity and the equations of $A_{\psi}$ type$_{\text{Kan}} [\Psi']$; the coe equations follow similarly.

\[\blacksquare\]

**Lemma 4.31** (Coherent expansion III). Suppose $A$ is a term and $\{A_{\psi}^{w} \}_{\psi : \Psi' \to \Psi}$ is a family of terms such that $A_{\psi}^{w} \equiv (A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}$ type$_{\text{Kan}} [\Psi]$ and $A_{\psi} \equiv A_{\psi}^{w}$ for all $\psi : \Psi' \to \Psi$. Then $A_{\psi} \equiv A_{\psi_{\text{id}}}^{w}$ type$_{\text{Kan}} [\Psi]$.

\[\text{Proof.}\] The presupposition $A_{\psi} \equiv A_{\psi_{\text{id}}}^{w}$ type$_{\text{pre}} [\Psi]$ is immediate by Lemma 4.17. Now, suppose $\psi : \Psi' \to \Psi$,

1. $\xi_i$ is valid,
2. $M \equiv M' \in A_{\psi} \[\Psi']$,
3. $N_i \equiv N_i' \in A_{\psi} \[\Psi', y \mid \xi_i, \xi_j\]$ for any $i, j$, and
4. $N_i (r/y) \equiv M \in A_{\psi} \[\Psi' \mid \xi_i\]$ for any $i$,

and show $hcom_{A_{\psi}^{w}}^{\bowtie r} (M; \xi_i \leftrightarrow y.N_i) \equiv hcom_{A_{\psi_{\text{id}}}^{w}}^{\bowtie r} (M'; \xi_i \leftrightarrow y.N_i') \in A_{\psi} \[\Psi']$. We apply Lemma 4.18 at $A_{\psi} \equiv A_{\psi_{\text{id}}}^{w}$ type$_{\text{pre}} [\Psi']$ to the term $hcom_{A_{\psi}^{w}}^{\bowtie r} (M; \xi_i \leftrightarrow y.N_i)$ and family of terms

\[
\{ hcom_{A_{\psi_{\text{id}}}^{w}}^{\bowtie r} (M; \xi_i \leftrightarrow y.N_i') \}_{\psi'}^{\Psi''}.
\]

We know $hcom_{A_{\psi}^{w}} \equiv hcom_{A_{\psi_{\text{id}}}^{w}}$, $hcom_{A_{\psi}^{w}} \equiv hcom_{(A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}}$, and $hcom_{A_{\psi_{\text{id}}}^{w}} \equiv hcom_{(A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}}$ in $A_{\psi} \[\Psi'']$ by $A_{\psi_{\text{id}}}^{w} \equiv (A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}$ type$_{\text{Kan}} [\Psi'']$ and $A_{\psi_{\text{id}}}^{w} \equiv (A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}$ type$_{\text{pre}} [\Psi'']$ (both by transitivity through $(A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}$). We conclude that $hcom_{A_{\psi}^{w}} \equiv hcom_{A_{\psi_{\text{id}}}^{w}}$ in $A_{\psi} \[\Psi']$, and the desired result follows by $A_{\psi}^{w} \equiv (A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}$ type$_{\text{Kan}} [\Psi']$. The remaining hcom equations of Definition 4.29 follow by transitivity and $A_{\psi_{\text{id}}}^{w}$ type$_{\text{Kan}} [\Psi]$.

Now, suppose $\psi : (\Psi', x) \to \Psi$ and $M \equiv M' \in A_{\psi} \langle r/x \rangle \[\Psi]\prime \equiv A_{\psi} \langle r/x \rangle \[\Psi']$ and show $\coe_{x.A_{\psi}^{w}}^{\bowtie r} (M) \equiv \coe_{x.A_{\psi_{\text{id}}}^{w}} \langle r/x \rangle (M') \in A_{\psi} \langle r/x \rangle \[\Psi']$. We apply Lemma 4.18 at $A_{\psi} \langle r/x \rangle$ type$_{\text{pre}} [\Psi']$ to the term $\coe_{x.A_{\psi}^{w}}^{\bowtie r} (M)$ and family of terms $\{ \coe_{x.A_{\psi_{\text{id}}}^{w}}^{\bowtie r} \langle r/x \rangle (M') \}_{\psi'}^{\Psi''}$. As above, we obtain $\coe_{x.A_{\psi}^{w}} \equiv \coe_{x.A_{\psi_{\text{id}}}^{w}} \equiv A_{\psi}^{w} \equiv (A_{\psi_{\text{id}}}^{w})_{\psi \to \Psi}$ type$_{\text{Kan}} [\Psi', x]$. The remaining coe equation of Definition 4.29 follows similarly. \[\blacksquare\]
Lemma 4.32 (Head expansion).

1. If $A'$ type$_{pre}$ [$\Psi$] and $A \mapsto^{\xi} A'$ then $A = A'$ type$_{pre}$ [$\Psi$].
2. If $M' \in A$ [$\Psi$] and $M \mapsto^{\xi} M'$ then $M = M'$ $\in A$ [$\Psi$].
3. If $A'$ type$_{Kan}$ [$\Psi$] and $A \mapsto^{\xi} A'$ then $A = A'$ type$_{Kan}$ [$\Psi$].

Proof. Part (1) follows from Lemma 4.17 applied to the term $A$ and family $\{A'\psi\}_{\psi}$, observing that $A'\psi$ type$_{pre}$ [$\Psi'$] and $A\psi \mapsto^{*} A'\psi$ for all $\psi$. Parts (2–3) follow similarly, appealing instead to Lemma 4.18 and Lemma 4.31 respectively.

Using homogeneous composition, which fills open boxes in constant types, and coercion, which varies the type of a single element, we can define a heterogeneous composition that fills open boxes in a varying type. In fact, we have already seen an instance of heterogeneous composition in our definition of coercion in path types (Section 3.2).

Theorem 4.33 (Heterogeneous composition). If $A \equiv B$ type$_{Kan}$ [$\Psi$, $y$],

1. $\bar{\xi}_i$ is valid,
2. $M \equiv M' \in A(\langle r/y \rangle)$ [$\Psi$],
3. $N_i \equiv N'_j \in A[\Psi, y | \xi_i, \xi_j]$ for any $i, j$, and
4. $N_i(\langle r/y \rangle) \equiv M \in A(\langle r/y \rangle)$ [$\Psi | \xi_i$] for any $i$,

then

1. $\text{com}^{y_{\sim r} r'}_{y,A}(M; \bar{\xi}_i \mapsto y.N_i) \equiv \text{com}^{y_{\sim r} r'}_{y,B}(M'; \bar{\xi}_i \mapsto y.N'_i) \in A(\langle r'/y \rangle)$ [$\Psi$];
2. if $r = r'$ then $\text{com}^{y_{\sim r} r'}_{y,A}(M; \bar{\xi}_i \mapsto y.N_i) \equiv M \in A(\langle r/y \rangle)$ [$\Psi$]; and
3. if $\xi_i$ then $\text{com}^{y_{\sim r} r'}_{y,A}(M; \bar{\xi}_i \mapsto y.N_i) \equiv N_i(\langle r'/y \rangle) \in A(\langle r'/y \rangle)$ [$\Psi$].

Proof. By hypothesis (3), $N_i \psi \equiv N'_j \psi \in A\psi$ [$\Psi'$] for all $\psi : \Psi' \rightarrow (\Psi, y)$ satisfying $\xi_i$ and $\xi_j$. By $A \equiv B$ type$_{Kan}$ [$\Psi$, $y$] we have $(\text{co}e^y_{y,A}(N_i))\psi \equiv (\text{co}e^y_{y,B}(N'_i))\psi \in A(\langle r'/y \rangle)\psi$ [$\Psi'$], and therefore $\text{co}e^y_{y,A}(N_i) \equiv \text{co}e^y_{y,B}(N'_i) \in A[\Psi, y | \xi_i, \xi_j]$. Similarly, by hypothesis (4) we conclude $(\text{co}e^y_{y,A}(N_i))(\langle r/y \rangle) \equiv \text{co}e^y_{y,A}(M) \in A(\langle r'/y \rangle)$ [$\Psi | \xi_i$], and by hypothesis (2), $\text{co}e^y_{y,A}(M) \equiv \text{co}e^y_{y,B}(M') \in A(\langle r'/y \rangle)$ [$\Psi$]. Therefore, by hypothesis (1) and
A \doteq B \text{ type}_{\text{Kan}} [\Psi, y]:

\[
\text{hcom}_{A(r'/y)}(\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(M); \xi_i \leftrightarrow y.\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(N_i)) \doteq \text{hcom}_{B(r'/y)}(\text{coe}_{y,B}^{\sim_{\text{\text{\sim}}} r'}(M'); \xi_i \leftrightarrow y.\text{coe}_{y,B}^{\sim_{\text{\text{\sim}}} r'}(N'_i)) \in A(r'/y) [\Psi]
\]

Part (1) follows by Lemma 4.32 on each side. Moreover, when \( r = r' \),

\[
\text{hcom}_{A(r'/y)}(\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(M); \xi_i \leftrightarrow y.\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(N_i)) \doteq \text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(M)
\]

\[\doteq M \in A(r'/y) [\Psi]\]

and when \( \xi_i \),

\[
\text{hcom}_{A(r'/y)}(\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(M); \xi_i \leftrightarrow y.\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(N_i)) \doteq (\text{coe}_{y,A}^{\sim_{\text{\text{\sim}}} r'}(N_i))(r'/y)
\]

\[\doteq N_i(r'/y) \in A(r'/y) [\Psi].\]

Parts (2–3) follow from the above facts and Lemma 4.32.

Our definition of coercion in \( x.\text{hcom}_{q_{\text{Kan}}}(A; \xi \leftrightarrow z.B_i) \) requires filling open boxes of possibly-invalid shape—namely, the subset of \( \xi_i \) not containing \( x \) (Section 4.4.11). Such box fillers are not instances of homogeneous composition in the sense of Definition 4.29; surprisingly, however, they are definable by recursion on the list of equations.

**Theorem 4.34** (Generalized homogeneous composition). If \( A \doteq B \text{ type}_{\text{Kan}} [\Psi] \),

1. \( M \doteq M' \in A [\Psi] \),

2. \( N_i \doteq N'_i \in A [\Psi, y \mid \xi_i, \xi_j] \) for any \( i, j \), and

3. \( N_i(r/y) \doteq M \in A [\Psi \mid \xi_i] \) for any \( i \),

then

1. \( \text{ghcom}_{A}^{\sim_{\text{\text{\sim}}} r'}(M; \xi_i \leftrightarrow y.N_i) \doteq \text{ghcom}_{B}^{\sim_{\text{\text{\sim}}} r'}(M'; \xi_i \leftrightarrow y.N'_i) \in A [\Psi] \);

2. if \( r = r' \) then \( \text{ghcom}_{A}^{\sim_{\text{\text{\sim}}} r'}(M; \xi_i \leftrightarrow y.N_i) \doteq M \in A [\Psi] \); and

3. if \( \xi_i \) then \( \text{ghcom}_{A}^{\sim_{\text{\text{\sim}}} r'}(M; \xi_i \leftrightarrow y.N_i) \doteq N_i(r'/y) \in A [\Psi] \).

\(^4\)Following Cohen et al. [CCHM18], Angiuli et al. [Ang+19] notate this filtering operation \( \forall x \).
Proof. By induction on the length of $\xi_i$. If $\xi_i = \cdot$, part (1) requires $\text{ghcom}^{\sim r'}_A(M; \cdot) \doteq \text{ghcom}^{\sim r'}_B(M'; \cdot) \in A[\Psi]$, which is immediate by Lemma 4.32 on both sides; part (2) is immediate by Lemma 4.32 on the left; and part (3) is impossible.

Now consider $\text{ghcom}^{\sim r'}_A(M; s = s' \leftrightarrow y.N, \xi_i \leftrightarrow y.\bar{N}_i)$, armed with the inductive hypothesis that ghcoms with one fewer attached face satisfy properties (1–3). By Lemma 4.32 we must show (the binary form of)

$$
\text{hcom}^{\sim r'}_A(M; s = \varepsilon \leftrightarrow z.T, s = s' \leftrightarrow y.N, \xi_i \leftrightarrow y.\bar{N}_i) \in A[\Psi] \\
T_{\varepsilon} \doteq \text{hcom}^{\sim s'}_A(M; s' = \varepsilon \leftrightarrow y.N, s' = \bar{\varepsilon} \leftrightarrow y.\text{ghcom}^{\sim r''}_A(M; \xi_i \leftrightarrow y.\bar{N}_i), \bar{\xi}_i \leftrightarrow y.\bar{N}_i)
$$

writing $\bar{\varepsilon}$ for 1 when $\varepsilon = 0$ and vice versa. Let us first show $T_{\varepsilon} \in A[\Psi, z \mid s = \varepsilon]$ by Definition 4.29, noting that the composition is valid by $s' = \varepsilon, s' = \bar{\varepsilon}$ and that

1. $M \in A[\Psi \mid s = \varepsilon]$ by $M \in A[\Psi]$,

2. $N \in A[\Psi, y \mid s = \varepsilon, s' = \varepsilon]$ (by $N \in A[\Psi, y \mid s = s']$, because $s = s'$ whenever $s = \varepsilon, s' = \varepsilon$), $N \doteq N_i \in A[\Psi, y \mid s = \varepsilon, s' = \varepsilon, \bar{\xi}_i]$ (by $N \doteq N_i \in A[\Psi, y \mid s = s', \xi_i]$), and $N(r/y) \doteq M \in A[\Psi \mid s = \varepsilon, s' = \varepsilon]$ (by $N(r/y) \doteq M \in A[\Psi \mid s = s']$), and

3. $\text{ghcom}^{\sim s'}_A(M, \bar{\xi}_i \leftrightarrow y.\bar{N}_i) \in A[\Psi, y \mid s = \varepsilon, s' = \bar{\varepsilon}]$ by inductive hypothesis (1), $\text{ghcom}^{\sim s'}_A(M, \xi_i \leftrightarrow y.N_i) \in A[\Psi, y \mid s = \varepsilon, s' = \bar{\varepsilon}, \bar{\xi}_i]$ by inductive hypothesis (3), and $\text{ghcom}^{\sim s'}_A(r/y) \doteq M \in A[\Psi, y \mid s = \varepsilon, s' = \bar{\varepsilon}]$ by inductive hypothesis (2).

The remaining equations are immediate. To check $\text{hcom}^{\sim r'}_A \in A[\Psi]$ it suffices to observe that $T_{\varepsilon} \in A[\Psi, z \mid s = \varepsilon]$ (by the above); $T_{\varepsilon} \doteq N(z/y) \in A[\Psi, z \mid s = \varepsilon, s = s']$ (by the $s' = \varepsilon$ face of $T_{\varepsilon}$); $T_{\varepsilon} \doteq N(z/y) \in A[\Psi, z \mid s = \varepsilon, \xi_i]$ (by the $\xi_i$ face of $T_{\varepsilon}$); $T_{\varepsilon}\langle r/z \rangle \doteq M \in A[\Psi \mid s = \varepsilon]$ (by $r = z(r/z)$ in $T_{\varepsilon}$); and the $\bar{s} = \bar{\varepsilon}$ faces ensure the composition is valid. Part (1) follows by repeating this argument on the right side, and parts (2–3) follow by Definition 4.29. $\square$

Theorem 4.35 (Generalized heterogeneous composition). If $A \doteq B \in A[r/y]$, then

1. $M \doteq M' \in A(r/y)[\Psi]$,

2. $N_i \doteq N'_i \in A[\Psi, y \mid \xi_i, \bar{\xi}_i]$ for any $i, j$, and

3. $N_i\langle r/y \rangle \doteq M \in A(r/y)[\Psi] \mid \xi_i]$ for any $i$,

then

1. $\text{gcom}^{\sim r'}_{y,A}(M; \bar{\xi}_i \leftrightarrow y.\bar{N}_i) \doteq \text{gcom}^{\sim r'}_{y,B}(M'; \bar{\xi}_i \leftrightarrow y.\bar{N}_i) \in A\langle r'/y \rangle[\Psi]$;

2. if $r = r'$ then $\text{gcom}^{\sim r'}_{y,A}(M; \bar{\xi}_i \leftrightarrow y.\bar{N}_i) \doteq M \in A(r/y)[\Psi]$; and
3. if \( \xi_i \) then \( gcom_{y,A}^{\tau \mapsto r'}(M; \xi_i \mapsto y.N_i) \equiv N_i(r'/y) \in A(r'/y) \ [\Psi] \).

**Proof.** Our implementation of \( gcom \) by \( ghcom \) and \( coe \) mirrors exactly our implementation of \( com \) by \( hcom \) and \( coe \). The proof is therefore identical to that of Theorem 4.33, appealing to Theorem 4.34 instead of Definition 4.29. □

### 4.4 Semantics of types

We now establish the basic properties of each type former in Cartesian cubical type theory. Many of these properties are the standard rules of type theory (as in Figure 2.4) generalized to arbitrary \( \Psi \). Others govern novel type formers—path types, \( S^1 \), \( V \)-types, and formal composites of types—or specify when types are Kan.

Unlike in Section 2.5, formation and introduction rules do not follow directly from the definitions of Figure 4.3, but require value-coherence of each type’s underlying \( \Psi \)-PER. The Kan formation rules require us to verify that the operational semantics of \( hcom \) and \( coe \) at each type satisfy the conditions of Definition 4.29. We prove elimination and computation rules by coherent expansion, checking that the interval substitution instances of each elimination form compute coherently on introduction forms. Finally, uniqueness rules follow from induction on the value elements of each type, as in Idealized Nuprl. We prove only the closed, binary form of each rule, as one can mechanically extend these to open terms by commuting term substitutions past constructors, using the fact that judgments are closed under interval substitutions.

We present the rules of Cartesian cubical type theory in the appendices of this dissertation, both as a computational type theory (in the style of Nuprl or RedPRL) in Appendix A, and as an intensional type theory (in the style of Agda, Coq, or \texttt{redtt}) in Appendix B. (See Section 2.4 for a discussion of computational and intensional type theories.) The former exposes particularities of our computational semantics, while we expect the latter to admit denotational semantics in Cartesian cubical sets similar to those of Angiuli et al. [Ang+19]. The results of this section establish consistency (Theorem 4.58) of both type theories, as well as canonicity for the computational calculus (Theorems 4.63 and 4.77).

#### 4.4.1 Dependent functions

Recall from Figure 4.3 that for \( i \in \{0, 1, \ldots, \omega\} \) and \( \kappa \in \{\text{pre}, \text{Kan}\} \),

\[
\tau^\kappa_i(\Psi, (a:A) \to B, (a:A') \to B', \{(\lambda a.N, \lambda a.N') \mid a : A \triangleright N \equiv N' \in B [\Psi]\})
\]

if and only if \( A \sim A' \downarrow \alpha \in \tau^\kappa_i [\Psi] \), \( a : A \triangleright B \sim B' \downarrow \beta \in \tau^\kappa_i [\Psi], \text{Coh}(\alpha), \text{and CohFam}(\beta) \); expanding definitions, these in turn hold if and only if \( \tau^\kappa_i \models (A \triangleright A' \text{ type}_{\text{pre}} [\Psi]) \) and \( \tau^\kappa_i \models \)}
(a : A ⇒ B ≡ B’ type_{pre} [Ψ]). Moreover, (a:A) → B ∼ (a:A’) → B’ ⊣ [(a:A) → B] ∈ τ_i^κ [Ψ] because interval substitutions preserve judgments and (a:A) → B val_{Δ}.

In the remainder of this section, all judgments are relative to τ_i^κ. Relative to τ_{pre}^κ, Rule 4.36 is the ordinary formation rule for dependent functions. In our proof of Rule 4.103, however, we will invoke Rule 4.36 at arbitrary τ_i^κ in order to observe that every universe U_i^κ is closed under dependent functions.

**Rule 4.36 (Pre-type formation).** If A ≡ A’ type_{pre} [Ψ] and a : A ⇒ B ≡ B’ type_{pre} [Ψ] then (a:A) → B ≡ (a:A’) → B’ type_{pre} [Ψ].

**Proof.** We have already observed that (a:A) → B ∼ (a:A’) → B’ ⊣ [(a:A) → B] ∈ τ_i^κ [Ψ]; it remains to show Coh([(a:A) → B]). Suppose ψ : Ψ’ → Ψ and [(a:A) → B]_ψ(λa.N, λa.N’), in which case case a : Aψ ≡ N ≡ N’ ∈ Bψ [Ψ’]. We must show λa.N ∼ λa.N’ ∈ [(a:A) → B]_ψ [Ψ’]. By λa.N val_{Δ}, it suffices to see that for all ψ’ : Ψ” → Ψ’, [(a:A) → B]_ψ’(λa.Nψ’, λa.N’ψ’), which holds by a:Aψψ’ ≡ Nψ’≡N’ψ’ ∈ Bψψ’ [Ψ’]. □

Because we have defined the value elements of (a:A) → B according to its introduction principle, value-coherence amounts to the introduction rule itself.


**Proof.** The presuppositions of our hypothesis and Rule 4.36 imply (a:A) → B type_{pre} [Ψ]. The result follows by Coh([(a:A) → B]) at id_Ψ. □


**Proof.** The presupposition of our first hypothesis, (a:A) → B type_{pre} [Ψ], implies a : A ⇒ B type_{pre} [Ψ]; the presupposition B[N/a] type_{pre} [Ψ] follows immediately.

For all ψ : Ψ’ → Ψ, we know Mψ ⊣ λa.O_{ψ} and [(a:A) → B]_ψ(λa.O_{id_ψ}, λa.O_ψ), and therefore a : Aψ ≡ O_{id_ψ} ∈ Bψ [Ψ]. We apply coherent expansion (Lemma 4.18) to (M N), B[N/a] type_{pre} [Ψ], and {O_ψ[Nψ/a]}_ψ, using Mψ Nψ →* (λa.O_{ψ}) Nψ → O_ψ[Nψ/a] and O_ψ[Nψ/a] → (O_{id_ψ}[N/a])ψ ∈ Bψ[Nψ/a] [Ψ’]. We conclude that M N ≡ O_{id_ψ}[N/a] ∈ B[N/a] [Ψ], as desired. □

**Rule 4.39 (Elimination).** If M ≡ M’ ∈ (a:A) → B [Ψ] and N ≡ N’ ∈ A [Ψ] then M N ≡ M’ N’ ∈ B[N/a] [Ψ].

Rule 4.40 (Uniqueness). If $M \in (a:A) \rightarrow B[^\Psi]$ then $M \triangleq \lambda a.M \ a \in (a:A) \rightarrow B[^\Psi]$.

Proof. By Lemma 4.16, $M \Downarrow \lambda a.O$ and $M \triangleq \lambda a.O \in (a:A) \rightarrow B[^\Psi]$; by transitivity and Rule 4.37 it suffices to show $a : A \Rightarrow O \equiv M \ a \in B[^\Psi]$, that is, if $\psi : \Psi' \rightarrow \Psi$ and $N \equiv N' \in A\psi[^\Psi]$ then $O\psi[N/a] \equiv M\psi N' \in B\psi[N/a][^\Psi]$. By Lemma 4.38, $O\psi[N'/a] \equiv M\psi N' \in B\psi[N'/a][^\Psi]$, where $M\psi \Downarrow \lambda a.O_\psi$. The result follows by $B\psi[N/a] \equiv B\psi[N'/a]$ type$_\text{pre}[^\Psi]$ and $a : A\psi \Rightarrow O_{id_\psi} \equiv O_\psi \in B\psi[^\Psi']$, the latter by $[(a:A) \rightarrow B][\psi(\lambda a.O_\psi, \lambda a.O_\psi)]$.

Computation rules that correspond to cubically-stable evaluation steps follow directly from head expansion; as in Idealized Nuprl, we omit such proofs, and omit such rules from Appendix A. We include the following proof for illustrative purposes:

Rule 4.41 (Computation). If $a : A \Rightarrow M \in B[^\Psi]$ and $N \in A[^\Psi]$ then $(\lambda a.M) N \equiv M[N/a] \in B[N/a][^\Psi]$.

Proof. By Lemma 4.32, $M[N/a] \in B[N/a][^\Psi]$, and $(\lambda a.M) N \mathrel{\rightarrow_{\text{head}}} M[N/a]$.

Finally, we must show that the dependent function type of two Kan types is again Kan. In our proof of Rule 4.102, we will invoke Rule 4.42 in order to observe that the (inductively defined collection of) elements of $U_i^Kan$ are Kan types.

Rule 4.42 (Kan type formation). If $A \equiv A' \text{ type}_\text{Kan}[^\Psi]$ and $a : A \Rightarrow B \equiv B' \text{ type}_\text{Kan}[^\Psi]$ then $(a:A) \rightarrow B \equiv (a:A') \rightarrow B'$ type$_\text{Kan}[^\Psi]$.

Proof. Rule 4.36 establishes the presupposition $(a:A) \rightarrow B \equiv (a:A') \rightarrow B'$ type$_\text{pre}[^\Psi]$. It remains to check the Kan conditions of Definition 4.29; we begin with homogeneous composition. Suppose $\psi : \Psi' \rightarrow \Psi$, $\xi_i$ is valid,

1. $M' \in (a:A\psi) \rightarrow B\psi[^\Psi']$,
2. $N_i \equiv N'_i \in (a:A\psi) \rightarrow B\psi[^\Psi', y | \xi_i, \xi_j]$ for any $i, j$, and
3. $N_i(r/y) \equiv M \in (a:A\psi) \rightarrow B\psi[^\Psi' | \xi_i]$ for any $i$; show

$hcom^\rightarrow_{(a:A\psi) \rightarrow B\psi}(M; \xi_i \leftarrow y.N_i) \equiv hcom^\rightarrow_{(a:A\psi) \rightarrow B\psi}(M'; \xi_i \leftarrow y.N'_i) \in (a:A\psi) \rightarrow B\psi[^\Psi']$. By Lemma 4.33 on both sides and Rule 4.37, it suffices to show

$a : A\psi \Rightarrow hcom^{\rightarrow_{\rightarrow\rightarrow\psi}}_{B\psi}[M a; \xi_i \leftarrow y.N_i a] \equiv hcom^{\rightarrow_{\rightarrow\rightarrow\psi}}_{B\psi}(M' a; \xi_i \leftarrow y.N'_i a) \in B\psi[^\Psi']$

which is to say that for any $\psi' : \Psi'' \rightarrow \Psi'$ and $N' \equiv N' \in A\psi'\psi'[^\Psi'']$,

$hcom^{\rightarrow_{\rightarrow\rightarrow\psi}}_{B\psi\psi'[N/a]}(M\psi' N; \xi_i\psi' \leftarrow y.N_i\psi' N) \equiv hcom^{\rightarrow_{\rightarrow\rightarrow\psi}}_{B\psi\psi'[N'/a]}(M\psi' N'; \xi_i\psi' \leftarrow y.N_i\psi' N') \in B\psi\psi'[N/a][^\Psi''].$
We have \( B\psi\psi'[N/a] = B'\psi\psi'[N'/a] \) type\(_{\text{Kan}}\) \([\Psi'']\) by \( a : A \Rightarrow B \Rightarrow B' \) type\(_{\text{Kan}}\) \([\Psi]\), so the result follows from Definition 4.29 if we show that \( \xi_i\psi' \) is valid,

1. \( M\psi' N \cong M'\psi' N' \in B\psi\psi'[N/a] [\Psi''] \),

2. \( N_i\psi' N \cong N'_i\psi' N' \in B\psi\psi'[N/a] [\Psi'', y | \xi_i\psi', \xi_j\psi'] \) for any \( i, j \), and

3. \( N_i(r/y)\psi' N \cong M\psi' N' \in B\psi\psi'[N/a] [\Psi'' | \xi_i\psi'] \) for any \( i \).

These follow from our hypotheses and a restricted elimination rule (obtained from Rule 4.39 and Definition 4.27)—if \( M \cong M' \in (a:A) \to B [\Psi | \Xi] \) and \( N \cong N' \in A [\Psi | \Xi] \) then \( M N \cong M' N' \in B[N/a] [\Psi | \Xi] \).

Next, show that if \( r \cong r' \) then \( \text{hcom}^{_{(a:A') \to B\psi}}(M; \xi_i \mapsto y.N_i) \cong M \in (a:A') \to B\psi [\Psi'] \).

By Lemma 4.32 on the left and Rule 4.40 on the right, it suffices to show

\[
\lambda a.\text{hcom}^{_{(a:A) \to B\psi}}(M a; \xi_i \mapsto y.N_i) \cong \lambda a.M a \in (a:A) \to B\psi [\Psi'].
\]

By Rule 4.37, it suffices to show that for any \( \psi' : \Psi'' \to \Psi' \) and \( N \cong N' \in A\psi\psi'[\Psi''], \)

\[
\text{hcom}^{_{B\psi\psi'[N/a]}}(M\psi' N; \xi_i\psi' \mapsto y.N_i\psi' N) \cong M\psi' N' \in B\psi\psi'[N/a] [\Psi'].
\]

By \( B\psi\psi'[N/a] \) type\(_{\text{Kan}}\) \([\Psi'']\) and \( r \cong r' \) on the left, it suffices to show \( M\psi' N \cong M\psi' N' \in B\psi\psi'[N/a] [\Psi''], \) which holds by Rule 4.39.

The final hcom property states that if \( \xi_i \) holds then \( \text{hcom}^{_{(a:A') \to B\psi}}(M; \xi_i \mapsto y.N_i) \cong N_i(r'/y) \in (a:A') \to B\psi [\Psi'] \).

As before, by Lemma 4.32 on the left, Rule 4.40 on the right, and Rule 4.37, show for any \( \psi' : \Psi'' \to \Psi' \) and \( N \cong N' \in A\psi\psi'[\Psi''] \) that

\[
\text{hcom}^{_{B\psi\psi'[N/a]}}(M\psi' N; \xi_i\psi' \mapsto y.N_i\psi' N) \cong N_i(r'/y)\psi' N' \in B\psi\psi'[N/a] [\Psi'].
\]

This follows by \( B\psi\psi'[N/a] \) type\(_{\text{Kan}}\) \([\Psi'']\) and \( \xi_i\psi' \) on the left, and Rule 4.39.

Now we must show that dependent functions also support coercion. Suppose \( \psi : (\Psi', x) \to \Psi \) and \( M \cong M' \in (a:A\psi(r/x)) \to B\psi(r/x) [\Psi'], \) and show \( \text{coe}^{_{(a:A') \to B\psi}}(M) \cong \text{coe}^{_{x.(a:A') \to B\psi}}(M') \in (a:A\psi(r'/x)) \to B\psi(r'/x) [\Psi'] \).

