Greatest HITs: Higher inductive types in coinductive definitions via induction under clocks

Magnus Baunsgaard Kristensen
Department of Computer Science
IT University of Copenhagen
Denmark

Rasmus Ejlers Møgelberg
Department of Computer Science
IT University of Copenhagen
Denmark

Andrea Vezzosi
Department of Computer Science
IT University of Copenhagen
Denmark

Abstract

We present Clocked Cubical Type Theory, the first type theory combining multi-clocked guarded recursion with the features of Cubical Type Theory. Guarded recursion is an abstract form of step-indexing, which can be used for construction of advanced programming language models. In its multi-clocked version, it can also be used for coinductive programming and reasoning, encoding productivity in types. Combining this with Higher Inductive Types (HITs) the encoding extends to coinductive types that are traditionally hard to represent in type theory, such as the type of finitely branching labelled transition systems.

Among our technical contributions is a new principle of *induction under clocks*, providing computational content to one of the main axioms required for encoding coinductive types. This principle is verified using a denotational semantics in a presheaf model.

ACM Reference Format:
Magnus Baunsgaard Kristensen, Rasmus Ejlers Møgelberg, and Andrea Vezzosi. 2022. Greatest HITs: Higher inductive types in coinductive definitions via induction under clocks. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 33 pages. https://doi.org/10.1145/3531130.3533359

1 Introduction

Homotopy type theory [50] is the extension of Martin-Löf type theory [37] with the univalence axiom and higher inductive types. It can be seen as a foundation for mathematics, allowing for synthetic approaches to mathematics, as exemplified by synthetic homotopy theory, as well as more direct formalisations of mathematics, closer to the traditional set-theory based developments. Higher Inductive Types (HITs) play a key role in both approaches: In synthetic homotopy theory these are used to describe spaces such as the spheres and the torus, and in the more direct formalisations, HITs express free structures and quotients, both of which have traditionally been hard to represent and use in type theory.

The kind of mathematics that is particularly important to the LICS community often involves reasoning about recursion, and in particular programming with and reasoning about coinductive types. Doing this in type theory has historically been difficult for a number of reasons. One is that the definitions of the functors of interest in coalgebra often involve constructions such as finite or countable powerset (for modelling non-determinism), or quotients. All of these are known to be representable as HITs [16, 27, 50]. Another reason is that programming and reasoning about coinductive types involves productivity checking, which in most proof assistants is implemented as non-modular syntactic checks that can lead to considerable overhead for programmers [25]. A third reason is that bisimilarity, which is the natural notion of identity for coinductive types, rarely (at least provably) coincides with the built-in identity types of type theory, and so proofs cannot be transported along bisimilarity proofs.

Multi-clocked guarded recursion [6] allows for the productivity requirement for coinductive definitions to be encoded in types. The key ingredient in this is a modal operator $\triangleright$ (pronounced 'later') indexed by clocks κ, encoding a notion of time-steps in types. Guarded recursive types such as $\mathrm{Str}^\kappa \simeq \mathbb{N} \times \triangleright^\kappa \mathrm{Str}^\kappa$ are recursive types in which the recursion variable occurs only under a $\triangleright$. In the case of Str^κ the type equivalence above states that the head of a guarded stream is available now, but the tail only after a time step on clock κ. Guarded streams can be defined using a fixed point operator fix^κ of type $(\triangleright^\kappa A \to A) \to A$. In the case of Str^κ the delay in the domain of the input precisely captures the productivity requirement for programming with streams. Guarded recursive types can themselves be encoded as fixed points on a universe. Using quantification over clocks, the coinductive type of streams can be encoded as $\mathrm{Str} \stackrel{\mathrm{def}}{=} \forall\kappa.\mathrm{Str}^\kappa$.

Guarded recursion [43] can also be used as an abstract approach to step-indexing techniques [4, 5]. In particular, the type variables in guarded recursive types can also appear in negative positions, and so can be used to provide models of untyped lambda calculus or solve the advanced type equations needed for modelling higher-order store [10] and advanced notions of separation logic [34]. A type theory

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
LICS '22, August 2–5, 2022, Haifa, Israel
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9351-5/22/08…$15.00
https://doi.org/10.1145/3531130.3533359

combining guarded recursion with homotopy type theory will therefore provide a powerful framework, not just for coinductive reasoning but also for reasoning about recursion and advanced programming language features beyond the reach of traditional domain theory.

1.1 Clocked Cubical Type Theory

This paper presents Clocked Cubical Type Theory (CCTT), the first type theory to combine all the above mentioned features: Multi-clocked guarded recursion, HITs and univalence. Rather than basing this on the original formulation of homotopy type theory we build on Cubical Type Theory (CTT) [21], a variant of HoTT based on the cubical model of type theory, and implemented in the Cubical Agda proof assistant [54]. One reason for this choice is that the operational properties of CTT as well as the concrete implementation give hope for an implementation of our type theory. In fact we have already extended the implementation of Guarded Cubical Agda [53] to support most of the new constructions of CCTT. Another is that the path types of CTT make for a simple implementation of the extensionality principle for the $\triangleright^\kappa$ modality [9].

CCTT extends CTT with the constructions of Clocked Type Theory (CloTT) [7], a type theory for multi-clocked guarded recursion. CloTT uses a Fitch-style approach to programming with the $\triangleright$ modality. This means that elements of modal type are introduced by abstracting over tick variables $\alpha : \kappa$ and eliminated by application to ticks. One benefit of this is that let-expressions are avoided, and these do not interact well with dependent types in general. For example, one can define a dependently typed generalisation of applicative action [38] operator of type

$$\triangleright^\kappa(\textstyle\prod (x : A) .B) \to \prod (y : \triangleright^\kappa A) .\triangleright (\alpha : \kappa).B[y\ [\alpha]/x] \quad (1)$$

In this term the tick variable α should be thought of as evidence that time has passed, which is used in subterms like $y\ [\alpha]$ to access the element of type A delivered by $y : \triangleright^\kappa A$ in the next time step. CloTT has a single tick constant $\diamond$ that can be used for a controlled elimination of $\triangleright$. Bahr. et al. [7] define a strongly normalising operational semantics for CloTT, in which $\diamond$ unfolds fixed point operations. Since tick variables α can be substituted by $\diamond$, their identity is crucial for normalisation. For reasoning in the type theory, however, it is often necessary that $y\ [\alpha]$ does not depend on α up to path equality. This is referred to as the *tick irrelevance axiom*.

When adding such axioms to a univalent type theory one should be take care to avoid introducing extra structure that one will eventually need to reason about. Ideally, an axiom like tick irrelevance should state that the type of ticks is propositional, i.e., that any two elements can be identified, any two paths between elements can be identified, and so on for higher dimensions. In CCTT, there appears to be no such way of formulating the axiom, because ticks do not form a type. This paper shows how to obtain tick irrelevance by enriching the language of ticks with paths $\mathsf{tirr}\ u\ v$ between any pair of ticks u, v. This has the additional benefit that it allows for rules giving computational content to the axiom.

1.2 Induction under clocks

The encoding of coinductive streams from guarded streams mentioned above generalises to an encoding of coinductive solutions to type equations of the form $X \simeq F(X)$ for all functors F commuting with clock quantification in the sense that the canonical map

$$F(\forall\kappa.X) \to \forall\kappa.F(X) \quad (2)$$

is an equivalence of types. Note that this is a semantic condition, rather than the more standard grammars for coinductive types. For this to be useful, one must have a large supply of such functors, including functors formed using inductive and higher inductive types. This property has previously been obtained via axioms [6]. Here we formulate a principle of *induction under clocks*, which gives computational content to these axioms also in the case of higher inductive types.

Consider for example the finite powerset $\mathsf{P_f}$ which can be defined as a HIT given by constructors for empty set, singletons, union, equalities stating associativity, commutativity and idempotency of union as well as an axiom forcing $\mathsf{P_f}(X)$ to be a hset [27]. In this case induction under clocks states that to inhabit a family over an element of type $\forall\kappa.\mathsf{P_f}(X)$ it suffices to inductively describe the cases for elements of the form $\lambda\kappa.\mathsf{con}_i(p)$ for each constructor con_i for $\mathsf{P_f}$, i.e., for $\lambda\kappa.\emptyset, \lambda\kappa.\{a\}, \lambda\kappa.x\ [\kappa] \cup y\ [\kappa]$ and for equality constructors such as $\lambda\kappa.\mathsf{idem}(x\ [\kappa], r)$, where idem implements idempotency of union. In these cases, x, y are assumed to be of type $\forall\kappa.\mathsf{P_f}(X)$. Using this, one can construct an inverse to (2) for $F = \mathsf{P_f}$, and also prove that both compositions are identities. As a consequence, the coinductive solution to $\mathsf{LTS} \simeq \mathsf{P_f}(A \times \mathsf{LTS})$ can be encoded as $\mathsf{LTS} \stackrel{\mathsf{def}}{=} \forall\kappa.\mathsf{LTS}^\kappa$ where $\mathsf{LTS}^\kappa \simeq \mathsf{P_f}(A \times \triangleright^\kappa \mathsf{LTS}^\kappa)$ is a guarded recursive type. LTS is the type of A-labelled finitely branching transition systems. We moreover prove that path equality for this type coincides with bisimilarity.

The results for LTS mentioned above hold for all types of labels A that are *clock-irrelevant*, meaning that the map $A \to \forall\kappa.A$ is an equivalence for κ fresh. In previous type theories for multi-clocked guarded recursion clock-irrelevance of all types has been taken as an axiom. Most types are clock-irrelevant, but ensuring this for universes requires special care both syntactically and semantically [12]. In this paper we opt for a simpler solution, taking clock-irrelevance to be a side condition to the correctness of the encoding of coinductive types. Using the principle of induction under clocks we show that HITs constructed using clock-irrelevant types are themselves clock-irrelevant, an important step towards reintroducing clock irrelevance as an axiom.

1.3 Denotational semantics

Our type theory is justified by a denotational semantics in the form of a presheaf model. This model combines previous models of CTT [21, 44] and CloTT [36]. A key challenge for the construction of this model is the definition of a composition structure on the operation modelling $\triangleright$. We give a general theorem for defining composition structures for dependent right adjoints [19] which can be used to model other type theories combining Fitch-style modal operators [17] with Cubical Type Theory.

Overview of paper. The paper is organised as follows. We start by recalling Cubical Type Theory in section 2, and section 3 extends this to CCTT. The encoding of coinductive types using guarded recursion is recalled in section 4 which also details the example of labelled transition systems. The syntax for HITs in CCTT is presented in section 5, which also states the principle of induction under clocks. The denotational semantics of Clocked Cubical Type Theory is sketched in section 6. Finally, we discuss related work in section 7, and conclude and discuss future work in section 8.

2 Cubical Type Theory

Cubical Type Theory [21] is an extension of Martin-Löf type theory which gives a computational interpretation to extensionality principles like function extensionality and univalence. It does so by internalizing the notion from Homotopy Type Theory that equalities are paths, by explicitly representing them as maps from an interval object. The interval, $\mathbb{I}$, is not a type but contexts can contain $i : \mathbb{I}$ assumptions and there is a judgement, $\Gamma \vdash r : \mathbb{I}$ which specifies that r is built according to this grammar

$$r, s ::= 0 \mid 1 \mid i \mid 1 - r \mid r \wedge s \mid r \vee s$$

where i is taken from the assumptions in Γ. Equality of two interval elements, $\Gamma \vdash r \equiv s : \mathbb{I}$ corresponds to the laws of De Morgan algebras. A good intuition is to think of $\mathbb{I}$ as the real interval $[0, 1]$ with $r \wedge s$ and $r \vee s$ given by minimum and maximum operations.

Given $x, y : A$, we write $\mathsf{Path}_A(x, y)$ for the type of paths between x and y, which correspond to maps from $\mathbb{I}$ into A which are equal to x or y when applied to the endpoints $0, 1$ of the interval $\mathbb{I}$.

$$\frac{\Gamma, i : \mathbb{I} \vdash t : A}{\Gamma \vdash \lambda i.\, t : \mathsf{Path}_A(t[0/i], t[1/i])}$$

$$\frac{\Gamma \vdash p : \mathsf{Path}_A(a_0, a_1) \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash p\, r : A}$$

We sometimes write $=_A$ or simply $=$ as infix notation for path equality, and use $\equiv$ for judgemental equality. Path types support the β and η equalities familiar from function types, but we also have that a path applied to 0 or 1 reduces according to the endpoints specified in its type.

$$(\lambda i.\, t)\, r \equiv t[r/i] \qquad p \equiv (\lambda i.\, p\, i) \qquad p\, 0 \equiv a_0 \qquad p\, 1 \equiv a_1$$

The path witnessing reflexivity is defined as $\mathsf{refl}\, x \stackrel{\text{def}}{=} \lambda i.\, x : \mathsf{Path}_A(x, x)$, given any $x : A$. It is also straightforward to provide a proof of function extensionality, as it is just a matter of reordering arguments:

$$\frac{\Gamma \vdash p : \Pi(x : A).\, \mathsf{Path}_B(f\, x, g\, x)}{\Gamma \vdash \lambda i.\, \lambda x.\, p\, x\, i : \mathsf{Path}_{\Pi(x:A).B}(f, g)}$$

Other constructions like transitivity, or more generally transport along path, require a further primitive operation called composition, which we now recall.

Define first the *face lattice* $\mathbb{F}$, as the free distributive lattice with generators $(i = 0)$ and $(i = 1)$, and satisfying the equality $(i = 0) \wedge (i = 1) = 0_{\mathbb{F}}$. Explicitly, elements $\Gamma \vdash \varphi : \mathbb{F}$ are given by the following grammar, where i ranges over the interval variables from Γ:

$$\varphi, \psi ::= 0_{\mathbb{F}} \mid 1_{\mathbb{F}} \mid (i = 0) \mid (i = 1) \mid \varphi \wedge \psi \mid \varphi \vee \psi$$

Given $\Gamma \vdash \varphi : \mathbb{F}$ we can form the restricted context Γ, φ. Types and terms in context Γ, φ are called *partial*, matching the intuition that they are only defined for elements of Γ that satisfy the constraints specified by φ. For example while a type $i : \mathbb{I} \vdash A$ type corresponds to a whole line, a type $i : \mathbb{I}, (i = 0) \vee (i = 1) \vdash B$ type is merely two disconnected points. Given a partial element $\Gamma, \varphi \vdash u : A$ we write $\Gamma \vdash t : A[\varphi \mapsto u]$ to mean the conjunction of $\Gamma \vdash t : A$ and $\Gamma, \varphi \vdash t \equiv u : A$. Likewise we write $\Gamma \vdash A[\varphi \mapsto B]$ type to mean $\Gamma \vdash A$ type and $\Gamma, \varphi \vdash A \equiv B$ type.

Composition is given by the following typing rule:

$$\frac{\begin{array}{c}\Gamma \vdash \varphi : \mathbb{F} \qquad \Gamma, i : \mathbb{I} \vdash A \text{ type} \\ \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash u_0 : A[0/i][\varphi \mapsto u[0/i]]\end{array}}{\Gamma \vdash \mathsf{comp}^i_A\, [\varphi \mapsto u]\, u_0 : A[1/i][\varphi \mapsto u[1/i]]}$$

the inputs u and u_0 represent the sides and the base of an open box, while composition gives us the lid, i.e. the side opposite to the base. The behaviour of composition is specified by u whenever φ is satisfied, and otherwise depends on the type A, consequently when we introduce new type formers we will also give a corresponding equation for comp. As an example we can prove transitivity for paths with the following term:

$$\frac{\Gamma \vdash p : \mathsf{Path}_A(x, y) \qquad \Gamma \vdash q : \mathsf{Path}_A(y, z)}{\Gamma \vdash \lambda i.\, \mathsf{comp}^j\, [i = 0 \mapsto x, i = 1 \mapsto q\, j]\, (p\, i) : \mathsf{Path}_A(x, z)}$$

The standalone syntax for partial elements is as a list of faces and terms $[\varphi_1\, u_1, \ldots, \varphi_n\, u_n]$, but we will write $[\vee_i \varphi_1 \mapsto [\varphi_1\, u_1, \ldots, \varphi_n\, u_n]]$ as $[\varphi_1 \mapsto u_1, \ldots, \varphi_n \mapsto u_n]$. We refer to Cohen et al. [21] for more details on partial elements, composition, and the *glueing construction*. The latter is what allows to derive the univalence principle as a theorem, by turning a partial equivalence into a total one. Writing $A \simeq B$ for the type stating that A and B are equivalent [50], univalence

states that the canonical map of type $A =_{\mathsf{U}} B \to A \simeq B$ is itself an equivalence for any types $A, B : \mathsf{U}$. We will use univalence in the rest of the paper, but we will not need to refer to a specific implementation, so we omit the details in this brief overview.

From Homotopy Type Theory [50] we also recall the notion of homotopy *proposition* which are types for which any two elements are path equal, and homotopy *set* which are types whose path types are propositions. Given any type A its propositional truncation $\|A\|_0$ is the least proposition that admits a map from A. It can be defined as an Higher Inductive Type (HIT), which are inductive types defined by providing generators for both the type itself and its path equality. Another example of a HIT is the finite powerset [27] of a type A mentioned in the introduction. We will discuss HITs in more detail in section 5.

3 Clocked Cubical Type Theory

CCTT extends CTT with the constructions of CloTT [7, 36], a type theory with Nakano style guarded recursion, multiple clocks and ticks. This means that the delay type operator is associated with clocks: $\triangleright^\kappa A$ is the type of data of type A which is available in the next time step on clock κ. There are no operations on clocks, they can only be assumed by placing assumptions such as $\kappa : \mathsf{clock}$ in the context, and abstracted to form elements of type $\forall\kappa.A$. The status of clock is similar to $\mathbb{I}$ in the sense that it is not a type, and so, e.g., clock $\to$ clock is not a wellformed type. It is sometimes convenient to assume a single clock constant κ_0, and this can be achieved by a precompilation adding this to the context. Similarly to function types, path equality in $\forall\kappa.A$ is equivalent to pointwise equality, and clock quantification preserves truncation levels.

Ticks are evidence that time has passed on a given clock. Tick assumptions in a context separate a judgement such as $\Gamma, \alpha : \kappa, \Gamma' \vdash t : A$ into a part (Γ) of assumptions arriving before the time tick α on the clock κ and the rest of the judgement that happens after. The introduction rule for $\triangleright\,(\alpha : \kappa).A$ in Figure 1 can therefore be read as stating that if t has type A one time step after the assumptions in Γ, then $\lambda(\alpha:\kappa).t$ is delayed data of type A. Because terms can appear in types, α can appear in A, and must therefore be mentioned in the type of $\lambda(\alpha:\kappa).t$. In many cases, however, it does not, and we will then write simply $\triangleright^\kappa A$ for $\triangleright\,(\alpha:\kappa).A$.

Elimination for $\triangleright\,(\alpha:\kappa).A$ is by application to ticks. There are two forms of ticks: simple ticks and forcing ticks. The judgement for simple ticks $\Gamma \vdash u : \kappa \rightsquigarrow \Gamma'$ states that u is a tick on the clock κ in context Γ with *residual context* Γ'. The residual context consists of the assumptions that were available before the tick u. In the case of a tick variable α, these are the assumptions to the left of α as well as the *timeless* assumptions to the right of α. Timeless assumptions are interval and clock assumptions ($i : \mathbb{I}$ and $\kappa : \mathsf{clock}$) as well as all faces, and $\mathsf{TL}(\Gamma)$ computes the timeless assumptions of Γ. The assumption $\Gamma'' \sqsubseteq \Gamma$ defined by the rule

$$\Gamma, \mathsf{TL}(\Gamma') \sqsubseteq \Gamma, \Gamma'$$

allows for further trimming of the residual context. The term $t\,_{\Gamma'}[u]$ applies t to the simple tick u. The assumption that t is well-typed in the residual context Γ' should be read as stating that it has the type $\triangleright\,(\alpha:\kappa).A$ before the tick u, and can therefore be opened after the tick u to produce something of type $A[u/\alpha]$. We will often omit the residual context Γ' from the term, writing simply $t\,[u]$ for $t\,_{\Gamma'}[u]$. Different choices of Γ' give the same term up to judgemental equality.