By Lemma 4.32 on both sides and Rule 4.37, we must show for any \( \psi' : \Psi'' \to \Psi' \) and \( N \cong N' \in A\psi\psi'(r'/x) [\Psi''] \) that

\[
\text{coe}^{_{x.B\psi\psi'[N/a]}}(M\psi' \text{coe}^{_{x.A\psi\psi'}}(N)) \cong \text{coe}^{_{x.B\psi\psi'[N'/a]}}(M\psi' \text{coe}^{_{x.A\psi\psi'}}(N')) \in B\psi\psi'(r'/x)[N/a] [\Psi''].
\]

By \( A\psi' = A'\psi' \) type\(_{\text{Kan}}\) \([\Psi'', x] \), \( \text{coe}^{_{x.A\psi\psi'}}(N) \cong \text{coe}^{_{x.A\psi\psi'}}(N') \in A\psi\psi'(r'/x) [\Psi''] \); therefore, the above instances of \( B\psi\psi' \) and \( B'\psi\psi' \) are equal Kan types. By Rule 4.39,

\[
M\psi' \text{coe}^{_{x.A\psi\psi'}}(N) \cong M\psi' \text{coe}^{_{x.A\psi\psi'}}(N') \in B\psi\psi'(r'/x)[\text{coe}^{_{x.A\psi\psi'}}(N)/a] [\Psi'']
\]
4.4.2 Dependent pairs

\(\psi\) and interval substitutions preserve judgments. It remains to show relative to expanding definitions, these in turn hold if and only if

\[\frac{M \models \psi}{\frac{\text{proof.}}{\text{By Lemma 4.32 on the left, Rule 4.40 on the right, and Rule 4.37, it suffices to show for any }}\]

\[\text{Lemma 4.12 and coe so the above coe are equal in } B[\psi']^r(\langle r'\psi' / x \rangle)\] is the result follows by Lemma 4.12 and \(\text{coe}_{x:A\psi'}(N) \equiv N \in A[\psi']^r(\langle r'\psi' / x \rangle) \bar{\Psi}'\].

Finally, show that if \(r = r'\) then \(\text{coe}_{x:A\psi'}^r(N) \equiv M \in (a:A\psi'(r' / x)) \rightarrow B[\psi'(r' / x)] \bar{\Psi}'\). By Lemma 4.32 on the left, Rule 4.40 on the right, and Rule 4.37, it suffices to show for any \(\psi' : \Psi'' \rightarrow \Psi'\) and \(N \equiv N' \in A[\psi']^r(\langle r'\psi' / x \rangle) \bar{\Psi}''\] that

\[\text{coe}_{x:A\psi}^r\text{coe}_{x:A\psi'}^r(N) \equiv M\psi' N' \in B[\psi']^r(\langle r'\psi' / x \rangle)\bar{\Psi}'\] is immediate by Rule 4.43. (Note that the hypothesis \(a : A \Rightarrow B \equiv B'\) does not follow from the presumption \(B[M/a] \equiv B[\Psi']\).)

\[\text{Rule 4.43 (Pretype formation). If } A \equiv A' \text{ type}_{pre} \Psi \text{ and } a : A \Rightarrow B \equiv B' \text{ type}_{pre} \Psi \text{ then } (a:A) \times B \equiv (a:A') \times B' \text{ type}_{pre} \Psi.\]

\[\text{Rule 4.44 (Introduction). If } a:A \Rightarrow B \text{ type}_{pre} \Psi, M \equiv M' \in A[\Psi], \text{ and } N \equiv N' \in B[M/a] \Psi', \text{ then } \langle M, N \rangle \equiv \langle M', N' \rangle \in (a:A) \times B[\Psi].\]

\[\text{Rule 4.45 (Elimination). If } P \equiv P' \in (a:A) \times B[\Psi] \text{ then } \text{fst}(P) \equiv \text{fst}(P') \in A[\Psi] \text{ and } \text{snd}(P) \equiv \text{snd}(P') \in B[\text{fst}(P)/a] [\Psi].\]
Proof. For any $\psi : \Psi' \to \Psi$, $P \psi \downarrow \langle M, N \rangle$, $M \in A \psi [\Psi']$, and $N \psi \in B \psi [M / a] [\Psi']$. For part (1), apply coherent expansion to $\text{fst}(P)$ with family $\{M \psi\}_\psi$; then $(M_{id \psi}) \psi = M \psi \in A \psi [\Psi]$ by $P \in (a : A) \times B [\Psi]$ at $id \psi, \psi$. By Lemma 4.18, $\text{fst}(P) \equiv M_{id \psi} \in A [\Psi]$, and part (1) follows by $M_{id \psi} = M_{id \psi}' \in A [\Psi]$ and a symmetric argument on the right side.

For part (2), apply coherent expansion to $\text{snd}(P)$ with family $\{N \psi\}_\psi$. We have $(N_{id \psi}) \psi = N \psi \in B \psi [(M_{id \psi}) \psi / a] [\Psi']$ by $P \in (a : A) \times B [\Psi]$ at $id \psi, \psi$, so by Lemma 4.18, $\text{snd}(P) \equiv N_{id \psi} \in B [M_{id \psi} / a] [\Psi]$. Part (2) follows by $B[M_{id \psi} / a] \equiv B[\text{fst}(P) / a]$ type pre $[\Psi]$ (by $a : A \Rightarrow B \equiv B'$ type pre $[\Psi]$ and $M_{id \psi} \equiv \text{fst}(P) \in A [\Psi]$), $N_{id \psi} \equiv N_{id \psi}' \in B [M_{id \psi} / a] [\Psi]$, and a symmetric argument on the right side.

□

Rule 4.46 (Uniqueness). If $P \in (a : A) \times B [\Psi]$ then $P \equiv (\text{fst}(P), \text{snd}(P)) \in (a : A) \times B [\Psi]$.

Proof. By Lemma 4.16, $P \downarrow \langle M, N \rangle$, $P \equiv \langle M, N \rangle \in (a : A) \times B [\Psi]$, $M \in A [\Psi]$, and $N \in B [M / a] [\Psi]$. By Rule 4.44 and Lemma 4.15 and transitivity, we show $[A]^{\downarrow} (M, \text{fst}(P))$ and $[B]^{\downarrow} (N, \text{snd}(P))$. This is immediate by $\text{fst}(P) \mapsto^* \text{fst}(\langle M, N \rangle) \mapsto^* M$ and $\text{snd}(P) \mapsto^* \text{snd}(\langle M, N \rangle) \mapsto^* N$.

□

Rule 4.47 (Kan type formation). If $A \equiv A'$ type $\text{Kan} [\Psi]$ and $a : A \Rightarrow B \equiv B'$ type $\text{Kan} [\Psi]$ then $(a : A) \times B \equiv (a : A') \times B'$ type $\text{Kan} [\Psi]$.

Proof. We begin with homogeneous composition. Suppose $\psi : \Psi' \to \Psi$, $\overrightarrow{\xi}$ is valid,

1. $M \equiv M' \in (a : A \psi) \times B \psi [\Psi']$,
2. $N_i \equiv N_j \in (a : A \psi) \times B \psi [\Psi', y \mid \xi_i, \xi_j]$ for any $i, j$, and
3. $N_i \langle r / y \rangle \equiv M \in (a : A \psi) \times B \psi [\Psi' \mid \xi_i]$ for any $i$; show

$h\text{com}_{(a : A \psi) \times B \psi}^{\text{r} \to \psi} (M; \overrightarrow{\xi}, \overrightarrow{y} \langle N \rangle) \equiv h\text{com}_{(a : A \psi) \times B \psi}^{\text{r} \to \psi} (M'; \overrightarrow{\xi}, \overrightarrow{y} \langle N \rangle) \in (a : A \psi) \times B \psi [\Psi']$. By Lemma 4.32 on both sides and Rule 4.44, it suffices to show (the binary forms of)

$h\text{com}_{A \psi}^{\text{r} \to \psi} (\text{fst}(M); \xi_i \leftrightarrow y.\text{fst}(N)) \in A \psi [\Psi']$

$\text{com}_{B \psi[F/a]} \equiv y.\text{snd}(N) \in B \psi [h\text{com}_{A \psi}^{\text{r} \to \psi} / a] [\Psi']$

where $F \equiv h\text{com}_{A \psi}^{\text{r} \to \psi} (\text{fst}(M); \xi_i \leftrightarrow y.\text{fst}(N))$.

We have $h\text{com}_{A \psi} \in A \psi [\Psi']$ and $F \in A \psi [\Psi', z]$ by $A$ type $\text{Kan} [\Psi]$ and Rule 4.45. We establish $\text{com}_{z \mapsto B \psi[F/a]} \equiv B \psi [h\text{com}_{A \psi} / a] [\Psi']$ by Theorem 4.33, using $B \psi [F / a]$ type $\text{Kan} [\Psi', z]$, $F \langle r / z \rangle \equiv h\text{com}_{A \psi}$,

1. $\text{snd}(M) \equiv B \psi [F \langle r / z \rangle / a] [\Psi']$ by $F \langle r / z \rangle \equiv \text{fst}(M) \in A \psi [\Psi']$ and Rule 4.45,
2. $N_i \equiv N_j \in B \psi [F \langle y / z \rangle / a] [\Psi', y \mid \xi_i, \xi_j]$ by $F \langle y / z \rangle \equiv \text{fst}(N) \in A \psi [\Psi', y \mid \xi_i]$ and Rule 4.45, and
3. \( \text{snd}(N_i(r/y)) \doteq \text{snd}(M) \in B \psi[F(r/z)/a]\) [\(\Psi' \mid \xi_i\)] by \(F(r/z) \doteq \text{fst}(M) \in A \psi[\Psi']\) and Rule 4.45.

Next, we must show that if \(r = r'\) then \(\text{hcom}_{(a:A \psi) \times B \psi} \doteq M \in (a:A \psi) \times B \psi[\Psi']\). By Lemma 4.32, \(\text{hcom}_{(a:A \psi) \times B \psi} \doteq (\text{hcom}_{A \psi}, \text{com}_{z:B \psi[F/a]}) \in (a:A \psi) \times B \psi[\Psi']\). By Definition 4.29 and Theorem 4.33, \(\text{hcom}_{A \psi} \doteq \text{fst}(M) \in A \psi[\Psi']\), \(\text{com}_{z:B \psi[F/a]} \doteq \text{snd}(M) \in B \psi[F(r/z)/a][\Psi']\), and \(B \psi[F(r/z)/a] \doteq B \psi[\text{fst}(N_i(r'/y))/a][\Psi'],\) The result follows by Rule 4.46.

Finally, show that if \(\xi_i\), \(\text{hcom}_{(a:A \psi) \times B \psi} \doteq N_i(r'/y) \in (a:A \psi) \times B \psi[\Psi']\); this follows from \(\text{hcom}_{A \psi} \doteq \text{fst}(N_i(r'/y)) \in A \psi[\Psi'],\) \(\text{com}_{z:B \psi[F/a]} \doteq \text{snd}(N_i(r'/y)) \in B \psi[\text{fst}(N_i(r'/y))/a][\Psi'],\) and \(B \psi[F(r'/z)/a] \doteq B \psi[\text{fst}(N_i(r'/y))/a][\Psi']\). The result follows by Rule 4.45. We also know that \(\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (M) \doteq \text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (M') \in ((a:A \psi) \times B \psi)(r'/x)[\Psi']\). By Lemma 4.32 and Rule 4.44, it suffices to show (the binary form of)

\[
\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (\text{fst}(M)) \in A \psi(r'/x)[\Psi']
\]

We know that \(\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (\text{fst}(M)) \in A \psi(r'/x)[\Psi']\) and \(B \psi[\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (\text{fst}(M))/a][\Psi']\) by \(A \psi|_{\text{type}_{\text{Kan}}}[\Psi', x], a : A \psi \Rightarrow B \psi|_{\text{type}_{\text{Kan}}}[\Psi', x]\), and Rule 4.45. We also know that \(\text{snd}(M) \in B \psi(r'/x)[\text{fst}(M)/a][\Psi']\) and \(\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (\text{fst}(M))/a)[\Psi']\) and the result follows.

Finally, show that if \(r = r'\) then \(\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (M) \doteq M \in ((a:A \psi) \times B \psi)(r'/x)[\Psi']\). By Lemma 4.32 and Rules 4.44 and 4.46, this follows from \(\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (M) \doteq \text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (M') \in A \psi(r'/x)[\Psi']\) and \(\text{coe}_{x:A \psi|_{\text{type}_{\text{Kan}}}} (M) \doteq \text{snd}(M) \in B \psi(r'/x)[\text{fst}(M)/a][\Psi']\).

### 4.4.3 Paths

For \(i \in \{0, 1, \ldots, \omega\}\) and \(\kappa \in \{\text{pre}, \text{Kan}\}\), \(\tau_i^\kappa(\Psi, \text{Path}_{x:A}(P_0, P_1), \text{Path}_{x:A}(P'_0, P'_1), \varphi)\) where

\[
\varphi = \{(\langle x \rangle M, \langle x \rangle M') \mid (M \sim M' \in \alpha[\Psi, x] \land (\forall \varepsilon. M(\varepsilon/x) \sim P_\varepsilon \in \alpha(\varepsilon/x)[\Psi])\}
\]

if and only if \(A \sim A' \downarrow \alpha \in \tau_i^\kappa[\Psi, x], \text{Coh}(\alpha), \text{and } P_\varepsilon \sim P'_\varepsilon \in \alpha(\varepsilon/x)[\Psi]\) for \(\varepsilon \in \{0, 1\}\), or equivalently, \(\tau_i^\kappa \models (A \equiv A' \downarrow \alpha \in \text{pre}_x[\Psi, x])\) and \(\tau_i^\kappa \models (P_\varepsilon \equiv P'_\varepsilon \in A(\varepsilon/x)[\Psi])\) for \(\varepsilon \in \{0, 1\}\). Relative to \(\tau_i^\kappa:\)

**Rule 4.48** (Pretype formation). If \(A \equiv A' \downarrow \alpha \in \text{pre}_x[\Psi, x]\) and \(P_\varepsilon \equiv P'_\varepsilon \in A(\varepsilon/x)[\Psi]\) for \(\varepsilon \in \{0, 1\}\), then \(\text{Path}_{x:A}(P_0, P_1) \equiv \text{Path}_{x:A}(P'_0, P'_1)\) \(\text{type}_{\text{pre}}[\Psi]\).

**Proof.** We have \(\text{Path}_{x:A}(P_0, P_1) \sim \text{Path}_{x:A}(P'_0, P'_1)\downarrow \{\text{Path}_{x:A}(P_0, P_1)\} \in \tau_i^\kappa[\Psi]\) because \(\text{Path}_{x:A}(P_0, P_1) \text{ val}_{\bot\Psi}\). It remains to show \(\text{Coh}(\{\text{Path}_{x:A}(P_0, P_1)\})\). Suppose that \(\psi : \Psi' \rightarrow \Psi\) and \(\{\text{Path}_{x:A}(P_0, P_1)\} \psi(\langle x \rangle M, \langle x \rangle M')\). Then \(M \equiv M' \in A \psi[\Psi', x]\) and \(M(\varepsilon/x) \equiv P_\varepsilon \psi \in A \psi(\varepsilon/x)[\Psi']\). by \(\langle x \rangle M \text{ val}_{\bot\Psi}\), \(\langle x \rangle M \equiv \langle x \rangle M' \in \{\text{Path}_{x:A}(P_0, P_1)\} \psi[\Psi]\). \(\Box\)
Rule 4.49 (Introduction). If $M \doteq M' \in A[\Psi, x]$ and $M(\varepsilon/x) \doteq P_\varepsilon \in A(\varepsilon/x) [\Psi]$ for $\varepsilon \in \{0, 1\}$, then $(x)M \doteq (x)M' \in \text{Path}_{x,A}(P_0, P_1) [\Psi].$


Rule 4.50 (Elimination).

1. If $M \doteq M' \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$ then $M@r \doteq M'@r \in A(r/x) [\Psi].$

2. If $M \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$ then $M@\varepsilon \doteq P_\varepsilon \in A(\varepsilon/x) [\Psi].$

Proof. Apply coherent expansion to $M@r$ with family $\{M_\psi(\alpha) \mid M_\psi \downarrow (x)M_\psi\}'$. By $M \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$ at $\alpha$, $M$ we know $(M_\alpha)\psi \doteq M_\alpha \in A_\psi[\Psi', x]$, so $(M_{\alpha_\psi})\psi(\alpha) \doteq M_{\alpha_\psi}(\psi) \in A(\alpha/x) [\Psi]$. Thus by Lemma 4.18, $M@r \doteq M_{\alpha_\psi}(\alpha) \in A(\alpha/x) [\Psi]$; part (1) follows by the same argument on the right side and $M_{\alpha_\psi} \doteq M_{\alpha_\psi} \in A[\Psi, x]$. Part (2) follows from $M@\varepsilon \doteq M_{\alpha_\psi}(\varepsilon) \in A(\varepsilon/x) [\Psi]$ and $M_{\alpha_\psi}(\varepsilon) \doteq P_\varepsilon \in A(\varepsilon/x) [\Psi].$ □

Rule 4.51 (Uniqueness). If $M \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$, $M \doteq (x)(M@x) \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$. Proof. By Lemma 4.16, $M \downarrow (x)N$ and $M \doteq (x)N \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$. By Rule 4.50, $M@x \doteq (x)N@x \in A[\Psi, x]$, so by Lemma 4.32 on the right, $M@x \doteq N \in A[\Psi, x]$. By Rule 4.49, $(x)(M@x) \doteq (x)N \in \text{Path}_{x,A}(P_0, P_1) [\Psi]$, and the result follows by transitivity. □

Rule 4.52 (Kan type formation). If $A \doteq A' \text{ type}_\text{Kan} [\Psi, x]$ and $P_\varepsilon \doteq P'_\varepsilon \in A(\varepsilon/x) [\Psi]$ for $\varepsilon \in \{0, 1\}$, then $\text{Path}_{x,A}(P_0, P_1) \doteq \text{Path}_{x,A'}(P'_0, P'_1) \text{ type}_\text{Kan} [\Psi]$

Proof. We begin with homogeneous composition. Suppose $\psi : \Psi' \rightarrow \Psi, \xi_i$ is valid,

1. $M \doteq M' \in \text{Path}_{x,A}(P_0, P_1) [\Psi'],$

2. $N_i \doteq N'_i \in \text{Path}_{x,A}(P_0, P_1) [\Psi', y | \xi_i, \xi_j]$ for any $i, j$, and

3. $N_i(\alpha/y) \doteq M \in \text{Path}_{x,A}(P_0, P_1) [\Psi' | \xi_i]$ for any $i$; show

$$\begin{align*}
\text{hcom}_{(\text{Path}_{x,A}(P_0, P_1))\psi}^r(M; \xi_i \leftarrow y.N_i) \\
\doteq \text{hcom}_{(\text{Path}_{x,A'}(P_0, P_1))\psi}^r(M'; \xi_i \leftarrow y.N'_i) \in (\text{Path}_{x,A}(P_0, P_1))\psi [\Psi'].
\end{align*}$$

By Lemma 4.32 and Rule 4.49 on both sides it suffices to show

$$\begin{align*}
\text{hcom}_{A\psi}^r(M@x; x = \varepsilon \leftarrow P_\varepsilon \psi, \xi_i \leftarrow y.N_i@x) \\
\doteq \text{hcom}_{A\psi}^r(M'@x; x = \varepsilon \leftarrow P'_\varepsilon \psi, \xi_i \leftarrow y.N'_i@x) \in A\psi[\Psi', x]
\end{align*}$$

and $(\text{hcom}_{A\psi})(\varepsilon/x) \doteq P_\varepsilon \psi \in A\psi(\varepsilon/x) [\Psi'].$ By our hypotheses and Rule 4.50,
1. \( M@x \triangleq M'@x \in A[\Psi', x] \),

2. \( P_{\xi} \psi \triangleq P'_{\xi} \psi \in A[\Psi', x \mid x = \varepsilon] \) and \( P_{\xi} \psi \triangleq M@x \in A[\Psi', x \mid x = \varepsilon] \),

3. \( N_{\xi} @x \triangleq N'_{\xi} @x \in A[\Psi', y, x \mid x \in \xi, \xi'] \), \( N_{\xi} @x \triangleq P'_{\xi} \psi \in A[\Psi', x, y \mid x \in \xi, \xi', x = \varepsilon] \), and

\[ \text{so } hcom_{A[\psi]} = hcom_{A[\psi']} \in A[\Psi', x] \text{ and } (hcom_{A[\psi']})(x/y) \triangleq P_{\xi} \psi \in A[\Psi'] \]

Next, show if \( r = r' \), \( hcom_{(Path_{x,A}(P_0, P_1))}[\psi](M; \xi_i \leftarrow y.N_i) \triangleq M \in (Path_{x,A}(P_0, P_1))[\Psi'] \). By Rule 4.49 and Definition 4.29 the left side equals \( (x)(M@x) \); the result follows by Rule 4.51.

Finally, if \( \xi_i \), \( hcom_{(Path_{x,A}(P_0, P_1))}[\psi](M; \xi_i \leftarrow y.N_i) \triangleq N_i(r'/y) \in (Path_{x,A}(P_0, P_1))[\Psi'] \). The left side equals \( (x)(N_i(r'/y))@x ; \) the result follows by Rule 4.51.

For coercion, let \( \psi : (\Psi', y) \to \Psi \) and \( M \triangleq M' \in (Path_{x,A}(P_0, P_1))\psi(r'/y)[\Psi'] \); show \( \text{coerc}_{\psi} \triangleq \psi \in (Path_{x,A}(P_0, P_1))\psi(r'/y)[\Psi'] \). Applying Lemma 4.32 on both sides and then Rule 4.49, it suffices to show

\[ \text{com}_{\psi}[\psi](M@x; x = \varepsilon \leftarrow y.P_{\xi} \psi) \triangleq \text{com}_{\psi}[\psi](M'@x; x = \varepsilon \leftarrow y.P_{\xi} \psi) \in A[\psi(r'/y)[\Psi', x]] \]

and \( (\text{com}_{\psi}[\psi])(x/y) \triangleq P_{\xi} \psi \triangleq A[\psi(r'/y)[\Psi', x]] \). By our hypotheses and Rule 4.50, \( M@x \triangleq M'@x \in A[\psi(r'/y)[\Psi', x]] \), \( P_{\xi} \psi \triangleq P'_{\xi} \psi \in A[\Psi', x, y \mid x = \varepsilon] \), and \( P_{\xi} \psi(r'/y) \triangleq M@x \in A[\psi(r'/y)[\Psi', x \mid x = \varepsilon]] \), so by Theorem 4.33, \( \text{com}_{\psi}[\psi] \triangleq \text{com}_{\psi}[\psi] \in A[\psi(r'/y)[\Psi', x]] \) and \( (\text{com}_{\psi}[\psi])(x/y) \triangleq P_{\xi} \psi \triangleq A[\psi(r'/y)[\Psi', x]] \).

Finally, show that if \( r = r' \), \( \text{coerc}_{\psi} \triangleq \psi \in (Path_{x,A}(P_0, P_1))\psi(r'/y)[\Psi'] \). Again the left side equals \( (x)(M@x) \), and Rule 4.51 completes the proof.

The above proofs generalize straightforwardly to extension types (Section 3.5).

### 4.4.4 Equalities

For \( i \in \{0, 1, \ldots, \omega\} \), \( t_i^{\text{pre}}(\Psi, \text{Eq}_A(M, N), \text{Eq}_{A'}(M', N'), \{(*, \star) \mid M \sim N \in A[\Psi]\}) \) if and only if \( A \sim A' \mid \alpha \in t_i^{\text{pre}}[\Psi] \), \( \text{Coh}(\alpha), M \sim M' \in A[\Psi] \), and \( N \sim N' \in A[\Psi] \); or equivalently, \( t_i^{\text{pre}}[\Psi] = (A \sim A' \mid \alpha \in \text{type}_{A}[\Psi]) \), \( t_i^{\text{pre}}[\Psi] = (M \sim M' \in A[\Psi]) \), and \( t_i^{\text{pre}}[\Psi] = (N \sim N' \in A[\Psi]) \). Relative to \( t_i^{\text{pre}}[\Psi] \).

**Rule 4.53** (Pretype formation). If \( A \sim A' \mid \text{type}_{A}[\Psi], M \sim M' \in A[\Psi], \) and \( N \sim N' \in A[\Psi] \), then \( \text{Eq}_A(M, N) \triangleq \text{Eq}_{A'}(M', N') \mid \text{type}_{A}[\Psi] \).

**Proof.** Again, \( \text{Eq}_A(M, N) \sim \text{Eq}_{A'}(M', N') \mid \text{Eq}_A(M, N) \in t_i^{\text{pre}}[\Psi] \) by \( \text{Eq}_A(M, N) \in \text{val}_{\Psi} \). To see \( \text{Coh}(\text{Eq}_A(M, N)) \), suppose \( \psi : \Psi' \to \Psi \) and \( \text{Eq}_A(M, N)[\psi](*, \star) \). Then \( N \psi \triangleq N' \psi \in A[\Psi'] \), and \( * \sim \star \in \text{Eq}_A(M, N)[\psi] \) holds by \( \star \in \text{val}_{\Psi} \).  


Rule 4.54 (Introduction). If \( M = N \in A [\Psi] \) then \( \star \in \text{Eq}_A (M, N) [\Psi] \).

Proof. Immediate by Rule 4.53.

Rule 4.55 (Elimination). If \( E \in \text{Eq}_A (M, N) [\Psi] \) then \( M = N \in A [\Psi] \).

Proof. Then \( \llbracket \text{Eq}_A (M, N) \rrbracket (E, E) \) so \( E \Downarrow \star \) and \( M = N \in A [\Psi] \).

Rule 4.56 (Uniqueness). If \( E \in \text{Eq}_A (M, N) [\Psi] \) then \( E = \star \in \text{Eq}_A (M, N) [\Psi] \).

Proof. Immediate by Lemma 4.16.

As we observed at the beginning of this chapter, equality types are not generally Kan, because coercion in \( \times \text{Eq}_A (M, N) \) requires all paths in \( A \) to be trivial. (In Chapter 5 we will discuss the converse—if all paths in \( A \) are trivial in an appropriate sense, then \( \times \text{Eq}_A (M, N) \) supports coercion.) On the other hand, one can always homogeneously compose:

\[
\text{hcom}_{\text{Eq}_A (M, N)}^r (M', \xi_i \leftrightarrow y.N'_i) := \star
\]

because \( M', N'_i \) equal \( \star \) by Rule 4.56.

4.4.5 Void

For \( i \in \{0, 1, \ldots, \omega\} \) and \( \kappa \in \{\text{pre}, \text{Kan}\} \), \( \tau^\kappa_i (\Psi, \text{void}, \text{void}, \{\} ) \). Relative to \( \tau^\kappa_i \):

Rule 4.57 (Pretype formation). \( \text{void type}_{\text{pre}} [\Psi] \).

Proof. By \( \text{void val}_{\text{pre}} \), \( \text{void} \sim \text{void} \Downarrow \llbracket \text{void} \rrbracket \in \tau^\kappa_i [\Psi] \) where every \( \llbracket \text{void} \rrbracket \Psi \) is empty; \( \text{Coh}(\llbracket \text{void} \rrbracket) \) holds vacuously.

Theorem 4.58 (Consistency). There is no \( M \) such that \( M \in \text{void} [\Psi] \).

Proof. Suppose \( M \in \text{void} [\Psi] \); then \( \llbracket \text{void} \rrbracket (M, M) \), which is impossible.

Theorem 4.58 implies the absurdity rule in Appendix A— if \( \Gamma \Rightarrow M \in \text{void} [\Psi] \) then \( \Gamma \Rightarrow N \in A [\Psi] \)—because the premise only holds if no \( \gamma \sim \gamma' \in \Gamma [\Psi] \) exist.

Rule 4.59 (Kan type formation). \( \text{void type}_{\text{Kan}} [\Psi] \).

Proof. Each Kan condition supposes that \( M = M' \in \text{void} [\Psi'] \), and therefore holds vacuously by Theorem 4.58.
4.4.6 Booleans

For $i \in \{0, 1, \ldots, \omega\}$ and $\kappa \in \{\text{pre, Kan}\}$, $\tau_i^\kappa(\Psi, \text{bool}, \text{bool}, \{(\text{true}, \text{true}), (\text{false}, \text{false})\})$. Relative to $\tau_i^\kappa$:

**Rule 4.60** (Pretype formation). $\text{bool type}_{\text{pre}} [\Psi]$.  

*Proof.* Once again, $\text{bool} \sim \text{bool} \Downarrow \llbracket \text{bool} \rrbracket \in \tau_i^\kappa [\Psi]$ by $\text{bool val}_{\mathcal{G}}$. For $\text{Coh}(\llbracket \text{bool} \rrbracket)$, we must show $\text{true} \sim \text{true} \in \llbracket \text{bool} \rrbracket [\Psi]$ and $\text{false} \sim \text{false} \in \llbracket \text{bool} \rrbracket [\Psi]$, which hold because $\text{true val}_{\mathcal{G}}$, $\text{false val}_{\mathcal{G}}$, and for all $\Psi'$, $\llbracket \text{bool} \rrbracket_{\Psi'}(\text{true}, \text{true})$ and $\llbracket \text{bool} \rrbracket_{\Psi'}(\text{false}, \text{false})$. □

**Rule 4.61** (Introduction). $\text{true} \in \llbracket \text{bool} \rrbracket [\Psi]$ and $\text{false} \in \llbracket \text{bool} \rrbracket [\Psi]$.  

*Proof.* Immediate by Rule 4.60. □

**Rule 4.62** (Elimination). If $\psi : \text{bool} \Rightarrow A$ type$_{\text{pre}} [\Psi]$, $M \vdash M' \in \text{bool} [\Psi]$, $T \vdash T' \in A[\text{true}/b] [\Psi]$, and $F \vdash F' \in A[\text{false}/b] [\Psi]$, then $(M; T, F) \equiv (M'; T', F') \in A[M/b] [\Psi]$.  

*Proof.* Apply coherent expansion on the left with family $\{\text{if}(M_{\psi}; T_{\psi}, F_{\psi}) | M_{\psi} \Downarrow M'_{\psi}\}_{b_{\psi}}$. We must show $\text{if}(M_{\psi}; T_{\psi}, F_{\psi}) \equiv \text{if}((M_{\text{id}_{\psi}})_{\psi}; T_{\psi}, F_{\psi}) \in A[M_{\psi}/b] [\Psi']$. By the definition of $\llbracket \text{bool} \rrbracket_{\psi}$, either $M_{\psi} = \text{true}$ or $M_{\psi} = \text{false}$; in either case $M_{\text{id}_{\psi}} = M_{\psi}$ because $\llbracket \text{bool} \rrbracket_{\psi}((M_{\text{id}_{\psi}})_{\psi}; M_{\psi})$. Suppose both equal true; show $\text{if}(\text{true}; T_{\psi}, F_{\psi}) \in A[M_{\psi}/b] [\Psi']$. By Lemma 4.32 and $T_{\psi} \in A[\psi][\text{true}/b] [\Psi']$ (by hypothesis) it suffices to show $A[\text{true}/b] \Rightarrow A_{\psi}[M_{\psi}/b] \Rightarrow A[\text{true}/b]$ type$_{\text{pre}} [\Psi']$, which follows from $M_{\psi} \equiv \text{true} \in \llbracket \text{bool} \rrbracket [\Psi']$ (using Lemma 4.16). The $M_{\psi} = \text{false}$ case is symmetric.

We conclude by Lemma 4.18 that if $(M; T, F) \equiv (M_{\text{id}_{\psi}}; T, F) \in A[M/b] [\Psi]$. By transitivity, Lemma 4.16, and the same argument on the right, it suffices to show $\text{if}(M_{\text{id}_{\psi}}; T', F') \equiv (M'_{\text{id}_{\psi}}; T', F') \in A[M_{\text{id}_{\psi}}/b] [\Psi]$. By $M \equiv M' \in \text{bool} [\Psi]$, either $M_{\text{id}_{\psi}} = M_{\text{id}_{\psi}}'$, true or $M_{\text{id}_{\psi}} = M_{\text{id}_{\psi}}' = \text{false}$, and in either case the result follows by Lemma 4.32. □

Our computational type theory (Appendix A) contains an untyped evaluation rule (Lemma 4.16) and therefore satisfies canonicity. Unlike in Chapter 2, that rule applies only to closed terms, because unstable evaluation does not commute with substitution.

**Theorem 4.63** (Canonicity I). If $M \in \text{bool} [\Psi]$ then either $M \Downarrow \text{true}$ and $M \equiv \text{true} \in \text{bool} [\Psi]$ or $M \Downarrow \text{false}$ and $M \equiv \text{false} \in \text{bool} [\Psi]$. Furthermore, the rules of Appendix A suffice to derive these equations.

*Proof.* By $M \sim M \in \text{bool} [\Psi]$ at $\text{id}_{\psi}$, $\text{id}_{\psi}$, either $M \Downarrow \text{true}$ or $M \Downarrow \text{false}$; the desired equation follows by Lemma 4.16 (a rule of Appendix A) and determinacy of evaluation. □

**Rule 4.64** (Kan type formation). $\text{bool type}_{\text{Kan}} [\Psi]$.  

*Proof.* For homogeneous composition, suppose that $\eta_{\xi}$ is valid.
4.4.7 Natural numbers

For \( i \in \{0, 1, \ldots, \omega \} \) and \( \kappa \in \{ \text{pre, Kan} \} \), \( \tau_\kappa^i(\Psi, \text{nat}, \text{nat}, \mathbb{N}_\Psi) \) where \( \mathbb{N} \) is the least context-indexed relation with \( \mathbb{N}_\Psi(z, z) \) and \( \mathbb{N}_\Psi(s(M), s(M')) \) whenever \( M \sim M' \in \mathbb{N}[\Psi] \). Relative to \( \tau_\kappa^i \):

**Rule 4.65** (Pretype formation). \( \text{nat} \text{type}_{\text{pre}}[\Psi] \).

*Proof.* By \( \text{nat} \text{val}_{\mathbb{N}} \), \( \text{nat} \sim \text{nat} \downarrow \mathbb{N} \in \tau_0^i[\Psi] \). To prove \( \text{Coh}(\mathbb{N}) \), we must show \( z \sim z \in \mathbb{N}[\Psi] \) (immediate by \( z \text{val}_{\mathbb{N}} \)) and \( s(M) \sim s(M') \in \mathbb{N}[\Psi] \) when \( M \sim M' \in \mathbb{N}[\Psi] \) (which holds by \( s(M) \text{val}_{\mathbb{N}} \) and \( M' \sim M' \sim \psi \in \mathbb{N}[\Psi'] \) for all \( \Psi' \rightarrow \Psi \)). \( \square \)

**Rule 4.66** (Introduction). \( z \in \text{nat}[\Psi] \) and \( M = M' \in \text{nat}[\Psi] \) then \( s(M) = s(M') \in \text{nat}[\Psi] \).

*Proof.* By Rule 4.65. \( \square \)

**Rule 4.67** (Elimination). If \( n : \text{nat} \Rightarrow A \text{type}_{\text{pre}}[\Psi] \), \( M = M' \in \text{nat}[\Psi] \), \( Z = Z' \in A[z/n] [\Psi] \), and \( n : \text{nat}, a : A \Rightarrow S = S' \in A[s(n)/n] [\Psi] \), then \( \text{nat} \text{rec}(M; Z, n.a.S) = \text{nat} \text{rec}(M'; Z', n.a.S') \in A[M/n] [\Psi] \).

*Proof.* In essence, we "proceed by induction on \(- \sim - \in \mathbb{N} [-]\)," a relation obtained by lifting the context-indexed relation

\[
\mathbb{N} := \mu R.((\Psi, z, z) \cup ((\Psi, s(M), s(M')) \mid M \sim M' \in R[\Psi]))
\]
from values to terms (via various interval substitutions). More precisely, we show (1) the
elimination rule lifts from values to terms, (2) the elimination rule holds for values, and
thus (3) the elimination rule holds for terms.
Define $\Phi_\Psi(M_0, M'_0)$ to hold when $\mathbb{N}_\Psi(M_0, M'_0)$ and for all $n : \text{nat} \Rightarrow A \text{type}_{\text{pre}} [\Psi]$, $Z \vdash Z' \in A[z/n] [\Psi]$, and $n : \text{nat}, a : A \Rightarrow S \equiv S' \in A[s(n)/n] [\Psi]$, we have $\text{natre}(M_0; Z, n.a.S) \equiv \text{natre}(M'_0; Z', n.a.S') \in A[M_0/n] [\Psi]$.

1. If $M \sim M' \in \Phi [\Psi]$ then the elimination rule holds for $M, M'$.

We have $M \sim M' \in \mathbb{N} [\Psi]$ by $\Phi \subseteq \mathbb{N}$ and monotonicity of candidate judgments. Apply coherent expansion to $\text{natre}(M; Z, n.a.S)$ at $A[M/n] \text{type}_{\text{pre}} [\Psi]$ with family
$\{ \text{natre}(M_\psi; Z_\psi, n.a.S_\psi) \mid M_\psi \Downarrow M_{\psi'} \}^{\psi'}$. Then for all $\psi : \Psi' \rightarrow \Psi$ we obtain
$\text{natre}(M_\psi; Z_\psi, n.a.S_\psi) \in A[\psi[M'/n] [\Psi']$ from $\Phi^{\psi}_\Psi(M, M')$ (by $M \sim M' \in \Phi [\Psi]$). We must show
\[
\text{natre}(M_\psi; Z_\psi, n.a.S_\psi) \equiv \text{natre}((M_{id_\psi})_\psi; Z_\psi, n.a.S_\psi) \in A[\psi[M'/n] [\Psi']
\]
but by Lemma 4.15 and $(M_{id_\psi})_\psi \equiv M_\psi \in \text{nat} [\Psi']$ it suffices to show these natre
are related by $\llbracket A[\psi[M'/n] \rrbracket$, which follows from $\Phi^{\psi}_\Psi((M_{id_\psi})_\psi, M_\psi)$.

2. If $\mathbb{N}_\Psi(M_0, M'_0)$ then $\Phi_\Psi(M_0, M'_0)$.

Let $N$ be the function of which $\mathbb{N}$ is the least pre-fixed point; we prove $\Phi$ is also a
pre-fixed point of $N$ (that is, $N(\Phi) \subseteq \Phi$). Suppose $N(\Phi)_\Psi(M_0, M'_0)$. Then either:

a) $M_0 = M'_0 = z$.
Show $\text{natre}(z; Z, n.a.S) \equiv \text{natre}(z; Z', n.a.S') \in A[z/n] [\Psi]$, which is immediate
by $Z \equiv Z' \in A[z/n] [\Psi]$ and Lemma 4.32 on both sides.

b) $M_0 = s(M), M'_0 = s(M')$, and $M \sim M' \in \Phi [\Psi]$.
Show $\text{natre}(s(M); Z, n.a.S) \equiv \text{natre}(s(M'); Z', n.a.S') \in A[s(M)/n] [\Psi]$. By
Lemma 4.32 on both sides, it suffices to show
\[
S[M/n][\text{natre}(M; Z, n.a.S)/a] \\
\equiv S'[M'/n][\text{natre}(M'; Z', n.a.S')/a] \in A[s(M)/n] [\Psi].
\]
We have $M \equiv M' \in \text{nat} [\Psi]$ and $\text{natre}(M; Z, n.a.S) \equiv \text{natre}(M'; Z', n.a.S') \in A[M/n] [\Psi]$ by $M \sim M' \in \Phi [\Psi]$, so the result follows by $n : \text{nat}, a : A \Rightarrow S \equiv S' \in A[s(n)/n] [\Psi]$.

3. Assume $M \sim M' \in \mathbb{N} [\Psi]$; by monotonicity and $\mathbb{N} \subseteq \Phi, M \sim M' \in \Phi [\Psi]$. Thus the
elimination rule holds for $M, M'$, completing the proof. □

Rule 4.68 (Kan type formation). $\text{natre}_{\text{Kan}} [\Psi]$. 

4.4.8 Circle

For \( i \in \{0, 1, \ldots, \omega\} \) and \( \kappa \in \{\text{pre, Kan}\} \), \( \tau_\kappa^\circ(\Psi, \mathcal{S}^1, \mathcal{S}^1, \mathbb{C}_\Psi) \) where \( \mathbb{C} \) is the least context-indexed relation with:

1. \( \mathbb{C}_\Psi(\text{base}, \text{base}) \),
2. \( \mathbb{C}_{(\Psi, x)}(\text{loop}_x, \text{loop}_x) \), and
3. \( \mathbb{C}_\Psi(\text{hcom}^\circ_{\mathcal{S}^1} \langle M; \xi_i \leftarrow y.N_i \rangle, \text{hcom}_\mathcal{S}^1 \circ \langle M'; \xi_i \leftarrow y.N'_i \rangle) \) whenever
   
   - a) \( r \neq r', -\xi_i \) for all \( i \), and \( \xi_i \) is valid,
   - b) \( M \sim M' \in \mathbb{C}[\Psi] \),
   - c) \( N_i \psi \sim N_j' \psi \in \mathbb{C}[\Psi'] \) for all \( i, j \) and \( \psi : \Psi' \rightarrow (\Psi, y) \) satisfying \( \xi_i, \xi_j \), and
   - d) \( N_i(\tau / y) \psi \sim M \psi \in \mathbb{C}[\Psi'] \) for all \( i \) and \( \psi : \Psi' \rightarrow \Psi \) satisfying \( \xi_i \).

By \( \mathcal{S}^1 \) \( \text{val}_\mathcal{S} \) it is immediate that \( \mathcal{S}^1 \sim \mathcal{S}^1 \downarrow \mathbb{C} \in \tau_\kappa^\circ \[ \Psi \] \). Relative to \( \tau_\kappa^\circ \):

Lemma 4.69. If \( \xi_i \) is valid,

1. \( M \sim M' \in \mathbb{C}[\Psi] \),
2. \( N_i \psi \sim N_j' \psi \in \mathbb{C}[\Psi'] \) for all \( i, j \) and \( \psi : \Psi' \rightarrow (\Psi, y) \) satisfying \( \xi_i, \xi_j \), and
3. \( N_i(\tau / y) \psi \sim M \psi \in \mathbb{C}[\Psi'] \) for all \( i \) and \( \psi : \Psi' \rightarrow \Psi \) satisfying \( \xi_i \),

then \( \text{hcom}^\circ_{\mathcal{S}^1} \langle M; \xi_i \leftarrow y.N_i \rangle \sim \text{hcom}_\mathcal{S}^1 \circ (\langle M'; \xi_i \leftarrow y.N'_i \rangle) \in \mathbb{C}[\Psi] \).

Proof. Abbreviating the above hcoms \( L \) and \( R \), we must show for any \( \psi_1 : \Psi_1 \rightarrow \Psi \) and \( \psi_2 : \Psi_2 \rightarrow \Psi_1 \), that \( L \psi_1 \bot L_1, R \psi_1 \bot R_1 \), and \( \mathbb{C}_\Psi \) relates \( L_1 \psi_2, L \psi_1 \psi_2, R_1 \psi_2, \) and \( R \psi_1 \psi_2 \). We proceed by cases on the first step taken by \( L \psi_1 \) and \( L \psi_1 \psi_2 \).

1. \( r \psi_1 = r' \psi_1 \).

Then \( L \psi_1 \hookrightarrow_{\mathbb{S}} M \psi_1, R \psi_1 \hookrightarrow_{\mathbb{S}} M' \psi_1 \), and the result follows from assumption (1).
2. \( r'_\psi \neq r''_\psi, \xi_j \psi_1 \) holds and \( -\xi_i \psi_1 \) for all \( i < j \), and \( r'_\psi \psi_2 = r''_\psi \psi_2 \).

Then \( L_\psi \mapsto N_j(r'/y)_\psi \), \( L_\psi \psi_2 \mapsto M'_\psi \psi_2 \), \( R_\psi \mapsto N'_j(r'/y)_\psi \), and \( R_\psi \psi_2 \mapsto M'_\psi \psi_2 \). Because \( \psi_1 \) satisfies \( \xi_j \), by (2) and (3), \( N_j \psi_1 \sim N'_j \psi_1 \in C [\Psi, y] \) and \( N_j(r/y)_\psi \sim M_\psi \psi_2 \in C [\Psi] \). By the former at \( L_\psi, \psi_2, C^\Psi_{\psi_2} (N_j(r/y)_\psi \psi_1, L_\psi \psi_2) \) and \( C^\Psi_{\psi_2} (L_\psi \psi_2, R_\psi \psi_2) \). By the latter at \( \psi_2, \text{id}_{\Psi_2}, C^\Psi_{\psi_2} (N_j(r/y)_\psi \psi_2, M_\psi \psi_2) \); by transitivity and \( r'_\psi \psi_2 = r''_\psi \psi_2 \) we have \( C^\Psi_{\psi_2} (L_\psi \psi_2, L_\psi \psi_2) \). Finally, by (1), \( C^\Psi_{\psi_2} (L_\psi \psi_2, R_\psi \psi_2) \).

3. \( r'_\psi \neq r''_\psi, \xi_j \psi_1 \) holds (and this is the least such \( i \)), \( r'_\psi \psi_2 \neq r''_\psi \psi_2 \), and \( \xi_j \psi_1 \psi_2 \) holds (and this is the least such \( j \leq i \)).

Then \( L_\psi \mapsto N_i(r'/y)_\psi \), \( L_\psi \psi_2 \mapsto N_i(r'/y)_\psi \psi_2 \), \( R_\psi \mapsto N'_i(r'/y)_\psi \), and \( R_\psi \psi_2 \mapsto N'_i(r'/y)_\psi \psi_2 \). In this case, \( \langle r'/y \rangle \psi_1 \psi_2 \) satisfies \( \xi_i, \xi_j \); the result holds because \( -\sim - \in C [\Psi_2] \) relates \( N_i(r'/y)_\psi \psi_1, N_j(r'/y)_\psi \psi_1, N'_j(r'/y)_\psi \psi_2, \) and \( N'_j(r'/y)_\psi \psi_1 \).

4. \( r'_\psi \neq r''_\psi, -\xi_i \psi_1 \) for all \( i \), and \( r'_\psi \psi_2 = r''_\psi \psi_2 \).

Then \( L_\psi \psi_2 \mapsto M'_\psi \psi_2, R_\psi \psi_2 \mapsto M'_\psi \psi_2 \). In this case, \( L_\psi = L_\psi \psi_2 \) and \( R_\psi = R_\psi \psi_2 \), so the result follows by (1).

5. \( r'_\psi \neq r''_\psi, -\xi_i \psi_1 \) for all \( i \), \( r'_\psi \psi_2 \neq r''_\psi \psi_2 \), and \( \xi_j \psi_1 \psi_2 \) holds (the least such \( j \)).

Then \( L_\psi \psi_2 \mapsto N_i(r'/y)_\psi \psi_2, R_\psi \psi_2 \mapsto N'_i(r'/y)_\psi \psi_2 \). The result follows because \( L_\psi = L_\psi \psi_2, R_\psi = R_\psi \psi_2 \), and because \( \langle r'/y \rangle \psi_1 \psi_2 \) satisfies \( \xi_j, N_j(r'/y)_\psi \psi_2 \sim N'_j(r'/y)_\psi \psi_2 \in C [\Psi_2] \).

6. \( r'_\psi \neq r''_\psi, -\xi_i \psi_1 \) for all \( i \), \( r'_\psi \psi_2 \neq r''_\psi \psi_2 \), and \( -\xi_j \psi_1 \psi_2 \) for all \( j \).

Then \( L_\psi \psi_2 \mapsto M'_\psi \psi_2 \). \( R_\psi \psi_2 \); it suffices to show \( C^\Psi_{\psi_2} (L_\psi \psi_2, R_\psi \psi_2) \).

We know \( \xi_i \psi_1 \psi_2 \) is valid and \( r'_\psi \psi_2 \neq r''_\psi \psi_2 \) for all \( i \), so the result follows immediately by the third clause of the definition of \( C \).

\[ \square \]

**Rule 4.70** (Pretype formation). \( S^1 \text{ type}_{\psi} [\Psi] \).

**Proof.** We must show Coh(\( C \)). There are three cases:

1. base \sim base \in C [\Psi].
   Immediate because base val_{\psi}.

2. loop_{x} \sim loop_{x} \in C [\Psi, x].

Let \( \psi_1 : \Psi \rightarrow (\Psi, x) \) and \( \psi_2 : \Psi_2 \rightarrow \Psi_1 \); show \( \text{loop}_{x\psi_1} \downarrow M_1 \) and \( C^\Psi_{\psi_2} (M_1 \psi_2, \text{loop}_{x\psi_1} \psi_2) \).

If \( x\psi_1 = \epsilon \) then \( M_1 = \text{base} \), \( \text{loop}_{x\psi_1} \psi_2 \mapsto \text{base} \), and \( C^\Psi_{\psi_2} (\text{base}, \text{base}) \). If instead \( x\psi_1 = x' \) and \( x' \psi_2 = \epsilon \), then \( M_1 = \text{loop}_{x'}, \text{loop}_{x'\psi_2} \mapsto \text{base} \).
and \(C_{\Psi}(\text{base, base})\). Otherwise, \(x\psi_1 = x'\) and \(x'\psi_2 = x''\), so \(M_1 = \text{loop}_{x'}\) and \(C_{\Psi}(\text{loop}_{x''}, \text{loop}_{x''})\).

3. \(\text{hcom}_{g_1}^{r'\sim r'}(M; \xi_i \mapsright y.N_i) \sim \text{hcom}_{g_1}^{r'\sim r'}(M'; \xi_i \mapsright y.N'_i) \in C[\Psi]\) where...

This is a special case of Lemma 4.69 in which \(r \neq r'\) and \(\neg \xi_i\) for all \(i\). \(\square\)

**Rule 4.71** (Introduction). \(\text{base} \in \mathbb{S}^1[\Psi], \text{loop}_\epsilon \equiv \text{base} \in \mathbb{S}^1[\Psi], \text{and} \text{loop}_r \in \mathbb{S}^1[\Psi].\)

**Proof.** The first follows from \(\text{Coh}(C)\), the second from \(\text{loop}_\epsilon \mapsright \mathbb{S} \text{base}\) and Lemma 4.32, and the third from \(\text{Coh}(C)\) if \(r = x\) and Lemma 4.32 if \(r = \epsilon\). \(\square\)

**Rule 4.72** (Kan type formation). \(\mathbb{S}^1 \text{type}_{\text{Kan}}[\Psi].\)

**Proof.** For homogeneous composition, suppose \(\xi_i\) is valid,

1. \(M \equiv M' \in \mathbb{S}^1[\Psi']\),

2. \(N_i \equiv N'_i \in \mathbb{S}^1[\Psi', y | \xi_i, \xi_j] \) for any \(i, j\), and

3. \(N_i(r/y) \equiv M \in \mathbb{S}^1[\Psi' | \xi_i] \) for any \(i\); show

\[
\text{hcom}_{g_1}^{r'\sim r'}(M; \xi_i \mapsright y.N_i) \equiv \text{hcom}_{g_1}^{r'\sim r'}(M'; \xi_i \mapsright y.N'_i) \in \mathbb{S}^1[\Psi'],
\]

which is immediate by Lemma 4.69. Next, we must show if \(r = r'\) that \(\text{hcom}_{g_1}^{r'\sim r'}(M; \xi_i \mapsright y.N_i) \equiv M \in \mathbb{S}^1[\Psi']\); this is immediate by \(\text{hcom}_{g_1}^{r'\sim r'}(M; \xi_i \mapsright y.N_i) \mapsright \mathbb{S} M\) and Lemma 4.32. Finally, show if \(\xi_i\) holds that \(\text{hcom}_{g_1}^{r'\sim r'}(M; \xi_i \mapsright y.N_i) \equiv N_i(r/y) \in \mathbb{S}^1[\Psi'].\) Each side is an element of \(\mathbb{S}^1,\) so by Lemma 4.16 it suffices to show \(\text{C}_{\Psi}(\text{hcom}_{g_1}^{r'\sim r'}(M; \xi_i \mapsright y.N_i), N_i(r/y))\). If \(r = r'\) then \(\text{hcom} \mapsright M\) and the result follows by \(N_i(r/y) \equiv M \in \mathbb{S}^1[\Psi' | \xi_i]\), because \(\text{id}_{\Psi'}\) satisfies \(\xi_i\). Otherwise, let \(\xi_j\) be the first true equation. Then \(\text{hcom} \mapsright N_j(r/y)\) and the result follows from \(N_i \equiv N_j \in \mathbb{S}^1[\Psi', y | \xi_i, \xi_j]\).

For coercion, suppose that \(M \equiv M' \in \mathbb{S}^1[\Psi']\) and show \(\text{coe}_{X, \mathbb{S}^1}^{r'\sim r'}(M) \equiv \text{coe}_{X, \mathbb{S}^1}^{r'\sim r'}(M') \in \mathbb{S}^1[\Psi']\). This is immediate by \(\text{coe}_{X, \mathbb{S}^1}^{r'\sim r'}(M) \mapsright \mathbb{S} M\) and Lemma 4.32 on both sides. Similarly, if \(r = r'\) then \(\text{coe}_{X, \mathbb{S}^1}^{r'\sim r'}(M) \equiv M \in \mathbb{S}^1[\Psi']\) by Lemma 4.32 on the left. \(\square\)

**Rule 4.73** (Computation).

1. If \(P \in B[\Psi]\) then \(\mathbb{S}^1-\text{elim}_{c,A}(\text{base}; P, x.L) \equiv P \in B[\Psi]\).

2. If \(L \in B[\Psi, x]\) and \(L(\epsilon/x) \equiv P \in B(\epsilon/x)[\Psi]\) for \(\epsilon \in \{0, 1\}\), \(\mathbb{S}^1-\text{elim}_{c,A}(\text{loop}_r; P, x.L) \equiv L(r/x) \in B(r/x)[\Psi]\).
Proof. Like previous computation rules, part (1) is immediate by Lemma 4.32; in contrast, part (2) holds only under certain typing hypotheses, and not by untyped computation. If \( r = \varepsilon \), part (2) holds by Lemma 4.32 and \( L(\varepsilon / x) \equiv P \in B(\varepsilon / x) \, [\Psi] \). If \( r = y \), apply coherent expansion to the left with \( \{ P \psi \mid y \psi = \varepsilon \} \psi' \cup \{ L \psi(z / x) \mid y \psi = z \} \psi' \). The id\(_\Psi\) element of this family is \( L(y/x) \); when \( y \psi = \varepsilon \), \( L(y/x) \psi \equiv P \psi \in B(y/x) \psi \, [\Psi'] \) (by \( y \psi = (\varepsilon / x) \psi \)).

To prove the elimination rule, as in Rule 4.67, we proceed by induction on \( \mathbb{C} \), which we defined in Section 4.2 as the least pre-fixed point of a function we will notate \( C \). Our inductive hypothesis \( \Phi_{\Psi}(M_0, M'_0) \) holds when:

1. \( \mathbb{C}_{\Psi}(M_0, M'_0) \) and

2. The elimination rule holds for \( M_0 \) and \( M'_0 \). That is: for all \( c : \mathbb{S}^1 \Rightarrow A = A' \) type\(_{\text{Kan}}\) \([\Psi] \), \( P \equiv P' \in A[\text{base} / c] \, [\Psi] \), \( L \equiv L' \in A[\text{loop}_x / c] \, [\Psi, x] \), and \( L(\varepsilon / x) \equiv P \in A[\text{base} / c] \, [\Psi] \) for \( \varepsilon \in \{0, 1\} \), \( \mathbb{S}^1\)-elim\(_{A, \Psi}(M_0, P ; x, L) \equiv \mathbb{S}^1\)-elim\(_{A, \Psi}(M'_0, P' ; x, L') \in A[M_0 / c] \, [\Psi] \).

Lemma 4.74. If \( M \sim M' \in \Phi \, [\Psi] \) then whenever \( c : \mathbb{S}^1 \Rightarrow A = A' \) type\(_{\text{Kan}}\) \([\Psi] \), \( P \equiv P' \in A[\text{base} / c] \, [\Psi] \), \( L \equiv L' \in A[\text{loop}_x / c] \, [\Psi, x] \), and \( L(\varepsilon / x) \equiv P \in A[\text{base} / c] \, [\Psi] \) for \( \varepsilon \in \{0, 1\} \), \( \mathbb{S}^1\)-elim\(_{A, \Psi}(M_0, P ; x, L) \equiv \mathbb{S}^1\)-elim\(_{A, \Psi}(M'_0, P' ; x, L') \in A[M_0 / c] \, [\Psi] \).

Proof. Apply coherent expansion to the left with family \( \{ \mathbb{S}^1\text{-elim}_{A, \Psi}(M_0 ; P_\Psi, x, L_\Psi) \mid M_\Psi \parallel M_0 \}_{\Psi'} \), by showing that

\[
\mathbb{S}^1\text{-elim}_{A, \Psi}(M_0 ; P_\Psi, x, L_\Psi) \equiv \mathbb{S}^1\text{-elim}_{A, \Psi}((M_{id_\Psi})_\Psi ; P_\Psi, x, L_\Psi) \in (A[M_\Psi / c])_{\Psi'}.
\]

The left is an element of this type by \( \Phi_{\Psi}(M_\Psi, M_\Psi) \) and \( A_{\Psi}[M_\Psi / c] \equiv \mathbb{A}_{\Psi}[M_{id_\Psi} / c] \) type\(_{\text{Kan}}\) \([\Psi'] \) (by \( M_\Psi \equiv M_{id_\Psi} \in \mathbb{S}^1 \, [\Psi'] \)); the right is an element by \( \Phi_{\Psi}(M_{id_\Psi}, M_{id_\Psi}) \) and \( A[M_{id_\Psi} / c] \equiv A[M / c] \) type\(_{\text{Kan}}\) \([\Psi] \). The equality follows by \( (M_{id_\Psi})_{\Psi} \parallel M_2 \), \( \Phi_{\Psi}(M_\Psi, M_2) \), and Lemma 4.16. Thus, by Lemma 4.18, \( \mathbb{S}^1\)-elim\(_{A, \Psi}(M_0 ; P, x, L) \equiv \mathbb{S}^1\)-elim\(_{A, \Psi}(M_{id_\Psi} ; P, x, L) \in A[M / c] \, [\Psi] \).

By a symmetric argument on the right, \( A[M / c] \equiv A'[M' / c] \) type\(_{\text{Kan}}\) \([\Psi] \) (by \( M \equiv M' \in \mathbb{S}^1 \, [\Psi] \)), and transitivity, it remains only to establish that \( \mathbb{S}^1\)-elim\(_{A, \Psi}(M_{id_\Psi} ; P, x, L) \equiv \mathbb{S}^1\)-elim\(_{A', \Psi}(M'_{id_\Psi} ; P', x, L') \in A[M / c] \, [\Psi] \); this holds by \( \Phi_{\Psi}(M_{id_\Psi}, M'_{id_\Psi}) \) and \( A[M_{id_\Psi} / c] \equiv A[M / c] \) type\(_{\text{Kan}}\) \([\Psi] \).
1. $C(\Phi)_\Psi(\text{base, base})$.

Then $C(\text{base, base})$ by definition, and the elimination rule holds by Lemma 4.32.

2. $C(\Phi)(\Psi, y)(\text{loop}_y, \text{loop}_y)$.

Then $C(\Psi, y)(\text{loop}_y, \text{loop}_y)$ by definition, and the elimination rule holds by Rule 4.73 on both sides (setting $B = A[\text{loop}_y/c]$, and by $A[\text{loop}_x/c](\epsilon/x) = A[\text{base}/c]$ type$_{\text{Kan}}[\Psi]$) and $L(y/x) = L'(y/x) \in A[\text{loop}_y/c][\Psi]$.

3. $C(\Phi)_\Psi(\text{hcom}^{r \dashv \psi'}_{\Psi}(M; \xi_i \leftarrow y.N_i), \text{hcom}^{r \dashv \psi'}_{\Psi}(M'; \xi_i \leftarrow y.N'_i))$ where

   a) $r \neq r'$, $-\xi_i$ for all $i$, and $\xi_i$ is valid,
   b) $M \sim M' \in \Phi[\Psi]$,
   c) $N_i \psi \sim N'_i \psi \in \Phi[\Psi]$ for all $i, j$ and $\psi : \Psi' \rightarrow (\Psi, y)$ satisfying $\xi_i, \xi_j$, and
   d) $N_i(r/y)\psi \sim M \psi \in \Phi[\Psi']$ for all $i$ and $\psi : \Psi' \rightarrow \Psi$ satisfying $\xi_i$.

By construction, $\Phi \subseteq \mathbb{C}$, so by monotonicity of candidate judgments, $C(\Psi)$ relates the above hcoms. By Lemma 4.74 and $M \sim M' \in \Phi[\Psi]$, $\mathcal{S}^1_\text{elim}_{\text{c},A}(M, \ldots) \doteq \mathcal{S}^1_\text{elim}_{\text{c},A}(M', \ldots) \in A[M/c][\Psi]$. For all $\psi$ satisfying $\xi_i, \xi_j, N_i \psi \sim N'_i \psi \in \Phi[\Psi]$, so by Lemma 4.74, $\mathcal{S}^1_\text{elim}_{\text{c},A}(N_i, \ldots) \doteq \mathcal{S}^1_\text{elim}_{\text{c},A}(N'_i, \ldots) \in A[N_i/c][\Psi, y \mid \xi_i, \xi_j]$.

Similarly, $\mathcal{S}^1_\text{elim}_{\text{c},A}(M) \doteq \mathcal{S}^1_\text{elim}_{\text{c},A}(N_i(r/y)) \in A[M/c][\Psi \mid \xi_i]$.