As basic examples of programming with simple ticks we mention the dependently typed generalisation of applicative action (1), an equivalence of types

$$\triangleright^\kappa(\Sigma(x : A).B(x)) \simeq \Sigma(x : \triangleright^\kappa A).\triangleright\,(\alpha:\kappa).(B(x\,[\alpha])) \tag{3}$$

and the unit

$$\lambda x.\lambda(\alpha:\kappa).x : A \to \triangleright^\kappa A \tag{4}$$

Note that the latter uses the rule for variables which allows these to be introduced from anywhere in the context, also across ticks. Some Fitch style modal type theories do not allow such introductions [17, 19]. The timelessness of the interval is needed to type the extensionality principle

$$\mathsf{Path}_{\triangleright^\kappa A}(x, y) \simeq \triangleright\,(\alpha:\kappa).\mathsf{Path}_A(x\,[\alpha], y\,[\alpha])$$

where the left to right direction $\lambda p.\lambda(\alpha:\kappa).\lambda i.p\;i\;[\alpha]$ requires $p\;i\;[\alpha]$ to be welltyped in a context where i occurs after α.

3.1 Forcing ticks

Unrestricted elimination of $\triangleright$ is unsound, since any type A with a map $\triangleright^\kappa A \to A$ can be inhabited using fixed points. However, it is safe to eliminate $\triangleright^\kappa$ in contexts of the form $\Gamma, \kappa : \mathsf{clock}$, i.e., where no variables mention κ in their type. To close such an elimination rule under substitutions, the clock variable κ must be abstracted in the term.

Application to forcing ticks does exactly that: If κ' is a clock in Γ then $\Gamma \vdash (\kappa', \diamond) \rightsquigarrow \Gamma$ is a forcing tick and so if $\Gamma, \kappa : \mathsf{clock} \vdash t : \triangleright(\alpha : \kappa).A$ then

$$\Gamma \vdash (\kappa.t)\,_\Gamma[(\kappa', \diamond)] : A[(\diamond : \kappa')/(\alpha:\kappa)]$$

This construction binds κ in t, and $A[(\diamond : \kappa')/(\alpha:\kappa)]$ uses a special simultaneous substitution of the pair $(\diamond, \kappa')$ for the pair (α, κ). In particular $[(\diamond : \kappa')/(\alpha:\kappa)]$ commutes as expected with type and term formers, replacing each of α and κ with $\diamond$ and κ' as intended, but it will also have to turn a simple tick application like $t\,[\alpha]$ into a forcing tick application $(\kappa.t)\;[(\kappa', \diamond)]$ to preserve typing. This substitution operation was originally described by Mannaa et al. [36].

Forcing ticks can be used to define the force operator of Atkey and McBride [6]

$$\mathsf{force} \stackrel{\mathsf{def}}{=} \lambda x.\lambda\kappa.(\kappa.x\,[\kappa])\;[(\kappa, \diamond)] : \forall\kappa.\triangleright^\kappa A \to \forall\kappa.A \tag{5}$$

which is used in the encoding of coinductive types. Replacing the delayed substitution of κ' for κ in $(\kappa.t)\,_\Gamma[(\kappa', \diamond)]$ with an

Context and type formation rules

$$\frac{\Gamma \vdash \qquad \kappa \notin \Gamma}{\Gamma, \kappa : \mathsf{clock} \vdash} \qquad \frac{\Gamma \vdash \kappa : \mathsf{clock} \qquad \alpha \notin \Gamma}{\Gamma, \alpha : \kappa \vdash} \qquad \frac{\Gamma, \alpha : \kappa \vdash A \text{ type}}{\Gamma \vdash \triangleright\,(\alpha{:}\kappa).A \text{ type}} \qquad \frac{\Gamma, \kappa : \mathsf{clock} \vdash A \text{ type}}{\Gamma \vdash \forall \kappa.A \text{ type}}$$

Typing rules

$$\frac{\Gamma, \kappa : \mathsf{clock} \vdash t : A}{\Gamma \vdash \lambda\kappa.t : \forall\kappa.A} \qquad \frac{\Gamma \vdash t : \forall\kappa.A \qquad \Gamma \vdash \kappa' : \mathsf{clock}}{\Gamma \vdash t[\kappa'] : A[\kappa'/\kappa]} \qquad \frac{\Gamma, \alpha : \kappa \vdash t : A}{\Gamma \vdash \lambda(\alpha{:}\kappa).t : \triangleright\,(\alpha{:}\kappa).A}$$

$$\frac{\Gamma \vdash u : \kappa \rightsquigarrow \Gamma' \qquad \Gamma' \vdash t : \triangleright\,(\alpha{:}\kappa).A}{\Gamma \vdash t\,{}_{\Gamma'}[u] : A[u/\alpha]} \qquad \frac{\Gamma \vdash (\kappa', u) \rightsquigarrow \Gamma' \qquad \Gamma', \kappa : \mathsf{clock} \vdash t : \triangleright(\alpha : \kappa).A}{\Gamma \vdash (\kappa.t)\,{}_{\Gamma'}[(\kappa', u)] : A[(u : \kappa')/(\alpha{:}\kappa)]} \qquad \frac{\Gamma \vdash t : \triangleright^{\kappa} A \to A}{\Gamma \vdash \mathsf{dfix}^{\kappa}\, t : \triangleright^{\kappa} A}$$

$$\frac{\Gamma \vdash t : \triangleright^{\kappa} A \to A}{\Gamma \vdash \mathsf{pfix}^{\kappa}\, t : \triangleright\,(\alpha{:}\kappa).\mathsf{Path}_A((\mathsf{dfix}^{\kappa} t)\,[\alpha], t(\mathsf{dfix}^{\kappa} t))} \qquad \frac{\Gamma, \alpha : \kappa \vdash A : \mathsf{U}}{\Gamma \vdash \overline{\triangleright}\,(\alpha{:}\kappa).A : \mathsf{U}} \qquad \frac{\Gamma, \kappa : \mathsf{clock} \vdash A : \mathsf{U}}{\Gamma \vdash \overline{\forall}\kappa.A : \mathsf{U}}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash t : B} \qquad \frac{\kappa : \mathsf{clock} \in \Gamma}{\Gamma \vdash \kappa : \mathsf{clock}} \qquad \frac{x : A \in \Gamma}{\Gamma \vdash x : A}$$

Simple ticks

$$\frac{\Gamma'' \sqsubseteq \Gamma}{\Gamma, \alpha : \kappa, \Gamma' \vdash \alpha : \kappa \rightsquigarrow \Gamma'', \mathsf{TL}(\Gamma')} \qquad \frac{\Gamma \vdash u : \kappa \rightsquigarrow \Gamma' \qquad \Gamma \vdash v : \kappa \rightsquigarrow \Gamma' \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash \mathsf{tirr}(u, v, r) : \kappa \rightsquigarrow \Gamma'}$$

Forcing ticks

$$\frac{\Gamma' \sqsubseteq \Gamma \qquad \Gamma' \vdash \kappa' : \mathsf{clock}}{\Gamma \vdash (\kappa', \diamond) \rightsquigarrow \Gamma'} \qquad \frac{\Gamma'' \sqsubseteq \Gamma}{\Gamma, \alpha : \kappa', \Gamma' \vdash (\kappa', \alpha) \rightsquigarrow \Gamma'', \mathsf{TL}(\Gamma')}$$

$$\frac{\Gamma \vdash (\kappa', u) \rightsquigarrow \Gamma' \qquad \Gamma \vdash (\kappa', v) \rightsquigarrow \Gamma' \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash (\kappa', \mathsf{tirr}(u, v, r)) \rightsquigarrow \Gamma'}$$

Timeless assumptions

$$\mathsf{TL}(\Gamma, x : A) = \mathsf{TL}(\Gamma) \qquad \mathsf{TL}(\Gamma, \alpha : \kappa) = \mathsf{TL}(\Gamma) \qquad \mathsf{TL}(\Gamma, i : \mathbb{I}) = \mathsf{TL}(\Gamma), i : \mathbb{I}$$
$$\mathsf{TL}(\Gamma, \kappa : \mathsf{clock}) = \mathsf{TL}(\Gamma), \kappa : \mathsf{clock} \qquad \mathsf{TL}(\Gamma, \varphi) = \mathsf{TL}(\Gamma), \varphi \qquad \mathsf{TL}(\cdot) = \cdot$$

Figure 1. Selected typing rules for CCTT. The rules for ticks use the relation defined as $\Gamma, \mathsf{TL}(\Gamma') \sqsubseteq \Gamma, \Gamma'$

Judgemental equalities on terms

$$t\,{}_{\Gamma}[u] \equiv t\,{}_{\Gamma'}[u] \qquad (\kappa.t)\,{}_{\Gamma}[(\kappa, u)] \equiv (\kappa.t)\,{}_{\Gamma'}[(\kappa, u)] \qquad (\lambda\kappa.t)[\kappa'] \equiv t[\kappa'/\kappa]$$
$$\lambda\kappa.(t[\kappa]) \equiv t \qquad (\lambda(\alpha{:}\kappa).t)\,[u] \equiv t\,[u/\alpha] \qquad \lambda(\alpha{:}\kappa).(t\,[\alpha]) \equiv t$$
$$(\kappa.\lambda(\alpha{:}\kappa).t)\,[(\kappa', u)] \equiv t[(u : \kappa')/(\alpha{:}\kappa)] \qquad (\kappa.t)\,[(\kappa', u)] \equiv (t[\kappa'/\kappa])\,[u] \qquad (\kappa.\mathsf{dfix}^{\kappa}\, t)\,[(\kappa', \diamond)] \equiv t(\mathsf{dfix}^{\kappa}\, t)[\kappa'/\kappa]$$
$$\mathsf{El}\,(\overline{\forall}\kappa.A) \equiv \forall\kappa.\mathsf{El}\,(A) \qquad (\kappa.\mathsf{pfix}^{\kappa}\, t)\,[(\kappa', \diamond)]\, r \equiv t(\mathsf{dfix}^{\kappa}\, t)[\kappa'/\kappa] \qquad \mathsf{El}\,(\overline{\triangleright}\,(\alpha{:}\kappa).A) \equiv \triangleright\,(\alpha{:}\kappa).\mathsf{El}\,(A)$$

Judgemental equalities on ticks

$$\mathsf{tirr}(u, v, 0) \equiv u \qquad \mathsf{tirr}(\diamond, \diamond, r) \equiv \diamond \qquad \mathsf{tirr}(u, v, 1) \equiv v$$

Figure 2. Selected judgemental equality rules of CCTT. The three η rules are subject to the standard conditions of κ and α, respectively, not appearing in t. As a consequence of the first two rules, the residual context is omitted for tick application in the rules below. The 8th axiom listed assumes that $\diamond$ does not appear in u, so that u can be considered a simple tick.

actual substitution, gives the elimination rule used by Bahr. et al. [7]. Here we opt for an explicit substitution because it is easier to type check and give semantics to, which is why Mannaa et al. [36] made a similar choice. Bahr. et al. [7] designed their application rule for $\diamond$ to obtain a normalisation result for their theory. We expect that using the present rule, one can prove the same property for CCTT.

3.2 Tick irrelevance

Since fixed points unfold when applied to $\diamond$, the identity of ticks is crucial for the operational properties of Clocked Type Theory, in particular for ensuring strong normalisation. However, on the extensional level the identity of ticks is irrelevant, as stated by the *tick irrelevance axiom*

$$\mathsf{tirr}^\kappa : \Pi(x : \triangleright^\kappa A).\triangleright\,(\alpha:\kappa).\triangleright\,(\beta:\kappa).\mathsf{Path}_A(x\,[\alpha], x\,[\beta]) \quad (6)$$

This can be inhabited in CCTT using the novel construction tirr building a path between any pair of ticks by defining

$$\mathsf{tirr}^\kappa(t) \stackrel{\text{def}}{=} \lambda(\alpha:\kappa).\lambda(\beta:\kappa).\lambda i.t\,[\mathsf{tirr}(\alpha,\beta,i)]$$

Similar axioms can be constructed for forcing tick applications. One important application of tirr is in showing that force (5) is a type equivalence with inverse $\mathsf{force}^{-1} \stackrel{\text{def}}{=} \lambda x.\lambda\kappa.\lambda(\alpha:\kappa).x\,[\kappa]$: If $x : \forall\kappa.\triangleright^\kappa A$, then

$$\begin{aligned}\mathsf{force}^{-1}(\mathsf{force}\,x) &\equiv \lambda\kappa.\lambda(\alpha:\kappa).(\kappa.x\,[\kappa])\,[(\kappa,\diamond)]\\ &= \lambda\kappa.\lambda(\alpha:\kappa).(\kappa.x\,[\kappa])\,[(\kappa,\alpha)]\\ &\equiv \lambda\kappa.\lambda(\alpha:\kappa).x\,[\kappa]\,[\alpha] \equiv x\end{aligned}$$

where the path equality is witnessed by

$$\lambda i.\lambda\kappa.\lambda(\alpha:\kappa).(\kappa.x\,[\kappa])\,[(\kappa,\mathsf{tirr}(\diamond,\alpha,i))].$$

The ability of tirr to form a path between any two ticks allows us to prove coherence laws between different uses of tick irrelevance. For example we can construct fillers for boxes whose boundary is given by tick applications as in the lemma below.

Lemma 3.1. *Let* $\Gamma' \vdash t : \triangleright\,(\alpha:\kappa).A$ *and* $\Gamma, i_0, \dots, i_n : \mathbb{I}, \varphi \vdash u : \kappa \rightsquigarrow \Gamma'$ *where* $\varphi = \bigvee_{k,b}(i_k = b)$ *for* $k = 0, \dots, n$ *and* $b = 0, 1$. *Then we have* $\Gamma, i_0, \dots, i_n : \mathbb{I} \vdash B[\varphi \mapsto A[u/\alpha]]$ type *and a term* $\Gamma, i_0, \dots, i_n : \mathbb{I} \vdash \mathsf{filler}(t,u) : B[\varphi \mapsto t\,[u]]$.

For example, the lemma constructs a filler of the diagram below, thus constructing a path from the composition of $\mathsf{tirr}^\kappa(t)\,[\alpha]\,[\beta]$ and $\mathsf{tirr}^\kappa(t)\,[\beta]\,[\gamma]$ to $\mathsf{tirr}^\kappa(t)\,[\alpha]\,[\gamma]$,

$$\begin{array}{ccc} A & \xrightarrow{\mathsf{tirr}^\kappa(t)\,[\beta]\,[\gamma]} & A \\ {\scriptstyle \mathsf{tirr}^\kappa(t)\,[\alpha]\,[\beta]}\big\downarrow & & \big\downarrow{\scriptstyle \mathsf{tirr}^\kappa(t)\,[\alpha]\,[\gamma]} \\ A & \xrightarrow{t\,[\alpha]} & A \end{array}$$

The rules for judgemental equality (Figure 2) include standard β and η-rules for functions, clock abstraction and tick abstraction. Note that there are two β equalities for tick application, one for each kind of ticks. There is also a rule stating that if a forcing tick does not contain $\diamond$, application to it reduces to application to a simple tick. One consequence of this is that an η-rule for forcing tick application can be derived as follows

$$\lambda(\alpha:\kappa').((\kappa.t)\,[(\alpha,\kappa')]) \equiv \lambda(\alpha:\kappa').(t[\kappa'/\kappa]\,[\alpha]) \equiv t[\kappa'/\kappa]$$

Although we do not prove canonicity for our type theory, we set up equalities that we believe are necessary to compute canonical forms in contexts of free clock and interval variables, but not free tick variables. For example, $\mathsf{tirr}(\diamond,\diamond,r) \equiv \diamond$ ensures that in a context with no tick variables all ticks are equal to $\diamond$.

3.3 Fixed points

The operator $\mathsf{dfix}^\kappa : (\triangleright^\kappa A \to A) \to \triangleright^\kappa A$ computes fixed points of productive functions. Using this, one can construct Nakano's fixed point operator as

$$\mathsf{fix}^\kappa \stackrel{\text{def}}{=} \lambda f.f(\mathsf{dfix}^\kappa f) : (\triangleright^\kappa A \to A) \to A \quad (7)$$

To ensure termination, fixed points do not unfold judgementally, except when applied to $\diamond$. As in Guarded Cubical Type Theory [9] we add a path pfix^κ between a fixed point and its unfolding. Note that the right hand endpoint of $\mathsf{pfix}^\kappa t$ is $\mathsf{fix}^\kappa t$ and so,

$$\mathsf{fix}^\kappa t \equiv t(\lambda(\alpha:\kappa).\mathsf{dfix}^\kappa t\,[\alpha]) = t(\lambda(\alpha:\kappa).\mathsf{fix}^\kappa t) \quad (8)$$

With this we can prove that fixed points are unique, in fact the following stronger statement is true.

Lemma 3.2. *The types* $\Sigma(x : A).\mathsf{Path}_A(x, f\,(\lambda(\alpha:\kappa).x))$ *and*

$$\Sigma(x : \triangleright^\kappa A).\triangleright\,(\alpha:\kappa).\mathsf{Path}_A(x\,[\alpha], f(x))$$

are contractible for every $f : \triangleright^\kappa A \to A$.

By lemma 3.2 the pair $(\mathsf{dfix}^\kappa t, \mathsf{pfix}^\kappa t)$ is uniquely determined up to path, as it witnesses that $\mathsf{dfix}^\kappa t$ is the fixed point of $\lambda x.\lambda(\_:\kappa).t\,x$. Moreover both dfix^κ and pfix^κ unfold when applied to $\diamond$, which means they will not be stuck terms in a context with no tick variables.

The principles above can be used to encode guarded recursive types as fixed points on the universe. For example, assuming codes for products and natural numbers, we define

$$\overline{\mathsf{Str}^\kappa}(\mathbb{N}) \stackrel{\text{def}}{=} \mathsf{fix}^\kappa(\lambda X.\overline{\mathbb{N}}\overline{\times}\overline{\triangleright}\,(\alpha:\kappa).X\,[\alpha])$$

and then by (8) $\overline{\mathsf{Str}^\kappa}(\mathbb{N}) = \overline{\mathbb{N}}\overline{\times}\overline{\triangleright}\,(\alpha:\kappa).\overline{\mathsf{Str}^\kappa}(\mathbb{N})$ so, defining $\mathsf{Str}^\kappa(\mathbb{N}) \stackrel{\text{def}}{=} \mathsf{El}\,(\overline{\mathsf{Str}^\kappa}(\mathbb{N}))$ gives the type equivalence

$$\mathsf{Str}^\kappa(\mathbb{N}) \simeq \mathbb{N} \times \triangleright^\kappa \mathsf{Str}^\kappa(\mathbb{N}) \quad (9)$$

Note that we do not assume the *clock irrelevance axiom*

$$\frac{\Gamma \vdash t : \forall\kappa.A \qquad \kappa \notin \mathsf{fc}(A)}{\Gamma \vdash \mathsf{cirr}^\kappa t : \forall\kappa'.\forall\kappa''.\mathsf{Path}_A(t[\kappa'], t[\kappa''])} \quad (10)$$

often assumed in type theories with multi-clock guarded recursion [7, 11]. Instead we will use the following definition.

Definition 3.3. A type A is *clock-irrelevant* if the canonical map $A \to \forall\kappa.A$ (for κ fresh) is an equivalence.

If A is clock-irrelevant it satisfies axiom (10), and assuming a clock constant the two statements are equivalent. Clock irrelevance is needed for the correctness of encodings of coinductive types. For example, using (9) one can prove

$$\forall\kappa.\mathsf{Str}^\kappa(\mathbb{N}) \simeq (\forall\kappa.\mathbb{N}) \times \forall\kappa.(\triangleright^\kappa \mathsf{Str}^\kappa(\mathbb{N})) \simeq \mathbb{N} \times \forall\kappa.\mathsf{Str}^\kappa(\mathbb{N})$$

assuming $\mathbb{N}$ clock-irrelevant, and using that force is an equivalence. In this paper, rather than assuming all types clock irrelevant, we will prove clock-irrelevance for a large class of types and use this to construct final coalgebras for a correspondingly large class of functors.