Apply coherent expansion to $\mathcal{S}^1_\text{elim}_{\text{c},A}(\text{hcom}^{r \dashv \psi'}_{\Psi}(M; \xi_i \leftarrow y.N_i); P, x.L)$ at type $A[\text{hcom}^{r \dashv \psi'}_{\Psi}(M; \xi_i \leftarrow y.N_i)/c]$ type$_{\text{Kan}}[\Psi]$ with family:

\[
\begin{align*}
\mathcal{S}^1_\text{elim}_{\text{c},A}(M \psi; P \psi, x.L \psi) & \quad r \psi = r' \psi \\
\mathcal{S}^1_\text{elim}_{\text{c},A}(N_i(r'/y) \psi; P \psi, x.L \psi) & \quad r \psi \neq r' \psi, \text{ least } j \text{ s.t. } \xi_j \psi \\
\text{com}_{z.A\psi[F/c]}(\mathcal{S}^1_\text{elim}_{\text{c},A}(M \psi; P \psi, x.L \psi); \xi_i \psi \leftarrow y.T_i) & \quad \text{otherwise }
\end{align*}
\]

We must check all three cross-cases, noting that $\text{id}_\Psi$ falls in the third category above. In the first case, when $r \psi = r' \psi$:

\[
\text{com}_{z.A\psi[F/c]}(\mathcal{S}^1_\text{elim}_{\text{c},A}(M \psi); \xi_i \psi \leftarrow y.T_i) \doteq \mathcal{S}^1_\text{elim}_{\text{c},A}(M \psi) \in A[\text{hcom} \psi/c][\Psi]
\]

This holds by Theorem 4.33, $A[\text{hcom} \psi/c] = A[\psi[F/c](r' \psi/z), A[\psi[F/c](r' \psi/z) \doteq A[M/c] \psi$ type$_{\text{Kan}}[\Psi]$], and $A[\psi[F/c] \doteq A[N_i(z/y)/c] \psi$ type$_{\text{Kan}}[\Psi \mid z \mid \xi_i \psi]$. In the
second case, when \( r \psi \neq r' \psi \), \( \xi_j \psi \) holds, and \( \neg \xi_i \psi \) for \( i < j \):

\[
\text{com}_{z.A \psi[F/c]}(\S^1 \text{-elim}_{c.A \psi}(M \psi); \xi_i \psi \leftarrow y.T_i)
\]

\[
\hat{=} \S^1 \text{-elim}_{c.A \psi}(N_j(r'/y) \psi) \in A[\text{hcom}\psi/c] [\Psi']
\]

As before, this holds by Theorem 4.33. Finally, when \( r \psi \neq r' \psi \) and \( \neg \xi_i \psi \) for all \( i \):

\[
\text{com}_{z.A \psi[F/c]}(\S^1 \text{-elim}_{c.A \psi}(M \psi); \xi_i \psi \leftarrow y.T_i) \in A[\text{hcom}\psi/c] [\Psi']
\]

Once again this holds by Theorem 4.33. Therefore, by Lemma 4.18:

\[
\text{com}_{z.A \psi[F/c]}(\S^1 \text{-elim}_{c.A}(M; P, x.L); \xi_i \leftarrow y.\S^1 \text{-elim}_{c.A}(N_i; P, x.L))
\]

\[
\hat{=} \S^1 \text{-elim}_{c.A}(\text{hcom}_{\S^1}(M; \xi_i \leftarrow y.N_i); P, x.L) \in A[\text{hcom}/c] [\Psi].
\]

By transitivity and a symmetric argument on the right, it suffices to show that two
coms are equal, which follows from Theorem 4.33. \( \square \)

**Rule 4.76 (Elimination).** If \( c : \S^1 \Rightarrow A \cong A' \text{type}_{\text{Kan}} [\Psi], M \cong M' \in \S^1 [\Psi], P \cong P' \in A[\text{base}/c] [\Psi], L \cong L' \in A[\text{loop}_{x/c}] [\Psi, x], \text{and } L(e/x) \cong P \in A[\text{base}/c] [\Psi] \text{ for } e \in \{0, 1\}, \)
then \( \S^1 \text{-elim}_{c.A}(M; P, x.L) \hat{=} \S^1 \text{-elim}_{c.A}(M'; P', x.L') \in A[M/c] [\Psi]. \)

**Proof.** By Lemma 4.75, \( \Phi \) is a pre-fixed point of \( C \), and therefore \( C \subseteq \Phi \) (because \( C \) is its least pre-fixed point). By monotonicity of the candidate judgments, \( M \sim M' \in \Phi [\Psi] \); the result follows from Lemma 4.74. \( \square \)

By restricting homogeneous composition to valid shapes \( \xi_i \), we obtain a sharper canonicity result than Cohen et al. [CCHM18], in which 0-dimensional elements of higher inductive types evaluate to constructors, not homogeneous compositions.

**Theorem 4.77 (Canonicity II).** If \( M \in \S^1 [\cdot] \) then \( M \not\downarrow \text{base} \) and \( M \hat{=} \text{base} \in \S^1 [\cdot] \); moreover, the rules of Appendix A derive these equations.

**Proof.** By definition, \( \xi \downarrow (M, M) \). The only generator of \( C \) is base, because there exist no valid shapes \( \xi_i \) with \( \text{Fl} (\xi_i) = \cdot \) and \( \neg \xi_i \) for all \( i \). (By Definition 4.28, all valid shapes in an empty interval context must contain either \( 0 = 0 \) or \( 1 = 1 \).) Therefore \( M \not\downarrow \text{base} \), and the desired equation follows by Lemma 4.16 (which is a rule of Appendix A) and determinacy of evaluation. \( \square \)
4.4.9 V-types

Throughout this section, we use the following abbreviations:

\[
\begin{align*}
\text{Fiber}(A, B, F, M) &:= (a:A) \times \text{Path}_b(F a, M) \\
\text{isContr}(C) &:= (c:C) \times ((c':C) \rightarrow \text{Path}_c(c', c)) \\
\text{Equiv}(A, B) &:= (f:A \rightarrow B) \times ((b:B) \rightarrow \text{isContr}(*)) \\
\end{align*}
\]

For \( i \in \{0, 1, \ldots, \omega \} \) and \( \kappa \in \{\text{pre, Kan}\} \), \( \tau_i^\kappa((\Psi, x), V^\xi(A, B, E), V^\xi(A', B', E'), \varphi) \) if and only if \( \tau_i^\kappa \models (A \doteq A' \text{ type}_{\text{pre}} [\Psi, x \mid x = 0]), \tau_i^\kappa \models (B \doteq B' \text{ type}_{\text{pre}} [\Psi, x]), \tau_i^\kappa \models (E \doteq E' \in \text{Equiv}(A, B)) [\Psi, x \mid x = 0] \), and \( \varphi(\text{Vin}_x(M, N), \text{Vin}_x(M', N')) \) whenever

1. \( \tau_i^\kappa \models (N \doteq N' \in B [\Psi, x]) \),
2. \( \tau_i^\kappa \models (M \doteq M' \in A [\Psi, x \mid x = 0]) \), and
3. \( \tau_i^\kappa \models (\text{fst}(E) M \doteq N \in B [\Psi, x \mid x = 0]) \).

(Note that \( \tau_i^\kappa \models (\text{Equiv}(A, B) \text{ type}_{\text{pre}} [\Psi, x \mid x = 0]) \) by the formation, introduction, and elimination rules for dependent functions, dependent pairs, and paths.) Relative to \( \tau_i^\kappa \):

**Rule 4.78** (Pretype formation).

1. If \( A \text{ type}_{\text{pre}} [\Psi] \) then \( V_0(A, B, E) \doteq A \text{ type}_{\text{pre}} [\Psi] \).
2. If \( B \text{ type}_{\text{pre}} [\Psi] \) then \( V_1(A, B, E) \doteq B \text{ type}_{\text{pre}} [\Psi] \).
3. If \( A \doteq A' \text{ type}_{\text{pre}} [\Psi | r = 0], B \doteq B' \text{ type}_{\text{pre}} [\Psi], \) and \( E \doteq E' \in \text{Equiv}(A, B) [\Psi | r = 0] \), then \( V_r(A, B, E) \doteq V_r(A', B', E') \text{ type}_{\text{pre}} [\Psi] \).

**Proof.** Parts (1–2) hold by Lemma 4.32. For part (3), we must first show \( V_r(A, B, E) \sim V_r(A', B', E') \) \( \downarrow \) \( \in \tau_i^\kappa [\Psi] \), which is to say that, abbreviating these terms \( L \) and \( R \), for all \( \psi_1 : \Psi_1 \rightarrow \Psi \) and \( \psi_2 : \Psi_2 \rightarrow \Psi_1, L \psi_1 \downarrow L, R \psi_1 \downarrow R, (\tau_i^\kappa)^b(\Psi_2, L \psi_1, L \psi_1 \psi_2, \ldots), (\tau_i^\kappa)^b(\Psi_2, R \psi_1, R \psi_1 \psi_2, \ldots), \) and \( (\tau_i^\kappa)^b(\Psi_2, L_1 \psi_2, R_1 \psi_2, \ldots) \). We proceed by cases on the first step taken by \( L \psi_1 \) and \( L \psi_1 \psi_2 \).

1. \( r \psi_1 = 0 \).

Then \( L \psi_1 \rightarrow A \psi_1, R \psi_1 \rightarrow A' \psi_1 \), and the result follows by \( A \psi_1 \doteq A' \psi_1 \text{ type}_{\text{pre}} [\Psi_1] \).

2. \( r \psi_1 = 1 \).

Then \( L \psi_1 \rightarrow B \psi_1, R \psi_1 \rightarrow B' \psi_1 \), and the result follows by \( B \doteq B' \text{ type}_{\text{pre}} [\Psi] \).
3. $r_{\psi_1} = x$ and $r_{\psi_1\psi_2} = 0$.

Then $L_{\psi_1}$ val, $L_{\psi_1\psi_2} \mapsto A_{\psi_1\psi_2}$, $R_{\psi_1}$ val, $R_{\psi_1\psi_2} \mapsto A'_{\psi_1\psi_2}$, and the result follows by $A_{\psi_1\psi_2} \doteq A'_{\psi_1\psi_2}$ type$_{pre}$ $[\Psi_2]$.

4. $r_{\psi_1} = x$ and $r_{\psi_1\psi_2} = 1$.

Then $L_{\psi_1}$ val, $L_{\psi_1\psi_2} \mapsto B_{\psi_1\psi_2}$, $R_{\psi_1}$ val, $R_{\psi_1\psi_2} \mapsto B'_{\psi_1\psi_2}$, and the result follows by $B \doteq B'$ type$_{pre}$ $[\Psi]$.

5. $r_{\psi_1} = x$ and $r_{\psi_1\psi_2} = x'$.

Then $L_{\psi_1}$ val, $L_{\psi_1\psi_2}$ val, $R_{\psi_1}$ val, $R_{\psi_1\psi_2}$ val, and by $A_{\psi_1\psi_2} \doteq A'_{\psi_1\psi_2}$ type$_{pre}$ $[\Psi_2 \mid x' = 0]$, $B_{\psi_1\psi_2} \doteq B'_{\psi_1\psi_2}$ type$_{pre}$ $[\Psi_2]$, and $E_{\psi_1\psi_2} \doteq E'_{\psi_1\psi_2} \in$ Equiv($A_{\psi_1\psi_2}, B_{\psi_1\psi_2}$) $[\Psi_2 \mid x' = 0]$, we have $\tau^x(L_{\psi_2}, V_{\chi}(A_{\psi_1\psi_2}, B_{\psi_1\psi_2}, E_{\psi_1\psi_2}), V_{\chi}(A'_{\psi_1\psi_2}, B'_{\psi_1\psi_2}, E'_{\psi_1\psi_2}), \ldots)$.

It remains only to prove Coh($[V_r(A, B, E)]$). For any $\psi : \Psi' \rightarrow \Psi$, suppose that $[V_r(A, B, E)](M_0, N_0)$, and show $M_0 \sim N_0 \in [V_r(A, B, E)] [\Psi']$. If $r_{\psi} = 0$, coherence follows from $[V_0(A, B, E)] = [A]\Psi_1$; if $r_{\psi} = 1$, coherence follows from $[V_1(A, B, E)] = [B]\Psi_1$. Otherwise, $[V_s(A, B, E)](\text{Vin}_s(M, N), \text{Vin}_s(M', N'))$, or equivalently, $N \doteq N' \in B\Psi \ [\Psi']$, $M \doteq M' \in A\Psi \ [\Psi' \mid x = 0]$, and $\text{fstd}(E\Psi) M \doteq N \in B\Psi \ [\Psi' \mid x = 0]$. We proceed by cases on the first step taken by the $\psi_1$ and $\psi_1\psi_2$ instances of the left:

1. $x_{\psi_1} = 0$.

Then $L_{\psi_1} \mapsto_{\infty} M_{\psi_1}$, $R_{\psi_1} \mapsto_{\infty} M'_{\psi_1}$, and the result follows by $[V_0(A_{\psi_1}, \ldots)] = [A_{\psi_1}]$ and $M_{\psi_1} \doteq M'_{\psi_1} \in A_{\psi_1} [\Psi_1]$.

2. $x_{\psi_1} = 1$.

Then $L_{\psi_1} \mapsto_{\infty} N_{\psi_1}$, $R_{\psi_1} \mapsto_{\infty} N'_{\psi_1}$, and the result follows by $[V_1(A_{\psi_1}, \ldots)] = [B_{\psi_1}]$ and $N \doteq N' \in B\Psi \ [\Psi']$.

3. $x_{\psi_1} = x'$ and $x_{\psi_1\psi_2} = 0$.

Then $L_{\psi_1}$ val, $L_{\psi_1\psi_2} \mapsto M_{\psi_1\psi_2}$, $R_{\psi_1}$ val, $R_{\psi_1\psi_2} \mapsto M'_{\psi_1\psi_2}$, and the result follows by $[V_0(A_{\psi_1\psi_2}, \ldots)] = [A_{\psi_1\psi_2}]$ and $M_{\psi_1\psi_2} \doteq M'_{\psi_1\psi_2} \in A_{\psi_1\psi_2} [\Psi_2]$.

4. $x_{\psi_1} = x'$ and $x_{\psi_1\psi_2} = 1$.

Then $L_{\psi_1}$ val, $L_{\psi_1\psi_2} \mapsto N_{\psi_1\psi_2}$, $R_{\psi_1\psi_2} \mapsto N'_{\psi_1\psi_2}$, and the result follows by $[V_1(A_{\psi_1\psi_2}, \ldots)] = [B_{\psi_1\psi_2}]$ and $N \doteq N' \in B\Psi \ [\Psi']$.

5. $x_{\psi_1} = x'$ and $x_{\psi_1\psi_2} = x''$.

Then $L_{\psi_1}$ val, $L_{\psi_1\psi_2}$ val, $R_{\psi_1}$ val, $R_{\psi_1\psi_2}$ val, and by $N_{\psi_1\psi_2} \doteq N'_{\psi_1\psi_2} \in B_{\psi_1\psi_2} [\Psi_2]$, $M_{\psi_1\psi_2} \doteq M'_{\psi_1\psi_2} \in A_{\psi_1\psi_2} [\Psi_2 \mid x'' = 0]$, and $\text{fstd}(E_{\psi_1\psi_2})$ $M_{\psi_1\psi_2} \doteq N_{\psi_1\psi_2} \in B\Psi \ [\Psi_2 \mid x'' = 0]$, $[V_{\chi''}(A_{\psi_1\psi_2}, \ldots)](\text{Vin}_{\chi''}(M_{\psi_1\psi_2}, N_{\psi_1\psi_2}), \text{Vin}_{\chi''}(M'_{\psi_1\psi_2}, N'_{\psi_1\psi_2})). \Box$
Rule 4.79 (Introduction).

1. If \( M \in A[\Psi] \) then \( \mathsf{Vin}_0(M, N) \equiv M \in A[\Psi] \).
2. If \( N \in B[\Psi] \) then \( \mathsf{Vin}_1(M, N) \equiv N \in B[\Psi] \).
3. If \( M \equiv M' \in A[\Psi \mid r = 0], N \equiv N' \in B[\Psi], E \in \mathsf{Equiv}(A, B)[\Psi \mid r = 0], \) and \( \mathsf{fst}(E) M \equiv N \in B[\Psi \mid r = 0], \) then \( \mathsf{Vin}_r(M, N) \equiv \mathsf{Vin}_r(M', N') \in \mathcal{V}_r(A, B, E)[\Psi] \).

Proof. Parts (1–2) hold by Lemma 4.32. For part (3), if \( r = 0 \) or \( r = 1 \), the result follows from parts (1–2) and Rule 4.78. If \( r = x \), the result is immediate by \( \mathsf{Coh}(\mathcal{V}_x(A, B, E)) \). \( \square \)

Rule 4.80 (Elimination).

1. If \( M \in A[\Psi] \) and \( F \in A \rightarrow B[\Psi] \), then \( \mathsf{Vproj}_0(M, F) \equiv F M \in B[\Psi] \).
2. If \( M \in B[\Psi] \) then \( \mathsf{Vproj}_1(M, F) \equiv M \in B[\Psi] \).
3. If \( M \equiv M' \in \mathcal{V}_r(A, B, E)[\Psi] \) and \( F \equiv \mathsf{fst}(E) \in A \rightarrow B[\Psi \mid r = 0], \) then \( \mathsf{Vproj}_r(M, F) \equiv \mathsf{Vproj}_r(M', \mathsf{fst}(E)) \in B[\Psi] \).

Proof. Parts (1–2) hold by Lemma 4.32. For part (3), if \( r = 0 \) or \( r = 1 \), the result follows from parts (1–2), Rule 4.39, and Rule 4.78. If \( r = x \), apply coherent expansion to the left with family

\[
\begin{aligned}
F \psi & \quad M \psi \\
F \psi & \quad M \psi \\
N \psi & \quad x' \psi = x', M \psi \Downarrow \mathsf{V}_{\mathcal{V}_x}(O \psi, N \psi)
\end{aligned}
\]

where \( O \psi \in A[\Psi \mid x' = 0], N \psi \in B[\Psi], \) and \( \mathsf{fst}(E \psi) O \psi \equiv N \psi \in B[\Psi \mid x' = 0] \). First, show that if \( x \psi = 0, F \psi M \psi \equiv (N_{id_{\psi}}) \psi \in B[\Psi] \). By Lemma 4.16, \( M \equiv \mathsf{Vin}_x(O_{id_{\psi}}, N_{id_{\psi}}) \in \mathcal{V}_x(A, B, E)[\Psi] \), so by Rule 4.79, \( M \psi \equiv (O_{id_{\psi}}) \psi \in A \psi [\Psi'] \). By assumption, \( F \psi \equiv \mathsf{fst}(E \psi) \in A \psi \rightarrow B \psi [\Psi'] \). This case is completed by Rule 4.39 and \( \mathsf{fst}(E \psi) (O_{id_{\psi}}) \psi \equiv (N_{id_{\psi}}) \psi \in B \psi [\Psi'] \). Next, show that if \( x \psi = 1, M \psi \equiv (N_{id_{\psi}}) \psi \in B \psi [\Psi'] \). This case is immediate by Rule 4.79 and \( M \equiv \mathsf{Vin}_x(O_{id_{\psi}}, N_{id_{\psi}}) \in \mathcal{V}_x(A, B, E)[\Psi] \) under \( \psi \). Finally, show that if \( x \psi = x', N \psi \equiv (N_{id_{\psi}}) \psi \in B \psi [\Psi'] \). By \( M \in \mathcal{V}_x(A, B, E)[\Psi] \) under \( id_{\psi}, \psi \) we have \( \mathcal{V}_x(A, B, E)(V_{\mathcal{V}_x}(O \psi, N \psi), V_{\mathcal{V}_x}(V_{\mathcal{V}_x}(O_{id_{\psi}}), (N_{id_{\psi}}) \psi)) \), completing this case.

By Lemma 4.18 we conclude \( \mathsf{V}_{\mathcal{V}_x}(M, F) \equiv N_{id_{\psi}} \equiv B[\Psi] \), and by a symmetric argument, \( \mathsf{V}_{\mathcal{V}_x}(M', \mathsf{fst}(E)) \equiv N'_{id_{\psi}} \). We complete the proof with transitivity and \( N_{id_{\psi}} \equiv N'_{id_{\psi}} \in B[\Psi] \). By \( \mathcal{V}_x(A, B, E)(\mathcal{V}_x(O_{id_{\psi}}, N_{id_{\psi}}), V_{\mathcal{V}_x}(O_{id_{\psi}}, N'_{id_{\psi}})) \). \( \square \)

Rule 4.81 (Computation). If \( M \in A[\Psi \mid r = 0], N \in B[\Psi], F \in A \rightarrow B[\Psi \mid r = 0], \) and \( F M \equiv N \in B[\Psi \mid r = 0], \) then \( \mathsf{Vproj}_r(\mathsf{Vin}_r(M, N), F) \equiv N \in B[\Psi] \).
Proof. If \( r = 0 \) then by Lemma 4.32 it suffices to show \( F \Vin_0(M, N) \models N \in B [\Psi] \), which follows from Rules 4.39 and 4.79 and our hypothesis \( F M \models N \in B [\Psi] \). If \( r = 1 \) the result holds by Lemma 4.32. If \( r = x \) we apply coherent expansion to the left with family

\[
\begin{cases}
F \psi \Vin_0(M \psi, N \psi) & x \psi = 0 \\
N \psi & x \psi = 1 \text{ or } x \psi = x'
\end{cases}
\]

If \( x \psi = 0 \) then \( F \psi \Vin_0(M \psi, N \psi) \models N \psi \in B \psi [\Psi'] \) by Rules 4.39 and 4.79 and \( F M \models N \in B [\Psi | x = 0] \). If \( x \psi \neq 0 \) then \( N \psi \in B \psi [\Psi'] \) and the result follows by Lemma 4.18.

**Rule 4.82 (Uniqueness).** If \( N \in \Vin_r(A, B, E) [\Psi] \) and \( M \models N \in A [\Psi | r = 0] \), then \( \Vin_r(M, \Vproj_r(N, \fst(E))) \models N \in \Vin_r(A, B, E) [\Psi] \).

Proof. If \( r = 0 \) or \( r = 1 \) the result is immediate by Lemma 4.32 and Rule 4.78. If \( r = x \) then by Lemma 4.16, \( N \models \Vin_x(M', P') \in \Vin_x(A, B, E) [\Psi] \) where \( M' \models A [\Psi | x = 0] \), \( P' \models B [\Psi] \), and \( \fst(E) M' \models P' \models B [\Psi | x = 0] \). By Rule 4.79 it suffices to show that \( M \models M' \models A [\Psi | x = 0] \), \( \Vproj_x(N, \fst(E)) \models P' \models B [\Psi] \), and \( \fst(E) M' \models P' \models B [\Psi | x = 0] \) (which is immediate). To show \( N \models M' \models A [\Psi | x = 0] \) it suffices to prove \( N \models M' \models A [\Psi | x = 0] \), which follows from \( N \models \Vin_x(M', P') \in \Vin_x(A, B, E) [\Psi] \) and Rules 4.78 and 4.79. To show \( \Vproj_x(N, \fst(E)) \models P' \models B [\Psi] \), by Rule 4.80 it suffices to check \( \Vproj_x(\Vin_x(M', P'), \fst(E)) \models P' \models B [\Psi] \), which holds by Rule 4.81.

**Lemma 4.83.** If \( A \models A' \text{ type}_\text{Kan} [\Psi | x = 0] \), \( B \models B' \text{ type}_\text{Kan} [\Psi] \), \( E \models E' \models \text{Equiv}(A, B) [\Psi | x = 0] \), \( \vec{\xi}_i \) is valid,

1. \( M \models M' \in \Vin_x(A, B, E) [\Psi] \),
2. \( N_i \models N'_j \in \Vin_x(A, B, E) [\Psi, y | \vec{\xi}_i, \vec{\xi}_j] \) for any \( i, j \), and
3. \( N_i(r/y) \models M \in \Vin_x(A, B, E) [\Psi | \vec{\xi}_i] \) for any \( i \),

then

1. \( \text{hcom}^{\vec{r} \to r'}_{\Vin_x(A, B, E)}(M; \vec{\xi}_i \leftarrow_{y} N_i) \models \text{hcom}^{\vec{r} \to r'}_{\Vin_x(A', B', E')}(M'; \vec{\xi}_i \leftarrow_{y} N'_i) \in \Vin_x(A, B, E) [\Psi] \);
2. if \( r = r' \) then \( \text{hcom}^{\vec{r} \to r'}_{\Vin_x(A, B, E)}(M; \vec{\xi}_i \leftarrow_{y} N_i) \models M \in \Vin_x(A, B, E) [\Psi] \); and
3. if \( \vec{\xi}_i \) then \( \text{hcom}^{\vec{r} \to r'}_{\Vin_x(A, B, E)}(M; \vec{\xi}_i \leftarrow_{y} N_i) \models N_i(r'/y) \in \Vin_x(A, B, E) [\Psi] \).
Proof. For part (1), apply coherent expansion to $\text{hcom}_{V_x(A, B, E)}^{r \rightarrow r'}(M; \xi_i \leftrightarrow y. N_i)$ with family

$$
\begin{cases}
\text{hcom}_{A_y}^{r \rightarrow r'}(M y; \xi_i y \leftrightarrow y. N_i y) & x y = 0 \\
\text{hcom}_{B_y}^{r \rightarrow r'}(M y; \xi_i y \leftrightarrow y. N_i y) & x y = 1
\end{cases}
$$

$(\text{Vin}_x(O(r/y), \text{hcom}_{B_y}^{r \rightarrow r'}(V_{\text{proj}}(M, \text{fst}(E)); \overrightarrow{T}))) y$ $x y = x'$

Consider $y = \text{id}_y$. Using rules for dependent functions, dependent types, and univalence:

1. $O \in A [\Psi, y \mid x = 0]$ and $O(r/y) \equiv M \in A [\Psi \mid x = 0]$ (by $V_x(A, B, E) \equiv A \text{type}_{pre} [\Psi \mid x = 0]$).

2. $V_{\text{proj}}(M, \text{fst}(E)) \in B [\Psi]$ where $V_{\text{proj}}(M, \text{fst}(E)) \equiv \text{fst}(E) M \in B [\Psi \mid x = 0]$ and $V_{\text{proj}}(M, \text{fst}(E)) \equiv M \in B [\Psi \mid x = 1]$.

3. $V_{\text{proj}}(N_i, \text{fst}(E)) \equiv V_{\text{proj}}(N_i, \text{fst}(E)) \in B [\Psi, y \mid \xi_i, \xi_j]$ and $V_{\text{proj}}(M, \text{fst}(E)) \equiv V_{\text{proj}}(M, \text{fst}(E)) \in B [\Psi \mid \xi_i]$.

4. $\text{fst}(E) O \in B [\Psi, y \mid x = 0]$, $\text{fst}(E) O \equiv V_{\text{proj}}(N_i, \text{fst}(E)) \in B [\Psi, y \mid x = 0, \xi_i]$ (both equal $\text{fst}(E) N_i$), and $\text{fst}(E) O(r/y) \equiv V_{\text{proj}}(M, \text{fst}(E)) \in B [\Psi \mid x = 0]$ (both equal $\text{fst}(E) M$).

5. $\text{hcom}_{B}^{r \rightarrow r'}(M; \xi_i \leftrightarrow y. N_i) \in B [\Psi, y \mid x = 1]$ (by $V_x(A, B, E) \equiv B \text{type}_{pre} [\Psi \mid x = 1]$), $\text{hcom}_{B}^{r \rightarrow r'}(M; \xi_i \leftrightarrow y. N_i) \equiv V_{\text{proj}}(N_i, \text{fst}(E)) \in B [\Psi, y \mid x = 1, \xi_i]$ (both equal $N_i$), and $\text{hcom}_{B}^{r \rightarrow r'}(M; \xi_i \leftrightarrow y. N_i) \equiv V_{\text{proj}}(M, \text{fst}(E)) \in B [\Psi \mid x = 1]$ (both equal $M$).

6. By the above, $\text{hcom}_{B}^{r \rightarrow r'}(V_{\text{proj}}(M, \text{fst}(E)); \overrightarrow{T}) \in B [\Psi]$ and $\text{hcom}_{B} \equiv \text{fst}(E) O(r'/y) \in B [\Psi \mid x = 0]$, so $\text{Vin}_x(O(r'/y), \text{hcom}_{B}^{r \rightarrow r'}(V_{\text{proj}}(M, \text{fst}(E)); \overrightarrow{T})) \in V_x(A, B, E) [\Psi]$.

When $x y = x'$, coherence is immediate. When $x y = 0$, $\text{Vin}_0(O(r'/y), \ldots) \equiv \text{hcom}_{A y} \in A y [\Psi']$ as required. When $x y = 1$, $\text{Vin}_1(\ldots, \text{hcom}_{B y}^{r \rightarrow r'}(\ldots; \overrightarrow{T})) \equiv \text{hcom}_{B y} \in B y [\Psi']$ as required. Part (1) follows by Lemma 4.18 and a symmetric argument on the right.

For part (2), show $\text{Vin}_x(O(r'/y), \text{hcom}_{B}^{r \rightarrow r'}(V_{\text{proj}}(M, \text{fst}(E)); \overrightarrow{T})); \overrightarrow{T})) \equiv M \in V_x(A, B, E) [\Psi]$ when $r = r'$. By the above, $\text{Vin}_x(\ldots) \equiv \text{Vin}_x(M, V_{\text{proj}}(M, \text{fst}(E))) \in V_x(A, B, E) [\Psi]$, so the result follows by Rule 4.82.
For part (3), we must show $\text{Vin}_x(O(r'/y), \text{hcom}_B^{r\Rightarrow r'}(\text{Vproj}_x(M, \text{fst}(E))); \overrightarrow{T}) \doteq N_r(r'/y) \in V_x(A, B, E)[\Psi]$ when $\xi$. But we have $\text{Vin}_x(\ldots) \doteq \text{Vin}_x(N_r(r'/y), \text{Vproj}_x(N_r(r'/y), \text{fst}(E))) \in V_x(A, B, E)[\Psi]$, so the result again follows by Rule 4.82. □

**Lemma 4.84.** If $x \neq y$,

1. $A \doteq A'$ type$_{\text{Kan}}[\Psi, y \mid x = 0]$,
2. $B \doteq B'$ type$_{\text{Kan}}[\Psi, y]$,
3. $E \doteq E' \in \text{Equiv}(A, B)[\Psi, y \mid x = 0]$, and
4. $M \doteq M' \in (V_x(A, B, E))\langle r/y \rangle[\Psi]$,

then

1. $\text{coe}_{y, V_x(A, B, E)}^{r\Rightarrow r'}(M) \doteq \text{coe}_{y, V_x(A', B', E')}^{r\Rightarrow r'}(M') \in (V_x(A, B, E))\langle r'/y \rangle[\Psi]$ and
2. $\text{coe}_{y, V_x(A, B, E)}^{r\Rightarrow r'}(M) \doteq M \in (V_x(A, B, E))\langle r/y \rangle[\Psi]$.

**Proof.** We apply coherent expansion to $\text{coe}_{y, V_x(A, B, E)}^{r\Rightarrow r'}(M)$ with family

$$
\begin{align*}
\text{coe}^{r\Rightarrow r'}_{y, A} & (M) \quad x\psi = 0 \\
\text{coe}^{r\Rightarrow r'}_{y, B} & (M) \quad x\psi = 1 \\
(V_{\text{x}}(\text{coe}_{y, A}^{r\Rightarrow r'}(M), \text{com}^{r\Rightarrow r'}_{y, B}(\text{Vproj}_x(M, \text{fst}(E\langle r/y \rangle)); \overrightarrow{T}))) & (x\psi = x')
\end{align*}
$$

Consider $\psi = \text{id}_\Psi$.

1. $\text{Vproj}_x(M, \text{fst}(E\langle r/y \rangle)) \in B\langle r/y \rangle[\Psi]$ (by our hypothesis $M \in V_x(A\langle r/y \rangle, \ldots)[\Psi]$), with boundary $\text{Vproj}_x(M, \text{fst}(E\langle r/y \rangle)) \doteq \text{fst}(E\langle r/y \rangle) M \in B\langle r/y \rangle[\Psi \mid x = 0]$ and $\text{Vproj}_x(M, \text{fst}(E\langle r/y \rangle)) \doteq M \in B\langle r/y \rangle[\Psi \mid x = 1]$.

2. $\text{fst}(E) \text{coe}_{y, A}^{r\Rightarrow y}(M) \in B[\Psi, y \mid x = 0]$ because $\text{fst}(E) \in A \rightarrow B[\Psi, y \mid x = 0]$ and $\text{coe}_{y, A}^{r\Rightarrow y}(M) \in A[\Psi, y \mid x = 0]$ (by $M \in A\langle r/y \rangle[\Psi \mid x = 0]$). Under $\langle r/y \rangle$ this $\doteq \text{fst}(E\langle r/y \rangle) M \in B\langle r/y \rangle[\Psi \mid x = 0]$.

3. $\text{coe}_{y, B}^{r\Rightarrow y}(M) \in B[\Psi, y \mid x = 1]$ (by $M \in B\langle r/y \rangle[\Psi \mid x = 1]$) and $\text{coe}_{y, B}^{r\Rightarrow r}(M) \doteq M \in B\langle r/y \rangle[\Psi \mid x = 1]$. 
Lemma 4.85. If

1. \( A \doteq A' \) \( \text{type}_{\text{Kan}} \) \( [\Psi, x \mid x = 0] \),
2. \( B \doteq B' \) \( \text{type}_{\text{Kan}} \) \( [\Psi, x] \),
3. \( E \doteq E' \in \text{Equiv}(A, B) \) \( [\Psi, x \mid x = 0] \), and
4. \( M \doteq M' \in (\mathcal{V}_x(A, B, E))\langle r/x \rangle \) \( [\Psi] \),

then

1. \( \text{coer}_{x, \mathcal{V}_x(A, B, E)}^r(M) \doteq \text{coer}_{x, \mathcal{V}_x(A', B', E')}^r(M') \in (\mathcal{V}_x(A, B, E))\langle r/x \rangle \) \( [\Psi] \) and
2. \( \text{coer}_{x, \mathcal{V}_x(A, B, E)}^r(M) \doteq M \in (\mathcal{V}_x(A, B, E))\langle r/x \rangle \) \( [\Psi] \).

Proof. For part (1), by Lemma 4.32 on both sides, it suffices to show (the binary form of) \( \text{Vin}_r(\text{fst}(O), P) \in (\mathcal{V}_x(A, B, E))\langle r/x \rangle \) \( [\Psi] \), where

\[
N := \text{coer}_{x, B}^r(\text{Vproj}, (M, \text{fst}(E\langle r/x \rangle)))
\]

\[
F := \text{Fiber}(A\langle r/x \rangle, B\langle r/x \rangle, E\langle r/x \rangle), N)
\]

\[
C := \text{snd}(E\langle r/x \rangle) \ N
\]

\[
O := \text{hcom}_{F}^{1,\text{eq}_0}(\text{fst}(C); r = 0 \leftrightarrow (\text{snd}(C) \langle M, \langle \_ \rangle(\text{fst}(E\langle 0/x \rangle) \ M)) @ z, r = 1 \leftrightarrow \_ \text{fst}(C))
\]

\[
P := \text{hcom}_{B\langle r/x' \rangle}^{1,\text{eq}_0}(N; r' = 0 \leftrightarrow z.\text{snd}(O) @ z, r' = 1 \leftrightarrow \_N, T)
\]

\[
T := r = r' \leftrightarrow \_ \text{Vproj}, (M, \text{fst}(E\langle r/x \rangle))
\]

1. \( N \in B\langle r/x \rangle \) \( [\Psi] \) by \( \text{Vproj}, (M, \text{fst}(E\langle r/x \rangle)) \in B\langle r/x \rangle \) \( [\Psi] \).
2. $F \text{ type}_{\text{Kan}} [\Psi]$.

3. $C \in \text{isContr}(F) [\Psi \mid r' = 0]$ by $\text{snd}(E) \in (b:B) \rightarrow \text{isContr}((A, B, \text{fst}(E), b)) [\Psi, x \mid x = 0]$.

4. $O \in F [\Psi \mid r' = 0]$ because
   
   a) $(\text{snd}(C) \langle M, \langle \_ \rangle(\text{fst}(E_0/x_M)) \rangle) @ z \in F [\Psi, z \mid r' = 0, r = 0]$ because we have $\text{snd}(C) \langle M, \ldots \rangle \in \text{Path}_F(\langle M, \langle \_ \rangle(\text{fst}(E_0/x_M)) \rangle, \text{fst}(C)) [\Psi \mid r' = 0, r = 0]$, by
      
      i. $\text{snd}(C) \in (c':F) \rightarrow \text{Path}_F(c', \text{fst}(C)) [\Psi \mid r' = 0]$.
      
      ii. $M \in A(0/x) [\Psi \mid r = 0]$ by $\langle V_x(A, B, E) \rangle \cdot r/x = A(0/x)$ $\text{type}_{\text{Kan}} [\Psi \mid r = 0]$,.
      
      iii. $\text{fst}(E_0/x_M) M \in B(0/x) [\Psi \mid r = 0]$.
      
      iv. $\langle \_ \rangle(\text{fst}(E_0/x_M)) M \in \text{Path}_{B(0/x)}(\text{fst}(E_0/x_M) M, N) [\Psi \mid r' = 0, r = 0]$ by $N \doteq \text{fst}(E_0/x_M) M \in B(0/x) [\Psi \mid r' = 0, r = 0]$, and
      
      v. $F \doteq (a:A(0/x)) \times \text{Path}_{B(0/x)}(\text{fst}(E_0/x_M) a, N)$ $\text{type}_{\text{Kan}} [\Psi \mid r' = 0]$.
   
   b) $\text{fst}(C) \in F [\Psi \mid r' = 0]$, and
   
   c) $\text{fst}(C) \doteq (\text{snd}(C) \langle M, \ldots \rangle) @ 1 \in F [\Psi \mid r' = 0, r = 0]$ by the right endpoint specified above.

5. $P \in B(\langle r'/x \rangle) [\Psi]$ because
   
   a) $\text{snd}(O) @ z \in B(\langle r'/x \rangle) [\Psi, z \mid r' = 0]$.
   
   b) $N \in B(\langle r'/x \rangle) [\Psi \mid r' = 1]$.
   
   c) $\text{Vproj}_x(M, \text{fst}(E(\langle r'/x \rangle))) \in B(\langle r'/x \rangle) [\Psi \mid r = r']$.
   
   d) $N \in B(\langle r'/x \rangle) [\Psi]$.
   
   e) $N \doteq \text{snd}(O) @ 1 \in B(\langle r'/x \rangle) [\Psi \mid r' = 0]$ by the right endpoint specified in $F$.
   
   f) $N \doteq \text{Vproj}_x(M, \text{fst}(E(\langle r'/x \rangle))) \in B(\langle r'/x \rangle) [\Psi \mid r = r']$, and
   
   g) $\text{snd}(O) @ z \doteq \text{Vproj}_x(M, \text{fst}(E(\langle r'/x \rangle))) \in B(\langle r'/x \rangle) [\Psi, z \mid r' = 0, r = r']$ because $O \doteq (\text{snd}(C) \langle M, \ldots \rangle) @ 0 \doteq \langle M, \_ \rangle(\text{fst}(E_0/x_M) M) (by \ r = 0 \ \text{and} \ \text{the specified left endpoint}) \ \text{and} \ \text{thus} \ \text{snd}(O) @ z \doteq \text{fst}(E_0/x_M) M$.

Then $\text{Vin}_r(\text{fst}(O), P) \in (V_x(A, B, E)) \langle r'/x \rangle [\Psi]$ because $\text{fst}(O) \in A(\langle r'/x \rangle) [\Psi \mid r' = 0]$, $P \in B(\langle r'/x \rangle) [\Psi]$, and $\text{fst}(E(\langle r'/x \rangle)) \text{fst}(O) \doteq P \in B(\langle r'/x \rangle) [\Psi \mid r' = 0] (by \ P \doteq \text{snd}(O) @ 0 \in B(\langle r'/x \rangle) [\Psi \mid r' = 0] \ \text{and \ \the \ specified \ left \ endpoint})$, completing part (1).

Part (2) follows from Rule 4.82, $P \doteq \text{Vproj}_x(M, \text{fst}(E(\langle r'/x \rangle))) \in B(\langle r'/x \rangle) [\Psi \mid r = r']$, and $\text{fst}(O) \doteq M \in A(\langle r'/x \rangle) [\Psi \mid r = r', r' = 0] (by \ O \doteq (\text{snd}(C) \langle M, \ldots \rangle) @ 0 \doteq \langle M, \ldots \rangle)$. □
In the proof of Lemma 4.85, we include the \( r = 1 \) face of \( O \) and the \( r' = 1 \) face of \( P \) only to ensure all compositions are valid; one could instead use ghcom (Theorem 4.34).

**Rule 4.86** (Kan type formation).

1. If \( A \) type\(_{\text{Kan}}\) \([\Psi]\) then \( V_0(A, B, E) \cong A \) type\(_{\text{Kan}}\) \([\Psi]\).

2. If \( B \) type\(_{\text{Kan}}\) \([\Psi]\) then \( V_1(A, B, E) \cong B \) type\(_{\text{Kan}}\) \([\Psi]\).

3. If \( A \cong A' \) type\(_{\text{Kan}}\) \([\Psi | r = 0]\), \( B \cong B' \) type\(_{\text{Kan}}\) \([\Psi]\), and \( E \cong E' \in \text{Equiv}(A, B) \) \([\Psi | r = 0]\), then \( V_r(A, B, E) \cong V_r(A', B', E') \) type\(_{\text{Kan}}\) \([\Psi]\).

**Proof.** Parts (1–2) follow from Lemma 4.32. Part (3) requires homogeneous composition and coercion; for homogeneous composition, suppose we are given \( \psi : \Psi' \to \Psi \) and a valid composition scenario in \( V_r(\psi, B, \psi, E\psi) \). If \( r\psi = 0 \) (resp., 1) then the composition is in \( A\psi \) (resp., \( B\psi \)) and homogeneous composition follows from \( A\psi \cong A'\psi \) type\(_{\text{Kan}}\) \([\Psi']\) (resp., \( B\psi \cong B'\psi \) type\(_{\text{Kan}}\) \([\Psi']\)). Otherwise, \( r\psi = x \) and homogeneous composition exists by Lemma 4.83 at \( A\psi \cong A'\psi \) type\(_{\text{Kan}}\) \([\Psi' \mid x = 0]\), \( B\psi \cong B'\psi \) type\(_{\text{Kan}}\) \([\Psi]\), and \( E\psi \cong E'\psi \in \text{Equiv}(A\psi, B\psi) \) \([\Psi' \mid x = 0]\).

For coercion, suppose \( \psi : (\Psi', x) \to \Psi \) and \( M \cong M' \in (V_r(\psi, B, E))\psi(s/x) \) \([\Psi']\). If \( r\psi = 0 \) (resp., 1), coercion follows from \( A\psi \cong A'\psi \) type\(_{\text{Kan}}\) \([\Psi', x]\) (resp., \( B\psi \cong B'\psi \) type\(_{\text{Kan}}\) \([\Psi', x]\)). If \( r\psi = x \), coercion follows from Lemma 4.85; otherwise, it follows from Lemma 4.84. \( \Box \)

### 4.4.10 Compositions of pretypes

As discussed in Section 3.4, in order for the pretype universe (resp., Kan type universe) to be Kan, we must equip homogeneous compositions of pretypes (resp., Kan types) with the structure of a pretype (resp., Kan type). Let us say that \( A, \xi_i \leftarrow y.B_i \) and \( A', \xi_i \leftarrow y.B'_i \) are (equal) pretype compositions \( r \rightsquigarrow r' \) whenever \( \xi_i \) is valid,

1. \( A \cong A' \) type\(_{\text{pre}}\) \([\Psi]\),

2. \( B_i \cong B'_j \) type\(_{\text{pre}}\) \([\Psi, y \mid \xi_i, \xi_j]\) for any \( i, j \), and

3. \( B_i(r/y) \cong A \) type\(_{\text{pre}}\) \([\Psi \mid \xi_i]\) for any \( i \).

For \( i \in \{0, 1, \ldots, \omega\} \), \( \tau_i^{\text{pre}}(\Psi, \text{hcom}_{\text{pre}}^r(A; \xi_i \leftarrow y.B_i), \text{hcom}_{\text{pre}}^{r'}(A'; \xi_i \leftarrow y.B'_i), \emptyset) \) when \( A, \xi_i \leftarrow y.B_i \) and \( A', \xi_i \leftarrow y.B'_i \) are equal pretype compositions \( r \rightsquigarrow r' \) relative to \( \tau_i^{\text{pre}} \) with \( r \neq r' \) and \( \neg \xi_i \) for all \( i \). Then, relative to \( \tau_i^{\text{pre}} \):

**Rule 4.87** (Pretype formation). If \( A, \xi_i \leftarrow y.B_i \) and \( A', \xi_i \leftarrow y.B'_i \) are equal pretype compositions \( r \rightsquigarrow r' \), then
4.4.11 Compositions of Kan types

We complete part (3) with

if \( r = r' \) then \( B_i \langle r/y \rangle \preceq B_i \preceq \langle r'/y \rangle \) type\( \preceq \) \( \Psi \).

Proof. We abbreviate these \( \text{hcom}^r \) terms \( L \) and \( R \). Part (2) holds by Lemma 4.32. For part (3), if \( r = r' \), the result holds by Lemma 4.32 and \( B_i \langle r/y \rangle \preceq A \) type\( \preceq \) \( \Psi \), otherwise, there is a least \( j \) such that \( \xi_j \) holds. Apply coherent expansion to \( L \) with family

\[
\begin{align*}
A & \quad r\psi = r'\psi \\
B_j \langle r'/y \rangle \psi & \quad r\psi \neq r'\psi, \xi_j \psi, \text{ and } \forall l < j, \neg \xi_l \psi.
\end{align*}
\]

If \( r\psi = r'\psi \) then \( B_j \langle r/y \rangle \psi \preceq A \psi \) type\( \preceq \) \( \Psi \). If \( r\psi \neq r'\psi \), there is some least \( l \) such that \( \xi_l \psi \); then \( B_j \langle r'/y \rangle \psi \preceq B_j \langle r'/y \rangle \psi \) type\( \preceq \) \( \Psi \). By Lemma 4.17, \( L \preceq B_j \langle r'/y \rangle \) type\( \preceq \) \( \Psi \), and we complete part (3) with \( B_j \langle r'/y \rangle \preceq B_j \langle r'/y \rangle \) type\( \preceq \) \( \Psi \).

For part (1), we conclude \( L \sim R \triangleleft [L] \in \tau_i^\text{pre} \Psi \) by the same argument as in Lemma 4.69, because the definition and operational semantics of \( \text{hcom}_\Psi \) and \( \text{hcom}_{\kappa \text{pre}} \) are identical. To see \( \text{Coh}([L]) \), suppose \( [L]_{\psi}(M_0, N_0) \) for any \( \psi : \Psi \rightarrow \Psi \). If \( r\psi = r'\psi \), coherence follows by part (2); if \( \xi_j \psi \), coherence follows by part (3); otherwise, \( [L]_{\psi} \) is the empty relation, which contradicts \( [L]_{\psi}(M_0, N_0) \).

4.4.11 Compositions of Kan types

For \( i \in \{0, 1, \ldots, \omega \} \) and \( \kappa \in \{\text{pre}, \text{Kan}\}, \)

\[
\tau_i^\kappa(\Psi, \text{hcom}^r \preceq \text{hcom}_{\text{Kan}}^r (A; \xi_i \leftarrow y.B_i), \text{hcom}^r \preceq \text{hcom}_{\text{Kan}}^r (A'; \xi_i \leftarrow y.B_i'), \varphi)
\]

when \( A, \xi_i \leftarrow y.B_i \) and \( A', \xi_i \leftarrow y.B_i' \) are equal pretype compositions \( r \preceq r' \) relative to \( \tau_i^\text{Kan} \) with \( r \neq r' \) and \( \neg \xi_i \) for all \( i \), and \( \varphi(\text{box}^r \preceq \text{box}^{r'} (M; \xi_i \leftarrow N_i), \text{box}^r \preceq \text{box}^{r'} (M'; \xi_i \leftarrow N_i')) \) whenever

1. \( \tau_i^\text{Kan} \models (M \models M' \in A \Psi) \).
2. \( \tau_i^\text{Kan} \models (N_i \models N_j' \in B_i \langle r'/y \rangle \Psi | \xi_i, \xi_j) \) for all \( i, j \), and
3. \( \tau_i^\text{Kan} \models (M \models \text{coer}_{y.B_i} (N_i) \in A \Psi | \xi_i) \) for all \( i \).

Note that \( \tau_i^\text{pre} \) is closed under \( \text{hcom}_{\text{Kan}}(-) \) of types in \( \tau_i^\text{Kan} \), because \( [\text{hcom}_{\text{Kan}}(-)] \) is only value-coherent for Kan types, and we require \( \tau_i^\text{Kan} \subseteq \tau_i^\text{pre} \). In this section, we
Lemma 4.88. If \( A, \xi_i \leftrightarrow y.B_i \) and \( A', \xi_i \leftrightarrow y.B'_i \) are equal Kan type compositions \( r \rightsquigarrow r' \), then

1. \( A \dot= A' \text{ type}_{\text{Kan}}[\Psi] \).
2. \( B_i \dot= B'_i \text{ type}_{\text{Kan}}[\Psi, y | \xi, \xi_j] \) for any \( i, j \), and
3. \( B_i(r/y) \dot= A \text{ type}_{\text{Kan}}[\Psi | \xi] \) for any \( i \).

Proof. As in Rule 4.87, part (1) follows from the argument of Lemma 4.69. Part (2) holds by Lemma 4.32. For part (3), if \( r = r' \), the result follows by Lemma 4.32 and \( B_i(r/y) \dot= A \text{ type}_{\text{Kan}}[\Psi | \xi] \). Otherwise, there is a least \( j \) such that \( \xi_j \) holds. Apply coherent expansion to the left with family

\[
\begin{align*}
A\psi & \\
B_j(r'/y)\psi & r\psi \neq r'\psi, \xi_j\psi, \text{ and } \forall l < j. \sim \xi_l\psi.
\end{align*}
\]

If \( r\psi = r'\psi \) then \( B_j(r/y)\psi = B_j(r'/y)\psi = A\psi \text{ type}_{\text{Kan}}[\Psi] \). If \( r\psi \neq r'\psi \), there is some least \( l \leq j \) such that \( \xi_l\psi \); then \( B_j(r'/y)\psi = B_j(r'/y)\psi = A \text{ type}_{\text{Kan}}[\Psi'] \). By Lemma 4.31, \( \text{hcom}_{\text{Kan}}[\Psi] = B_j(r'/y) \text{ type}_{\text{Kan}}[\Psi] \), and part (3) follows by \( B_i(r/y) \dot= B_j(r'/y) \text{ type}_{\text{Kan}}[\Psi] \). \( \square \)

Lemma 4.89. If \( A, \xi_i \leftrightarrow y.B_i \) is a Kan type composition \( r \rightsquigarrow r' \), then

1. \( M \dot= M' \in A[\Psi] \),
2. \( N_i \dot= N'_i \in B_i(r'/y) [\Psi | \xi, \xi_j] \) for any \( i, j \), and
3. \( \text{coer}_{y, B_i}^r(N_i) = M \in A[\Psi | \xi] \) for any \( i \),

then \( \text{box}_{y, B_i}^r(M; \xi_i \leftrightarrow N_i) \sim \text{box}_{y, B_i}^r(M'; \xi_i \leftrightarrow N'_i) \in [\text{hcom}_{\text{Kan}}(A; \xi_i \leftrightarrow y.B_i)][\Psi] \).

Proof. We focus on the unary case; the binary case follows similarly. For any \( \psi_1 : \Psi_1 \rightarrow \Psi \) and \( \psi_2 : \Psi_2 \rightarrow \Psi_1 \) we must show \( \text{box}_{\psi_1} \downarrow X_1 \) and \( [\text{hcom}]_{\psi_1 \psi_2}^{\downarrow} (X_1 \psi_2, \text{box}_{\psi_1} \psi_2) \). We proceed by cases on the first step taken by \( \text{box}_{\psi_1} \) and \( \text{box}_{\psi_1} \psi_2 \).
1. \( r \psi_1 = r' \psi_1 \).

Then \( \text{box}_1 \vdash M A \psi_1, \text{hcom}_1 \psi_1 \psi_2 = [A] \psi_1 \psi_2 \) by Lemma 4.88, and by \( M \in A [\Psi] \), \( [A] \psi_1 \psi_2 (X_1 \psi_2, M \psi_1 \psi_2) \).

2. \( r \psi_1 \neq r' \psi_1, \xi_j \psi_1 \) holds (where this is the least such \( j \)), and \( r \psi_1 \psi_2 = r' \psi_1 \psi_2 \).

Then \( \text{box}_1 \vdash N \psi_1, \text{box}_1 \psi_1 \psi_2 \vdash M \psi_1 \psi_2, \) and again \( \text{hcom}_1 \psi_1 \psi_2 = [A] \psi_1 \psi_2 \) by Lemma 4.88. By \( B_j \langle \psi \rangle \psi_1 \psi_2 \models A \psi_1 \psi_2 \) type\_Kan \([\Psi]_2 \) and \( N \psi_1 \in B_j \langle \psi \rangle \psi_1 \psi_2 \) \([\Psi]_1 \) at \( \text{id}_{\psi_1, \psi_2} \) we have \( [A] \psi_1 \psi_2 (X_1 \psi_2, N \psi_1 \psi_2) \). We also have \( [A] \psi_1 \psi_2 (N \psi_1 \psi_2, M \psi_1 \psi_2) \) by \( (\text{coh}_{y, B_j}(N)) \psi_1 \psi_2 \models M \psi_1 \psi_2 \in A \psi_1 \psi_2 \) \([\Psi]_2 \) and \( (\text{coh}_{y, B_j}(N)) \psi_1 \psi_2 \models N \psi_1 \psi_2 \in A \psi_1 \psi_2 \) \([\Psi]_2 \); the result follows by transitivity.

3. \( r \psi_1 \neq r' \psi_1, \xi_j \psi_1 \) holds (least such), \( r \psi_1 \psi_2 \neq r' \psi_1 \psi_2 \), and \( \xi_j \psi_1 \psi_2 \) holds (least such).

Then \( \text{box}_1 \vdash N \psi_1, \text{box}_1 \psi_1 \psi_2 \vdash N \psi_1 \psi_2, \) and \( \text{hcom}_1 \psi_1 \psi_2 = [B_j \langle \psi \rangle \psi_1 \psi_2] \psi_1 \psi_2 \) by Lemma 4.88. The result follows by \( N \psi_1 \in B_j \langle \psi \rangle \psi_1 \psi_2 \) \([\Psi]_1 \) and \( N \psi_1 \psi_2 \models N \psi_1 \psi_2 \in B_j \langle \psi \rangle \psi_1 \psi_2 \) \([\Psi]_2 \).

4. \( r \psi_1 \neq r' \psi_1, \neg \xi_i \psi_1 \) for all \( i \), and \( r \psi_1 \psi_2 = r' \psi_1 \psi_2 \).

Then \( \text{box}_1 \text{val}, \text{box}_1 \psi_1 \psi_2 \models M \psi_1 \psi_2, \) \( \text{hcom}_1 \psi_1 \psi_2 = [A] \psi_1 \psi_2 \) by Lemma 4.88, and the result follows by \( M \in A [\Psi] \).

5. \( r \psi_1 \neq r' \psi_1, \neg \xi_i \psi_1 \) for all \( i \), \( r \psi_1 \psi_2 \neq r' \psi_1 \psi_2 \), and \( \xi_j \psi_1 \psi_2 \) holds (least such).

Then \( \text{box}_1 \text{val}, \text{box}_1 \psi_1 \psi_2 \models N \psi_1 \psi_2, \) \( \text{hcom}_1 \psi_1 \psi_2 = [B_j \langle \psi \rangle \psi_1 \psi_2] \psi_1 \psi_2 \) by Lemma 4.88, and the result follows by \( N \psi_1 \psi_2 \models B_j \langle \psi \rangle \psi_1 \psi_2 \) \([\Psi]_2 \).

6. \( r \psi_1 \neq r' \psi_1, \neg \xi_i \psi_1 \) for all \( i \), and \( r \psi_1 \psi_2 \neq r' \psi_1 \psi_2 \), and \( \neg \xi_j \psi_1 \psi_2 \) for all \( j \).

Then \( \text{box}_1 \text{val} \) and \( \text{box}_1 \psi_1 \psi_2 \text{ val}; \) the result follows by the definition of \( \text{hcom}_1 \). □

**Rule 4.90** (Pretype formation). If \( A, \xi_i \leftrightarrow y.B_i \) and \( A', \xi_i \leftrightarrow y.B'_i \) are equal Kan type compositions \( r \rightsquigarrow r' \), then

1. \( \text{hcom}_{\text{pre}}(A; \xi_i \leftrightarrow y.B_i) \models \text{hcom}_{\text{pre}}(A'; \xi_i \leftrightarrow y.B'_i) \text{ type}_\text{pre} [\Psi] \),
2. if \( r = r' \) then \( \text{hcom}_{\text{pre}}(A; \xi_i \leftrightarrow y.B_i) \models A \text{ type}_\text{pre} [\Psi] \), and
3. if \( \xi_i \) then \( \text{hcom}_{\text{pre}}(A; \xi_i \leftrightarrow B_i) \models B_i \langle \psi \rangle \text{ type}_\text{pre} [\Psi] \).

**Proof.** Parts (2–3) hold immediately by Lemma 4.88. For part (1), by Lemma 4.88 it suffices to show \( \text{Coh}([\text{hcom}_1]) \). Let \( \psi : \Psi' \to \Psi \) and \( [\text{hcom}_1 \psi]_0 (M_0, N_0) \). If \( r \psi = r' \psi \) then \( M_0 \sim N_0 \in [\text{hcom}_1 \psi]_0 [\Psi'] \) by \( [\text{hcom}_1 \psi]_0 = [A] \psi \) and \( \text{Coh}([A]) \). If \( \xi_i \psi \) holds for some \( i \), then \( M_0 \sim N_0 \in [\text{hcom}_1 \psi]_0 [\Psi'] \) by \( [\text{hcom}_1 \psi]_0 = [B_j \langle \psi \rangle \psi_1 \psi_2] \) and \( \text{Coh}([B_j \langle \psi \rangle \psi_1 \psi_2]) \). If \( r \psi \neq r' \psi \) and \( \neg \xi_i \psi \) for all \( i \), then \( M_0 \) and \( N_0 \) are boxes and the result holds by Lemma 4.89. □
Rule 4.91 (Introduction). If $A, \overline{\xi_i \leftarrow y. B_i}$ is a Kan type composition $r \rightsquigarrow r'$,

1. $M \doteq M' \in A[\Psi]$,
2. $N_i \doteq N'_j \in B_1(r'/y) [\Psi \mid \xi_i, \xi_j]$ for any $i, j$, and
3. $\text{coe}^{r \rightsquigarrow r'}_{y. B_i}(N_i) \doteq M \in A[\Psi \mid \xi_i]$ for any $i$.

then

1. $\text{box}^{r \rightsquigarrow r'}(M; \overline{\xi_i \leftarrow N_i}) \doteq \text{box}^{r \rightsquigarrow r'}(M'; \overline{\xi_i \leftarrow N'_i}) \in \text{hcom}_{q_{\text{kan}}}^{r \rightsquigarrow r'}(A; \overline{\xi_i \leftarrow y. B_i}) [\Psi]$;
2. if $r = r'$ then $\text{box}^{r \rightsquigarrow r'}(M; \overline{\xi_i \leftarrow N_i}) \doteq M \in A[\Psi]$; and
3. if $\xi_i$ then $\text{box}^{r \rightsquigarrow r'}(M; \overline{\xi_i \leftarrow N_i}) \doteq N_i \in B_1(r'/y) [\Psi]$.

Proof. Part (1) holds by Lemma 4.89 and Rule 4.90; part (2) holds by Lemma 4.32. For part (3), if $r = r'$, the result holds by Lemma 4.32. Otherwise, there is a least $j$ such that $\xi_j$ holds, and we apply coherent expansion to the left with family

$$
\begin{aligned}
M \psi & \quad r \psi = r' \psi \\
N_k \psi & \quad r \psi \neq r' \psi, \, \xi_k \psi, \text{ and } \forall k' < k. \neg \xi_k \psi.
\end{aligned}
$$

If $r \psi = r' \psi$ then $M \psi \doteq N_j \psi \in B_1(r'/y) \psi [\Psi']$ by $M \psi \doteq (\text{coe}^{r \rightsquigarrow r'}_{y. B_j}(N_j)) \psi \in A \psi [\Psi']$, $(\text{coe}^{r \rightsquigarrow r'}_{y. B_j}(N_j)) \psi \doteq N_j \psi \in B_1(r'/y) \psi [\Psi']$, and $B_1(r'/y) \psi \doteq A \psi \text{type}_{\text{kan}} [\Psi']$. If $r \psi \neq r' \psi$ then $N_k \psi \doteq N_j \psi \in B_1(r'/y) \psi [\Psi']$ by $N_k \psi \doteq N_j \psi \in B_1(r'/y) \psi [\Psi']$ and $B_1 \psi \doteq B_j \psi \text{type}_{\text{kan}} [\Psi', y]$. Thus by Lemma 4.31 we have $\text{hcom} \doteq N_j \in B_1(r'/y) [\Psi]$, and part (3) follows by $N_j \doteq N_i \in B_1(r'/y) [\Psi]$. □

Rule 4.92 (Elimination). If $A, \overline{\xi_i \leftarrow y. B_i}$ and $A, \overline{\xi_i \leftarrow y. B'_i}$ are equal Kan type compositions $r \rightsquigarrow r'$ and $M \doteq M' \in \text{hcom}_{q_{\text{kan}}}^{r \rightsquigarrow r'}(A; \overline{\xi_i \leftarrow y. B_i}) [\Psi]$, then

1. $\text{cap}^{r \rightsquigarrow r'}(M; \overline{\xi_i \leftarrow y. B_i}) \doteq \text{cap}^{r \rightsquigarrow r'}(M'; \overline{\xi_i \leftarrow y. B'_i}) \in A [\Psi]$;
2. if $r = r'$ then $\text{cap}^{r \rightsquigarrow r'}(M; \overline{\xi_i \leftarrow y. B_i}) \doteq M \in A [\Psi]$; and
3. if $\xi_i$ then $\text{cap}^{r \rightsquigarrow r'}(M; \overline{\xi_i \leftarrow y. B_i}) \doteq \text{coe}^{r \rightsquigarrow r'}_{y. B_i}(M) \in A [\Psi]$.