4 Encoding coinductive types

In the previous section we saw that the type $\forall\kappa.\mathsf{Str}^\kappa(\mathbb{N})$ satisfies the type equivalence expected of a type of streams. In this section we prove that (assuming $\mathbb{N}$ clock-irrelevant) this type also satisfies the universal property characterising streams, as a special case of a more general property. In Section 5 we will show that $\mathbb{N}$ is indeed clock-irrelevant (as a special case of Theorem 5.3) as are a large collection of inductive and higher inductive types.

Here we essentially adapt the final coalgebra construction from Møgelberg [39]. Given a type I consider the category[1] $I \to \mathsf{U}$ of I-indexed types whose type of objects is $I \to \mathsf{U}$ and whose morphisms from X to Y is the type

$$X \to Y \stackrel{\text{def}}{=} \Pi(i : I).\, \mathsf{El}\,(X\,i) \to \mathsf{El}\,(Y\,i)$$

An *I-indexed endofunctor* is an endofunctor on $I \to \mathsf{U}$.

Let F be an I-indexed endofunctor, an *F-coalgebra* is a pair (X, f) such that $X : I \to \mathsf{U}$ and $f : X \to FX$. We say an F-coalgebra (Y, g) is *final* if for all coalgebras (X, f) the following type is contractible

$$\Sigma(h : X \to Y).\, g \circ h = F(h) \circ f$$

Using contractibility we address the fact that there can be more than one proof of path equality, as is also done by Ahrens et al. [3].

If $X : \forall\kappa.\ (I \to \mathsf{U})$ write $\forall\kappa.X$ for $\lambda(i : I).\overline{\forall}\kappa.(X\ [\kappa]\ i)$.

Definition 4.1. An I-indexed endofunctor F *commutes with clock quantification* if for all families X the canonical map $\mathsf{can}_F : F\,(\forall\kappa.X\ [\kappa]) \to \forall\kappa.\,F\,(X\ [\kappa])$ is a pointwise equivalence.

For example, if $A : \mathsf{U}$ the constant functor to A commutes with clock quantification if and only if A is clock-irrelevant.

Lemma 4.2. *The collection of endofunctors commuting with clock quantification is closed under composition, pointwise product, pointwise Π, pointwise Σ over clock irrelevant types, and pointwise universal quantification over clocks. If F commutes with clock quantification then the guarded recursive type X satisfying $X \simeq F(\triangleright^\kappa X)$ is clock irrelevant. Any path type of a clock irrelevant type is clock irrelevant.*

The following theorem states that all such endofunctors have final coalgebras. The idea of this originates with Atkey and McBride [6], and our proof is an adaptation of the proof by Møgelberg [39] of a similar statement in extensional type theory. It refers to the endofunctor $\triangleright^\kappa$, defined as the pointwise extension of the functor $\mathsf{U} \to \mathsf{U}$ mapping A to $\overline{\triangleright}\,(\alpha : \kappa).A$ for a fresh α.

Theorem 4.3. *Let F be an I-indexed endofunctor which commutes with clock quantification, and let $\nu^\kappa(F)$ be the guarded recursive type satisfying $\nu^\kappa(F) \simeq F(\triangleright^\kappa(\nu^\kappa(F))$. Then $\nu(F) \stackrel{def}{=} \forall\kappa.\nu^\kappa(F)$ has a final F-coalgebra structure.*

Example 4.4. In Section 5 we will prove that $\mathbb{N}$ is clock-irrelevant, and hence the functor $F(X) = \mathbb{N} \times X$ commutes with clock quantification. Theorem 4.3 then states correctness of the encoding of streams as $\forall\kappa.\mathsf{Str}^\kappa(\mathbb{N})$.

Example 4.5. Consider the functor $F(X) = \mathsf{P}_\mathsf{f}(A \times X)$, where P_f is the finite powerset functor defined as a HIT [27], and A is assumed to be a clock-irrelevant set. In Section 5 we will prove that P_f commutes with clock quantification, which implies that F does the same. Theorem 4.3 then gives an encoding of the final coalgebra for F. This type plays an important role in automata theory as it describes the finitely branching A-labelled transition systems. Let LTS^κ denote fixed point for $F \circ \triangleright^\kappa$, let $\mathsf{LTS} \stackrel{\text{def}}{=} \forall\kappa.\mathsf{LTS}^\kappa$ be the coinductive type and let $\mathsf{ufld} : \mathsf{LTS} \to \mathsf{P}_\mathsf{f}(A \times \mathsf{LTS})$ be the final coalgebra structure.

Suppose now $x, y : \mathsf{LTS}$ and $R : \mathsf{LTS} \times \mathsf{LTS} \to \mathsf{U}$. Define

$$\mathsf{Bisim}_\forall(R)(x, y) = \mathsf{Sim}_\forall(R)(x, y) \times \mathsf{Sim}_\forall(R)(y, x)$$

where $\mathsf{Sim}_\forall(R)(x, y)$ is

$$\Pi x', a.(a, x') \in \mathsf{ufld}(x) \to \exists y'.((a, y') \in \mathsf{ufld}(y)) \times R(x', y')$$

where $\in$ is defined by induction [27]. By Lemma 4.2, LTS is clock irrelevant, and an easy induction on $X : \mathsf{P}_\mathsf{f}(A \times \mathsf{LTS})$ shows that the predicate $(a, x') \in X$ is clock irrelevant. In Section 5 we will prove that propositional truncation commutes with clock quantification, and since $\exists$ in homotopy type theory is defined as the composition of truncation and Σ [50], putting all this together shows that $\mathsf{Bisim}_\forall(-)$ defines a $\mathsf{LTS} \times \mathsf{LTS}$-indexed endofunctor commuting with clock quantification, and so Theorem 4.3 gives an encoding of bisimilarity as a coinductive type.

Møgelberg and Veltri [41] prove that path equality coincides with bisimilarity for guarded recursive types. Using their results we can prove the following.

Theorem 4.6. *Two elements of the type* LTS *from Example 4.5 are bisimilar if and only if they are path equal. Since both these are propositions, they are equivalent as types.*

[1] Here we mean the naive internal notion where the type of arrows is not necessarily truncated, but we also do not require higher coherences. Same for functor below.

$$\frac{}{\vdash \cdot \square} \qquad \frac{\vdash \Psi \ \square}{\vdash \Psi, i : \mathbb{I} \ \square} \qquad \frac{\Gamma_0 \vdash}{\Gamma_0 \vdash \cdot \ \mathsf{tel}}$$

$$\frac{\Gamma_0 \vdash \Gamma \ \mathsf{tel} \qquad \Gamma_0, \Gamma \vdash A \ \mathsf{type}}{\Gamma_0 \vdash \Gamma, x : A \ \mathsf{tel}}$$

Figure 3. Telescope judgements.

The results of Example 4.5 and Theorem 4.6 are liftings of results by Møgelberg and Veltri [41] for guarded recursive types to coinductive types. One benefit of passing to coinductive types is the existence of non-causal functions. For example, it is not possible to write an endofunction on $\mathsf{Str}^\kappa(\mathbb{N})$ filtering out every other element of the input stream. Instead this can be written as an endofunction on $\mathsf{Str}(\mathbb{N})$ [6]. Similar examples can be imagined for Theorem 4.5, for example, a function that collapses an LTS over labels to one over pairs of labels. Another benefit is that weak bisimilarity can be defined as an equivalence relation on coinductive types. Møgelberg and Paviotti [40] construct a weak bisimilarity relation on the guarded recursive type $\mathsf{L}^\kappa X \simeq X + \triangleright^\kappa \mathsf{L}^\kappa X$, and lift this to a relation on a model of FPC. The weak bisimilarity relation on $\mathsf{L}^\kappa X$ is not transitive, in fact, its transitive closure relates all pairs of values, but the transitive closure of the induced relation on $\mathsf{L}X \stackrel{\text{def}}{=} \forall \kappa. \mathsf{L}^\kappa X$ coincides with the one studied by Capretta [13] for the coinductive delay monad.

5 Higher Inductive Types

We now extend CCTT with higher inductive types, adapting a schema for these from Cavallo and Harper [14] to our version of CTT. For simplicity we exclude indexed families, but the schema is still general enough to cover examples like spheres, pushouts, W-suspensions [47], (higher) truncations, as well as finite and countable powersets. We first present the schema and the introduction rules for HITs, then (subsection 5.2) we present our principle of induction under clocks, which generalises the elimination rule for HITs.

5.1 Introduction and formation

The judgements of Figure 4 capture declarations of the form

$$\begin{array}{l}\mathsf{data}\ \mathsf{H}\ (\delta : \Delta)\ \mathsf{where}\\ \quad\vdots\\ \quad \ell_i : (\gamma : \Gamma) \to (\overline{x : \overline{\Xi \to \mathsf{H}\,\delta}}) \to (\overline{i : \Psi}) \to \mathsf{H}\,\delta[\varphi \mapsto e]\\ \quad\vdots\end{array}$$

which list constructors and their types for a new datatype H, taking parameters in the telescope Δ. On top of the declared constructors every HIT has an introduction form for *homogeneous composition*, written $\mathsf{hcomp}^i_{\mathsf{H}\,\delta}\,[\varphi \mapsto u]\,u_0$, where δ is not allowed to depend on i, as well as a transport operation. Following Coquand et al. [22], the composition structure for

Constructor declarations (presuppose $\cdot \vdash \Delta \ \mathsf{tel}$)

$$\frac{\mathscr{K} = (\ell_1, C_1) \ldots (\ell_n, C_n) \qquad (\forall i)\ \Delta; (\ell_1, C_1) \ldots (\ell_{i-1}, C_{i-1}) \vdash C_i \ \mathsf{constr}}{\Delta \vdash \mathscr{K} \ \mathsf{constrs}}$$

$$\frac{\begin{array}{c}\Delta \vdash \Gamma \ \mathsf{tel} \quad (\forall k)\ \Delta, \Gamma \vdash \Xi_k \ \mathsf{tel} \quad \vdash \Psi \ \square \quad \Psi \vdash \varphi : \mathbb{F}\\ \delta : \Delta, \Gamma, \overline{\Xi \to \mathsf{H}\,\delta}, \Psi, \varphi \vdash_{\mathscr{K}, \mathsf{H}\,\delta} e : \mathsf{H}\,\delta\\ e = [\varphi_1\ M_1 \ldots \varphi_m\ M_m] \quad (\forall k)\ M_k \in \mathsf{Bndr}(\mathscr{K}; \overline{\Xi \to \mathsf{H}\,\delta})\end{array}}{\Delta; \mathscr{K} \vdash (\Gamma; \overline{\Xi}; \Psi; \varphi; e)\ \mathsf{constr}}$$

Grammar for boundary terms

$$\begin{array}{rcl} N, M \in \mathsf{Bndr}(\mathscr{K}; \Theta) & ::= & x\ \overline{u}\\ & \mid & \mathsf{con}_\ell(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r})\\ & \mid & \mathsf{hcomp}^j_{\mathsf{H}\,\delta}\,[\varphi \mapsto M]\,M_0\end{array}$$

Formation

$$\frac{\Gamma \vdash \delta : \Delta}{\Gamma \vdash \mathsf{H}\,\delta \ \mathsf{type}}$$

Introductions

$$\frac{(\ell, (\Gamma; \overline{\Xi}; \Psi; \varphi; e)) \in \mathscr{K} \qquad \Gamma_0 \vdash \delta : \Delta \qquad \Gamma_0 \vdash \overline{t} : \Gamma[\delta] \qquad \Gamma_0 \vdash \overline{a} : \overline{\Xi[\delta, \overline{t}] \to \mathsf{H}\,\delta} \qquad \Gamma_0 \vdash \overline{r} : \Psi}{\Gamma_0 \vdash \mathsf{con}_\ell(\overline{t}, \overline{a}, \overline{r}) : \mathsf{H}\,\delta[\varphi[\overline{r}] \mapsto e[\overline{t}, \overline{a}, \overline{r}]]}$$

$$\frac{\Gamma \vdash \delta : \Delta \qquad \Gamma \vdash \varphi : \mathbb{F} \qquad \Gamma \vdash u_0 : \mathsf{H}\,\delta[\varphi \mapsto u[0/i]]}{\Gamma \vdash \mathsf{hcomp}^i_{\mathsf{H}\,\delta}\,[\varphi \mapsto u]\,u_0 : \mathsf{H}\,\delta[\varphi \mapsto u[1/i]]}$$

$$\frac{\Gamma \vdash \varphi : \mathbb{F} \qquad \Gamma, i : \mathbb{I} \vdash \delta : \Delta \qquad \Gamma, i : \mathbb{I}, \varphi \vdash \delta \equiv \delta[0] : \Delta \qquad \Gamma \vdash u_0 : \mathsf{H}\,\delta[0]}{\Gamma \vdash \mathsf{trans}^i_{\mathsf{H}\,\delta}\,\varphi\,u_0 : \mathsf{H}\,\delta[1][\varphi \mapsto u_0]}$$

Figure 4. Schema for Higher Inductive Types, $\delta : \Delta \vdash \mathsf{H}\,\delta$ type. In the typing of the boundary e the subscript $\mathscr{K}, \mathsf{H}\,\delta$ indicates that this judgement can refer to the labels from $\mathscr{K}$ and $\mathsf{H}\,\delta$ itself. The grammar for boundary terms assumes $x \in \Theta$, that $\overline{u}, \overline{t}$ not mention variables in Θ and that $\ell \in \mathscr{K}$. For further judgemental equalities for trans see Appendix A.5.

H is given by combining these two operations. We omit the details for lack of space, but just recall that from the homogeneous composition operator one can derive a *homogeneous filling* operator $\mathsf{hfill}^i_A\,[\varphi \mapsto u]\,u_0 : \mathsf{Path}_A(u_0, \mathsf{hcomp}^i_A\,[\varphi \mapsto u]\,u_0)$ which provides a filler for the box specified by u and u_0 and closed by homogeneous composition [31].

A constructor for a HIT H is specified by a tuple $\Delta; \mathscr{K} \vdash (\Gamma, \overline{\Xi}, \Psi, \varphi, e)\ \mathsf{constr}$ where $\mathscr{K}$ lists the previously declared constructors. The telescope Γ lists the non-recursive arguments, and each Ξ in $\overline{\Xi}$ is the arity for a recursive argument. Path constructors further take a non-empty telescope

of interval variables Ψ and have a boundary e specified as a partial element over the formula φ. The judgements for telescopes are given in Figure 3. Note that these imply that all of Δ, Γ, and Ξ only contain variables of proper types, i.e., no interval, face, clock or tick assumptions. Only the boundary is allowed to refer to H and the previous constructors from $\mathscr{K}$. We signify this by adding those as subscripts to the typing judgement for e. Cavallo and Harper require the components M of the boundary to conform to a dedicated typing judgement, which restricts their shape and makes it possible to correctly specify the inputs to the dependent eliminator for H. For conciseness we replicate those restrictions with a grammar, $\mathsf{Bndr}(\mathscr{K}, \Theta)$, which specifies that a boundary term must be built either from an applied recursive argument $x\,\overline{u}$, a previous constructor, or a use of hcomp. We also assume a code $\overline{\mathsf{H}}\,\delta : \mathsf{U}_i$ whenever the types in all of the Γ and Ξ telescopes in $\mathscr{K}$ have a code in U_i as well.

Remark 1. *Formally, boundary terms are a separate syntactic entity which we silently include in ordinary terms. In particular, they have a separate equality judgement which is used when typing systems and compositions of boundary terms. This equality is the congruence relation induced by reducing compositions and constructors to their boundary terms. It is likely that this equality can be proved to coincide with the one induced by equality on terms. This approach is similar to the one of Cavallo and Harper [14].*

We now give some concrete examples. For readability we write $(\ell, (\Gamma; \overline{(x : \Xi \to \mathsf{H}\,\delta)}; \Psi; [\varphi_1 \mapsto M_1, \ldots, \varphi_n \mapsto M_n]))$ in place of $(\ell, (\Gamma, \overline{\Xi}, \Psi, \vee_i \varphi_i, [\varphi_1 \mapsto M_1, \ldots, \varphi_n \mapsto M_n]))$. We also write ℓ rather than con_ℓ.

Example 5.1. Pushout. The context Δ contains the data of a pushout, i.e., $\Delta \stackrel{\text{def}}{=} (A\,B\,C : \mathsf{U})(f : \mathsf{El}\,(C \to A))(g : \mathsf{El}\,(C \to B))$. The pushout has two point constructors, and a path.

- $(\mathsf{inl}, ((a : \mathsf{El}\,A); \cdot; \cdot; []))$
- $(\mathsf{inr}, ((b : \mathsf{El}\,B), \cdot; \cdot; []))$
- $\left(\mathsf{push}, \left((c : \mathsf{El}\,C); \cdot; (i : \mathbb{I}); \begin{bmatrix} i = 0 \mapsto \mathsf{inl}(f\,c, \cdot, \cdot) \\ i = 1 \mapsto \mathsf{inr}(g\,c, \cdot, \cdot) \end{bmatrix}\right)\right)$

Sphere. We can encode the circle and higher spheres, $\mathbb{S}^n$ for $n \geq 1$, as a point and an n-dimensional surface

- $(\mathsf{base}, (\cdot; \cdot; \cdot; []))$
- $\left(\mathsf{surface}, \left(\cdot; \cdot; \bar{i} : \mathbb{I}^n; \begin{bmatrix} i_1 = 0 \vee i_1 = 1 \mapsto \mathsf{base}(\cdot; \cdot; \cdot) \\ \ldots \\ i_n = 0 \vee i_n = 1 \mapsto \mathsf{base}(\cdot; \cdot; \cdot) \end{bmatrix}\right)\right)$

Note that this defines $\mathbb{S}^n$ for each external n. A type of n-spheres internally indexed over $n : \mathbb{N}$ can be defined using suspensions [50].

Propositional and Higher Truncation. Let $\Delta \stackrel{\text{def}}{=} (A : \mathsf{U})$, and write propositional truncation as $\|A\|_0$.

- $(\mathsf{in}, (\mathsf{El}\,A; \cdot; \cdot; []))$
- $\left(\mathsf{squash}, \left(\cdot; (a_0, a_1 : \|A\|_0); (i : \mathbb{I}); \begin{bmatrix} i = 0 \mapsto a_0 \\ i = 1 \mapsto a_1 \end{bmatrix}\right)\right)$

For higher truncations $\|A\|_n$, where $n \geq 0$, we use the hub and spoke construction [50, Sect. 7.3], as the schema does not allow quantifying over paths of the HIT itself.[2] Instead of squash then we have the following two constructors:

- $(\mathsf{hub}, (\cdot; (f : \mathbb{S}^{n+1} \to \|A\|_n); \cdot; []))$
- $(\mathsf{spoke}, ((x : \mathbb{S}^{n+1}); (f : \mathbb{S}^{n+1} \to \|A\|_n); (i : \mathbb{I}); e))$

where $e \stackrel{\text{def}}{=} [i = 0 \mapsto f\,x, i = 1 \mapsto \mathsf{hub}(\cdot, f, \cdot)]$.

Finite Powerset. The finite powerset $\mathsf{P_f}(A)$ is defined in context $\Delta \stackrel{\text{def}}{=} (A : \mathsf{U})$ using the following constructors

- $(\emptyset, (\cdot; \cdot; \cdot; []))$
- $(\{-\}, (a : \mathsf{El}\,A; \cdot; \cdot; []))$
- $(\cup, (\cdot; X : \mathsf{P_f}(A), Y : \mathsf{P_f}(A), ;))$
- $\left(\mathsf{nl}, \left(\cdot, (X : \mathsf{P_f}(A)), (i : \mathbb{I}), \begin{bmatrix} i = 0 \mapsto \emptyset \cup X \\ i = 1 \mapsto X \end{bmatrix}\right)\right)$
- $\left(\mathsf{as}, \left(\cdot, (X, Y, Z : \mathsf{P_f}(A)), (i : \mathbb{I}), \begin{bmatrix} i = 0 \mapsto (X \cup Y) \cup Z \\ i = 1 \mapsto X \cup (Y \cup Z) \end{bmatrix}\right)\right)$
- $\left(\mathsf{comm}, \left(\cdot, (X, Y : \mathsf{P_f}(A)), (i : \mathbb{I}), \begin{bmatrix} i = 0 \mapsto X \cup Y \\ i = 1 \mapsto Y \cup X \end{bmatrix}\right)\right)$
- $\left(\mathsf{idem}, \left(\cdot, (X : \mathsf{P_f}(A)), (i : \mathbb{I}), \begin{bmatrix} i = 0 \mapsto X \cup X \\ i = 1 \mapsto X \end{bmatrix}\right)\right)$
- $(\mathsf{hub}, (\cdot; (f : \mathbb{S}^1 \to \mathsf{P_f}(A)); \cdot; []))$
- $(\mathsf{spoke}, ((s : \mathbb{S}^1); (f : \mathbb{S}^1 \to \mathsf{P_f}(A)); (i : \mathbb{I}); e))$

where $e \stackrel{\text{def}}{=} [i = 0 \mapsto f\,x, i = 1 \mapsto \mathsf{hub}(\cdot, f, \cdot)]$.