Proof. Part (2) holds by Lemma 4.32 and Rule 4.90. For part (3), if $r = r'$ then the result holds by part (2), $B_1 \text{type}_{\text{kan}} [\Psi, y]$, and $B_1(r/y) \doteq A \text{type}_{\text{kan}} [\Psi]$. Otherwise, $r \neq r'$ and there is a least $j$ such that $\xi_j$ holds. Apply coherent expansion to the left with family

$$
\begin{aligned}
M \psi & \quad r \psi = r' \psi \\
\text{coe}^{r \rightsquigarrow r'}_{y. B_k \psi}(M \psi) & \quad r \psi \neq r' \psi, \, \xi_k \psi, \text{ and } \forall i < k. \neg \xi_i \psi
\end{aligned}
$$

When $r \psi = r' \psi$, $(\text{coe}_{y, B_j}^r(M)) \psi \equiv M \psi \in A \psi[\Psi']$ by $M \in B_j(r'/y)[\Psi]$ (by Rule 4.90), $B_j \text{type}_{\text{Kan}}[\Psi', y]$, and $B_j(r/y) \psi \equiv A \psi \text{type}_{\text{Kan}}[\Psi']$. When $r \psi \neq r' \psi$ and $\xi_k \psi$ where $k$ is the least such, we have $(\text{coe}_{y, B_j}^{r \leftarrow r'}(M)) \psi \equiv (\text{coe}_{y, B_k}^{r \leftarrow r'}(M)) \psi \in A \psi[\Psi']$ by $B_j \psi = B_k \psi \text{type}_{\text{Kan}}[\Psi', y]$ and $B_j(r/y) \psi \equiv A \psi \text{type}_{\text{Kan}}[\Psi']$. Thus $\text{cap} \equiv (\text{coe}_{y, B_j}^{r \leftarrow r'}(M)) \equiv A \psi[\Psi]$ by Lemma 4.18, and part (3) follows by $\text{coe}_{y, B_j}^{r \leftarrow r'}(M) \equiv A \psi[\Psi]
abla\:y, M \psi \equiv \text{box}^{r \leftarrow r'}(O \psi; \xi_i \psi \mapsto N_i \psi) \in \text{hcom} \psi[\Psi']$ by Lemma 4.16. Apply coherent expansion to the left with family

$$
\begin{cases}
M \psi & r \psi = r' \psi \\
\text{coe}_{y, B_j}^{r \leftarrow r'}(M) & r \psi \neq r' \psi, \xi_j \psi, \text{and } \forall i < j. \neg \xi \psi \\
O \psi & r \psi \neq r' \psi \text{ and } \forall i. \neg \xi \psi
\end{cases}
$$

where $M \psi \downarrow \text{box}^{r \leftarrow r'}(O \psi; \xi_i \psi \mapsto N_i \psi)$.

When $r \psi = r' \psi$, $M \psi \equiv (O_{\text{id}_{\psi}}) \psi \equiv A \psi[\Psi']$ by $M \psi \equiv \text{box} \psi \in \text{hcom} \psi[\Psi']$, $\text{hcom} \psi \equiv A \psi \text{type}_{\text{Kan}}[\Psi']$ (by Rule 4.90), and $\text{box} \psi \equiv (O_{\text{id}_{\psi}}) \psi \equiv \text{hcom} \psi[\Psi']$ (by Rule 4.91). When $r \psi \neq r' \psi$ and $\xi_j \psi$ where $j$ is the least such, $(O_{\text{id}_{\psi}}) \psi \equiv (\text{coe}_{y, B_j}^{r \leftarrow r'}(M)) \psi \equiv A \psi[\Psi']$ by $M \psi \equiv (N_i; \text{id}_{\psi}) \psi \equiv B_j(r'/y)[\Psi']$ (by Rules 4.90 and 4.91) and $O_{\text{id}_{\psi}} \equiv (\text{coe}_{y, B_j}^{r \leftarrow r'}(N_i; \text{id}_{\psi})) \equiv A \psi[\Psi \mid \xi_j]$. When $r \psi \neq r' \psi$ and $\neg \xi \psi$ for all $i$, $(O_{\text{id}_{\psi}}) \psi \equiv O \psi \equiv A \psi[\Psi]$ by

$$
[\text{hcom}]((\text{box}^{r \leftarrow r'}(O_{\text{id}_{\psi}}; \xi_i \mapsto N_i; \text{id}_{\psi})); \text{box}^{r \leftarrow r'}(O \psi; \xi_i \mapsto N_i \psi))
$$

(by $M \equiv \text{hcom}[\Psi]$ at id_{\psi}, \psi). Therefore $\text{cap} \equiv O_{\text{id}_{\psi}} \equiv A \psi[\Psi]$ by Lemma 4.18, and part (1) follows by a symmetric argument on the right. □

**Rule 4.93 (Computation).** If $A, \xi_i \mapsto y. B_i$ is a Kan type composition $r \leftarrow r'$,

1. $M \equiv M' \in A[\Psi]$
2. $N_i \equiv N'_j \in B_j(r'/y)[\Psi \mid \xi_i, \xi_j]$ for any $i, j$, and
3. $\text{coe}_{y, B_j}^{r \leftarrow r'}(N_i) \equiv M \in A[\Psi \mid \xi_i]$ for any $i$,

then $\text{cap}^{r \leftarrow r'}(\text{box}^{r \leftarrow r'}(M; \xi_i \mapsto N_i); \xi_i \mapsto y.B_i) \equiv M \in A[\Psi]$.

**Proof.** Both terms are elements of $A$ by Rules 4.91 and 4.93, so by Lemma 4.15 it suffices to show $[A]^\theta(\text{cap}, M)$. If $r = r'$ then $\text{cap} \mapsto \text{box} \mapsto M$ and $[A]^\theta(M, M)$. If $r \neq r'$ and $\xi_i$ holds where $i$ is the least such, then $\text{cap} \mapsto \text{coe}_{y, B_i}^{r \leftarrow r'}(\text{box})$, and $[A]^\theta(\text{coe}_{y, B_i}^{r \leftarrow r'}(\text{box}), M)$ by box $\equiv N_i \in B_j(r'/y)[\Psi]$ and $\text{coe}_{y, B_i}^{r \leftarrow r'}(N_i) \equiv M \in A[\Psi]$ if $r \neq r'$ and $\neg \xi_i$ for all $i$, then $\text{cap} \mapsto M$ and $[A]^\theta(M, M)$. □
Rule 4.94 (Uniqueness). If \( A, \xi_i \hookrightarrow y.B_i \) is a Kan type composition \( r \rightsquigarrow r' \) and \( M \in \text{hcom}^{r \rightsquigarrow r'}_{\kappa} (A; \xi_i \hookrightarrow y.B_i) \) [\( \Psi \)], then

\[
\text{box}^{r \rightsquigarrow r'} (\text{cap}^{r \rightsquigarrow r'} (M; \xi_i \hookrightarrow y.B_i); \xi_i \hookrightarrow M) \simeq M \in \text{hcom}^{r \rightsquigarrow r'}_{\kappa} (A; \xi_i \hookrightarrow y.B_i) \) [\( \Psi \)].
\]

Proof. By \( \text{cap}^{r \rightsquigarrow r'} (M; \xi_i \hookrightarrow y.B_i) \in A [\Psi] \) (by Rule 4.92), \( M \in B_i (r'/y) \) [\( \Psi ] \xi_i]) (by Rule 4.90), \( \text{coe}^{r \rightsquigarrow r'} (M) \equiv \text{cap} \in A [\Psi ] \xi_i]) \) (by Rule 4.92), and Rule 4.91, \( \text{box} \in \text{hcom} [\Psi] \). Thus, by Lemma 4.15 it suffices to show \( \| \text{hcom} \| (\text{box}, M) \). If \( r = r' \) then \( \text{box} \hookrightarrow \text{cap} \hookrightarrow M \) and \( \| \text{hcom} \| (M, M) \). If \( r \neq r' \) and \( \neg \xi_i \) for all \( i \), then \( M \| \) \( \text{box}^{r \rightsquigarrow r'} (O; \xi_i \hookrightarrow N_j) \) and \( M \| \text{box}^{r \rightsquigarrow r'} (O; \xi_i \hookrightarrow N_j) \in \text{hcom} [\Psi] \).

The result follows by transitivity and Rule 4.91:

1. \( \text{cap}^{r \rightsquigarrow r'} (M; \xi_i \hookrightarrow y.B_i) \simeq O \in A [\Psi] \) by Lemma 4.15 and \( \hookrightarrow * O \),

2. \( M \simeq N_i \in B_i (r'/y) \) [\( \Psi ] \xi_i) \) by \( M \simeq \text{box}^{r \rightsquigarrow r'} (O; \xi_i \hookrightarrow N_i) \in \text{hcom} [\Psi] \) and Rule 4.91, and

3. \( \text{coe}^{r \rightsquigarrow r'} (M) \equiv \text{cap}^{r \rightsquigarrow r'} (M; \xi_i \hookrightarrow y.B_i) \in A [\Psi ] \xi_i] \) by Rule 4.92 as before. 

Lemma 4.95. If \( A, s_j = s'_j \hookrightarrow z.B_j \) and \( A', s_j = s'_j \hookrightarrow z.B'_j \) are equal Kan type compositions \( s \rightsquigarrow s' \) and, writing \( H := \text{hcom}^{s \rightsquigarrow s'}_{\kappa} (A; s_j = s'_j \hookrightarrow z.B_j) \), if \( \xi_i \) is valid,

1. \( M \simeq M' \in H [\Psi] \),

2. \( N_i \simeq N'_i \in H [\Psi, y \xi_i, \xi_i'] \) for any \( i, i' \), and

3. \( N_i (r/y) \simeq M \in H [\Psi, \xi_i] \) for any \( i \),

then

1. \( \text{hcom}^r_H (M; \xi_i \hookrightarrow y.N_i) \simeq \text{hcom}^{r \rightsquigarrow r'}_{\kappa} (M'; \xi_i \hookrightarrow y.N'_i) \in H [\Psi]; \)

2. if \( r = r' \) then \( \text{hcom}^r_H (M; \xi_i \hookrightarrow y.N_i) \simeq M \in H [\Psi]; \) and

3. if \( \xi_i \) then \( \text{hcom}^r_H (M; \xi_i \hookrightarrow y.N_i) \simeq N_i (r'/y) \in H [\Psi]. \)

Proof. If \( s = s' \) or \( s_j = s'_j \) for some \( j \), the results are immediate by parts (2–3) of Lemma 4.88. Otherwise, \( s \neq s' \) and \( s_j \neq s'_j \) for all \( j \); apply coherent expansion to \( \text{hcom}^r_H \) with the family defined in Figure 4.4. Consider \( \psi = \text{id}_\Psi \).
1. $O_i \equiv O_{i'} \in A[\Psi, y \mid \xi_i, \xi_{i'}]$ for all $i, i'$, by $N_i \doteq N_{i'} \in H[\Psi, y \mid \xi_i, \xi_{i'}]$.

2. $P_j \doteq P_{j'} \in B_j(s'/z)[\Psi, y \mid s_j = s'_j, s_{j'} = s'_{j'}]$ for all $j, j'$, by $H \doteq B_j(s'/z)$ type_{Kan} $[\Psi \mid s_j = s'_j]$ and $B_j(s'/z) \doteq B_{j'}(s'/z)$ type_{Kan} $[\Psi \mid s_j = s'_j, s_{j'} = s'_{j'}].$

3. $Q \in A[\Psi]$ by
   a) $O_i \equiv O_{i'} \in A[\Psi, y \mid \xi_i, \xi_{i'}]$ for all $i, i'$,
   b) $\text{coe}_{s_{B_j}}(P_j) \doteq \text{coe}_{s_{B_j'}}(P_{j'}) \in A[\Psi, y \mid s_j = s'_j, s_{j'} = s'_{j'}]$ for all $j, j'$, by $P_j \doteq P_{j'} \in B_j(s'/z)[\Psi, y \mid s_j = s'_j, s_{j'} = s'_{j'}], B_j \doteq B_{j'}$ type_{Kan} $[\Psi \mid s_j = s'_j, s_{j'} = s'_{j'}]$, and $B_j(s'/z) \doteq A$ type_{Kan} $[\Psi \mid s_j = s'_j]$,
   c) $\text{hcom}_{A}(M; \xi_i \rightarrow y.N_i) \in A[\Psi, y \mid s = s']$ by $H \doteq A$ type_{Kan} $[\Psi \mid s = s'].$
   d) $\text{cap}_{s_{B_j}}(M; s_j = s'_j \rightarrow z.B_j) \in A[\Psi]$ by $M \in H[\Psi]$.
   e) $O_i(r/y) \doteq \text{cap}_{s_{B_j}}(M; s_j = s'_j \rightarrow z.B_j) \in A[\Psi \mid \xi_i]$ for all $i$, by $N_i(r/y) \doteq M \in H[\Psi \mid \xi_i]$.
   f) $\text{coe}_{s_{B_j}}(P_j(r/y)) \doteq \text{coe}_{s_{B_j}}(M; s_j = s'_j \rightarrow z.B_j) \in A[\Psi \mid s_j = s'_j]$ for all $j$, by $P_j(r/y) \doteq M \in A[\Psi \mid s_j = s'_j]$ and $\text{cap}_{s_{B_j}}(M; s_j = s'_j \rightarrow z.B_j) \doteq \text{coe}_{s_{B_j}}(M) \in A[\Psi \mid s_j = s'_j]$.
   g) $\text{hcom}_{A}(M; \xi_i \rightarrow y.N_i) \doteq \text{cap}_{s_{B_j}}(M; s_j = s'_j \rightarrow z.B_j) \in A[\Psi \mid s = s']$ because both $\doteq M$.
   h) $O_i \equiv \text{coe}_{s_{B_j}}(P_j) \in A[\Psi, y \mid \xi_i, s_j = s'_j]$ for all $i, j$ because both $\doteq \text{coe}_{s_{B_j}}(N_i)$,
i) $O_i = \text{hcom}_{A}^{r \rightsquigarrow y}(M; \xi_i \mapsto y.N_i) \in A[\Psi, y | \xi_i, s = s']$ for all $i$ because both $\equiv N_i$.

j) $\text{coe}^{r \rightsquigarrow y}_{z.B_j}(P_j) \equiv \text{hcom}_{A}^{r \rightsquigarrow y}(M; \xi_i \mapsto y.N_i) \in A[\Psi, y | s_j = s'_j, s = s']$ for all $j$,

by $\text{coe}^{r \rightsquigarrow y}_{z.B_j}(P_j) \equiv \text{hcom}_{A}^{r \rightsquigarrow y}(M; \xi_i \mapsto y.N_i) \in A[\Psi, y | r \mapsto B_j(s'/z) \equiv A \text{type}_{\text{Kan}}[\Psi | s_j = s'_j, s = s']$ and $B_j(s'/z) \equiv A \text{type}_{\text{Kan}}[\Psi | s_j = s'_j, s = s']$.

4. $\text{box}^{r \rightsquigarrow y}(Q; s_j = s'_j \mapsto P_j(r'/y)) \in H[\Psi]$ because $Q \in A[\Psi], P_j(r'/y) \equiv P_j(r'/y) \in B_j(s'/z) [\Psi | s_j = s'_j, j']$ for all $j'$, and $\text{coe}^{r \rightsquigarrow y}_{z.B_j}(P_j(r'/y)) \equiv Q \in A[\Psi | s_j = s'_j]$ for all $j$.

When $s \psi \neq s' \psi$ and $s_j \psi \neq s'_j \psi$ for all $j$, coherence is immediate. When $s \psi = s' \psi$, $\text{box} \psi \equiv Q \psi \equiv (\text{hcom}_{A}^{r \rightsquigarrow y}(M; \xi_i \mapsto y.N_i)) \psi$. When $s \psi \neq s' \psi$ and $s_j \psi = s'_j \psi$ for the least such $j$, $\text{box} \psi \equiv P_j(r'/y) \psi = (\text{hcom}_{A}^{r \rightsquigarrow y}(M; \xi_i \mapsto y.N_i)) \psi$. By Lemma 4.18, $\text{hcom}_{H} \equiv \text{box} \equiv H[\Psi]$; part (1) follows by a symmetric argument on the right.

For part (2), if $r = r'$ then $Q \equiv \text{cap}^{r \rightsquigarrow y}(M; s_j = s'_j \mapsto z.B_j) \in A[\Psi]$ and $P_j(r'/y) \equiv M \in B_j(s'/z) [\Psi | s_j = s'_j]$ for all $j$, so $\text{box}^{r \rightsquigarrow y}(Q; s_j = s'_j \mapsto P_j(r'/y)) \equiv M \in H[\Psi]$ by Rule 4.94.

For part (3), if $\xi_i$ then $Q \equiv O_i(r'/y) = \text{cap}^{r \rightsquigarrow y}(N_j(r'/y); s_j = s'_j \mapsto z.B_j) \in A[\Psi]$ and $P_j(r'/y) \equiv N_j(r'/y)$ for all $j$, so $\text{box} \equiv N_j(r'/y) \in H[\Psi]$ by Rule 4.94.

\begin{lemma}
If $A, \xi_i \mapsto z.B_i$ and $A', \xi_i \mapsto z.B'_i$ are equal Kan type compositions $r \rightsquigarrow r'$ in $(\Psi, x)$ and, writing $H := \text{hcom}^{r \rightsquigarrow y}_{\text{Kan}}(A; \xi_i \mapsto z.B_i)$, we have $M \equiv M' \in H(r'/x) [\Psi]$, then

1. $\text{coe}^{r \rightsquigarrow y}_{x.H}(M') \equiv \text{coe}^{r \rightsquigarrow y}_{x.H}(M') \in H(r'/x) [\Psi]$ and

2. if $r = r'$ then $\text{coe}^{r \rightsquigarrow y}_{x.H}(M) \equiv M \in H(r'/x) [\Psi]$.

\end{lemma}

\begin{proof}
If $s = s'$ or $\xi_i$ for some $i$, the results are immediate by parts (2–3) of Lemma 4.88. Otherwise, $s \neq s'$ and $\xi_i$ for all $i$; apply coherent expansion to $\text{coe}^{r \rightsquigarrow y}_{x.H}(M)$ with the family defined in Figure 4.5. Consider $\psi = \text{id}_\Psi$.

1. $N_i \equiv N_i \in B_i[\Psi, x, z | \xi_i(r/x), \xi_i(r/x)]$ for all $i, j$ by $M \in B_i(s'/z)(r/x) [\Psi | \xi_i(r/x)]$.

2. $O \in A(r/x) [\Psi, z]$ by

a) (cap$_{z.B_i}$)$(r/x) \in A(r/x) [\Psi]$ by $M \in H(r'/x) [\Psi]$. 

\end{proof}
\[
\begin{align*}
&\text{coe}_{x,A}^{r\psi,\omega r}(M\psi) \quad \text{for all } \psi \in T, \quad Q\psi = s'\psi \\
&\text{coe}_{x,B_i(s'/z)}^{r\psi,\omega r}(M\psi) \quad s\psi \neq s'\psi, \text{least } \xi_i\psi \\
&\text{(box)}^{s\omega s'}(R; \xi_i \leftarrow Q_i(s'/z))(r'/x)\psi
\end{align*}
\]

\[
\begin{align*}
&N_i := \text{coe}_{x,B_i}^{s\omega s}(\text{coe}_{x,B_i}^{r\omega x}(M)) \\
&O := (\text{hcom}_{A}^{s\omega s}((\text{cap}_{x,A}^{r\omega x}(M; \xi_i \leftarrow z.B_i)); \xi_i \leftarrow z.\text{coe}_{x,B_i}^{s\omega s}(N_i)))(r/x) \\
P := \text{gcom}_{x,A}^{r\omega r}(O(s(r/x)/z); \xi_i \leftarrow x.N_i(s/z)|_{(x\neq \xi_i)}, T) \\
&T := s = s' \leftarrow x.\text{coe}_{x,A}^{r\omega x}(M)|_{(x\neq s',s')} \\
&Q_k := \text{gcom}_{x,A}^{s(r'/x)\omega s}(P; \xi_i \leftarrow z.N_i(r'/x)|_{(x\neq \xi_i)}, r = r' \leftarrow z.N_k(r'/x)) \\
&R := \text{hcom}_{A}^{s\omega s}(P; \xi_i \leftarrow z.\text{coe}_{x,B_i}^{s\omega s}(Q_i), r = r' \leftarrow z.O)
\end{align*}
\]

Figure 4.5: Family of reducts in Lemma 4.96.

3. \(P \in A(r'/x) [\Psi] \) by

a) \(O(s(r/x)/z) \in A(r/x) [\Psi], \)

b) \(N_i(s/z) = N_j(s/z) \in A[\Psi, x | \xi_i, \xi_j] \) for all \(i, j \) such that \(x \neq \xi_i, \xi_j \) by \(N_i(s/z) = N_j(s/z) \in B_i(s/z) [\Psi, x | \xi_i(r/x), \xi_j(r/x)] \) and \(B_i(s/z) = A \text{ type}_\text{Kan} [\Psi, x | \xi_i], \)

c) \(\text{coe}_{x,A}^{r\omega x}(M) \in A[\Psi, x | s = s'] \) if \(x \neq s, s' \) by \(H(r/x) = A(r/x) \text{ type}_\text{Kan} [\Psi | s(r/x) = s'(r/x)], \)

d) \(O(s(r/x)/z) = N_i(s/z)(r/x) \in A(r/x) [\Psi | \xi_i] \) for all \(i \) such that \(x \neq \xi_i \) by \(O(s(r/x)/z) = (\text{coe}_{x,B_i}^{s\omega s}(N_i))(s/z)(r/x) = N_i(s/z)(r/x) \in A(r/x) [\Psi | \xi_i(r/x)], \)

e) \(O(s(r/x)/z) \neq (\text{coe}_{x,A}^{r\omega x}(M))(r/x) \in A(r/x) [\Psi | s = s'] \) if \(x \neq s, s' \) because \(O(s(r/x)/z) = O(s/z) = \text{cap}(r/x) = M \in A(r/x) [\Psi | s = s'], \)

f) \(N_i(s/z) = \text{coe}_{x,A}^{s\omega s}(M) \in A[\Psi, x | \xi_i, s = s'] \) for all \(i \) such that \(x \neq \xi_i, s, s' \) by \(N_i(s/z) = \text{coe}_{x,B_i}^{s\omega x}(M) \in B_i(s'/z) [\Psi, x | \xi_i, s = s'] \) and \(B_i(s'/z) = A \text{ type}_\text{Kan} [\Psi, x | \xi_i], s = s'. \)

4. \(Q_k = Q_{k'} \in B_k(r'/x) [\Psi, z | \xi_k(r'/x), \xi_{k'}(r'/x)] \) for all \(k, k' \) by
5. $R(r'/x) \in A(r'/x) [\Psi]$ by

a) $P \in A(r'/x) [\Psi]$, 
b) $\text{coe}^{\text{z-as}(r'/x)}_{\text{z,B}_{j}(r'/x)}(Q_i) \equiv \text{coe}^{\text{z-as}(r'/x)}_{\text{z,B}_{j}(r'/x)}(Q_i) \in A(r'/x) [\Psi, z | \xi_i(r'/x), \xi_j(r'/x)]$ for all $i, j$ by $B_i \equiv B_j \text{typeKan} [\Psi, z | \xi_i, \xi_j]$ and $B_i(s/z)(r'/x) \equiv A(r'/x) \text{typeKan} [\Psi | \xi_i(r'/x)]$, 
c) $O \in A(r'/x) [\Psi, z | r = r']$, 
d) $P \equiv (\text{coe}^{\text{z-as}(Q_i)}_{\text{z,B}_{i}(r'/x)})(s/z)(r'/x) \in A(r'/x) [\Psi | \xi_i(r'/x)]$ for all $i$ by $Q_i(s/z)(r'/x) \equiv P \in B_i(s/z)(r'/x) [\Psi | \xi_i(r'/x)]$ and $B_i(s/z)(r'/x) \equiv A(r'/x) \text{typeKan} [\Psi | \xi_i(r'/x)]$, 
e) $P \equiv O(s/z)(r'/x) \in A(r'/x) [\Psi | r = r']$ by $O(s/z)(r'/x) = O(s(r'/x)/z)$, and 
f) $(\text{coe}^{\text{z-as}(Q_i)})(r'/x) \equiv O(r'/x) \in A(r'/x) [\Psi, z | \xi_i(r'/x), r = r']$ for all $i$ by $O(r'/x) = O((\text{coe}^{\text{z-as}(Q_i)})(r'/x) \in A(r'/x) [\Psi, z | \xi_i(r'/x)]$ and $Q_i(r'/x) \equiv N_i(r'/x) \in A(r'/x) [\Psi, z | \xi_i(r'/x), r = r']$.

6. $\text{box}^{(r'/x)\rightarrow (r'/x)}(R(r'/x); \xi_i(r'/x) \leftrightarrow Q_i(s^{r'/x}/z)) \in H(r'/x) [\Psi]$ by

a) $R(r'/x) \in A(r'/x) [\Psi]$, 
b) $Q_i(s^{r'/x}/z) \equiv Q_j(s^{r'/x}/z) \in B_i(s'/z)(r'/x) [\Psi | \xi_i(r'/x), \xi_j(r'/x)]$ for all $i, j$, and 
c) $(\text{coe}^{\text{z-as}(Q_i)(s'/z)})(r'/x) \equiv R(r'/x) \in A(r'/x) [\Psi | \xi_i(r'/x)]$ for all $i$ because $R(r'/x) \equiv (\text{coe}^{\text{z-as}(Q_i)})(s'/z)(r'/x) \in A(r'/x) [\Psi | \xi_i(r'/x)]$.

Consider $\psi : \Psi' \rightarrow \Psi$. When $s\psi \neq s'\psi$ and $-\xi_i\psi$ for all $i$, coherence is immediate. When $s\psi = s'\psi$, then by $s \neq s'$, we must have $x \not\equiv s, s'$ and thus $s^{r'/x}\psi = s^{r'/x}\psi$. 

also. Thus box(\(r'/x\))\(\psi \doteq R(r'/x)\)\(\psi \doteq P(r'/x)\)\(\psi \doteq (\text{coe}^{r \rightarrow x}_{x} A(M))(r'/x)\)\(\psi \in A(r'/x)\) [\(\Psi'\)] as required. When \(\psi \neq s'\psi\) and \(\xi_i\psi\) for the least such \(i\), again \(\alpha \neq \xi_i\) and box(\(r'/x\))\(\psi \doteq Q_i(s'/z)(r'/x)\)\(\psi \doteq (\text{coe}^{r \rightarrow x}_{x} A(s'/z))(M)\)\(\psi \in A(r'/x)\) [\(\Psi'\)]. By Lemma 4.18, \(\text{coe}^{r \rightarrow x}_{x} H(M)\) \(\doteq\) box(\(r'/x\)) \(\in H(r'/x)\) [\(\Psi'\)]; part (1) follows by a symmetric argument.

For part (2), if \(r = r'\) then \(R(r'/x) = O(s'/z)(r'/x) = (\text{cap}^{s \rightarrow s'}_{s} (M; \xi_i \rightarrow z. B_i))(r'/x)\) \(\in A(r'/x)\) [\(\Psi\)] and \(Q_i(s'/z)(r'/x) = N_i(s'/z)(r'/x) = M \in B_i(s'/z)(r'/x)\) [\(\Psi \mid \xi_i(r'/x)\)] for all \(i\), so box(\(r'/x\)) \(\doteq M \in H(r'/x)\) [\(\Psi\)] by Rule 4.94, and part (2) follows by transitivity. \(\square\)

**Rule 4.97** (Kan type formation). If \(A, \xi_i \rightarrow y. B_i\) and \(A', \xi_i \rightarrow y. B'_i\) are equal Kan type compositions \(r \rightarrow r'\), then

1. \(\text{hcom}^{r \rightarrow r'}_{\text{Kan}}(A; \xi_i \rightarrow y. B_i) = \text{hcom}^{r \rightarrow r'}_{\text{Kan}}(A'; \xi_i \rightarrow y. B'_i)\) \(\text{type}_{\text{Kan}}\) [\(\Psi\)].
2. if \(r = r'\) then \(\text{hcom}^{r \rightarrow r} \left( A; \xi_i \rightarrow y. B_i \right) = A \) \(\text{type}_{\text{Kan}}\) [\(\Psi\)], and
3. if \(\xi_i\) then \(\text{hcom}^{r \rightarrow r'}_{\text{Kan}}(A; \xi_i \rightarrow B_i) = B_i(r'/y)\) \(\text{type}_{\text{Kan}}\) [\(\Psi\)].

**Proof.** We already showed parts (2–3) in Lemma 4.88. For part (1), the hcom conditions hold by Lemma 4.95 instantiated at all \(\psi : \Psi' \rightarrow \Psi\) instances of this Kan type composition; the coe conditions hold by Lemma 4.96 at the \(\psi : (\Psi', x) \rightarrow \Psi\) instances. \(\square\)

### 4.4.12 Universes

For \(i \in \{0, 1, \ldots\}\) and \(j \in \{i + 1, i + 2, \ldots, \omega\}\):

\[
\tau^\Psi_j(\Psi, \mathcal{U}_i^\Psi, \mathcal{U}_i^\xi, \{(A_0, B_0) \mid \exists \varphi. \tau^\Psi_i(\Psi, A_0, B_0, \varphi)\})
\]

**Rule 4.98** (Pretype formation). For \(i \in \{0, 1, \ldots\}\) and \(j \in \{i + 1, i + 2, \ldots, \omega\}\), \(\tau^\Psi_j \models (\mathcal{U}_i^\Psi \text{ type}_{\Psi}')\).

**Proof.** We have \(\mathcal{U}_i^\Psi \sim \mathcal{U}_i^\xi \downarrow \llbracket \mathcal{U}_i^\Psi \rrbracket \in \tau^\Psi_j\) [\(\Psi\)] by \(\mathcal{U}_i^\Psi \text{ val}_{\Psi}\) and the definition of \(\tau^\Psi_j\). To prove \(\text{Coh}(\llbracket \mathcal{U}_i^\Psi \rrbracket)\), suppose that \(\llbracket \mathcal{U}_i^\Psi \rrbracket_{\Psi'}(A_0, B_0)\) holds and show \(A_0 \sim B_0 \in \llbracket \mathcal{U}_j^\Psi \rrbracket\) [\(\Psi'\)]. Then \(\tau^\Psi_i(\Psi', A_0, B_0, \varphi)\) for some \(\varphi\), and thus \(A_0 \sim B_0 \downarrow \alpha \in \tau^\Psi_i(\Psi')\) for some \(\alpha\) by the value-coherence property of \(\tau^\Psi_i\) (Theorem 4.9), completing our proof. \(\square\)

**Rule 4.99** (Cumulativity).

1. If \(\tau^\Psi_{i, \omega} \models (A \doteq B \in \mathcal{U}_i^\Psi\) [\(\Psi\)] and \(i \leq j\) then \(\tau^\Psi_{i, \omega} \models (A \doteq B \in \mathcal{U}_j^\Psi\) [\(\Psi\)].
2. If \(\tau^\Psi_{i, \omega} \models (A \doteq B \in \mathcal{U}_i^\text{Kan}\) [\(\Psi\)] then \(\tau^\Psi_{i, \omega} \models (A \doteq B \in \mathcal{U}_i^\text{pre}\) [\(\Psi\)].
Proof. For part (1), by Theorem 4.10, \( \tau_i^\alpha \subseteq \tau_j^\alpha \) and therefore \( \| \mathcal{U}_i^\alpha \| \subseteq \| \mathcal{U}_j^\alpha \| \). The result follows by monotonicity of the candidate judgments. For part (2), by Theorem 4.10, \( \tau_i^{\text{Kan}} \subseteq \tau_i^{\text{pre}} \) and the result follows similarly.