5.2 Induction under clocks

We now present the principle of induction under clocks. This is an elimination principle defining elements of dependent families of the form $\Gamma, h : \forall\overline{\kappa}.\mathsf{H}\,(\delta[\overline{\kappa}]) \vdash D$ type for a vector of clock variables $\overline{\kappa}$ by defining its action on elements of the form $h = \lambda\overline{\kappa}.\mathsf{con}_\ell(\overline{t}, \overline{a}, \overline{r})$ in such a way that boundary conditions are respected. In the case of an empty list of clock variables, this principle specialises to the elimination principle for HITs of Cavallo and Harper [14], which we refer to as the *usual elimination principle for* H.

Figure 5 presents the rule along with judgemental equalities. The figure first defines the judgement $\Gamma \vdash \mathscr{E} : \mathscr{K} \multimap^{\delta} D$ for an *elimination list* $\mathscr{E}$, which contains the premises necessary to handle the constructors in $\mathscr{K}$. The case for $\mathscr{K}, (\ell, (\Gamma, \overline{\Xi}, \Psi, \varphi, e))$ requires an elimination list of the form $\mathscr{E}, u$ where $\mathscr{E}$ is an elimination list for $\mathscr{K}$ and u is an element of the type family D at an index built with con_ℓ. In particular u is typed in a context extended with non-recursive arguments γ, recursive arguments $\overline{x}$, interval variables $\overline{i}$ for the constructor $\mathsf{con}_\ell(\gamma, \overline{x}, \overline{i})$, and also induction hypotheses $\overline{y}$ about the variables $\overline{x}$. The element u also needs to suitably match the boundary e whenever φ is satisfied. To express this

[2] We believe the semantics would support doing so, but it would complicate the schema.

Elimination lists

$$\frac{\Gamma_0 \vdash \delta : \forall\overline{\kappa}.\Delta \qquad \Gamma_0, h : \forall\overline{\kappa}.\mathsf{H}\,(\delta[\overline{\kappa}]) \vdash D\ \mathsf{type}}{\Gamma_0 \vdash \cdot : \cdot \rightharpoonup^{\delta} D}$$

$$\frac{\Gamma_0 \vdash \mathscr{E} : \mathscr{K} \rightharpoonup^{\delta} D \qquad \Gamma_0, \gamma : \forall\overline{\kappa}.\Gamma_i[\delta[\overline{\kappa}]], \overline{x} : \overline{(\forall\overline{\kappa}.\Xi_i[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]] \to \mathsf{H}\,(\delta[\overline{\kappa}]))}, \overline{y} : \overline{\Pi(\xi : \forall\overline{\kappa}.\Xi_i[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]]).D[\lambda\overline{\kappa}.x[\overline{\kappa}]\xi[\overline{\kappa}]]},\ \overline{i} : \Psi \vdash u(\gamma, x, y, \overline{i}) : D[\lambda\overline{\kappa}.\mathsf{con}_\ell(\gamma[\overline{\kappa}], \overline{x[\overline{\kappa}], \overline{i}})][\varphi \mapsto (\!|e|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y}}]}{\Gamma_0 \vdash \mathscr{E}, u : \mathscr{K}, (\ell, (\Gamma, \overline{\Xi}, \Psi, \varphi, e)) \rightharpoonup^{\delta} D}$$

Boundary interpretation

$$\begin{aligned}
(\!|e|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y}} &= (\!|e|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\cdot}\\
(\!|[\varphi_0\ M_0, \ldots, \varphi_n\ M_n]|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}} &= [\varphi_0\ (\!|M_0|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}}, \ldots, \varphi_n\ (\!|M_n|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}}]\\
(\!|x_j\ \overline{u}|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}} &= y_j\ \lambda\overline{\kappa}.\overline{u}[\delta[\overline{\kappa}], \gamma[\overline{\kappa}], \overline{i}, \hat{\gamma}[\overline{\kappa}]]\\
(\!|\mathsf{con}_\ell(\overline{t_\ell}, \overline{\lambda\xi.\,M}, \overline{r_\ell})|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}} &= \mathscr{E}_\ell[\lambda\overline{\kappa}.\overline{t_\ell}[\delta[\overline{\kappa}], \gamma[\overline{\kappa}], \overline{i}, \hat{\gamma}[\overline{\kappa}]], \overline{S}, \overline{R}, \overline{r_\ell}]\\
&\quad (S_i = \lambda\overline{\kappa}.\lambda\overline{\xi_i}.\,M[\delta[\overline{\kappa}], \gamma[\overline{\kappa}], \overline{x}[\overline{\kappa}], \overline{i}, \hat{\gamma}[\overline{\kappa}], \overline{\xi_i}], \quad R_i = \lambda\xi_i.(\!|M_i|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},(\hat{\gamma},\xi_i)})\\
(\!|\mathsf{hcomp}^j\ [\psi \mapsto M]\ M_0|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}} &= \mathsf{comp}^j_{D[v\,j]}\ [\psi \mapsto (\!|M|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},(\hat{\gamma},j)}]\ (\!|M_0|\!)^{\delta}_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{\gamma}}\\
&\quad (v = \mathsf{hfill}^j_{\forall\overline{\kappa}.\,\mathsf{H}\,(\delta[\overline{\kappa}])}[\psi \mapsto \lambda\overline{\kappa}.\,M[\delta[\overline{\kappa}], \gamma[\overline{\kappa}], \overline{x}[\overline{\kappa}], \overline{i}, \hat{\gamma}[\overline{\kappa}], j]]\ (\lambda\overline{\kappa}.\,M_0[\delta[\overline{\kappa}], \gamma[\overline{\kappa}], \overline{x}[\overline{\kappa}], \overline{i}, \hat{\gamma}[\overline{\kappa}]]))
\end{aligned}$$

Induction under clocks

$$\frac{\Gamma \vdash \delta : \forall\overline{\kappa}.\Delta \quad \Gamma, x : \forall\kappa.\mathsf{H}\,(\delta[\overline{\kappa}]) \vdash D\ \mathsf{type} \qquad \Gamma \vdash \mathscr{E} : \mathscr{K} \rightharpoonup^{\delta} D \qquad \Gamma \vdash u : \forall\overline{\kappa}.\mathsf{H}\,(\delta[\overline{\kappa}])}{\Gamma \vdash (H) - \mathsf{elim}_D(\mathscr{E}, u) : D[u]}$$

Judgemental equalities

$$\begin{aligned}
(H) - \mathsf{elim}_D(\mathscr{E}, \lambda\overline{\kappa}.\mathsf{con}_\ell(\overline{t}, \overline{a}, \overline{r})) &\equiv \mathscr{E}_\ell[\overline{\lambda\overline{\kappa}.t}, \overline{\lambda\overline{\kappa}.a}, \overline{\lambda\xi.\,(H) - \mathsf{elim}_D(\mathscr{E}, \lambda\overline{\kappa}.a(\xi[\overline{\kappa}]))}, \overline{r}]\\
(H) - \mathsf{elim}_D(\mathscr{E}, \lambda\overline{\kappa}.\mathsf{hcomp}[\varphi \mapsto u]\ u_0) &\equiv \mathsf{comp}^i_{D[v\,i]}[\varphi \mapsto (H) - \mathsf{elim}_D(\mathscr{E}, \lambda\overline{\kappa}.u)]\ (H) - \mathsf{elim}_D(\mathscr{E}, \lambda\overline{\kappa}.u_0)\\
&\quad (v = \mathsf{hfill}^i_{\forall\overline{\kappa}.\,\mathsf{H}\,(\delta[\overline{\kappa}])}[\varphi \mapsto \lambda\overline{\kappa}.\,u]\ (\lambda\overline{\kappa}.\,u_0))
\end{aligned}$$

Figure 5. The principle of induction under clocks. In the definition of the boundary interpretation the notation $\mathscr{E}_\ell$ refers to the component of $\mathscr{E}$ for the label ℓ. Also the environment $\hat{\gamma}$ contains the extra assumptions introduced in the con and hcomp cases: in the case of interval variables $i[\overline{\kappa}]$ stands for just i.

we transform $e : \mathsf{H}\,\delta$ into $(\!|e|\!)^{\delta}_{\mathscr{E},\overline{y}}$, the corresponding element of the family D, built using the elimination list $\mathscr{E}$ to handle the previous constructors, and the induction hypotheses $\overline{y}$ to handle the recursive arguments $\overline{x}$. This transformation satisfies a typing principle that we omit for space reasons.

In the following examples we will see how instantiating induction under clocks with 1 or 0 clocks respectively induces equivalences which prove that many HITs commute with clock abstraction. In particular, these examples verify the results used in section 4.

Example 5.2. Spheres. In this case induction under clocks unfolds to the principle

$$\frac{\Gamma, x : \forall\kappa.\mathbb{S}^n \vdash D\ \mathsf{type} \quad \Gamma \vdash u_b : D[\lambda\kappa.\mathsf{base}] \quad \Gamma \vdash t : \forall\kappa.\mathbb{S}^n \qquad \Gamma, \overline{i} : \mathbb{I}^n \vdash u_s : D[\lambda\kappa.\mathsf{surface}(i)][\bigvee_{0\le k<n}(i_k = 0 \vee i_k = 1) \mapsto u_b]}{\Gamma \vdash (\mathbb{S}^n) - \mathsf{elim}_D(u_b, u_s, t) : D[t]}$$

Using this, we define $\alpha : \forall\kappa.\mathbb{S}^n \to \mathbb{S}^n$ as follows

$$\alpha(t) \stackrel{\mathsf{def}}{=} (\mathbb{S}^n) - \mathsf{elim}_{\mathbb{S}^n}(\mathsf{base}, \lambda\overline{i}.\,\mathsf{surface}(\overline{i}), t)$$

We must verify the boundary condition which states that if a component of $\overline{i}$ is an endpoint then $u_s(\overline{i}) \equiv u_b$. This follows from the boundary condition for surface.

Since $s \equiv (\mathsf{const}\,s)\ [\kappa_0]$ for all $s : \mathbb{S}^n$, if α as defined above is a right inverse to const we will have produced an equivalence $\mathbb{S}^n \to \forall\kappa.\,\mathbb{S}^n$ and shown the sphere to be clock irrelevant. We can achieve this with another application of induction under a clock, inhabiting the type $s = \mathsf{const}(\alpha(s))$:

$$\Gamma \vdash \mathsf{refl}\ : \lambda\kappa.\,\mathsf{base} = \mathsf{const}(\mathsf{base})$$

$$\Gamma, \overline{i} : \mathbb{I}^n \vdash \mathsf{refl}\ : \lambda\kappa.\,\mathsf{surface}(\overline{i}) = \mathsf{const}(\mathsf{surface}(\overline{i}))$$

The boundary condition in this case states that the second case reduces to the first when $\overline{i}$ contains an endpoint. This follows from congruence of refl .

Propositional and Higher Truncation. To show that propositional truncation commutes with clock quantification we first define a map $\alpha : \forall\kappa.\, \|A\| \to \|\forall\kappa.\, A\|$ by induction under clocks as follows:

$$\Gamma, a : \forall\kappa.\, A \vdash \mathsf{in}(a) : \|\forall\kappa.\, A\|$$
$$\Gamma, x_0, x_1 : \forall\kappa.\, \|A\|, y_0, y_1 : \|\forall\kappa.\, A\|, i : \mathbb{I} \vdash$$
$$\mathsf{squash}(y_0, y_1, i) : \|\forall\kappa.\, A\|\,[(i = 0) \mapsto y_0, (i = 1) \mapsto y_1]$$

The above data defines a map α satisfying

$$\alpha(\lambda\kappa.\mathsf{in}(a\ [\kappa])) = \mathsf{in}(a)$$
$$\alpha(\lambda\kappa.\,\mathsf{squash}(x_0\ [\kappa], x_1\ [\kappa], r) = \mathsf{squash}(\alpha(x_0), \alpha(x_1), r)$$

Let $\beta : \|\forall\kappa.\, A\| \to \forall\kappa.\, \|A\|$ be the canonical map. To show that $\beta \circ \alpha = \mathsf{id}$, it suffices by induction under clocks to show

$$\beta(\alpha(\lambda\kappa.\mathsf{in}(a\ [\kappa]))) = \lambda\kappa.\mathsf{in}(a\ [\kappa])$$
$$\beta(\alpha(\lambda\kappa.\,\mathsf{squash}(x_0[\kappa], x_1[\kappa], r)) = \lambda\kappa.\mathsf{squash}(x_0[\kappa], x_1[\kappa], r)$$

In the latter case the left hand side reduces to

$$\lambda\kappa.\mathsf{squash}(\beta(\alpha(x_0))\ [\kappa], \beta(\alpha(x_1))\ [\kappa], r)$$

and so the case follows by induction. Note that in both these uses of induction under clocks, the boundary condition is satisfied. For example, in the latter case, when $r = 0$ the term reduces to the proof of $\beta(\alpha(x_0)) = x_0$ given by the induction hypothesis. Showing that $\alpha \circ \beta = \mathsf{id}$ follows from an application of the usual elimination principle.

For the higher truncations we apply induction under clocks again, starting by observing that we have terms as follows:

$$a : \forall\kappa.A \vdash \mathsf{in}(a) : \|\forall\kappa.\, A\|_n$$
$$x, y \vdash \mathsf{hub}(y \circ \mathsf{const}) : \|\forall\kappa.A\|_n$$
$$s : \forall\kappa.\, \mathbb{S}^{n+1}, x, y, i : \mathbb{I} \vdash \mathsf{spoke}(s\ [\kappa_0], y \circ \mathsf{const}, i) : \|\forall\kappa.A\|_n$$

where $x : \forall\kappa.\, (\mathbb{S}^{n+1} \to \|A\|_n)$ and $y : (\forall\kappa.\mathbb{S}^{n+1}) \to \|\forall\kappa.A\|_n$. For us to apply the principle, the third term would need to reduce the second when $i = 1$, which it does, and $y(s)$ when $i = 0$. When $i = 0$ we have instead $\mathsf{spoke}(s\ [\kappa_0], y \circ \mathsf{const}, i) = y(\lambda\kappa.\, s\ [\kappa_0])$. From the sphere example we know that $\mathbb{S}^{n+1}$ is clock irrelevant, and hence we have a path $p : \lambda\kappa.\, s\ [\kappa_0] = s$. This means that we can obtain the term needed for the induction as

$$\mathsf{hcomp}^j \begin{bmatrix} (i = 0) \mapsto y\ (p\ j) \\ (i = 1) \mapsto \mathsf{hub}(y \circ \mathsf{const}) \end{bmatrix} \mathsf{spoke}(s\ [\kappa_0], y \circ \mathsf{const}, i).$$

We omit the details of showing that this map is inverse to the canonical one.

Pushout. Using constructions similar to the examples above, one can prove that pushouts commute with $\forall\kappa$ using induction under clocks. More precisely, let $A, B, C : \forall\kappa.\mathsf{U}$, $f : \forall\kappa.C\ [\kappa] \to A\ [\kappa]$, and $g : \forall\kappa.C\ [\kappa] \to B\ [\kappa]$. Then the canonical map

$$(\forall\kappa.\, A) \sqcup_{(\forall\kappa.\, C)} (\forall\kappa.\, B) \to \forall\kappa.((A\ [\kappa]) \sqcup_{(C\ [\kappa])} (B\ [\kappa]))$$

is an equivalence.

Finite Powerset. A full account of the constructors for the finite powerset is not enlightening, so we instead exemplify the induction principle in the cases for singleton, union, and idempotence constructors:

$$\Gamma, x : \forall\kappa.A\ [\kappa] \vdash u_{\{-\}} : D[\lambda\kappa.\{x\ [\kappa]\}]$$
$$\Gamma, x, x' : \forall\kappa.\mathsf{P_f}(A\ [\kappa]), y : D(x), y' : D(x') \vdash$$
$$u_\cup(x, x', y, y') : D(\lambda\kappa.x\ [\kappa] \cup x'\ [\kappa])$$
$$\Gamma, x : \forall\kappa.\mathsf{P_f}(A\ [\kappa]), y : D(x), i : \mathbb{I} \vdash$$
$$u_{\mathsf{idem}}(x, y, i) : D(\lambda\kappa.\mathsf{idem}(x\ [\kappa], i))\ \begin{bmatrix} (i = 0) \mapsto u_\cup(x, x, y, y) \\ (i = 1) \mapsto y \end{bmatrix}$$

In addition it will contain the hub and spoke constructors derived from $\mathbb{S}^1$, towards obtaining a set type. Finally, this principle shows that there is an equivalence $\mathsf{P_f}(\forall\kappa.A) \simeq \forall\kappa.\mathsf{P_f}(A)$ as required for the encoding of labelled transition systems. The concrete calculations are analogous to those for the pushout or trunctation depending on the constructor.

As a step towards reintroducing clock-irrelevance as an axiom, one would need to show that the collection of clock irrelevant types is closed under HIT formation:

Theorem 5.3. *Let $\delta : \Delta \vdash H\delta$ type be a higher inductive type with constructors*

$$\mathsf{con}_\ell : (\gamma : \Gamma_\ell[\delta])(x : \Xi_\ell[\delta, \gamma] \to H\delta)(r : \Psi_\ell) \to H\delta[\varphi_\ell \mapsto e_\ell]$$

Then $H\delta$ is clock irrelevant if $\Gamma_i[\delta]$ is clock irrelevant for all ℓ.

As a special case $\mathbb{N}$ is clock-irrelevant as needed for the encoding of streams of $\mathbb{N}$.

6 Denotational semantics

Previous work has defined a model of CTT in presheaves over a cube category [21, 44] and a model of CloTT in covariant presheaves over a category of time objects $\mathscr{T}$ [36]. We combine these by considering the category $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$ of covariant presheaves over $\mathscr{C} \times \mathscr{T}$, where $\mathscr{C}$ is the opposite category of the usual choice of cube category. Let U be a Hofmann-Streicher universe [30] in $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$.

Following the approach of Licata et al. [35], we construct the model using the internal type theory of $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$. For this, we must provide an interval object $\mathbb{I}$ and a cofibration object $\mathbb{F}$ satisfying certain axioms. We refer to this data as a *cubical model*, and recall that presheaves over $\mathscr{C} \times \mathscr{D}$ admits a cubical model [24] for any small $\mathscr{D}$, where $\mathbb{I}$ and $\mathbb{F}$ are taken to be the inclusion of those from $\mathsf{PSh}(\mathscr{C})$.

Recall [44] that a *CCHM fibration* (A, α) over a context $\Gamma : \mathsf{U}$ is a family $A : \Gamma \to \mathsf{U}$ equipped with a fibration structure α. The denotational semantics is given by the category with families (CwF) [26] Fib constructed as follows: Contexts are global elements of U, families are global CCHM fibrations, and elements of (A, α) are elements of A.

To model HITs we follow Coquand et al. [22]. The principle of induction under clocks is justified semantically using that

$\forall\kappa.(-)$ can be modelled as an ω^{op}-limit [12] and that the structure maps of this limit cannot change the outermost constructor in HITs, allowing for an inductive proof.