As in Chapter 2, we have defined the elements of \( \mathcal{U}_i^\alpha \) inductively; therefore, to see that the elements of \( \mathcal{U}_i^{\text{Kan}} \) we must induct on the definition of \( \tau_i^{\text{Kan}} \).

**Lemma 4.100.** Let \( \Phi(x)(\tau) := \{ (\Psi, A_0, B_0, \varphi) \mid \tau \models (A_0 = B_0 \text{ type}_\alpha [\Psi]) \} \). If \( \tau \) is a cubical type system and \( A \sim B \downarrow \alpha \in \Phi(x)(\tau) [\Psi] \) then \( \tau \models (A \equiv B \text{ type}_\alpha [\Psi]) \).

**Proof.** We apply coherent expansion to \( A \) with family \( \{ A_\psi \mid A_\psi \downarrow A_\psi ^{\psi'} \} \). By our hypothesis at \( \psi, \text{id}_\psi \) we have \( \tau \models (A_\psi \equiv A_\psi ^{\psi'} \text{id}_\psi \text{ type}_\alpha [\Psi']) \).

Suppose \( \kappa = \text{pre} \). We already know \( A_\psi \) and \( (A_\psi ^{\psi'}) \) are pretypes, so by Lemma 4.14 it suffices to show that their values are equal pretypes, which follows from our hypothesis at \( \text{id}_\psi, \psi \). Thus, by Lemma 4.17, \( \tau \models (A \equiv A_0 \text{ type}_{\text{pre}} [\Psi]) \) where \( A \downarrow A_0 \).

Suppose \( \kappa = \text{Kan} \). Both \( A_\psi \) and \( (A_\psi ^{\psi'}) \) are Kan, so by Lemma 4.30 it suffices to show their values under all \( \psi' : \Psi' \to \Psi \) are equally Kan, which follows from our hypothesis at \( \psi, \psi' \) and \( \text{id}_\psi, \psi' \). Thus, by Lemma 4.31, \( \tau \models (A \equiv A_0 \text{ type}_{\text{Kan}} [\Psi]) \) where \( A \downarrow A_0 \).

In each case, we similarly obtain \( \tau \models (B \equiv B_0 \text{ type}_\alpha [\Psi]) \) where \( B \downarrow B_0 \), and the result follows from \( \tau \models (A_0 \equiv B_0 \text{ type}_\alpha [\Psi]) \) by our hypothesis at \( \text{id}_\psi, \text{id}_\Psi \). □

**Lemma 4.101.**

1. If \( \tau_i^{\text{pre}} \models (A \equiv B \in \mathcal{U}_i^{\text{Kan}} [\Psi]) \) then \( \tau_i^{\text{Kan}} \models (A \equiv B \text{ type}_{\text{Kan}} [\Psi]) \).

2. If \( \tau_i^{\text{pre}} \models (A \equiv B \in \mathcal{U}_i^{\text{pre}} [\Psi]) \) then \( \tau_i^{\text{pre}} \models (A \equiv B \text{ type}_{\text{pre}} [\Psi]) \).

**Proof.** Let \( \Phi_i := \{ (\Psi, A_0, B_0, \varphi) \mid \tau_i \models (A_0 \equiv B_0 \text{ type}_\alpha [\Psi]) \} \). Our proof proceeds by mutual strong induction on \( i \).

1. We show \( K(\nu_i, \Phi_i^{\text{Kan}}) \subseteq \Phi_i^{\text{Kan}} \); then \( \tau_i^{\text{Kan}} \subseteq \Phi_i^{\text{Kan}} \), so \( A \sim B \downarrow \alpha \in \Phi_i^{\text{Kan}} [\Psi] \), and part (1) follows from Lemma 4.100. We check each type former separately. Suppose \( \text{Fun}(\Phi_i^{\text{Kan}})(\Psi, (a:A) \to B, (a:A') \to B', \varphi) \). Then \( A \sim A' \downarrow \alpha \in \Phi_i^{\text{Kan}} [\Psi] \), which by Lemma 4.100 implies \( \tau_i^{\text{Kan}} \models (A \equiv A' \text{ type}_{\text{Kan}} [\Psi]) \); similarly, \( \tau_i^{\text{Kan}} \models (a : A \gg B \equiv B' \text{ type}_{\text{Kan}} [\Psi]) \). By Rule 4.42, \( \tau_i^{\text{Kan}} \models ((a:A) \to B \equiv (a:A') \to B' \text{ type}_{\text{Kan}} [\Psi]) \).

Other cases are similar except for universes, where we must show for \( j < i \) that \( \tau_i^{\text{Kan}} \models (\mathcal{U}_j^{\text{type}_{\text{Kan}} [\Psi]}) \). To see that \( \mathcal{U}_j^{\text{pre}} \) admits homogeneous composition, suppose we are given a composition scenario in \( \mathcal{U}_j^{\text{pre}} \) relative to \( \tau_i^{\text{Kan}} \) (and hence, by monotonicity, relative also to \( \tau_i^{\text{pre}} \)). By inductive hypothesis (2), this is a composition of pretypes in \( \tau_j^{\text{pre}} \). By Rule 4.87, \( \text{hcom}_{\mathcal{U}_j^{\text{pre}}} \) of this scenario is a pretype in \( \tau_j^{\text{pre}} \).

Therefore \( \tau_i^{\text{Kan}} \models (\text{hcom}_{\mathcal{U}_j^{\text{pre}}} \in \mathcal{U}_j^{\text{pre}} [\Psi]) \) as required. Composition in \( \mathcal{U}_j^{\text{Kan}} \) follows
by a similar argument, appealing instead to inductive hypothesis (1) and Rule 4.90. Coercion for universes is immediate by \( \text{coe}_{x, U_j^\kappa}(A) \Rightarrow_{\omega} A \).

2. We show \( P(v_i, \tau_i^{\text{Kan}}, \Phi_i^{\text{pre}}) \subseteq \Phi_i^{\text{pre}} \), then \( \tau_i^{\text{pre}} \subseteq \Phi_i^{\text{pre}} \), and part (2) follows from Lemma 4.100. Most cases are analogous to part (1), with a few exceptions: for HKAN, we appeal to inductive hypothesis (1) and Rule 4.90, and for UPRE and UKAN we obtain \( \tau_i^{\text{pre}} \models (U_j^\kappa \text{ type}_{\pre} [\Psi]) \) immediately for all \( j < i \) by Rule 4.98.

\[ \Box \]

**Rule 4.102** (Elimination). If \( \tau_\omega^{\text{pre}} \models (A \equiv B \in U_i^\kappa [\Psi]) \) then \( \tau_\omega^{\text{pre}} \models (A \equiv B \text{ type}_\kappa [\Psi]) \).

**Proof.** By Lemma 4.101, \( \tau_i^{\kappa} \models (A \equiv B \text{ type}_\kappa [\Psi]) \); we obtain \( \tau_\omega^{\text{pre}} \models (A \equiv B \text{ type}_\kappa [\Psi]) \) from \( \tau_i^{\kappa} \subseteq \tau_\omega^{\text{pre}} \) (by Theorem 4.10) and monotonicity of the pretype and Kan type judgments. \( \Box \)

**Rule 4.103** (Introduction). Relative to \( \tau_\omega^{\text{pre}} \):

1. If \( A \equiv A' \in U_i^\kappa [\Psi] \) and \( a : A \gg B \equiv B' \in U_i^\kappa [\Psi] \) then \( (a : A) \rightarrow B \equiv (a : A') \rightarrow B' \in U_i^\kappa [\Psi] \).
2. If \( A \equiv A' \in U_i^\kappa [\Psi] \) and \( a : A \gg B \equiv B' \in U_i^\kappa [\Psi] \) then \( (a : A) \times B \equiv (a : A') \times B' \in U_i^\kappa [\Psi] \).
3. If \( A \equiv A' \in U_i^\kappa [\Psi, x] \) and \( P_\epsilon = P_\epsilon' \in A[\epsilon/x] [\Psi] \) for \( \epsilon \in \{0, 1\} \), then Path\(_{x:A}(P_0, P_1) \equiv \text{Path}_{x:A}(P_0, P_1)' \in U_i^\kappa [\Psi] \).
4. If \( A \equiv A' \in U_i^{\text{pre}} [\Psi], M \equiv M' \in A [\Psi], \) and \( N \equiv N' \in A [\Psi], \) then Eq\(_A(M, N) \equiv \text{Eq}_{A'}(M', N') \in U_i^{\text{pre}} [\Psi] \).
5. void \( \in U_i^\kappa [\Psi] \).
6. bool \( \in U_i^\kappa [\Psi] \).
7. nat \( \in U_i^\kappa [\Psi] \).
8. \( \mathbb{S}^1 \in U_i^\kappa [\Psi] \).
9. If \( A \equiv A' \in U_i^\kappa [\Psi \mid r = 0], B \equiv B' \in U_i^\kappa [\Psi], \) and \( E \equiv E' \in \text{Equiv}(A, B) [\Psi \mid r = 0] \), then \( \forall_r(A, B, E) \equiv \forall_r(A', B', E') \in U_i^\kappa [\Psi] \).
10. If \( j > i \) then \( U_i^\kappa \subseteq U_j^{\text{pre}} [\Psi] \).

**Proof.** We establish the presuppositions of parts (1–4) and (9) by Rule 4.102. In part (1), for example, from Rule 4.102 and \( \tau_\omega^{\text{pre}} \models (A \in U_j^\kappa [\Psi]) \) we obtain the presupposition \( \tau_\omega^{\text{pre}} \models (A \text{ type}_\kappa [\Psi]) \) of \( \tau_\omega^{\text{pre}} \models (a : A \gg B \equiv B' \in U_j^\kappa [\Psi]) \).

For part (1), by \( \tau_\omega^{\text{pre}} \models (A \equiv A' \in U_j^\kappa [\Psi]) \) and Lemma 4.101, \( \tau_j^{\kappa} \models (A \equiv A' \text{ type}_\kappa [\Psi]) \).

By \( \tau_\omega^{\text{pre}} \models (a : A \gg B \equiv B' \in U_j^\kappa [\Psi]) \), Lemma 4.101, and the fact that \( \tau_\omega^{\text{pre}} \) and \( \tau_j^{\kappa} \) give rise
to equal \( \llbracket A \rrbracket \) (by Theorem 4.10 and the unicity property of type systems), \( \tau_i^\kappa \models (a : A \to B \equiv B' : \text{type}_\kappa [\Psi]) \). Therefore, by Rule 4.36, \((a:A) \to B \sim (a:A') \to B' \downarrow \in \tau_i^\kappa [\Psi] \) and thus \((a:A) \to B \sim (a:A') \to B' \in \llbracket \mathcal{U}_i^\kappa \rrbracket [\Psi] \), completing part (1). Parts (2–10) are analogous. □

**Rule 4.104** (Kan type formation). \( \tau_\omega^{\text{pre}} \models (\mathcal{U}_i^\kappa : \text{type}_{\text{Kan}} [\Psi]) \).

*Proof.* By Rule 4.103, \( \tau_\omega^{\text{pre}} \models (\mathcal{U}_i^\kappa \in \mathcal{U}_i^{\text{Kan}} [\Psi]) \); the result follows by Rule 4.102. □

**Theorem 4.105** (Univalence). There exist terms \( \mathsf{ua}_\omega \) and \( \mathsf{ua}_\beta \) such that, relative to \( \tau_\omega^{\text{pre}} \), and for all \( i \in \{0, 1, \ldots \} \):

\[
\mathsf{ua}_\omega \in (A:B; \mathcal{U}_i^{\text{Kan}}) \to \text{Equiv}(A, B) \to \text{Path}_{\text{Kan}}(A, B) [\cdot]
\]

\[
\mathsf{ua}_\beta \in (A:B; \mathcal{U}_i^{\text{Kan}}) \to (e: \text{Equiv}(A, B)) \to (a:A) \to \text{Path}_{\text{Kan}}(\text{coe}_{\mathsf{ua}_\omega e}^0 \Rightarrow_1 \text{coe}_{\mathsf{ua}_\omega e}^1, \text{fst}(e) a) [\cdot]
\]

Therefore, every \( \mathcal{U}_i^{\text{Kan}} \) is a univalent universe \([\text{Lic16}]\).

*Proof.* Define:

\[
\mathsf{ua}_\omega A B e := \langle x \rangle \mathcal{V}_x(A, B, e)
\]

\[
\mathsf{ua}_\beta A B e a := \langle x \rangle \text{coe}^0_{\mathsf{ua}_\omega e}^1 (\text{fst}(e) a)
\]

Using previously-proven rules, one can routinely verify the required typing judgments. The only obstacle is to see that the left endpoint of \( \mathsf{ua}_\beta \) is correct, which requires unfolding the definition of coercion in \( \mathcal{V} \)-types. Using the variable names of Figure 4.2:

\[
\text{coe}^0_{\mathsf{ua}_\omega e}^1 (a) \equiv \text{coe}^0_{\mathsf{ua}_\omega e}^1 (\text{Vin}_1(O, P)) \\
\equiv \text{hcom}^0_{\mathsf{ua}_\beta} (N; \ldots, 1 \mapsto \_N, \ldots) \\
\equiv \text{coe}^0_{\mathsf{ua}_\beta} (\text{Vproj}_0(a, \text{fst}(e))) \\
\equiv \text{coe}^0_{\mathsf{ua}_\beta} (\text{fst}(e) a) \in B [\cdot]
\]

□
Computational-style rules

This appendix proposes rules for Cartesian cubical computational type theory. As discussed in Section 2.4, computational type theories such as Nuprl [Con+85] and Idealized Nuprl (Figure 2.4) are deductive systems for establishing judgments in a computational semantics, characterized by their inclusion of equality reflection and untyped computation rules. Readers can experiment with Cartesian cubical computational type theory in the RedPRL proof assistant [Ang+18]. Notably, RedPRL derives less benefit from untyped computation rules than Nuprl: cubically-unstable evaluation is sound only in an empty context (Lemma 4.16), and path endpoint projections (Rule 4.50) require typing information.

The judgments below were defined in Section 4.3, with the exception of ⇓, ↦−→, and ψ : Ψ′ → Ψ, defined in Section 4.1. Contexts Ψ and Ξ are unordered, and equations in Ξ are symmetric; κ ranges over {pre, Kan} and ε ranges over {0, 1}. Each rule is annotated with the lemma containing its proof, and all judgments are relative to the cubical type system τpreω defined in Section 4.2. We suppress ambient hypotheses in these rules for clarity; every judgment implicitly contains additional hypotheses □ except when specifically noted. Metavariables marked with † abbreviate large terms defined in Figure 4.2.

Structural rules

\[
\begin{align*}
\frac{A \in \mathcal{U}_i^\kappa \ [\Psi]}{a : A \gg a \in A \ [\Psi]} \quad & \quad (4.23) \\
\frac{M \models M' \in A \ [\Psi]}{M \psi \models M' \psi \in A \psi \ [\Psi]} \quad & \quad (4.13) \\
\frac{a : A \gg M \models M' \in B \ [\Psi]}{M \models M' \in A' \ [\Psi]} \quad & \quad (4.12) \\
\frac{M' \in A \ [\Psi]}{M \mapsto_{\triangleleft} M' \in A \ [\Psi]} \quad & \quad (4.32) \\
\frac{\cdot \gg M \models M \in A \ [\Psi]}{M \downarrow M_0 \in A \ [\Psi]} \quad & \quad (4.16)
\end{align*}
\]

\^There are some discrepancies between this appendix and RedPRL: the latter interleaves term and interval variables in a single context, and uses sequent calculus–style left rules rather than elimination rules.
\[
\frac{M \doteq M' \in A[\Psi]}{M \doteq M' \in A[\Psi]} \quad (4.27)
\]
\[
\frac{M \doteq M' \in A[\Psi]}{M \doteq M' \in A[\Psi, r = r]} \quad (4.27)
\]
\[
\frac{M \doteq M' \in A[\Psi, 0 = 1]}{M \doteq M' \in A[\Psi, 0 = 1]} \quad (4.27)
\]
\[
\frac{M(r/x) \doteq M'(r/x) \in A(r/x)[\Psi \mid \Xi(r/x)]}{M \doteq M' \in A[\Psi, x \mid \Xi, x = r]} \quad (4.27)
\]

**Kan operations**

\[
\frac{(r = 0), (r = 1) \in \xi_i}{\xi_i \text{ valid}} \quad (4.28)
\]
\[
\frac{(r = r) \in \xi_i}{\xi_i \text{ valid}} \quad (4.28)
\]
\[
\frac{A \in U_k^{\text{Kan}}[\Psi, x]}{M \in A(r/x)[\Psi]} \quad (4.29)
\]
\[
\frac{\text{coer}_{x, A}^{\sim}(M) \in A(r'/x)[\Psi]}{\xi_i \text{ valid}} \quad (4.29)
\]
\[
\frac{\text{hcom}_{A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A[\Psi]}{\xi_i \text{ valid}} \quad (4.29)
\]
\[
\frac{\text{hcom}_{A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A[\Psi]}{\xi_i \text{ valid}} \quad (4.29)
\]
\[
\frac{\text{ghcom}_{y, A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A[\Psi]}{\xi_i \text{ valid}} \quad (4.34)
\]
\[
\frac{\text{com}_{y, A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A(r'/y)[\Psi]}{\xi_i \text{ valid}} \quad (4.33)
\]
\[
\frac{\text{com}_{y, A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A(r'/y)[\Psi]}{\xi_i \text{ valid}} \quad (4.33)
\]
\[
\frac{\text{ghcom}_{y, A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A[\Psi]}{\xi_i \text{ valid}} \quad (4.34)
\]
\[
\frac{\text{ghcom}_{y, A}^{\sim}(M; \xi_i \hookrightarrow y.N_i) \in A[\Psi]}{\xi_i \text{ valid}} \quad (4.34)
\]
Other rules

\[
\frac{A \in \mathcal{U}_i^\kappa [\Psi] \quad a : A \gg B \in \mathcal{U}_i^\kappa [\Psi]}{(a:A) \rightarrow B \in \mathcal{U}_i^\kappa [\Psi]} \quad (4.36) \quad \frac{a : A \gg M \in B [\Psi]}{\lambda a.M \in (a:A) \rightarrow B [\Psi]} \quad (4.37)
\]

\[
\frac{M \in (a:A) \rightarrow B [\Psi] \quad N \in A [\Psi]}{M \circ N \in B[N/a] [\Psi]} \quad (4.39) \quad \frac{M \in (a:A) \rightarrow B [\Psi]}{M \triangleq \lambda a.M \ a \in (a:A) \rightarrow B [\Psi]} \quad (4.40)
\]

\[
\frac{A \in \mathcal{U}_i^\kappa [\Psi] \quad a : A \gg B \in \mathcal{U}_i^\kappa [\Psi]}{(a:A) \times B \in \mathcal{U}_i^\kappa [\Psi]} \quad (4.43) \quad \frac{M \in (a:A) \rightarrow B [\Psi]}{\langle M, N \rangle \in (a:A) \times B [\Psi]} \quad (4.44)
\]

\[
\frac{P \in (a:A) \times B [\Psi]}{\text{fst}(P) \in A [\Psi]} \quad (4.45) \quad \frac{P \in (a:A) \times B [\Psi]}{\text{snd}(P) \in B[\text{fst}(P)/a] [\Psi]} \quad (4.45)
\]

\[
P \in (a:A) \times B [\Psi] \quad P \triangleq \langle \text{fst}(P), \text{snd}(P) \rangle \in (a:A) \times B [\Psi] \quad (4.46)
\]

\[
\frac{A \in \mathcal{U}_i^\kappa [\Psi, x]}{P_0 \in A(\emptyset/x) [\Psi]} \quad \frac{P_1 \in A(1/x) [\Psi]}{\text{Path}_{x:A}(P_0, P_1) \in \mathcal{U}_i^\kappa [\Psi]} \quad (4.48) \quad \frac{M \in A [\Psi, x]}{M(\emptyset/x) \triangleq P_0 \in A(\emptyset/x) [\Psi]} \quad \frac{M(1/x) \triangleq P_1 \in A(1/x) [\Psi]}{\langle x \rangle M \in \text{Path}_{x:A}(P_0, P_1) [\Psi]} \quad (4.49)
\]

\[
\frac{M \in \text{Path}_{x:A}(P_0, P_1) [\Psi]}{M \circ r \in A(r/x) [\Psi]} \quad (4.50) \quad \frac{M \in \text{Path}_{x:A}(P_0, P_1) [\Psi]}{M \circ \epsilon \triangleq P_\epsilon \in A(\epsilon/x) [\Psi]} \quad (4.50)
\]

\[
\frac{M \in \text{Path}_{x:A}(P_0, P_1) [\Psi]}{M \triangleq (x)(M \circ \epsilon) \in \text{Path}_{x:A}(P_0, P_1) [\Psi]} \quad (4.51)
\]

\[
\frac{A \in \mathcal{U}_i^\pre [\Psi]}{M \in A [\Psi] \quad N \in A [\Psi]} \quad \frac{\text{Eq}_A(M, N) \in \mathcal{U}_i^\pre [\Psi]}{\text{Eq}_A(M, N) [\Psi]} \quad (4.53) \quad \frac{M \triangleq N \in A [\Psi]}{\epsilon \in \text{Eq}_A(M, N) [\Psi]} \quad (4.54)
\]

\[
\frac{E \in \text{Eq}_A(M, N) [\Psi]}{M \triangleq N \in A [\Psi]} \quad (4.55) \quad \frac{E \triangleq \epsilon \in \text{Eq}_A(M, N) [\Psi]}{\text{Eq}_A(M, N) [\Psi]} \quad (4.56)
\]
<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4.57)</td>
<td>( \text{void} \in \mathcal{U}_i^\kappa [\Psi] )</td>
</tr>
<tr>
<td>(4.58)</td>
<td>( M \in \text{void} [\Psi] )</td>
</tr>
<tr>
<td>(4.60)</td>
<td>( \text{bool} \in \mathcal{U}_i^\kappa [\Psi] )</td>
</tr>
<tr>
<td>(4.61)</td>
<td>( \text{true} \in \text{bool} [\Psi] )</td>
</tr>
<tr>
<td>(4.61)</td>
<td>( \text{false} \in \text{bool} [\Psi] )</td>
</tr>
<tr>
<td>(4.62)</td>
<td>( b : \text{bool} \Rightarrow A \in \mathcal{U}_i^\kappa [\Psi] )</td>
</tr>
<tr>
<td>(4.62)</td>
<td>( M \in \text{bool} [\Psi] )</td>
</tr>
<tr>
<td>(4.62)</td>
<td>( T \in A[\text{true}/b] [\Psi] )</td>
</tr>
<tr>
<td>(4.62)</td>
<td>( F \in A[\text{false}/b] [\Psi] )</td>
</tr>
<tr>
<td>(4.62)</td>
<td>( \text{if}(M; T, F) \in A[M/b] [\Psi] )</td>
</tr>
<tr>
<td>(4.65)</td>
<td>( \text{nat} \in \mathcal{U}_i^\kappa [\Psi] )</td>
</tr>
<tr>
<td>(4.66)</td>
<td>( z \in \text{nat} [\Psi] )</td>
</tr>
<tr>
<td>(4.66)</td>
<td>( M \in \text{nat} [\Psi] )</td>
</tr>
<tr>
<td>(4.66)</td>
<td>( s(M) \in \text{nat} [\Psi] )</td>
</tr>
<tr>
<td>(4.67)</td>
<td>( n : \text{nat} \Rightarrow A \in \mathcal{U}_i^\kappa [\Psi] )</td>
</tr>
<tr>
<td>(4.67)</td>
<td>( M \in \text{nat} [\Psi] )</td>
</tr>
<tr>
<td>(4.67)</td>
<td>( Z \in A[z/n] [\Psi] )</td>
</tr>
<tr>
<td>(4.67)</td>
<td>( n : \text{nat}, a : A \Rightarrow S \in A[s(n)/n] [\Psi] )</td>
</tr>
<tr>
<td>(4.67)</td>
<td>( \text{natrec}(M; Z, n, a, S) \in A[M/n] [\Psi] )</td>
</tr>
<tr>
<td>(4.70)</td>
<td>( S_1 \in \mathcal{U}_i^\kappa [\Psi] )</td>
</tr>
<tr>
<td>(4.71)</td>
<td>( \text{base} \in S_1 [\Psi] )</td>
</tr>
<tr>
<td>(4.71)</td>
<td>( \text{loop} \in S_1 [\Psi] )</td>
</tr>
<tr>
<td>(4.76)</td>
<td>( c : S_1 \Rightarrow A \in \mathcal{U}_i^{\text{Kan}} [\Psi] )</td>
</tr>
<tr>
<td>(4.76)</td>
<td>( M \in S_1 [\Psi] )</td>
</tr>
<tr>
<td>(4.76)</td>
<td>( P \in A[\text{base}/c] [\Psi] )</td>
</tr>
<tr>
<td>(4.76)</td>
<td>( L \in A[\text{loop}_x/c] [\Psi, x] )</td>
</tr>
<tr>
<td>(4.76)</td>
<td>( \forall \epsilon ) ( L(\epsilon/x) \doteq P \in A[\text{base}/c] [\Psi] )</td>
</tr>
<tr>
<td>(4.76)</td>
<td>( S_1{-}\text{elim}_{c, A}(M; P, x, L) \in A[M/c] [\Psi] )</td>
</tr>
<tr>
<td>(4.73)</td>
<td>( L \in B [\Psi, x] )</td>
</tr>
<tr>
<td>(4.73)</td>
<td>( \forall \epsilon ) ( L(\epsilon/x) \doteq P \in B(\epsilon/x) [\Psi] )</td>
</tr>
<tr>
<td>(4.73)</td>
<td>( S_1{-}\text{elim}_{c, A}(\text{loop}_r; P, x, L) \doteq L(r/x) \in B(r/x) [\Psi] )</td>
</tr>
</tbody>
</table>
\[
\forall \xi_i \text{ valid} \\
M \in \mathcal{S}^1 [\Psi] \\
(Vi) \ \ N_i \vdash N_j \in \mathcal{S}^1 [\Psi, y \mid \xi_i, \xi_j] \\
(Vi) \ \ N_i \langle r/y \rangle \vdash M \in \mathcal{S}^1 [\Psi \mid \xi_i] \\
\]
(4.75)

\[
\text{isContr}(C) := (\langle c,C \rangle \times ((c':C) \to \text{Path} \_c(c',c))) \\
\text{Equiv}(A,B) := (\langle f:A \to B \rangle \times (b:B) \to \text{isContr}((a:A) \times \text{Path} \_b(f \ a, b))) \\
\]
\[
A \in \mathcal{U}_i^\xi [\Psi \mid r = 0] \\
B \in \mathcal{U}_i^\xi [\Psi] \\
E \in \text{Equiv}(A,B) [\Psi \mid r = 0] \\
\Rightarrow V_r(A,B,E) \in \mathcal{U}_i^\xi [\Psi] \\
\]
(4.78)

\[
M \in V_r(A,B,E) [\Psi] \\
V_{\text{proj}}(M, \text{fst}(E)) \in B [\Psi] \\
\]
(4.80)

\[
M \in A [\Psi \mid r = 0] \\
N \in B [\Psi] \\
F \in A \to B [\Psi \mid r = 0] \\
F M \vdash N \in B [\Psi \mid r = 0] \\
\Rightarrow V_{\text{proj}}(V_r(M,N),F) \vdash N \in B [\Psi] \\
\]
(4.81)

\[
N \in V_r(A,B,E) [\Psi] \\
\text{Vin}_r(N, V_{\text{proj}}(N, \text{fst}(E))) \vdash N \in V_r(A,B,E) [\Psi] \\
\]
(4.82)

\[
\text{hcom}_x^{r \to r'}(M; \xi_i \leftarrow y.N_i) \\
\text{Vin}_x(O^\xi \langle r'/y \rangle, \text{hcom}_x^{r \to r'}(V_{\text{proj}}(M, \text{fst}(E)); \xi_i \leftarrow y. V_{\text{proj}}(N_i, \text{fst}(E)), T)) \\
\in V_x(A,B,E) [\Psi] \\
\]
(4.83)
\[
A \in \mathcal{U}_i^{\text{Kan}}[\Psi, x, y \mid x = 0] \quad B \in \mathcal{U}_i^{\text{Kan}}[\Psi, x, y] \quad E \in \text{Equiv}(A, B)[\Psi, x, y \mid x = 0] \quad M \in \mathcal{V}_x(A\langle r/y \rangle, B\langle r/y \rangle, E\langle r/y \rangle)[\Psi, x] \\
\begin{array}{c}
\text{coe}_{y, \mathcal{V}_x(A, B, E)}^{r \sim r'}(M) \equiv \text{Vin}_x(\text{coe}_{y, A}^{r \sim r'}(M), \text{com}_{y, B}^{r \sim r'}(\text{proj}_x(M, \text{fst}(E\langle r/y \rangle)); \uparrow \downarrow)) \\
\quad \in \mathcal{V}_x(A\langle r'/y \rangle, B\langle r'/y \rangle, E\langle r'/y \rangle)[\Psi, x]
\end{array}
\tag{4.84}
\]

\[
\frac{i < j}{\mathcal{U}_i^k \in \mathcal{U}_j^{k'}}[\Psi] \quad A \in \mathcal{U}_i^k[\Psi] \quad A \in \mathcal{U}_i^{\text{pre}}[\Psi] \quad A \in \mathcal{U}_i^{\text{Kan}}[\Psi]
\tag{4.103}
\]

\[
\begin{array}{c}
\xi_i \text{ valid} \\
A \in \mathcal{U}_i^{\text{Kan}}[\Psi] \quad M \in A[\Psi]
\end{array}
\quad \begin{array}{c}
(\forall i, j) \quad B_i = B_j \in \mathcal{U}_i^{\text{Kan}}[\Psi, y \mid \xi_i, \xi_j] \\
(\forall i) \quad B_i\langle r/y \rangle \equiv A \in \mathcal{U}_i^{\text{Kan}}[\Psi \mid \xi_i]
\end{array} \quad \begin{array}{c}
(\forall i, j) \quad N_i = N_j \in B_i\langle r'/y \rangle[\Psi \mid \xi_i, \xi_j] \\
(\forall i) \quad \text{coe}_{y, B_i}^{r \sim r'}(N_i) = M \in A[\Psi \mid \xi_i]
\end{array}
\tag{4.91}
\]

\[
\begin{array}{c}
\text{box}^{r \sim r'}(M; \xi_i \leftrightarrow N_i) \equiv \text{hcom}_{\mathcal{U}_i^{\text{Kan}}}(A; \xi_i \leftrightarrow y.B_i)[\Psi] \\
\quad \equiv N_i \text{ when } \xi_i
\end{array}
\tag{4.92}
\]

\[
\begin{array}{c}
\xi_i \text{ valid} \\
A \in \mathcal{U}_i^{\text{Kan}}[\Psi] \quad M \in A[\Psi]
\end{array}
\quad \begin{array}{c}
(\forall i, j) \quad B_i = B_j \in \mathcal{U}_i^{\text{Kan}}[\Psi, y \mid \xi_i, \xi_j] \\
(\forall i) \quad B_i\langle r/y \rangle = A \in \mathcal{U}_i^{\text{Kan}}[\Psi \mid \xi_i]
\end{array} \quad \begin{array}{c}
(\forall i, j) \quad N_i = N_j \in B_i\langle r'/y \rangle[\Psi \mid \xi_i, \xi_j] \\
(\forall i) \quad \text{coe}_{y, B_i}^{r \sim r'}(N_i) = M \in A[\Psi \mid \xi_i]
\end{array}
\tag{4.93}
\]

\[
\begin{array}{c}
\text{cap}^{r \sim r'}(\text{box}^{r \sim r'}(M; \xi_i \leftrightarrow N_i); \xi_i \leftrightarrow y.B_i) \equiv M \in A[\Psi] \\
M \in \text{hcom}_{\mathcal{U}_i^{\text{Kan}}}(A; \xi_i \leftrightarrow y.B_i)[\Psi]
\end{array}
\tag{4.94}
\]
\( r_i = r'_i \) valid

\[
M \in \text{hcom}^{s \mapsto s'}_{\mathcal{K}_k}(A; s_j = s'_j \hookrightarrow z.B_j) [\Psi]
\]

\( (\forall i, j) \quad N_i \models N_j \in \text{hcom}^{s \mapsto s'}_{\mathcal{K}_k}(A; s_j = s'_j \hookrightarrow z.B_j) [\Psi, y | r_i = r'_i, r_j = r'_j] \)

\( (\forall i) \quad N_i(r/y) \models M \in \text{hcom}^{s \mapsto s'}_{\mathcal{K}_k}(A; s_j = s'_j \hookrightarrow z.B_j) [\Psi | r_i = r'_i] \)

(4.95)

\[
\text{hcom}^{r \mapsto r'}_{\mathcal{K}_k}(M; r_i = r'_i \hookrightarrow y.N_i) \models \text{box}^{s \mapsto s'}(Q^1; s_j = s'_j \hookrightarrow P^1_j(r'/y))
\]

\[
\in \text{hcom}^{s \mapsto s'}_{\mathcal{K}_k}(A; s_j = s'_j \hookrightarrow z.B_j) [\Psi]
\]

\( \xi_i \) valid

\[
A \in \mathcal{U}_k^{\text{Kan}}[\Psi, x]
\]

\( (\forall i, j) \quad B_i \models B_j \in \mathcal{U}_k^{\text{Kan}}[\Psi, x, z | \xi_i, \xi_j] \)

\( (\forall i) \quad B_i(r/z) \models A \in \mathcal{U}_k^{\text{Kan}}[\Psi, x | \xi_i] \)

\[
M \in \text{hcom}^{s(r/x) \mapsto s'(r/x)}_{\mathcal{K}_k}(A(r/x); \xi_i(r/x) \hookrightarrow z.B_i(r/x)) [\Psi]
\]

(4.96)

\[
\text{co}^{r \mapsto r'}_{x.h\text{com}^{s \mapsto s'}_{\mathcal{K}_k}(A; \xi_i \hookrightarrow z.B_i)}(M) \models \text{box}^{s/r'(x) \mapsto s'(r'/x)}(H^1; \xi_i(r'/x) \hookrightarrow Q^1_i(s'(r'/x)/z))
\]

\[
\in \text{hcom}^{s(r'/x) \mapsto s'(r'/x)}_{\mathcal{K}_k}(A(r'/x); \xi_i(r'/x) \hookrightarrow z.B_i(r'/x)) [\Psi]
\]
Intensional-style rules

This appendix proposes rules for intensional Cartesian cubical type theory. As discussed in Section 2.4, intensional type theories, such as those implemented by Agda [Agda] and Coq [Coq], are deductive systems whose equality judgments (and therefore, typing judgments) admit decision procedures.1 We conjecture but have not proven decidability or even canonicity for the system of rules below. Readers can experiment with similar intensional Cartesian cubical type theories in the redtt proof assistant [Red18], and can consult Angiuli et al. [Ang+19] for the denotational semantics of a related deductive system.