To model a Fitch-style modality like $\triangleright$ we must define a dependent adjunction [19]. A dependent adjunction consists of an endofunctor L and an action on types R such that elements of $L(\Gamma) \vdash A$ type correspond bijectively to elements of $\Gamma \vdash R(A)$ type. Both R and the bijective correspondence are required to be natural in Γ. Mannaa et al. [36] define a dependent right adjoint on the slice of $\mathsf{PSh}(\mathscr{T})$ over an object of clocks in that model, capturing the dependence of $\triangleright$ on a clock. This construction extends to $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$. To extend to Fib we show how to lift CCHM fibration structures from A to $R(A)$ by a general construction.

Definition 6.1. Let $\mathbb{C}$ be a cubical model, and let L be an endofunctor on $\mathbb{C}$ which preserves finite limits. We say that L *preserves the interval* if there is an isomorphism $L(\mathbb{I}) \cong \mathbb{I}$ which preserves endpoints. We say that L *preserves cofibrations* if it maps cofibrations to cofibrations and preserves pullbacks where the vertical maps are cofibrations.

Writing $\Gamma.A$ for the comprehension object for the family A over Γ, and $[\varphi]$ for the family classified by a cofibration φ, the condition of preserving cofibrations can be presented in the internal language as an operation mapping cofibrations $\Gamma \vdash \varphi : \mathbb{F}$ on Γ to cofibrations $\mathsf{L}\Gamma \vdash \mathsf{L}_{\mathbb{F}}(\varphi) : \mathbb{F}$ such that $\mathsf{L}\Gamma.[\mathsf{L}_{\mathbb{F}}(\varphi)] \cong \mathsf{L}(\Gamma.[\varphi])$ as subobjects of $\mathsf{L}\Gamma$ and satisfying $\mathsf{L}_{\mathbb{F}}(\varphi[\sigma]) = \mathsf{L}_{\mathbb{F}}(\varphi)[\mathsf{L}\sigma]$.

Theorem 6.2. *Let $\mathbb{C}$ be a cubical model, let $\mathsf{L} : \mathbb{C} \to \mathbb{C}$ be a functor preserving finite products, the interval and cofibrations, and let R be a dependent right adjoint to L. If a family A over $\mathsf{L}\Gamma$ carries a global composition structure, so does $\mathsf{R}A$ over Γ. Moreover, this assignment is natural in Γ.*

Note that this is a statement about *global* composition structures in the model. The theorem can not be proved in the internal logic of the topos, but can be proved in an extension of this using crisp type theory, similarly to the construction of universes for cubical type theory [35]. The reason is that the proof uses the bijective correspondence of dependent right adjoints which only applies to global terms.

Proof (sketch). A CCHM composition structure on $R(A)$ corresponds to an assignment of any $\sigma : \Delta \times \mathbb{I} \to \Gamma$ to a rule

$$\frac{\Delta \vdash \varphi : \mathbb{F} \qquad \Delta.\mathbb{I}.[\varphi[\mathsf{p}]] \vdash u : (\mathsf{R}A)[\sigma\mathsf{p}] \qquad \Delta \vdash u_e : (\mathsf{R}A)[\sigma \circ (\mathrm{id}, e)] \qquad \Delta.[\varphi] \vdash u_e[\mathsf{p}] = u[(\mathrm{id}_\Delta, e).[\varphi]]}{\Delta \vdash c^{\mathsf{R}A}_\sigma\, \varphi\, u\, u_e : (\mathsf{R}A)[\sigma \circ (\mathrm{id}, 1-e)]}$$

for $e = 0, 1$, which is natural in Δ and satisfies

$$\Delta.[\varphi] \vdash (c_\sigma\, \varphi\, u\, u_e)[\mathsf{p}] = u[(\mathrm{id}, 1-e).[\varphi]]$$

Using $(\mathsf{R}A)[\sigma\mathsf{p}] = \mathsf{R}(A[\mathsf{L}(\sigma\mathsf{p})])$ the assumptions can be transported along the bijective correspondence of dependent right adjoints to give terms

$$\mathsf{L}(\Delta.\mathbb{I}.[\varphi[\mathsf{p}]]) \vdash \overline{u} : A[\mathsf{L}(\sigma\mathsf{p})] \qquad \mathsf{L}\Delta \vdash \overline{u_e} : A[\mathsf{L}(\sigma \circ (\mathrm{id}, e))]$$

Since L preserves cofibrations the domain of $\overline{u}$ is isomorphic to $\mathsf{L}(\Delta).\mathbb{I}.[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])]$, so this data can be transformed into input data for the composition operation on A. The result of this composition can then be transported back along the bijective correspondence of the dependent right adjoint to give the output for the composition operation on RA. □

7 Related work

Guarded Cubical Type Theory [9] combines Cubical Type Theory with single-clocked guarded recursion. While this case is useful for many purposes, it cannot be used to encode coinductive types. Møgelberg and Veltri [41] extend Guarded Cubical Type Theory with ticks as in CloTT. They give a model of this calculus including HITs, and show that bisimilarity coincides with path equality for a large class of guarded recursive types. This paper can be seen as an extension of that with multiple clocks, allowing for these results to be lifted from guarded recursive types to coinductive types. Note that modelling the multiclocked case is much more complex than the single clock case. In particular, equipping $\triangleright$ with a composition structure is much more challenging, because the simple description of the left adjoint in the single clock case allowed for a simple construction. The extended language of ticks giving computation rules for clock irrelevance presented here is also new.

The encoding of coinductive types using guarded recursion was first described by Atkey and McBride [6] in the simply typed setting. Since then a number of dependent type theories have been developed for programming and reasoning with these [18, 39], of which CloTT is the most advanced. Bahr. et al. [7] prove syntactic properties of CloTT including strong normalisation and canonicity, but only for a pure calculus without identity or path types. The model of CloTT constructed by Mannaa et al. [36] considers extensional identity types, but no cubical features. Coinductive types can also be encoded using a combination of guarded recursion and a $\Box$-modality [18]. This approach has not been studied in combination with CTT yet, and also appears to be less flexible, e.g., it does not seem possible to define nested coinductive types.

Sized types [33] is another approach to encoding productivity in types, by annotating (co)inductive types with sizes indicating a bound on the size of the allowed elements. Specifically, sized types reduce both termination and productivity to (well-founded) induction on sizes. They have been extensively studied from the syntactic perspective [1, 2, 46] but are not well understood from the perspective of denotational semantics. Our view is that sized types are closer to working in the models of type theory, like the one provided here, and guarded recursion is a more abstract, principled perspective. This is supported by the model of guarded recursion

in sized types constructed by Veltri and van der Weide [52]. To our knowledge sized types have never been used to solve equations with negative occurrences, which is an important application of guarded recursion.

The coincidence of bisimilarity and path equality for streams as coinductive records has been proved in Cubical Agda [54], and it is likely that the proof can be extended to general M-types. Veltri [51] proves that bisimilarity implies path equality for the final coalgebra for the P_f using sized types in Cubical Agda. This coincidence should therefore be seen as a feature of Cubical Type Theory, rather than guarded recursion. On the other hand, when proving such results guarded recursion is a powerful framework for ensuring productivity of definitions, as illustrated by the examples in this paper.

The final coalgebra for the finite powerset functor can be constructed in set theory as a limit of an $\omega + \omega$-indexed sequence [55]. This construction has been formalised in Cubical Agda by Veltri [51] using the lesser limited principle of omniscience, a weak choice principle. From this perspective it is interesting that our model uses ω-indexed step-indexing only, and therefore LTS as constructed in section 4 is realised in the model as an ω-limit. This construction works because of the formulation of P_f as a HIT and because the ω-chain is constructed *externally*, using judgemental equality. In particular, the counter example constructed in Proposition 5 of [51] uses an ω-chain of elements whose projections are only equal up to path equality.

Multimodal dependent type theory [28] is a general framework for dependent modal type theories parametrised over *mode theories*. By instantiating this appropriately, one can recover e.g., the basic modal framework needed for internalising parametricity in type theory [8, 15], or for the combination of $\triangleright$ and $\Box$ used for guarded recursion by Clouston et al. [18]. Generalising this to multiple clocks seems to require a notion of dependent mode theory, as for example, the modal operator $\triangleright$ depends on the object of clocks.

8 Conclusion and future work

We have presented the type theory CCTT, and shown that the principle of induction under clocks can be used to construct a rich supply of functors for which coinductive types can be encoded using guarded recursion. This allows for simple programming with a wide range of coinductive types, including ones constructed using higher inductive types. We have seen by example how to prove coincidence of path equality with bisimilarity for these types. We believe this type theory is useful not just for coinductive reasoning, but also for reasoning about advanced programming language features using a form of synthetic guarded domain theory [40, 45]. In fact, an earlier version of CCTT has already been used for a semantic proof of applicative simulation being a congruence for a lambda calculus with finite non-determinism [42].

We are currently developing a prototype implementation of CCTT as an extension of Cubical Agda. The current implementation[3] includes clock, ticks, $\triangleright$ (and its composition structure), and fixed points, but not the principle of induction under clocks. We believe this can be simulated using Agda's rewrite features [20], but should in the long run be build into Agda's pattern matching. The proof of Theorem 4.3 has been verified in this.

We would also like to prove canonicity for CCTT, building on similar results for Cubical Type Theory [23, 32, 48] and Clocked Type Theory [7]. For Cubical Type Theory canonicity is proved for terms in a context of only interval variables. Bahr. et al. [7] prove that in Clocked Type Theory, terms of type $\mathbb{N}$ or Bool (the only inductive types considered in that paper) in contexts with free clock variables, but no other assumptions, reduce to introduction forms. Generalising this result to general HITs and contexts also containing interval variables should allow for terms of type $\forall\overline{\kappa}.H(\delta\ [\overline{\kappa}])$ to reduce to a form in which the β-rule for induction under clocks can be applied. On the other hand, we believe that it should not be necessary to include free tick variables, and so the only tick that needs to be considered in reductions is $\diamond$. Our equational rules have been designed with this in mind.

Unlike this paper, other type theories for multi-clocked guarded recursion [11, 36] take clock irrelevance (10) as an axiom. This requires that universes be indexed by clock contexts, and the $\triangleright$ modality be restricted to $\triangleright^\kappa : \mathsf{U}_\Delta \to \mathsf{U}_\Delta$ for $\kappa \in \Delta$, because an unrestricted $\triangleright^\kappa$ breaks clock irrelevance [12]. Bizjak and Møgelberg [12] show how to construct such universes of types that are clock-irrelevant in the sense of the map $A \to \forall\kappa.A$ being an isomorphism, rather than an equivalence. Future work includes constructing larger universes of types clock-irrelevant in the more liberal sense used in this paper.

Acknowledgment

This work was supported by a research grant (13156) from VILLUM FONDEN.

References

[1] Andreas Abel and Brigitte Pientka. 2013. Wellfounded Recursion with Copatterns: A Unified Approach to Termination and Productivity. In *Proceedings ICFP 2013*. ACM, 185–196.

[2] Andreas Abel, Andrea Vezzosi, and Theo Winterhalter. 2017. Normalization by evaluation for sized dependent types. *Proceedings of the ACM on Programming Languages* 1, ICFP (2017), 1–30.

[3] Benedikt Ahrens, Paolo Capriotti, and Régis Spadotti. 2015. Non-Wellfounded Trees in Homotopy Type Theory. In *13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 38)*, Thorsten Altenkirch (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 17–30. https://doi.org/10.4230/LIPIcs.TLCA.2015.17

[3] https://github.com/agda/guarded/tree/forcing-ticks

[4] Andrew W. Appel and David McAllester. 2001. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst* 23, 5 (2001), 657–683.

[5] Andrew W. Appel, Paul-André Melliès, Christopher D. Richards, and Jérôme Vouillon. 2007. A very modal model of a modern, major, general type system. In *POPL*. 109–122.

[6] Robert Atkey and Conor McBride. 2013. Productive coprogramming with guarded recursion. *ACM SIGPLAN Notices* 48, 9 (2013), 197–208.

[7] Patrick Bahr., Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. 2017. The clocks are ticking: No more delays!. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 1–12.

[8] Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. 2015. A presheaf model of parametric type theory. *Electronic Notes in Theoretical Computer Science* 319 (2015), 67–82.

[9] Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. 2019. Guarded Cubical Type Theory: Path Equality for Guarded Recursion. *Journal of Automated Reasoning* 63, 2 (2019), 211–253.

[10] Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. 2012. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science* 8, 4 (2012).

[11] Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus E Møgelberg, and Lars Birkedal. 2016. Guarded dependent type theory with coinductive types. In *International Conference on Foundations of Software Science and Computation Structures*. Springer, 20–35.

[12] Aleš Bizjak and Rasmus Ejlers Møgelberg. 2020. Denotational semantics for guarded dependent type theory. *Mathematical Structures in Computer Science* 30, 4 (2020), 342–378.

[13] Venanzio Capretta. 2005. General recursion via coinductive types. *Logical Methods in Computer Science* 1 (2005).

[14] Evan Cavallo and Robert Harper. 2019. Higher inductive types in cubical computational type theory. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–27.

[15] Evan Cavallo and Robert Harper. 2020. Internal Parametricity for Cubical Type Theory. In *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[16] James Chapman, Tarmo Uustalu, and Niccolò Veltri. 2019. Quotienting the delay monad by weak bisimilarity. *Mathematical Structures in Computer Science* 29, 1 (2019), 67–92.

[17] Ranald Clouston. 2018. Fitch-style modal lambda calculi. In *International Conference on Foundations of Software Science and Computation Structures*. Springer, 258–275.

[18] Ranald Clouston, Aleš Bizjak, Hans Bugge Grathwohl, and Lars Birkedal. 2015. Programming and Reasoning with Guarded Recursion for Coinductive Types. In *Proceedings of FoSSaCS*. 407–421.

[19] Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M. Pitts, and Bas Spitters. 2020. Modal Dependent Type Theory and Dependent Right Adjoints. *Mathematical Structures in Computer Science* 30, 2 (2020), 118–138.

[20] Jesper Cockx. 2020. Type Theory Unchained: Extending Agda with User-Defined Rewrite Rules. In *25th International Conference on Types for Proofs and Programs (TYPES 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 175)*, Marc Bezem and Assia Mahboubi (Eds.). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2:1–2:27. https://doi.org/10.4230/LIPIcs.TYPES.2019.2

[21] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[22] Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. On Higher Inductive Types in Cubical Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (Oxford, United Kingdom) *(LICS '18)*. Association for Computing Machinery, New York, NY, USA, 255–264. https://doi.org/10.1145/3209108.3209197

[23] Thierry Coquand, Simon Huber, and Christian Sattler. 2019. Homotopy Canonicity for Cubical Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 131)*, Herman Geuvers (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 11:1–11:23. https://doi.org/10.4230/LIPIcs.FSCD.2019.11

[24] Thierry Coquand, Fabian Ruch, and Christian Sattler. 2019. Constructive sheaf models of type theory. *Mathematical Structures in Computer Science* (2019), 1–24.

[25] Nils Anders Danielsson. 2010. Beating the Productivity Checker Using Embedded Languages. In *PAR*, Vol. 43. 29–48.

[26] Peter Dybjer. 1995. Internal type theory. In *International Workshop on Types for Proofs and Programs*. Springer, 120–134.

[27] Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. 2018. Finite sets in homotopy type theory. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8-9, 2018*, June Andronick and Amy P. Felty (Eds.). ACM, 201–214. https://doi.org/10.1145/3167085

[28] Daniel Gratzer, GA Kavvos, Andreas Nuyts, and Lars Birkedal. 2020. Multimodal dependent type theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. 492–506.

[29] Martin Hofmann. 1997. Syntax and semantics of dependent types. In *Extensional Constructs in Intensional Type Theory*. Springer, 13–54.

[30] Martin Hofmann and Thomas Streicher. 1999. Lifting Grothendieck universes. (1999). www.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf Unpublished.

[31] Simon Huber. 2017. A Cubical Type Theory for Higher Inductive Types. (2017). http://www.cse.chalmers.se/~simonhu/misc/hcomp.pdf

[32] Simon Huber. 2019. Canonicity for cubical type theory. *Journal of Automated Reasoning* 63, 2 (2019), 173–210.

[33] J. Hughes, L. Pareto, and A. Sabry. 1996. Proving the Correctness of Reactive Systems Using Sized Types. In *Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996*. 410–423.

[34] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal, and Derek Dreyer. 2018. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *Journal of Functional Programming* 28 (2018).

[35] R. D. Licata, I. Orton, A.M. Pitts, and B. Spitters. 2018. Internal Universes in Models of Homotopy Type Theory. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 108)*, Hélène Kirchner (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 22:1–22:17.

[36] Bassel Mannaa, Rasmus Ejlers Møgelberg, and Niccolò Veltri. 2020. Ticking clocks as dependent right adjoints: Denotational semantics for clocked type theory. *Logical Methods in Computer Science* 16 (2020).

[37] Per Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis, Napoli.

[38] Conor McBride and Ross Paterson. 2008. Applicative programming with effects. *Journal of functional programming* 18, 1 (2008), 1–13.

[39] Rasmus Ejlers Møgelberg. 2014. A type theory for productive coprogramming via guarded recursion. In *CSL-LICS*. 71:1–71:10.

[40] Rasmus E Møgelberg and Marco Paviotti. 2019. Denotational semantics of recursive types in synthetic guarded domain theory. *Mathematical Structures in Computer Science* 29, 3 (2019), 465–510.

[41] Rasmus Ejlers Møgelberg and Niccolò Veltri. 2019. Bisimulation as path type for guarded recursive types. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–29. https://doi.org/10.1145/

3290317

[42] Rasmus Ejlers Møgelberg and Andrea Vezzosi. 2021. Two Guarded Recursive Powerdomains for Applicative Simulation. In *Proceedings 37th Conference on Mathematical Foundations of Programming Semantics, MFPS 2021, Hybrid: Salzburg, Austria and Online, 30th August - 2nd September, 2021 (EPTCS, Vol. 351)*, Ana Sokolova (Ed.). 200–217. https://doi.org/10.4204/EPTCS.351.13

[43] Hiroshi Nakano. 2000. A modality for recursion. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE, 255–266.

[44] I. Orton and A.M. Pitts. 2018. Axioms for Modelling Cubical Type Theory in a Topos. *Logical Methods in Computer Science* Volume 14, Issue 4 (Dec 2018).

[45] Marco Paviotti, Rasmus Ejlers Møgelberg, and Lars Birkedal. 2015. A model of PCF in guarded type theory. *Electronic Notes in Theoretical Computer Science* 319 (2015), 333–349.

[46] Jorge Luis Sacchini. 2013. Type-Based Productivity of Stream Definitions in the Calculus of Constructions. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. 233–242.

[47] Kristina Sojakova. 2015. Higher Inductive Types as Homotopy-Initial Algebras. *SIGPLAN Not.* 50, 1 (Jan. 2015), 31–42. https://doi.org/10.1145/2775051.2676983

[48] Jonathan Sterling and Carlo Angiuli. 2021. Normalization for Cubical Type Theory. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–15. https://doi.org/10.1109/LICS52264.2021.9470719

[49] Jonathan Sterling and Robert Harper. 2018. Guarded Computational Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 879–888.

[50] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study.

[51] Niccolò Veltri. 2021. Type-Theoretic Constructions of the Final Coalgebra of the Finite Powerset Functor. In *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference) (LIPIcs, Vol. 195)*, Naoki Kobayashi (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 22:1–22:18. https://doi.org/10.4230/LIPIcs.FSCD.2021.22

[52] Niccolò Veltri and Niels van der Weide. 2019. Guarded Recursion in Agda via Sized Types. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. 32:1–32:19.

[53] Niccolò Veltri and Andrea Vezzosi. 2020. Formalizing π-Calculus in Guarded Cubical Agda. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs* (New Orleans, LA, USA) *(CPP 2020)*. Association for Computing Machinery, New York, NY, USA, 270–283. https://doi.org/10.1145/3372885.3373814

[54] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2019. Cubical Agda: a dependently typed programming language with univalence and higher inductive types. *Proceedings of the ACM on Programming Languages* 3, ICFP (2019), 1–29.