We intend these rules as a general reference for Cartesian cubical type theory, so we have omitted pretypes (and hence, strict equality types) and our validity condition on composition shapes (and hence, generalized composition). Conversely, we have added level annotations to our typehood judgment [Coq19] and type annotations to elimination forms (and an eliminator abort(−) for void), ensuring that our rules are generalized algebraic and thus admit a category of models [Car86].

In order to admit decidable equality judgments, we have eliminated several instances of equality reflection. First, \( \text{hcom}_{\text{nat}}^{\tau \rightarrow \rho} (M; \xi \leftrightarrow y.N_i) \) must compute by recursion on \( M, N_i \) rather than generically to \( M \); otherwise, whenever \( P : \text{Path}_{\text{nat}}(P_0, P_1) \), the composition \( \text{hcom}_{\text{nat}}^{\rho \rightarrow 
abla^1}(P_0; 0 = 0 \leftrightarrow y.P@y) \) equals both \( P_0 \) (by the rule for \( \text{hcom}_{\text{nat}} \)) and \( P_1 \) (by \( 0 = 0 \)), implying \( P_0 = P_1 \) judgmentally. Secondly, we omit the equality reflection rule itself. In the future, one might consider alternative formulations of strict equality compatible with decidable judgmental equality; possibilities include observational type theory [AMS07], XTT [SAG19], or even simply adjoining an axiom for uniqueness of identity proofs (at the expense of canonicity) [ACK16].

Judgments

The judgments of intensional Cartesian cubical type theory range over the syntactic sorts presented in Figure B.1. Following XTT [SAG19], we gather interval variables and constraints in a single context \( \Psi \), now representing an object of the augmented Cartesian cube category (which adjoins to \( \Box \) an initial object interpreting inconsistent contexts).

1. \( \Psi \text{ cube}_+ \) when \( \Psi \) is an augmented Cartesian cube.

2. \( \Psi \mid r : \nabla, \) presupposing \( \Psi \text{ cube}_+ \), when \( r \) is an interval term in \( \Psi \).

1Historically, intensional equality judgments were generated by \( \beta \) rules and decided by untyped means; modern decision procedures are type-sensitive and admit various \( \eta \) rules [ACD07].
Augmented cubes \[ \Psi ::= \cdot | \Psi, x | \Psi, \xi \]
Contexts \[ \Gamma ::= \cdot | \Gamma, a : A \]
Interval terms \[ r, s ::= x | \varepsilon \]
Interval constants \[ \varepsilon ::= 0 | 1 \]
Interval equations \[ \xi ::= r = r' \]
Universe levels \[ i, j ::= 0 \in \mathbb{N} \]
Types \[ A, B ::= (a : A) \to B | (a : A) \times B | \text{Path}_{x,A}(M, N) | \text{void} | \text{bool} | \text{nat} \]
Terms \[ M, N ::= \lambda a.M | \text{app}_{a:A,B}(M, N) | \langle M, N \rangle | \text{fst}_{a:A,B}(M) | \text{snd}_{a:A,B}(M) \]
\[ \langle x \rangle M | \text{iapp}_{x:A}(M, r) | \text{abort}_A(M) | \text{true} | \text{false} \]
\[ \text{if}_{b:A}(M; M_0, M_1) | z | s(M) | \text{natrec}_{n:A}(M; N_0, n.a.N_1) \]
\[ \text{base} | \text{loop}_r | \text{S}_{\text{elim},A}(M; N_0, n.x.N_1) | \text{Vin}_r(M, N) \]
\[ \text{Vproj}_{A,B}^{\text{r',r}}(M, N) | \text{coe}_{X}^{\text{r',r}}(M) | \text{hcom}_A^{\text{r',r}}(M; \xi_i \mapsto y.N_i) \]
\[ \text{com}_{Y}^{\text{r',r}}(M; \xi_i \mapsto y.N_i) | \text{box}_{A,Y}^{\text{r',r}}(M; \xi_i \mapsto N_i) \]
\[ \text{cap}_{A}^{\text{r',r'}}(M; \xi_i \mapsto y.B_i) \]

Figure B.1: Grammar of intensional Cartesian cubical type theory.

3. \[ \Psi | r = r' : \mathbb{I}, \] presupposing \[ \Psi | r, r' : \mathbb{I}, \] when \( r, r' \) are equal interval terms in \( \Psi \).

4. \[ \Psi | \Gamma \vdash \text{ctx}, \] presupposing \( \Psi | \Gamma \vdash \text{cube}_c, \) when \( \Gamma \) is a context in \( \Psi \).

5. \[ \Psi | \Gamma \vdash A \text{ type}_i, \] presupposing \( \Psi | \Gamma \vdash \text{ctx}, \) when \( A \) is a type at level \( i \) in \( \Gamma \).

6. \[ \Psi | \Gamma \vdash A = B \text{ type}_i, \] presupposing \( \Psi | \Gamma \vdash A, B \text{ type}_i, \) when \( A, B \) are equal types.

7. \[ \Psi | \Gamma \vdash M : A, \] presupposing \( \Psi | \Gamma \vdash A \text{ type}_i, \) when \( M \) is an element of \( A \) in \( \Gamma \).

8. \[ \Psi | \Gamma \vdash M = N : A, \] presupposing \( \Psi | \Gamma \vdash M, N : A, \) when \( M, N \) are equal elements.

In the following rules, we omit premises to equational rules, most type and term annotations (writing \( M N \) for \( \text{app}_{a:A,B}(M, N) \), \( M @ r \) for \( \text{iapp}_{x:A}(M, r) \), \text{et cetera} \), and all congruence rules. Moreover, we adopt the following notations:

1. \[ \Psi | \Gamma \vdash \mathcal{J} \text{ abbreviates any typehood, membership, or equality judgment}. \]
2. \[ \Psi | \Gamma \vdash A \text{ type}_i [\xi_i \mapsto B_i] \text{ abbreviates } \Psi | \Gamma \vdash A \text{ type}_i \text{ and } \Psi, \xi_i | \Gamma \vdash A = B \text{ type}_i. \]
3. \[ \Psi | \Gamma \vdash M : A [\xi_i \mapsto N_i] \text{ abbreviates } \Psi | \Gamma \vdash M : A \text{ and } \Psi, \xi_i | \Gamma \vdash M = N_i : A. \]
Structural rules

\[
\begin{array}{c}
\Theta \quad \Theta, x \Theta \quad \Theta, r, r' : \Xi \\
\cdot \Theta \quad \Theta, x \Theta \quad \Theta, r = r' \Theta
\end{array}
\]

\[
\begin{array}{c}
\Theta | 0 : \Xi \\
\Theta | 1 : \Xi \\
\Theta | x : \Xi
\end{array}
\]

Kan operations

\[
\begin{array}{c}
\Theta | r, r' : \Xi \\
\Theta, x | \Gamma \vdash A type_i \\
\Gamma | \vdash \coe^{\text{Type}_i}_{\text{Type}_i}(M) : A\langle r/x \rangle [r = r' \leftrightarrow M]
\end{array}
\]

\[
\begin{array}{c}
\Theta | r, r' : \Xi \\
\Theta | \Gamma \vdash M : A \\
\Psi, \xi_i, y | \Gamma \vdash N_i : A [y = r \leftrightarrow M, \xi_i \leftrightarrow N_i]
\end{array}
\]

\[
\begin{array}{c}
\Psi | \Gamma \vdash \text{hcom}_{A}^{r \mapsto r'}(M; \xi_i \leftrightarrow y.N_i) : A [r = r' \leftrightarrow M, \xi_i \leftrightarrow N_i(r'/y)]
\end{array}
\]

\[
\begin{array}{c}
\Psi | \Gamma \vdash M : A\langle r/y \rangle \\
\Psi, \xi_i, y | \Gamma \vdash N_i : A [y = r \leftrightarrow M, \xi_i \leftrightarrow N_i]
\end{array}
\]

\[
\begin{array}{c}
\Psi | \Gamma \vdash \text{com}_{g,A}^{r \mapsto r'}(M; \xi_i \leftrightarrow y.N_i) : A\langle r'/y \rangle [r = r' \leftrightarrow M, \xi_i \leftrightarrow N_i(r'/y)]
\end{array}
\]

\[
\begin{array}{c}
\Psi | \Gamma \vdash \text{com}_{g,A}^{r \mapsto r'}(M; \xi_i \leftrightarrow y.N_i) = \text{hcom}_{A(r'/y)}^{r \mapsto r'}(\coe^{r \mapsto r'}_{y,A}(M); \xi_i \leftrightarrow y.\coe^{r \mapsto r'}_{y,A}(N_i)) : A\langle r'/y \rangle
\end{array}
\]
### Other rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Psi \mid \Gamma \vdash A \text{ type}_i )</td>
<td>( \Psi \mid \Gamma, a : A \vdash B \text{ type}_i )</td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash (a : A) \rightarrow B \text{ type}_i )</td>
<td>( \Psi \mid \Gamma, a : A \vdash M : B )</td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash M : (a : A) \rightarrow B )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash N : A )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{app}_{a,A,B}(M, N) : B[N/a] )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash (\lambda a.M) N = M[N/a] : B[N/a] )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash M = \lambda a.M a : (a : A) \rightarrow B )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{coe}<em>{x,(a : A) \rightarrow B}(M) = \lambda a.\text{coe}</em>{x,x,A}(a) ) ( (M \text{ coe}_{x,A}(a)) : (a : A) \rightarrow B(r' / x) )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{hcom}<em>{(a : A) \rightarrow B}(M ; \xi_i \leftarrow y.N_i) = \lambda a.\text{hcom}</em>{B}(M a ; \xi_i \leftarrow y.N_i a) : (a : A) \rightarrow B )</td>
<td></td>
</tr>
</tbody>
</table>

### Other rules (continued)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Psi \mid \Gamma \vdash A \text{ type}_i )</td>
<td>( \Psi \mid \Gamma, a : A \vdash B \text{ type}_i )</td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash M : A )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash (a : A) \times B \text{ type}_i )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{app}_{a,A,B}(M) : B[fst(M)/a] )</td>
<td>( \Psi \mid \Gamma \vdash \text{fst}(\langle M, N \rangle) = M : A )</td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{snd}_{a,A,B}(M) : B[snd(M)/a] )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{snd}(\langle M, N \rangle) = N : B[M/a] )</td>
<td>( \Psi \mid \Gamma \vdash (\text{fst}(M), \text{snd}(M)) : (a : A) \times B )</td>
</tr>
<tr>
<td>( \widetilde{M_0}[x] := \text{coe}_{x,A}(\text{fst}(M)) )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{coe}<em>{x,(a : A) \times B}(M) = \langle \widetilde{M_0}[r'], \text{coe}</em>{x,B[M_0[x]/a]}(\text{snd}(M)) \rangle : (a : A) \rightarrow B(r' / x) )</td>
<td></td>
</tr>
<tr>
<td>( \widetilde{M_0}[z] := \text{hcom}_{x,B[M_0[z]/a]}(\text{snd}(M); \xi_i \leftarrow y.\text{snd}(N_i)) )</td>
<td></td>
</tr>
<tr>
<td>( \widetilde{M_1} := \text{com}_{z,B[M_0[z]/a]}(\text{snd}(M); \xi_i \leftarrow y.\text{snd}(N_i)) )</td>
<td></td>
</tr>
<tr>
<td>( \Psi \mid \Gamma \vdash \text{hcom}_{(a : A) \times B}(M ; \xi_i \leftarrow y.N_i) = \langle \widetilde{M_0}[r'], \widetilde{M_1} \rangle : (a : A) \rightarrow B )</td>
<td></td>
</tr>
</tbody>
</table>
\[ \Psi, x \vdash A \quad \Psi, x = \varepsilon \vdash N_r : A \quad \Psi, x \vdash M : A [x = \varepsilon \mapsto N_r] \]

\[ \Psi \vdash \text{Path}_x.A(N_0, N_1) \]

\[ \Psi \vdash \text{Path}_x.A(N_0, N_1) \]

\[ \Psi \vdash (x)M : \text{Path}_x.A(N_0, N_1) \]

\[ \Psi \vdash (x)M@r = M⟨r/x⟩ : A⟨r/x⟩ \]

\[ \tilde{M} : = \langle x \rangle \text{coe}_{y.\text{Path}_x.A(N_0, N_1)}(r_\rightarrow') (M@x; x = \varepsilon \mapsto y.N_r) \quad \tilde{M} : = \langle x \rangle \text{hcom}_{A}(r_\rightarrow')(M@x; x = \varepsilon \mapsto \_N_r, \_ξ_\mapsto y.Q_i@x) \]

\[ \Psi \vdash \text{void} \quad \Psi \vdash A \quad \Psi \vdash M : \text{void} \]

\[ \Psi \vdash \text{true} : \text{bool} \quad \Psi \vdash \text{false} : \text{bool} \]

\[ \Psi \vdash \text{if}_b.A(M; N_0, N_1) : A[M/b] \quad \Psi \vdash \text{if}(\text{true}; N_0, N_1) = N_0 : A[\text{true}/b] \]

\[ \Psi \vdash \text{if}(\text{false}; N_0, N_1) = N_1 : A[\text{false}/b] \quad \Psi \vdash \text{coe}_{x.\text{bool}}(r_\rightarrow')(M) = M : \text{bool} \]

\[ \Psi \vdash \text{hcom}_{\text{bool}}(\text{true}; \_ξ_\mapsto y.\text{true}) = \text{true} : \text{bool} \]

\[ \Psi \vdash \text{hcom}_{\text{bool}}(\text{false}; \_ξ_\mapsto y.\text{false}) = \text{false} : \text{bool} \]
\[
\begin{array}{c}
\Pi | \Gamma \vdash \text{nat type} \\
\Pi | \Gamma \vdash z : \text{nat} \\
\Pi | \Gamma \vdash s(M) : \text{nat}
\end{array}
\]

\[
\begin{array}{c}
\Pi | \Gamma, n : \text{nat} \vdash A \text{ type} \\
\Pi | \Gamma \vdash M : \text{nat} \\
\Pi | \Gamma \vdash n_0 : A[z/n] \\
\Pi | \Gamma, n : \text{nat}, a : A \vdash N_1 : A[s(n)/n] \\
\Pi | \Gamma \vdash \text{natrec}_{n, A}(M; N_0, n.a.N_1) : A[M/n] \\
\Pi | \Gamma \vdash \text{natrec}(z; N_0, n.a.N_1) = N_0 : A[z/n]
\end{array}
\]

\[
\begin{array}{c}
\Pi | \Gamma \vdash \text{coer}^\sim_{x.\text{nat}}(M) = M : \text{nat} \\
\Pi | \Gamma \vdash \text{hocom}^\sim_{\text{nat}}(z; \xi \leftrightarrow y.z) = z : \text{nat}
\end{array}
\]

\[
\begin{array}{c}
\Pi | \Gamma \vdash S^1 \text{ type} \\
\Pi | \Gamma \vdash \text{base} : S^1 \\
\Pi | \Gamma \vdash \text{loop}_{r} : S^1 \ [r = e \leftrightarrow \text{base}]
\end{array}
\]

\[
\begin{array}{c}
\Pi | \Gamma, c : S^1 \vdash A \text{ type} \\
\Pi | \Gamma \vdash M : S^1 \\
\Pi | \Gamma \vdash N_0 : A[\text{base}/c] \\
\Pi | \Gamma, x \vdash N_1 : A[\text{loop}_{x}/c] \ [x = e \leftrightarrow N_0] \\
\Pi | \Gamma \vdash S^1-\text{elim}_{c.A}(M; N_0, x.N_1) : A[M/c]
\end{array}
\]

\[
\begin{array}{c}
\Pi | \Gamma \vdash \text{coer}^\sim_{x.S^1}(M) = M : S^1 \\
\Pi | \Gamma \vdash S^1-\text{elim}_{c.A}(\text{base}; N_0, x.N_1) = N_0 : A[\text{base}/c]
\end{array}
\]

\[
\begin{array}{c}
\Pi | \Gamma \vdash S^1-\text{elim}_{c.A}(\text{loop}_{r}; N_0, x.N_1) = N_1(r/x) : A[\text{loop}_{r}/c]
\end{array}
\]

\[
\widetilde{M}[z] := \text{hocom}_{S^1}^\sim(\xi \leftrightarrow y.Q_i)
\]

\[
\widetilde{H} := \text{com}_{z.A[\widetilde{M}[z]/c]}^\sim(S^1-\text{elim}_{c.A}(M; N_0, x.N_1); \xi \leftrightarrow y.S^1-\text{elim}_{c.A}(Q_i; N_0, x.N_1))
\]

\[
\Pi | \Gamma \vdash S^1-\text{elim}_{c.A}(\widetilde{M}[r']; N_0, x.N_1) = \widetilde{H} : A[\widetilde{M}[r']/c]
\]
\[
\text{isContr}(C) := (c: C) \times ((c': C) \to \text{Path}_{-C}(c', c))
\]
\[
\text{Equiv}(A, B) := (f:A \to B) \times ((b:B) \to \text{isContr}((a:A) \times \text{Path}_{-B}(f a, b)))
\]

\[
\Psi, r = 0 \mid \Gamma \vdash A \text{ type}_i \quad \Psi \mid \Gamma \vdash B \text{ type}_i \quad \Psi, r = 0 \mid \Gamma \vdash E : \text{Equiv}(A, B)
\]
\[
\Psi \mid \Gamma \vdash V_r(A, B, E) \text{ type}_i \ [r = 0 \leftrightarrow A, r = 1 \leftrightarrow B]
\]

\[
\Psi, r = 0 \mid \Gamma \vdash M : A
\]
\[
\Psi, r = 0 \mid \Gamma \vdash E : \text{Equiv}(A, B) \quad \Psi \mid \Gamma \vdash N : B \ [r = 0 \leftrightarrow \text{fst}(E) M]
\]
\[
\Psi \mid \Gamma \vdash V_{r'}(M, N) : V_r(A, B, E) \ [r = 0 \leftrightarrow M, r = 1 \leftrightarrow N]
\]
\[
\Psi \mid \Gamma \vdash \text{Vproj}_{r', B}(M, E) : B \ [r = 0 \leftrightarrow \text{fst}(E) M, r = 1 \leftrightarrow M]
\]

\[
\Psi \mid \Gamma \vdash \text{Vproj}_r(V_{r'}(M, N), E) = N : B \quad \Psi \mid \Gamma \vdash \text{Vproj}_r(N, \text{Vproj}_r(N, E)) = N : V_r(A, B, E)
\]

\[
\tilde{N} := \text{coe}_{x, B}^{r, r'}(\text{Vproj}_r(M, E(r/x)))
\]
\[
\tilde{F} := (a:A(r'/x)) \times \text{Path}_{-B(r'/x)}(\text{fst}(E(r'/x)) a, \tilde{N})
\]
\[
\tilde{C} := \text{snd}(E(r'/x)) \tilde{N}
\]
\[
\tilde{O} := \text{hcom}_{\tilde{F}}^{1, 0}(\text{fst}(\tilde{C}); r = 0 \leftrightarrow z. (\text{snd}(\tilde{C}) \langle M, \langle \_ \rangle (\text{fst}(E(0/x)) M)) @ z)
\]
\[
\tilde{P} := \text{hcom}_{B(r'/x)}^{1, 0}(\tilde{N}; r' = 0 \leftrightarrow z. \text{snd}(\tilde{O}) @ z, r = r' \leftrightarrow \_ \text{Vproj}_r(M, E(r/x)))
\]
\[
\Psi \mid \Gamma \vdash \text{coe}_{x, V_{r'}(A, B, E)}^{r, r'}(M) = V_{r'}(\text{fst}(\tilde{O}), \tilde{P}) : V_{r'}(A(r'/x), B(r'/x), E(r'/x))
\]
\[
\tilde{T} := x = 0 \leftrightarrow y. \text{fst}(E) \text{coe}_{y, A}^{r, r'}(M), x = 1 \leftrightarrow y. \text{coe}_{y, B}^{r, r'}(M)
\]
\[
\tilde{C} := V_{x, y}(\text{coe}_{y, A}^{r, r'}(M), \text{com}_{y, B}^{r, r'}(\text{Vproj}_x(M, E(r/y)); \tilde{T}))
\]
\[
\Psi \mid \Gamma \vdash \text{coe}_{x, V_{r'}(A, B, E)}^{r, r'}(M) = \tilde{C} : V_{x, y}(A(r'/y), B(r'/y), E(r'/y))
\]
\[
\tilde{O}[y] := \text{hcom}_{A}^{r, r'}(M; \xi_i \equiv y. N_i)
\]
\[
\tilde{T} := x = 0 \leftrightarrow y. \text{fst}(E) \tilde{O}[y], x = 1 \leftrightarrow y. \text{hcom}_{B}^{r, r'}(M; \xi_i \equiv y. N_i)
\]
\[
\tilde{H} := V_{x, y}(\tilde{O}[r'], \text{hcom}_{B}^{r, r'}(\text{Vproj}_x(M, E); \xi_i \equiv y. \text{Vproj}_x(N_i, E, \tilde{T}))
\]
\[
\Psi \mid \Gamma \vdash \text{hcom}_{V_{x, V_{r'}(A, B, E)}}^{r, r'}(M; \xi_i \equiv y. N_i) = \tilde{H} : V_{x, y}(A, B, E)
\]

\[
i < j
\]
\[
\Psi \mid \Gamma \vdash \text{Type}_i \text{ type}_j\quad \Psi \mid \Gamma \vdash \text{coe}_{x, \text{Type}_i}^{r, r'}(A) = A : \text{Type}_i
\]
\( \Psi \mid r, r' : \mathbb{I} \quad \Psi \mid \Gamma \vdash A : \text{Type}_k \quad \Psi, \xi, y \mid \Gamma \vdash B_i : \text{Type}_k \quad [y = r \leftrightarrow A, \xi_j \leftrightarrow B_j] \)

\[ \Psi, \xi_i \mid \Gamma \vdash N_i : B_i[r'/y \mid \xi_j \leftrightarrow N_j] \quad \Psi \mid \Gamma \vdash : \text{hcom}_{\text{Type}_k}^r(A; \xi_i \leftrightarrow y.B_i) \quad [r = r' \leftrightarrow M, \xi_i \leftrightarrow N_i] \]

\[ \Psi \mid \Gamma \vdash \text{box}^r_{\text{Type}_k}(M; \xi_i \leftrightarrow N_i) : \text{hcom}_{\text{Type}_k}^r(A; \xi_i \leftrightarrow y.B_i) \]

\[ \Psi \mid \Gamma \vdash \text{cap}^r_{\text{Type}_k}(M; \xi_i \leftrightarrow N_i) : A[r = r' \leftrightarrow M, \xi_i \leftrightarrow \text{coer}_{y.B_i}(M)] \]

\[ \Psi \mid \Gamma \vdash \text{box}^r_{\text{Type}_k}(\text{cap}^r_{\text{Type}_k}(M; \xi_i \leftrightarrow y.B_i); \xi_i \leftrightarrow M) = M : \text{hcom}_{\text{Type}_k}^r(A; \xi_i \leftrightarrow y.B_i) \]

\[ \bar{N}_i[w, z] := \text{coer}_{x.B_i(w/x)}^{r}(\text{coer}_{x.B_i(s'/x)}^{r}(\text{coer}_{x.B_i(r/x)}^{r}(M))) \]

\[ \bar{T} := \xi_i(r/x) \leftrightarrow x.N_i[x, s]|_{(x \neq \xi_i)}, s = s' \leftrightarrow x.\text{coer}_{x.A}^{r}(M)|_{(x \neq s,s')} \]

\[ \bar{O}[z] := \text{hcom}_{x.A}^{r}(\text{cap}^{r}(r/x)_{x.A}(\text{coer}_{x.B_i(r'/x)}^{s(r/x)}(M)), \xi_i \leftrightarrow N_i(r', z)|_{(x \neq \xi_i)}, r = r' \leftrightarrow z.\text{coer}_{z.B_i(r'/x)}^{r}(M)) \]

\[ \bar{P} := \text{com}_{x.A}^{r}(\bar{O}[s(r/x)]; \xi_i \leftrightarrow x.N_i[x, s]|_{(x \neq \xi_i)}, s = s' \leftrightarrow x.\text{coer}_{x.A}^{r}(M)|_{(x \neq s,s')} \]

\[ \bar{Q}_k[z] := \text{com}_{z.B_i(r'/x)}^{s(r/x)}(\bar{P}; \xi_i \leftrightarrow z.N_i[r', z]|_{(x \neq \xi_i)}, r = r' \leftrightarrow z.\text{coer}_{z.B_i(r'/x)}^{r}(M)) \]

\[ \bar{H} := \text{hcom}_{A(r'/x)}^{(r/x)}(\bar{P}; \xi_i \leftrightarrow z.\text{coer}_{z.B_i(r'/x)}^{s(r/x)}(\bar{Q}[z]), r = r' \leftrightarrow z.\bar{O}[z]) \]

\[ \bar{C} := \text{box}_{(r'/x)_{A(r'/x)}}^{r}(\bar{H}; \xi_i \leftrightarrow \bar{Q}[s'(r'/x)]) \]

\[ \Psi \mid \Gamma \vdash \text{coer}_{x.\text{hcom}_{x.A}^{r}(\text{cap}^{r}(M; \xi_i \leftrightarrow y.B_i); \xi_i \leftrightarrow M)}^{r}(A; \xi_i \leftrightarrow z.B_i)) \]

\[ \bar{P}[y] := \text{hcom}_{B_i(s'/z)}^{r}(M; r_i = r'_i \leftrightarrow y.N_i) \]

\[ \bar{P}[y] := \text{hcom}_{B_i(s'/z)}^{r}(M; r_i = r'_i \leftrightarrow y.N_i) \]

\[ \bar{Q} := \text{hcom}_{A}^{r}(\text{cap}^{r}(M; \xi_j \leftrightarrow z.B_j); \bar{T}) \quad \bar{H} := \text{box}_{(r'/x)}^{(r/x)}(\bar{Q}; \xi_j \leftrightarrow \bar{P}[r']) \]

\[ \Psi \mid \Gamma \vdash \text{hcom}_{\text{Type}_k}^{r}(\text{cap}^{r}(M; \xi_j \leftrightarrow z.B_j); \bar{T}) \quad \bar{H} := \text{hcom}_{\text{Type}_k}^{r}(A; \xi_j \leftrightarrow z.B_j) \]
Bibliography


[Lic16] Dan Licata. *Weak univalence with “beta” implies full univalence*. Email to Homotopy Type Theory mailing list. 2016. URL: https://groups.google.com/d/msg/homotopytypetheory/j2KBIVDw53s/YTDK4D0NFQAJ.