[55] James Worrell. 2005. On the final sequence of a finitary set functor. *Theor. Comput. Sci.* 338, 1-3 (2005), 184–199. https://doi.org/10.1016/j.tcs.2004.12.009

A Appendix

A.1 Omitted proofs Section 3

Proof of Lemma 3.2. Given $f : \triangleright^\kappa A \to A$, let p be the corresponding proof of (8), then as mentioned the center of contraction for

$$\Sigma(x : A).\mathsf{Path}_A(x, f\ (\lambda(\alpha:\kappa).x))$$

will be the pair $(\mathsf{fix}^\kappa f, p)$. Then for any other such pair (h, p_h) we have to show $(\mathsf{fix}^\kappa f, p) = (h, p_h)$, which is equivalent to

$$\Sigma(q : \mathsf{fix}^\kappa f = h).\ q_* p = p_h$$

We define q by guarded recursion

$$\mathsf{fix}^\kappa\ \lambda(r : \triangleright^\kappa(\mathsf{fix}^\kappa f = h)).(p^{-1}, p_h^{-1})_*(\mathsf{ap}_f(\lambda i.\lambda(\alpha:\kappa).r\ [\alpha]\ i))$$

We then proceed to prove $q_* p = p_h$ by first observing that it is equivalent to

$$(p, p_h)_* q = \mathsf{ap}_f(\lambda i.\lambda(\alpha:\kappa).q\ [\alpha]\ i)$$

so that by expanding q by (8) on the left hand side and canceling the transports we obtain the right hand side and the proof is concluded.

To prove that $\Sigma(x : \triangleright^\kappa A).\triangleright\,(\alpha : \kappa).\mathsf{Path}_A(x\ [\alpha], f(x))$ is contractible, note that if B is contractible, so is $\triangleright^\kappa B$. Applying this to the first part of the lemma, we get that the following equivalent types are all contractible:

$$\begin{aligned}
&\triangleright^\kappa(\Sigma(x : A).\mathsf{Path}_A(x, f\ (\lambda(\beta:\kappa).x))) \\
&\quad \simeq \Sigma(x : \triangleright^\kappa A).\triangleright\,(\alpha:\kappa).\mathsf{Path}_A(x\ [\alpha], f\ (\lambda(\beta:\kappa).(x\ [\alpha]))) \\
&\quad \simeq \Sigma(x : \triangleright^\kappa A).\triangleright\,(\alpha:\kappa).\mathsf{Path}_A(x\ [\alpha], f\ (\lambda(\beta:\kappa).(x\ [\beta]))) \\
&\quad \simeq \Sigma(x : \triangleright^\kappa A).\triangleright\,(\alpha:\kappa).\mathsf{Path}_A(x\ [\alpha], f\ (x))
\end{aligned}$$

where the first equivalence is by (3) and the second by tick irrelevance. □

Proof of Lemma 3.1. We have

$$\Gamma, i_0, \ldots, i_n : \mathbb{I} \vdash u_0 \stackrel{\text{def}}{=} u[0/i_0] : \kappa \rightsquigarrow \Gamma'$$

and

$$\Gamma, i_0, \ldots, i_n : \mathbb{I}, \varphi, j : \mathbb{I} \vdash p(j) \stackrel{\text{def}}{=} \mathsf{tirr}(u_0, u, j) : \kappa \rightsquigarrow \Gamma'$$

Define $\Gamma, i_0, \ldots, i_n : \mathbb{I}, j : \mathbb{I} \vdash C$ type by

$$\mathsf{hfill}^j\ [\varphi \mapsto A[p(j)/\alpha]]\ A[u_0/\alpha]$$

and then take $B \stackrel{\text{def}}{=} C\,1$. Then we define $\mathsf{filler}(t, u)$ as

$$\mathsf{hcomp}^j_C\ [\varphi \mapsto t\ [p(j)]]\ (t\ [u_0])$$

□

Substitutions

$$\frac{\Gamma \vdash}{\Gamma \vdash [] : \cdot} \qquad \frac{\Gamma \vdash \sigma : \Gamma' \qquad \Gamma \vdash t : A\sigma}{\Gamma \vdash (\sigma, t) : \Gamma', x : A}$$

$$\frac{\Gamma \vdash \sigma : \Gamma' \qquad \Gamma \vdash \kappa' : \mathsf{clock}}{\Gamma \vdash (\sigma, \kappa') : \Gamma', \kappa : \mathsf{clock}} \qquad \frac{\Gamma \vdash \sigma : \Gamma' \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash (\sigma, r) : \Gamma', i : \mathbb{I}}$$

$$\frac{\Gamma \vdash \sigma : \Gamma' \qquad \Gamma' \vdash \varphi : \mathbb{F} \qquad \Gamma \vdash \varphi\sigma = 1_{\mathbb{F}} : \mathbb{F}}{\Gamma \vdash \sigma : \Gamma', \varphi}$$

$$\frac{\Gamma_0 \vdash \sigma : \Gamma' \qquad \Gamma \vdash u : \kappa\sigma \rightsquigarrow \Gamma_0}{\Gamma \vdash (\sigma, u) : \Gamma', \alpha : \kappa}$$

$$\frac{\Gamma_0 \vdash \sigma : \Gamma' \qquad \Gamma \vdash (\kappa', v) \rightsquigarrow \Gamma_0}{\Gamma \vdash \sigma, (\kappa', v) : \Gamma', \kappa : \mathsf{clock}, \alpha : \kappa}$$

Figure 6. Formation rules for substitutions.

A.2 Substitution for Tick Application

In figure 6 we present the formation rules for substitutions, based on the ones from Mannaa et al. [36], and extended to account for the new tick judgemnts and the contexts from cubical type theory. In what follows we explain how to apply a substitution to a tick application term.

Operation 1. *Given* $\Delta \vdash \sigma : \Gamma$ *and* $\Gamma \vdash u : \kappa \rightsquigarrow \Gamma'$ *we define an operation* $\mathsf{residual}(\sigma, u)$ *returning tuples of one of two forms*

- (Δ', σ') *such that* $\Delta \vdash u\sigma : \kappa\sigma \rightsquigarrow \Delta'$ *and* $\Delta' \vdash \sigma' : \Gamma'$, *and* $\Delta \vdash \sigma|_{\Gamma'} \equiv \sigma' : \Gamma'$.
- $(\Delta', \kappa'', \sigma')$ *such that* $\Delta \vdash (\kappa\sigma, u\sigma) \rightsquigarrow \Delta'$ *and*

$$\Delta', \kappa'' : \mathsf{clock} \vdash \sigma' : \Gamma',$$

such that $\kappa\sigma' = \kappa''$ *and* $\Delta \vdash \sigma|_{\Gamma'} \equiv \sigma'[\kappa\sigma/\kappa''] : \Gamma'$.

Construction. The tick u contains tick variables $\alpha_0 : \kappa \ldots \alpha_n : \kappa$, here given in the order they appear in Γ, so in particular we have $\Gamma = \Gamma_1, \alpha_0 : \kappa, \Gamma_2$ and $\Gamma' \sqsubseteq \Gamma_1, \mathsf{TL}(\Gamma_2)$. Then let us look at the restriction of σ to $\Gamma_1, \alpha_0 : \kappa$. We have two cases:

(i) (σ_1, v) with $\Delta' \vdash \sigma_1 : \Gamma_1$ and $\Delta_0 \vdash v : \kappa\sigma_1 \rightsquigarrow \Delta'$ with $\Delta_0 \sqsubseteq \Delta$.
(ii) $(\sigma_0, (\kappa', v))$ with $\Gamma_1 = \Gamma_0, \kappa : \mathsf{clock}$, and $\Delta' \vdash \sigma_0 : \Gamma_0$, and $\Delta_0 \vdash (\kappa', v) \rightsquigarrow \Delta'$, with $\Delta_0 \sqsubseteq \Delta$.

In either case the tick v lives in the context $\Delta_0 \sqsubseteq \Delta$ because other components of the substitution σ, e.g. for $\alpha_1 \ldots \alpha_n$, might have shrunk the context so.

In case (i) we extend σ_1 to $\Delta' \vdash \sigma_1' : \Gamma_1, \mathsf{TL}(\Gamma_2)$ because whenever we have one of $\Delta \vdash \kappa_i : \mathsf{clock}$, or $\Delta \vdash r : \mathbb{I}$, or $\Delta \vdash \varphi \equiv 1_{\mathbb{F}} : \mathbb{F}$ we also have the same in $\Delta' \sqsubseteq \Delta$. Finally we take σ' to be $\sigma_1'|_{\Gamma'}$, which agrees with $\sigma|_{\Gamma'}$ by construction. The constructed tuple will be (Δ', σ').

In case (ii) we extend σ_0 first to $\Delta', \kappa'' \vdash (\sigma_0, \kappa'') : \Gamma_0, \kappa : \mathsf{clock}$ then to a substitution $\Delta', \kappa'' \vdash \sigma_0' : \Gamma_1, \mathsf{TL}(\Gamma_2)$ as above, furthermore noting that $\mathsf{TL}(\Gamma_2)$ does not depend on κ. Finally we take σ' to be $\sigma_0'|_{\Gamma'}$. The substitution $\sigma'[\kappa'/\kappa'']$ then agrees with $\sigma|_{\Gamma'}$ by construction, and we also have $\kappa\sigma' \equiv \kappa''$. The constructed tuple will be $(\Delta', \kappa'', \sigma')$.

To show that we have the correct typing for $u\sigma$ we observe that Δ' is smaller as a subcontext of Δ than the ones the ticks $\alpha_1\sigma \dots \alpha_n\sigma$ target, so they can all be weakened to fit the typing $\Delta \vdash \alpha_i\sigma : \kappa\sigma_1 \rightsquigarrow \Delta'$. In case (i) then we are done by extending this observation to v. In case (ii) we can further derive $\Delta \vdash (\kappa\sigma_1, \alpha_i\sigma) \rightsquigarrow \Delta'$ which gives us what we want. □

For $\Delta \vdash \sigma : \Gamma$, and $\Gamma' \vdash t : \triangleright\,(\alpha : \kappa).\, A$, and $\Gamma \vdash u : \kappa \rightsquigarrow \Gamma'$, we have that substitution commutes with tick application in the sense that $(t\,{}_{\Gamma'}[u])\sigma$ equals

$$\begin{cases} t\sigma'\,{}_{\Delta'}[u\sigma] & \text{if } \mathsf{residual}(\sigma, u) = (\Delta', \sigma') \\ (\kappa''.\, t\sigma')\,{}_{\Delta'}[(\kappa\sigma, u\sigma)] & \text{if } \mathsf{residual}(\sigma, u) = (\Delta', \kappa'', \sigma') \end{cases}$$

We want to show that typing is preserved. For the first case, we can assume $\Delta' \vdash t\sigma' : \triangleright\,(\alpha : \kappa\sigma').A(\sigma', \alpha)$, so by the tick application rule we have $\Delta \vdash t\sigma'\,{}_{\Delta'}[u\sigma] : A(\sigma', \alpha)[u\sigma/\alpha]$, where the latter type is equal to $A[u/\alpha]\sigma$ as expected. For the second case, we can assume $\Delta', \kappa'' \vdash t\sigma' : \triangleright\,(\alpha : \kappa\sigma').A(\sigma', \alpha)$, so by the forcing tick application rule we have

$$\Delta \vdash (\kappa''.\, t\sigma')\,{}_{\Delta'}[(\kappa\sigma, u\sigma)] : A(\sigma', \alpha)[u\sigma : \kappa\sigma/\alpha : \kappa''],$$

where the latter type is equal to $A[u/\alpha]\sigma$ as expected.

Operation 2. *Given* $\Delta \vdash \sigma : \Gamma$ *and* $\Gamma \vdash (\kappa, u) \rightsquigarrow \Gamma'$ *we define an operation* $\mathsf{residual}(\sigma, (\kappa, u))$ *returning tuples of the form*

- (Δ', σ') *such that* $\Delta \vdash (\kappa\sigma, u\sigma) \rightsquigarrow \Delta'$ *and* $\Delta' \vdash \sigma' : \Gamma'$, *and* $\Delta \vdash \sigma|_{\Gamma'} \equiv \sigma' : \Gamma'$

Construction. Here the tick u might not contain any tick variables, in which case we take Δ' to be Δ and σ' to be $\sigma|_{\Gamma'}$. If u does contain tick variables then we have cases (i) and (ii) as in the construction of $\mathsf{residual}(-,-)$. We chose to use κ in the assumption $\Gamma \vdash (\kappa, u) \rightsquigarrow \Gamma'$ so that the names in the cases would line up with the previous construction.

In case (i) we construct the tuple (Δ', σ') as we did before, and the same reasoning extends to the well-typing of $\Delta \vdash (\kappa\sigma, u\sigma) \rightsquigarrow \Delta'$.

In case (ii) let us recall that here Γ is of the form $\Gamma_0, \kappa : \mathsf{clock}, \alpha_0 : \kappa, \Gamma_2$, with $\Gamma' \sqsubseteq \Gamma_0, \kappa : \mathsf{clock}, \mathsf{TL}(\Gamma_2)$. We also have $\Delta' \vdash (\sigma_0, \kappa') : \Gamma_0, \kappa : \mathsf{clock}$, which agrees with σ restricted to the same context. As before we can extend (σ_0, κ') to a substitution for $\Gamma_0, \kappa : \mathsf{clock}, \mathsf{TL}(\Gamma_2)$ by using the relevant components of σ, and finally obtain the desired σ' by restriction to Γ'. □

For $\Delta \vdash \sigma : \Gamma$, and $\Gamma', \kappa : \mathsf{clock} \vdash t : \triangleright\,(\alpha : \kappa).A$, and $\Gamma \vdash (\kappa', u) \rightsquigarrow \Gamma'$, we have that substitution commutes with forcing tick application in the following way:

$$((\kappa.t)\,{}_{\Gamma'}[(\kappa', u)])\sigma = (\kappa.\, t(\sigma', \kappa))\,{}_{\Delta'}[(\kappa'\sigma, u\sigma)]$$

where $\mathsf{residual}(\sigma, (\kappa', u)) = (\Delta', \sigma')$. We want to show that typing is preserved. We can assume

$$\Delta', \kappa : \mathsf{clock} \vdash t(\sigma', \kappa) : \triangleright\,(\alpha : \kappa).\, A(\sigma', \kappa, \alpha),$$

then by the forcing tick application rule we have

$$\Delta \vdash (\kappa.\, t(\sigma', \kappa))\,{}_{\Delta'}[(\kappa'\sigma, u\sigma)] : A(\sigma', \kappa, \alpha)[u\sigma : \kappa'\sigma/\alpha : \kappa],$$

where the latter type is equal to $A[u : \kappa'/\alpha : \kappa]\sigma$ as expected.

A.3 Omitted proofs Section 4

Proof of Lemma 4.2. The case of composition is clear, and products follow from the fact that $\forall\kappa.(A \times B) \simeq (\forall\kappa.A) \times (\forall\kappa.B)$. Likewise, the case of Π-types follows from the fact that $\forall\kappa.\Pi(a : A).B \simeq \Pi(a : A).\forall\kappa.B$ which can be proved by commuting two arguments. In the case of Σ, since $\forall\kappa.(-)$ behaves as a function space from a type of clocks, one can prove

$$\begin{aligned} \forall\kappa.\Sigma(a : A).B(a) &\simeq \Sigma(a : \forall\kappa.A).\forall\kappa.B(a[\kappa]) \\ &\simeq \Sigma(a : A).\forall\kappa.B(a) \end{aligned}$$

using the assumption that A is clock invariant in the last equivalence. The case for universal quantification over clocks uses $\forall\kappa.\forall\kappa'.A \simeq \forall\kappa'.\forall\kappa.A$.

In the case of guarded recursive types, first note that if F commutes with clock quantification, so does $\triangleright^\kappa F$. This can be proved using $\forall\kappa'.\, \triangleright^\kappa A \simeq \triangleright^\kappa\forall\kappa'.A$, the left to right map of which maps a to

$$\lambda(\alpha : \kappa).\lambda\kappa'.a[\kappa']\,[\alpha]$$

for α fresh. This map type checks because $\mathsf{TL}(\kappa' : \mathsf{clock}) = \kappa' : \mathsf{clock}$. Using this, we can prove by guarded recursion that X is clock irrelevant as follows

$$\begin{aligned} \forall\kappa'.X &\simeq \forall\kappa'.F(\triangleright^\kappa X) \\ &\simeq F(\forall\kappa'.\, \triangleright^\kappa X) \\ &\simeq F(\triangleright^\kappa\forall\kappa'.X) \\ &\simeq F(\triangleright^\kappa X) \end{aligned}$$

using the guarded recursion assumption in the last line.

In the case of path types, if $x, y : A$ then

$$\begin{aligned} \forall\kappa.(\mathsf{Path}_A(x, y)) &\simeq \mathsf{Path}_{\forall\kappa.A}(\lambda\kappa.x, \lambda\kappa.y) \\ &\simeq \mathsf{Path}_A(x, y) \end{aligned}$$

The first of these equivalences simply swaps the clock and interval argument, the second uses the assumption that A is clock invariant, which means precisely that $\lambda a.\lambda\kappa.a$ is an equivalence, and so preserves path types. □

We now give a proof of Theorem 4.3. It uses the following lemma establishing the existence of a final $F \circ \triangleright^\kappa$-coalgebra for any endofunctor F.

Lemma A.1. *Let F be an I-indexed endofunctor, then for all κ, the type $v^\kappa(F) \stackrel{def}{=} \mathsf{fix}^\kappa\,(\lambda X.F(\triangleright^\kappa X)) : I \to \mathsf{U}$ has a final $F \circ \triangleright^\kappa$-coalgebra structure, i.e., there is a map $\mathsf{out}^\kappa : v^\kappa(F) \to$*

$F(\triangleright^\kappa \nu^\kappa(F))$, such that for all maps $f : X \to F(\triangleright^\kappa X)$ the following type is contractible

$$\Sigma(h : X \to \nu^\kappa(F)).\, \mathrm{out}^\kappa \circ h = F(\triangleright^\kappa(h)) \circ f$$

Proof. The map out^κ is given by the equivalence between a fixpoint in the universe and its unfolding, so we also have an inverse $(\mathrm{out}^\kappa)^{-1}$. The functor $\triangleright^\kappa$ is locally contractible [10] in the sense that the action on morphisms factors as a composition of two maps

$$(X \to Y) \to \triangleright^\kappa(X \to Y) \to (\triangleright^\kappa X \to \triangleright^\kappa Y)$$

and so also the mapping $\lambda h.(\mathrm{out}^\kappa)^{-1} \circ F(\triangleright^\kappa(h)) \circ f$ factors as a composition

$$(X \to \nu^\kappa(F)) \to \triangleright^\kappa(X \to \nu^\kappa(F)) \to (X \to \nu^\kappa(F))$$

Then by uniqueness of fixpoints (Lemma 3.2) we get the contractibility of

$$\Sigma(h : X \to \nu^\kappa(F)).\, h = (\mathrm{out}^\kappa)^{-1} \circ F(\triangleright^\kappa(h)) \circ f$$

which in turn is equivalent to our goal by postcomposition with out^κ. □

Proof of Theorem 4.3. We define the coalgebra $\mathrm{out} : \nu(F) \to F\,\nu(F)$ as $F(\mathrm{force}) \circ \mathrm{can}_F^{-1} \circ \forall\kappa(\mathrm{out}^\kappa)$. Given any coalgebra $f : X \to F\,X$, we can extend it to $\tilde{f}^\kappa : X \to F\,(\triangleright^\kappa X)$ for any κ. Then by lemma A.1 we have that for any κ the type

$$\Sigma(h : X \to \nu^\kappa(F)).\, \mathrm{out}^\kappa \circ h = F(\triangleright^\kappa(h)) \circ \tilde{f}^\kappa$$

is contractible. By clock quantification preserving contractibility and commuting with Σ types we have that

$$\Sigma(h : \forall\kappa.\, X \to \nu^\kappa(F)).\, \forall\kappa.\, (\mathrm{out}^\kappa) \circ h\,[\kappa] = F(\triangleright^\kappa(h\,[\kappa])) \circ \tilde{f}^\kappa$$

is also contractible. Then by $\forall\kappa.\, X \to \nu^\kappa(F) \simeq X \to \nu(F)$ and that $(\lambda\kappa.\, F(\triangleright^\kappa(h\,[\kappa])) \circ \tilde{f}^\kappa) = \mathrm{can}_F \circ F(\mathrm{force}^{-1}) \circ F(h)$ we obtain the desired result. More details on the calculations can be found in [39]. □

Example 4.5 uses the following lemma.

Lemma A.2. *Let A be a clock irrelevant set and let $X : \mathsf{P}_\mathsf{f}(A)$, $a : A$. Then $a \in X$ is clock irrelevant.*

Proof. The proof is by induction on X, which is valid since statements of the form $\mathsf{IsEquiv}(f)$ are propositions. If $X = \{b\}$ then $a \in X$ is by definition $\mathrm{Path}_A(a, b)$, which is clock irrelevant by Lemma 4.2. If $X = Y \cup Z$ then $a \in X$ is $(a \in Y) \times (a \in Z)$ which is clock-irrelevant by induction and Lemma 4.2. □

We now give a proof of Theorem 4.6. We will write

$$\mathsf{ufld}^\kappa : \mathsf{LTS}^\kappa \to \mathsf{P}_\mathsf{f}(A \times \triangleright^\kappa \mathsf{LTS}^\kappa)$$

for the equivalence associated with the guarded recursive type LTS^κ. First note the following.

Lemma A.3. *The following diagram commutes up to path equality.*

$$\begin{array}{ccc} \mathsf{LTS} & \xrightarrow{\mathsf{ufld}} & \mathsf{P}_\mathsf{f}(A \times \mathsf{LTS}) \\ {\scriptstyle \mathsf{ev}_\kappa}\downarrow & & \downarrow{\scriptstyle \mathsf{P}_\mathsf{f}(A \times f)} \\ \mathsf{LTS}^\kappa & \xrightarrow{\mathsf{ufld}^\kappa} & \mathsf{P}_\mathsf{f}(A \times \triangleright^\kappa \mathsf{LTS}^\kappa) \end{array}$$

where $\mathsf{ev}_\kappa \stackrel{def}{=} \lambda x.x[\kappa]$ and $f = \lambda x.\lambda(\alpha : \kappa).x[\kappa]$.

Proof. The map ufld is defined to be the composition of the following maps

$$\begin{aligned} \forall\kappa.\mathsf{ufld}^\kappa &: \forall\kappa.\mathsf{LTS}^\kappa \to \forall\kappa.\mathsf{P}_\mathsf{f}(A \times \triangleright^\kappa \mathsf{LTS}^\kappa) \\ \varphi &: \forall\kappa.\mathsf{P}_\mathsf{f}(A \times \triangleright^\kappa \mathsf{LTS}^\kappa) \to \mathsf{P}_\mathsf{f}(A \times \forall\kappa.\, \triangleright^\kappa\, \mathsf{LTS}^\kappa) \\ \mathsf{P}_\mathsf{f}(A \times \mathsf{force}) &: \mathsf{P}_\mathsf{f}(A \times \forall\kappa.\, \triangleright^\kappa\, \mathsf{LTS}^\kappa) \to \mathsf{P}_\mathsf{f}(A \times \forall\kappa.\mathsf{LTS}^\kappa) \end{aligned}$$

where $\forall\kappa.\mathsf{ufld}^\kappa(x) = \lambda\kappa.\mathsf{ufld}^\kappa(x[\kappa])$, φ is the witness that $\mathsf{P}_\mathsf{f}(A \times (-))$ commutes with clock quantification, and force : $\forall\kappa.\, \triangleright^\kappa\, \mathsf{LTS}^\kappa \to \forall\kappa.\mathsf{LTS}^\kappa$ is the inverse to $\lambda x.\lambda(\alpha : \kappa).x$ up to path equality. By the latter, we get the following sequence of path equalities

$$\begin{aligned} \mathsf{P}_\mathsf{f}(A \times f) \circ \mathsf{ufld} &= \mathsf{P}_\mathsf{f}(A \times \mathsf{ev}_\kappa) \circ \varphi \circ \forall\kappa.\mathsf{ufld}^\kappa \\ &= \mathsf{ev}_\kappa \circ \forall\kappa.\mathsf{ufld}^\kappa \\ &= \mathsf{ufld}^\kappa \circ \mathsf{ev}_\kappa \end{aligned}$$

as desired. □

Proof of Theorem 4.6. Møgelberg and Veltri [41] prove that path equality coincides with bisimilarity for guarded recursive types. Using their results we can prove that, given $x, y : \mathsf{LTS}$

$$\begin{aligned} \mathrm{Path}_{\mathsf{LTS}}(x, y) &\simeq \forall\kappa.\mathrm{Path}_{\mathsf{LTS}^\kappa}(x[\kappa], y[\kappa]) \\ &\simeq \forall\kappa.\mathrm{Bisim}^\kappa(x[\kappa], y[\kappa]) \end{aligned} \tag{11}$$

where the first equivalence uses functional extensionality for universal quantification over clocks and the second is [41, Corollary 5.4]. Here $\mathrm{Bisim}^\kappa(x, y) = \mathrm{Sim}^\kappa(x, y) \times \mathrm{Sim}^\kappa(y, x)$ where

$$\begin{aligned} \mathrm{Sim}^\kappa(x, y) \simeq \Pi(x' : \triangleright^\kappa \mathsf{LTS}^\kappa, a : A).(a, x') \in \mathsf{ufld}^\kappa(x) \to \\ \exists y' : \triangleright^\kappa \mathsf{LTS}^\kappa.(a, y') \in \mathsf{ufld}^\kappa(y) \times \\ \triangleright\,(\alpha : \kappa).\mathrm{Sim}^\kappa(x'\,[\alpha], y'\,[\alpha]) \end{aligned}$$

We must compare this to bisimilarity of x and y which is defined as

$$\forall\kappa.(\mathrm{Sim}^\kappa_\forall(x, y) \times \mathrm{Sim}^\kappa_\forall(y, x))$$

where

$$\begin{aligned} \mathrm{Sim}^\kappa_\forall(x, y) \simeq \Pi(x' : \mathsf{LTS}, a : A).(a, x') \in \mathsf{ufld}(x) \to \\ \exists y' : \mathsf{LTS}.(a, y') \in \mathsf{ufld}(y) \times \\ \triangleright\,(\alpha : \kappa).\mathrm{Sim}^\kappa_\forall(x', y') \end{aligned}$$

By an easy guarded recursive argument one can show that $\mathrm{Sim}^\kappa_\forall$ is a reflexive relation, and from this it follows that path

equality implies bisimilarity. To prove the other implication it suffices to show that

$$\Pi(x, y : \mathsf{LTS}).\mathsf{Sim}^{\kappa}_{\forall}(x, y) \to \mathsf{Sim}^{\kappa}(x[\kappa], y[\kappa])$$

and this statement is proved by guarded recursion. So suppose $x, y : \mathsf{LTS}$ and $\mathsf{Sim}^{\kappa}_{\forall}(x, y)$. Suppose further that $x' : \triangleright^{\kappa}\mathsf{LTS}^{\kappa}$, $a : A$ and $(a, x') \in \mathsf{ufld}^{\kappa}(x[\kappa])$. By Lemma A.3 this means that $(a, x') \in \mathsf{P_f}(A \times f)(\mathsf{ufld}(x))$ where $f = \lambda x.\lambda(\alpha : \kappa).x[\kappa]$. By [41, Lemma 4.1] there then (merely, i.e. in the sense of $\exists$) exists an $x'' : \mathsf{LTS}$ such that $x' = f(x'')$, i.e.,

$$x' = \lambda(\alpha : \kappa).(x''[\kappa]) \tag{12}$$

and $(a, x'') \in \mathsf{ufld}(x)$. By the assumption that $\mathsf{Sim}^{\kappa}_{\forall}(x, y)$ there then merely exists a $y'' : \mathsf{LTS}$ such that $(a, y'') \in \mathsf{ufld}(y)$ and

$$\triangleright(\alpha : \kappa).\mathsf{Sim}^{\kappa}_{\forall}(x'', y''). \tag{13}$$

Setting $y' = f(y'')$ then, again by [41, Lemma 4.1] $(a, y') \in \mathsf{P_f}(A \times f)(\mathsf{ufld}(y))$ and so by Lemma A.3, $(a, y') \in \mathsf{ufld}^{\kappa}(y[\kappa])$. It remains to show that

$$\triangleright(\alpha : \kappa).\mathsf{Sim}^{\kappa}(x'\ [\alpha], y'\ [\alpha])$$

which reduces to

$$\triangleright(\alpha : \kappa).\mathsf{Sim}^{\kappa}(x''[\kappa], y''[\kappa])$$

using (12) and definition of y'. This follows by guarded recursion from (13). □

A.4 Omitted proofs section 5

The equational theory of boundary terms is given by $\equiv_b$ defined in Figure 7. This theory is used in the typing of boundary terms, when typing systems and homogenous compositions. More precisely, the requirement is that for a system of boundary conditions $[\varphi_1\ M_1 \dots \varphi_m\ M_m]$ to be wellformed, it must be the case that $\varphi_i \wedge \varphi_j$ implies $M_i \equiv_b M_j$, for any i, j. Similarly, for $\mathsf{hcomp}^{j}_{\mathsf{H}\,\delta}[\psi \mapsto M']\ M'_0$ to be well typed, we must have $M'[0/j] \equiv_b M'_0$.

The next lemma states that boundaries are well-typed.

Lemma A.4. *Let $e = [\varphi_0\ M_0, \dots, \varphi_{n_\ell}\ M_{n_\ell}]$ be the boundary condition for con_ℓ. Assume $\Gamma \vdash \mathscr{E} : \mathscr{K}_{<\ell} \rightharpoonup^{\delta} D$; then the following typing holds:*

$$\Gamma, \gamma : \forall\overline{\kappa}.\Gamma_\ell[\delta[\overline{\kappa}]], \overline{x} : \overline{\forall\overline{\kappa}.\Xi_\ell[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]] \to H(\delta[\overline{\kappa}])},$$
$$\overline{y} : \overline{\Pi(\xi : \forall\overline{\kappa}.\Xi[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]]).D[\lambda\overline{\kappa}.x[\overline{\kappa}](\xi[\overline{\kappa}])]}, \overline{i} : \Psi_\ell, \varphi_\ell \vdash$$
$$(\!|e|\!)^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}} : D[\lambda\overline{\kappa}.\mathsf{con}_\ell(\gamma[\overline{\kappa}], \overline{x[\overline{\kappa}]}, \overline{i})]$$

Proof. Throughout we assume that $\Gamma \vdash \delta : \forall\kappa.\Delta$ unless otherwise specified. Through an induction on the structure of the boundary terms, we prove the following more general typing:

$$\Gamma', \hat{\gamma} : \forall\overline{\kappa}.\hat{\Gamma}[\delta[\overline{\kappa}], \gamma\ [\kappa]] \vdash (\!|M|\!)^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}, \hat{\gamma}} : D[\lambda\overline{\kappa}.M[\gamma[\overline{\kappa}], \overline{x[\overline{\kappa}]}, \overline{i}, \hat{\gamma}[\overline{\kappa}]]]$$

where Γ' is defined as follows

$$\Gamma' \stackrel{\mathsf{def}}{=} \Gamma, \gamma : \forall\overline{\kappa}.\Gamma_i[\delta[\overline{\kappa}]], \overline{x} : \overline{\forall\overline{\kappa}.\Xi_i[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]] \to \mathsf{H}(\delta[\overline{\kappa}])},$$
$$\overline{y} : \overline{(\xi : \forall\overline{\kappa}.\Xi[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]]) \to D[\lambda\overline{\kappa}.x[\overline{\kappa}](\xi[\overline{\kappa}])]}, \overline{i} : \Psi_i, \varphi_i$$

and M is assumed to be a boundary term of type $\mathsf{H}\,\delta$ in context

$$\delta : \Delta, \gamma : \Gamma_i[\delta], \overline{x} : \Xi_i[\delta, \gamma] \to \mathsf{H}\,\delta, \overline{i} : \Psi_i, \varphi_i, \hat{\gamma} : \hat{\Gamma}[\delta, \gamma]$$

The desired typing is then the case where $\hat{\Gamma}$ above is empty, since we have that φ_k implies that $\lambda\overline{\kappa}.\mathsf{con}_i(\dots)$ reduces to $\lambda\overline{\kappa}.M_k$. The $\hat{\Gamma}$ crops up because the interpretation of constructors adds a $\xi : \forall\overline{\kappa}.\Xi[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]]$ to the context while the hcomp case adds an interval variable and a face restriction to the context for the system of the composition. In the very first step, unfolding the list of partial elements adds a face restriction as well. Concretely we proceed as follows in each case. Also, we write σ for the substitution $[\delta[\overline{\kappa}]/\delta, \gamma[\overline{\kappa}]/\gamma, \overline{x[\overline{\kappa}]}/\overline{x}, \hat{\gamma}[\overline{\kappa}]/\hat{\gamma}]$ and τ for the same but without the $\overline{x}$ component.

- Assume $M = x_j\,\overline{u}$. From the typing assumptions on M we have that

  $$\delta : \Delta, \gamma : \Gamma_i, \overline{x} : \overline{\Xi_i[\delta, \gamma] \to \mathsf{H}\,\delta}, \overline{i} : \Psi, \varphi_i, \hat{\Gamma} \vdash u : \Xi_{i,j}$$

  This means that $\xi = \lambda\overline{\kappa}.\overline{u}\tau$ has type $\forall\overline{\kappa}.\Xi_{i,j}[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]]$, which is exactly the input to y_j, so that $y_j\ \xi$ has type $D[\lambda\overline{\kappa}.\,x[\overline{\kappa}](\overline{u}\tau)]$, as desired.
- Assume $M = \mathsf{con}_j(\overline{t}, \overline{\lambda\xi.M'}, \overline{r})$. By inductive hypothesis we have that

  $$\Gamma', \hat{\gamma} : \forall\overline{\kappa}.\,\hat{\Gamma}[\delta[\overline{\kappa}], \gamma[\overline{\kappa}]], \xi : \forall\overline{\kappa}.\Xi_{j,k}[\delta[\overline{\kappa}], \overline{t}\tau] \vdash (\!|M'_k|\!)^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}, (\hat{\gamma}, \xi)} : D[\lambda\overline{\kappa}.M'_k\sigma]$$

  This gives us the input necessary to apply $\mathscr{E}_j$. The $\overline{t}$ family has a similar typing structure to $\overline{u}$ in the previous example so we get $\overline{t}' = \lambda\overline{\kappa}.\overline{t}\tau$ of type $\forall\overline{\kappa}.\Gamma_j[\delta[\overline{\kappa}]]$. We obtain maps

  $$S_k : \forall\overline{\kappa}.\,\Xi_{j,k}[\delta[\overline{\kappa}], \overline{t}'[\overline{\kappa}]] \to \mathsf{H}(\delta[\overline{\kappa}])$$
  $$S_k = \lambda\overline{\kappa}.\lambda\xi.M'_k\sigma$$

  directly from the typing assumptions and by inductive hypothesis we obtain maps

  $$R_k : (\xi : \forall\overline{\kappa}.\Xi_{j,k}[\delta[\overline{\kappa}], \overline{t}'[\overline{\kappa}]]) \to D[\lambda\overline{\kappa}.S_k[\overline{\kappa}](\xi[\overline{\kappa}])]$$
  $$R_k = \lambda\xi.(\!|M'_k|\!)^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}, (\hat{\gamma}, \xi)}$$

  This means we have the required data to apply $\mathscr{E}_j$, and by the definition of clock abstracted elimination lists we have that $\mathscr{E}_j[\overline{t}', \overline{S}, \overline{R}, \overline{r}]$ inhabits D over $\lambda\overline{\kappa}.M\sigma$.
- Assume $M = \mathsf{hcomp}^{j}_{\mathsf{H}\,\delta}[\psi \mapsto M']\ M'_0$. By inductive hypothesis and the typing assumptions for the hcomp to be well formed we have that $(\!|M'_0|\!)^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}, \hat{\gamma}}$ inhabits D over $\lambda\overline{\kappa}.\,M'_0\sigma$, and $(\!|M'|\!)^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}, \hat{\gamma}, j}$ inhabits D over $\lambda\overline{\kappa}.\,M'\sigma$ in the context extended by $j : \mathbb{I}$ and restricted by ψ. The v term provides a path between $\lambda\overline{\kappa}.\,M'_0\sigma$ and

Rules for equality of boundary terms

$$\frac{\Gamma \vdash \overline{u} \equiv \overline{v}}{\Gamma \vdash x\ \overline{u} \equiv_b x\ \overline{v}} \qquad \frac{\Gamma \vdash \overline{t} \equiv \overline{t'} \qquad \Gamma \vdash \overline{r} \equiv \overline{r'} \qquad \forall i.(\Gamma, \overline{\xi_i} \vdash M_i \equiv_b M_i')}{\Gamma \vdash \mathsf{con}_\ell(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}) \equiv_b \mathsf{con}_\ell(\overline{t'}, \overline{\lambda\xi.\, M'}, \overline{r'})}$$

$$\frac{\Gamma, j, \varphi \vdash M \equiv_b M' \qquad \Gamma \vdash M_0 \equiv_b M_0'}{\Gamma \vdash \mathsf{hcomp}^j_{\mathsf{H}\,\delta}\ [\varphi \mapsto M]\ M_0 \equiv_b \mathsf{hcomp}^j_{\mathsf{H}\,\delta}\ [\varphi \mapsto M']\ M_0'} \qquad \frac{e_\ell = [\varphi_1\ N_1 \dots \varphi_m\ N_m] \qquad \Gamma \vdash \varphi_i \equiv \top}{\Gamma \vdash \mathsf{con}_\ell(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}) \equiv_b N_i(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r})}$$

$$\frac{\Gamma \vdash \varphi \equiv \top}{\Gamma \vdash \mathsf{hcomp}^j_{\mathsf{H}\,\delta}\ [\varphi \mapsto M]\ M_0 \equiv_b M[1/j]}$$

Substitution operation

$$(x_j\ \overline{u})(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}) = M_j[\overline{u}[\overline{t}, \overline{r}]/\overline{\xi_j}]$$

$$\mathsf{con}_\ell(\overline{t'}, \overline{\lambda\xi'.\, M'}, \overline{r'})(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}) = \mathsf{con}_\ell(\overline{t'}[\overline{t}, \overline{r}], \overline{\lambda\xi'.\, M'(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}, \overline{\xi'})}, \overline{r'}[\overline{r}])$$

$$(\mathsf{hcomp}^j_{\mathsf{H}\,\delta}\ [\varphi \mapsto N]\ N_0)(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}) = \mathsf{hcomp}^j_{\mathsf{H}\,\delta}\ [\varphi \mapsto (N(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}, j))]\ (N_0(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r}))$$

Figure 7. The equational theory of boundary terms is the least equivalence relation generated by the rules above. The rule for reducing a constructor to its boundary uses the substitution also defined above.

its composition with $\lambda\overline{\kappa}.\, M'\sigma$, which means that the composition in $D[vj]$ provides a term over $\lambda\overline{\kappa}.\, M\sigma$ as desired. Finally, for the composition to be well typed, we must verify that

$$(\!|M'|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y},j}[0/j] \equiv (\!|M_0'|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}$$

An easy induction shows that the left hand side equals $(\!|M'[0/j]|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}$ and since $M_0' \equiv_b M'[0/j]$, this follows from Lemma A.5 below.

- Assume $M = [\varphi_0\ M_0, \dots, \varphi_{n_i}\ M_{n_i}]$. In this case we have the following typing by inductive hypothesis:

$$\Gamma', \varphi_k \vdash (\!|M_k|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}} : D[\lambda\overline{\kappa}.M_k\sigma]$$

Finally, in order to conclude that

$$[\varphi_1\ (\!|M_1|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}, \dots, \varphi_{n_i}\ (\!|M_{n_i}|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}]$$

defines a system, and so a partial element of $D[\lambda\overline{\kappa}.M\sigma]$, we must show that on faces of the form $\varphi_j \wedge \varphi_k$ the judgemental equality

$$(\!|M_j|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}} \equiv (\!|M_k|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}$$

holds. Since on this face $M_j \equiv_b M_k$, this follows from Lemma A.5 below. □

Lemma A.5. *If* $\Gamma \vdash M \equiv_b N$ *then* $\Gamma \vdash (\!|M|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}} \equiv (\!|N|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}$

Proof. The proof is by induction on the proof of $M \equiv_b N$. The interesting case is the reduction of a constructor to its boundary, which requires showing that $\varphi_i \equiv \top$ implies

$$(\!|\mathsf{con}_\ell(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r})|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}} \equiv (\!|N_i(\overline{t}, \overline{\lambda\xi.\, M}, \overline{r})|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}} \quad (14)$$

The left hand side of this equation is $\mathscr{E}_\ell[\overline{t'}, \overline{S}, \overline{R}, \overline{r}]$, which under the assumption that $\varphi_i \equiv \top$ equals

$$(\!|N_i|\!)^\delta_{\mathscr{E},\overline{x}\mapsto\overline{y},\hat{y}}[\overline{t'}, \overline{S}, \overline{R}, \overline{r}/\overline{y}, \overline{x}, \overline{y}, \overline{i}]$$

It thus suffices to prove equality of the above with the right hand side of (14). We omit the straight forward induction proof of this substitution property. □

Details of higher truncation. Recall that we defined a map $\alpha : \forall\kappa.\, \|A\|_n \to \|\forall\kappa.\, A\|_n$. For $f : \forall\kappa.\, \mathbb{S}^{n+1} \to \|A\|_n$ we let $f' = \lambda s.\, \alpha(\lambda\kappa.\, (f\ [\kappa])(s))$ and $p_s : \lambda\kappa.\, s\ [\kappa_0] = s$ is the path extracted from clock irrelevance of $\mathbb{S}^{n+1}$. Note that from the definition of α, we get the following reductions:

$$\begin{aligned}
\alpha(\lambda\kappa.\mathsf{in}(a\ [\kappa])) &\equiv \mathsf{in}(a)\\
\alpha(\lambda\kappa.\mathsf{hub}(f\ [\kappa])) &\equiv \mathsf{hub}(f')\\
\alpha(\lambda\kappa.\ \mathsf{spoke}(s\ [\kappa], f\ [\kappa], i)) &\equiv\\
&\mathsf{hcomp}^j[(i=0) \mapsto \alpha(\lambda\kappa.\ (f\ [\kappa])((p_s\ j)\ [\kappa])),\\
&\qquad (i=1) \mapsto \mathsf{hub}(f')]\\
&\qquad\ \mathsf{spoke}(s\ [\kappa_0], f', i)
\end{aligned}$$

For ease of reasoning, we write out the reductions for the canonical map β:

$$\begin{aligned}
\beta(\mathsf{in}(a)) &\equiv \lambda\kappa.\mathsf{in}(a\ [\kappa])\\
\beta(\mathsf{hub}(f)) &\equiv \lambda\kappa.\mathsf{hub}(\lambda s.\ (\beta(f(s)))\ [\kappa])\\
\beta(\mathsf{spoke}(s, f, i)) &\equiv \lambda\kappa.\ \mathsf{spoke}(s, \lambda s.\ (\beta(f(s)))\ [\kappa], i)
\end{aligned}$$

We show that the two maps are inverse to one another by induction, leaving out the trivial first case. For hub we reason as follows:

$$\begin{aligned}\alpha(\beta(\mathsf{hub}(f))) &\equiv \alpha(\lambda\kappa.\mathsf{hub}(\lambda s.\,(\beta(f(s)))\,[\kappa]))\\ &\equiv \mathsf{hub}(\lambda s.\,\alpha(\lambda\kappa.\,(\beta(f(s)))\,[\kappa]))\\ &\equiv \mathsf{hub}(\lambda s.\,\alpha(\beta(f(s))))\\ &= \mathsf{hub}(\lambda s.\,f(s))\\ &\equiv \mathsf{hub}(f)\end{aligned}$$

$$\begin{aligned}\beta(\alpha(\lambda\kappa.\mathsf{hub}(f\,[\kappa]))) &\equiv \beta(\mathsf{hub}(\lambda s.\,\alpha(\lambda\kappa.\,(f\,[\kappa])(s))))\\ &\equiv \lambda\kappa'.\,\mathsf{hub}(\lambda s.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])(s)))\,[\kappa'])\\ &= \lambda\kappa'.\,\mathsf{hub}(\lambda s.\,(\lambda\kappa.\,(f\,[\kappa])(s))\,[\kappa'])\\ &\equiv \lambda\kappa.\,\mathsf{hub}(\lambda s.\,(f\,[\kappa])(s))\\ &\equiv \lambda\kappa.\,\mathsf{hub}(f\,[\kappa])\end{aligned}$$

Note that the only non-judgemental equality is the inductively justified cancellation of the compositions i.e., the path $q : \lambda s.\,\alpha(\beta(f(s))) = f$ and $r : \lambda s.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])(s))) = \lambda s.\,\lambda\kappa.\,(f\,[\kappa])(s)$ under the hub constructor.

We now supply the first calculation for the spoke constructor:

$$\begin{aligned}&\alpha(\beta(\mathsf{spoke}(s,f,i))) \equiv \alpha(\lambda\kappa.\,\mathsf{spoke}(s,\lambda s.\,(\beta(f(s)))\,[\kappa],i))\\ &\equiv \mathsf{hcomp}^j\,[(i=0)\mapsto \alpha(\lambda\kappa.\,(\beta(f((p_{\mathsf{const}\,s}\;j\;[\kappa]))))\,[\kappa]),\\ &\qquad (i=1)\mapsto \mathsf{hub}(\lambda s.\,\alpha(\lambda\kappa.\,(\beta(f(s)))\,[\kappa]))]\\ &\qquad \mathsf{spoke}(s,\lambda s.\,\alpha(\lambda\kappa.\,(\beta(f(s)))\,[\kappa]),i)\\ &\qquad = \mathsf{spoke}(s,\lambda s.\,\alpha(\lambda\kappa.\,(\beta(f(s)))\,[\kappa]),i)\\ &\qquad \equiv \mathsf{spoke}(s,\lambda s.\,\alpha(\beta(f(s))),i)\\ &\qquad = \mathsf{spoke}(s,f,i)\end{aligned}$$

For us to apply induction with the above path it would need to reduce strictly to the recursive call at s, $\lambda j.\,q(j)(s)$, on $i=0$ and the hub case, $\lambda j.\,\mathsf{hub}(\lambda s.\,q(j)(s))$, on $i=1$. This is too tall an ask, but luckily it is sufficient for our purposes that we can show such reductions up to a path. The path used for the induction is then defined by an appropriate composition as in the definition of the map α.

The calculation above uses two non-trivial paths. The first is the reduction of a composition to its base, which we note restricts to $\lambda j.\,\alpha(\lambda\kappa.\,(\beta(f((p_{\mathsf{const}\,s}\;j\;[\kappa]))))\,[\kappa])$ and reflexivity on $i=0$ and $i=1$ respectively. The second is exactly an application of q from above to the function input, or more concretely the path $\lambda j.\mathsf{spoke}(s,\lambda s.\,q(j)(s)),i)$. The reductions of spoke then mean that this reduces to exactly the paths we are looking for. Since p_{const} is given by clock irrelevance at a constant function it is path equal to reflexivity by lemma A.8. Cancelling this path at $i=0$ and reflexivity at $i=1$ with a composition then yields the desired path.

$$\begin{aligned}&\beta(\alpha(\lambda\kappa.\,\mathsf{spoke}(s\,[\kappa],f\,[\kappa],i)))\\ &\equiv \beta(\mathsf{hcomp}^j\,[(i=0)\mapsto \alpha(\lambda\kappa.\,(f\,[\kappa])((p_s\;j)\;[\kappa])),\\ &\qquad (i=1)\mapsto \mathsf{hub}(\lambda s.\,\alpha(\lambda\kappa.\,(f\,[\kappa])(s))))]\\ &\qquad \mathsf{spoke}(s\,[\kappa_0],\lambda s.\,\alpha(\lambda\kappa.\,(f\,[\kappa])(s))),i))\\ &\equiv \mathsf{hcomp}^j\,[(i=0)\mapsto \beta(\alpha(\lambda\kappa.\,(f\,[\kappa])((p_s\;j)\;[\kappa]))),\\ &\qquad (i=1)\mapsto \beta(\mathsf{hub}(\lambda s.\,\alpha(\lambda\kappa.\,(f\,[\kappa])(s)))))]\\ &\qquad \beta(\mathsf{spoke}(s\,[\kappa_0],\lambda s.\,\alpha(\lambda\kappa.\,(f\,[\kappa])(s))),i)))\\ &\equiv \mathsf{hcomp}^j\,[(i=0)\mapsto \beta(\alpha(\lambda\kappa.\,(f\,[\kappa])((p_s\;j)\;[\kappa]))),\\ &\qquad (i=1)\mapsto \lambda\kappa'.\,\mathsf{hub}(\lambda s.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])(s)))\,[\kappa'])]\\ &\qquad \lambda\kappa'.\mathsf{spoke}(s\,[\kappa_0],\lambda s.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])(s))))\,[\kappa'],i)\\ &= \lambda\kappa'.\mathsf{spoke}(s\,[\kappa_0],\lambda s.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])(s))))\,[\kappa'],i)\\ &= \lambda\kappa'.\mathsf{spoke}(s\,[\kappa'],\lambda s.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])(s))))\,[\kappa'],i)\\ &= \lambda\kappa.\mathsf{spoke}(s\,[\kappa],f\,[\kappa],i)\end{aligned}$$

This time the boundary obligation is that on $i=0$ the above must reduce to $\lambda j.\lambda\kappa.\,r(j)(s\,[\kappa])\,[\kappa]$ and on $i=1$ it must reduce to $\lambda j.\,\lambda\kappa.\,\mathsf{hub}(\lambda s.\,r(j)(s\,[\kappa])\,[\kappa])$. The calculation is a composition of three non-trivial paths. We first reduce the composition to its base, which reduces to reflexivity on $i=1$ and $\lambda j.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])((p_s\;j)\;[\kappa])))$ on $i=0$. The next path is again the path extracted from clock irrelevance, this time the one that shows that $s\,[\kappa_0] = s\,[\kappa']$, which is exactly the inverse application of clock irrelevance. This reduces to reflexivity on $i=1$ and the path $\lambda j.\,\beta(\alpha(\lambda\kappa.\,(f\,[\kappa])((p_s^{-1}\;j)\;[\kappa])))$ on $i=0$. At this point the composite path has the shape $\mathsf{refl}\circ\lambda j.\,\beta(\alpha(p'(j)))\circ\lambda j.\,\beta(\alpha(p'^{-1}(j)))$ on $i=0$ and refl^3 on $(i=1)$, meaning that it reduces to refl up to a path in either case. The last path is then the inductively obtained path

$$\lambda j.\,\lambda\kappa.\mathsf{spoke}(s\,[\kappa],\lambda s.\,r(j)(s)\,[\kappa],i).$$

The desired reduction now again follows from an application of the equalities governing the behavior of spoke.

Proving Theorem 5.3. We proceed by constructing a map

$$(h : \forall\kappa.\,\mathsf{H}\,\delta) \to \forall\kappa.\,h\,[\kappa] = h\,[\kappa_0].$$

To do this we first show a slightly modified induction under clocks principle with constant δ and $\Gamma_{(-)}$ parameters. Using this modified principle we construct the map in two stages: first we provide a candidate case for each constructor and secondly we show that the candidate terms satisfy the appropriate boundary condition up to path equality. The latter will allow us to rectify the terms given by the former constructions via a composition to obtain the input for the modified principle, thus allowing us to define a map of the desired type. The new induction principle needs a modified version of the boundary condition, given by transporting the usual one along the equivalence between the two, which means that we need to, at each stage, cohere with the transport in the earlier stages.

Before beginning the proof we need to introduce some notation and prove a small lemma about the structure of HIT constructors.

Lemma A.6. *Let $H\,\delta$ be a HIT with constructors taking input*

$$\gamma^0 : \Gamma_i^0[\delta], \gamma^1 : \Gamma_i^1[\delta], \ldots, \quad x : \Xi_i^0[\delta, \overline{\gamma}] \to H\,\delta, \ldots \text{ and } \overline{i} : \Psi_i$$

and boundary conditions $\varphi_i \vdash e_i$. Then there exists an equivalent HIT $H'\delta$ with constructors taking input of the form

$$\gamma : \Gamma_i[\delta], x : \Xi_i[\delta, \overline{\gamma}] \to H'\delta \text{ and } \overline{i} : \Psi_i$$

with boundary conditions $\varphi_i \vdash e_i$ where φ_i is of the form $\bigvee(i = 0) \vee (i = 1)$ with i ranging over all variables in Ψ. We say that HITs satisfying these three criteria are of reduced form.

Proof. Modifying the Γ and $\Xi \to \mathsf{H}\,\delta$ input is trivial, simply take $\Gamma_i[\delta]$ to be an iterated Σ type consisting of the Γ_i^j and $\Xi_i[\delta, \gamma] \stackrel{\text{def}}{=} \Xi_i^j[\delta, \gamma] + \Xi_i^j[\delta, \gamma] + \ldots$. This procedure clearly yields an equivalent HIT, so we make this assumption freely.

We achieve boundaries of the desired shape by adding constructors to specify the full boundary, noting that $\bigvee(i = 0) \vee (i = 1)$ as above is the second largest element of the face lattice specifying the entire boundary of a cube but not the interior. We call it the total face relative to Ψ. Let con_i be a constructor of $\mathsf{H}\,\delta$ with boundary extent φ_i and interval input Ψ_i. We add a number of constructors to $\mathsf{H'}\,\delta$ recursively in the following way: Write φ_i in disjunctive normal form. For each $j \in \Psi_i$ if $(j = 0)$ and $(j = 1)$ appear as disjuncts of φ_i we add con_i as is. Say that $(j = 0)$ is missing; we then add a constructor $\mathsf{con}_i^{(j=0)}$ with the same Γ and Ξ as con_i. This new constructor then has interval input Ψ with the j variable deleted. Now consider the face $\varphi_i \wedge (j = 0)$. If this is the total face relative to Ψ without j, we define the boundary term of $\mathsf{con}_i^{(j=0)}$ to be the boundary of con_i restricted to $(j = 0)$ and proceed with the next interval variable in Ψ. If it is not the total face and is missing for instance $k \in \Psi$, we repeat the procedure, defining a new constructor $\mathsf{con}_i^{(j=0)\wedge(k=0))}$ in the same way. Having recursively defined this new constructor, we use it for the boundary at $k = 0$.

It is immediate that we can define mutually inverse maps between $\mathsf{H}\,\delta$ and $\mathsf{H'}\,\delta$ using their respective elimination principles. □

The point of the modified boundary shape is that constructors with such boundaries exactly correspond to constructors for heterogeneous, iterated path types. This means that for HITs of the above form we can treat the end result $(i : \Psi) \to A[\varphi \mapsto e]$ as a proper type, allowing it to appear in e.g., Σ-types. We treat it as we would path types, introducing terms of them by abstraction and eliminating from them by application. As for path types we can compose iterated paths, and we record this fact in the following lemma:

Lemma A.7.

$$\frac{\Gamma \vdash p : \Pi(i : \Psi).\, A[\varphi \mapsto u] \qquad \Gamma, (i : \Psi), \varphi \vdash q : u = v}{\Gamma \vdash (i.\,q)^*\, p : \Pi(i : \Psi).\, A[\varphi \mapsto v]}$$

Proof. $(i.\,q)^*\, p \stackrel{\text{def}}{=} \lambda i.\, \mathsf{hcomp}^j\, [\varphi \mapsto q\, j]\, (p\, i)$ □

We define the following types in context where $\delta : \Delta$, and D a family over $\forall\kappa.\mathsf{H}\,\delta$, where $\overline{u_{<\ell}} : \mathscr{E}_{<\ell}(\delta, D)$ as specified below, and $\delta : \forall\kappa.\Gamma_\ell(\delta)$:

$$\begin{aligned}
&\mathscr{P}_\ell(\delta, D, \overline{u_{<\ell}}, \gamma) \stackrel{\text{def}}{=} \\
&\quad \Pi(x : (\forall\kappa.\, \Xi_\ell[\delta, \gamma\,[\kappa]] \to \mathsf{H}\,\delta)) \\
&\qquad (y : \Pi(\xi : \forall\kappa.\, \Xi_\ell[\delta, \gamma\,[\kappa]]).D[\lambda\kappa.x\,[\kappa](\xi\,[\kappa])]) \\
&\qquad (i : \Psi_\ell). \\
&\qquad D[\lambda\kappa.\, \mathsf{con}_\ell(\gamma\,[\kappa], x\,[\kappa], i)][\varphi_\ell \mapsto (\!|e_\ell|\!)^{\delta}_{\overline{u_{<\ell}}, x \mapsto y}] \\
&\mathscr{E}_\ell(\delta, D, \overline{u_{<\ell}}) \stackrel{\text{def}}{=} \Pi(\gamma : \forall\kappa.\Gamma_\ell[\delta]).\, \mathscr{P}_\ell(\delta, D, \overline{u_{<\ell}}, \gamma) \\
&\mathscr{E}'_\ell(\delta, D, \overline{u_{<\ell}}) \stackrel{\text{def}}{=} \Pi(\gamma : \Gamma_\ell[\delta]).\, \mathscr{P}_\ell(\delta, D, \overline{u_{<\ell}}, \lambda_.\, \gamma)
\end{aligned}$$

where

$$\begin{aligned}
\mathscr{E}(\delta, D) = \Sigma(u_{\ell_0} : \mathscr{E}_{\ell_0}(\delta)).\, \Sigma(u_{\ell_1} : \mathscr{E}_{\ell_1}(\delta, u_{\ell_0})). \\
\ldots\, \mathscr{E}_{\ell_n}(\delta, (u_{\ell_0}, u_{\ell_1}, \ldots, u_{\ell_{n-1}}))
\end{aligned}$$

and $\mathscr{E}_{<\ell}(\delta, D)$ is the prefix of the above iterated Σ types containing fields only for the labels below ℓ. The above definition is well-founded because $\mathscr{E}_\ell(\delta, -)$ only refers to $\mathscr{E}_{\ell'}(\delta, -)$ for labels $\ell' < \ell$.

Lemma A.8. *For each ℓ we have*

$$\begin{aligned}
\mathsf{cirr}_{\Gamma_\ell} &: \Pi(\gamma : \forall\kappa.\Gamma_\ell[\delta]).\, \lambda_.\, \gamma\,[\kappa_0] = \gamma \\
\mathsf{cirr\text{–}coh}_{\Gamma_\ell} &: \Pi(\gamma : \Gamma_\ell[\delta]).\, \mathsf{refl} = \mathsf{cirr}_{\Gamma_\ell}\, (\lambda_.\, \gamma)
\end{aligned}$$

Proof. From clock irrelevance of Γ_ℓ we get a proof that the constant map from $\Gamma_\ell[\delta]$ to $\forall\kappa.\Gamma_\ell[\delta]$ has a right inverse. Moreover it has application to κ_0 as a left inverse with reflexivity as the proof. From this we obtain a proof that the constant map is an half adjoint equivalence, from which we can project $\mathsf{cirr}_{\Gamma_\ell}$ and $\mathsf{cirr\text{–}coh}_{\Gamma_\ell}$. □

Lemma A.9. *For each ℓ, δ and $u_{<\ell}$ we have an equivalence of types $f_\ell : \mathscr{E}'_\ell(\delta, D, u_{<\ell}) \simeq \mathscr{E}_\ell(\delta, D, u_{<\ell})$.*

Proof. We will use $\mathsf{cirr}_{\Gamma_\ell}$ to transport $\mathscr{P}_\ell(\delta, D, u_{<\ell}, \lambda\_.\, \gamma\,[\kappa_0])$ to $\mathscr{P}_\ell(\delta, D, u_{<\ell}, \gamma)$. Concretely we define

$$f_\ell(u') \stackrel{\text{def}}{=} \lambda\gamma.\, \mathsf{cirr}_{\Gamma_\ell}(\gamma)^*(u'\, (\gamma\,[\kappa_0]))$$

□

We then define $\mathscr{E}'(\delta, D)$ as the iterated sigma type

$$\begin{aligned}
\Sigma(u_{\ell_0} : \mathscr{E}'_{\ell_0}(\delta, D)).\Sigma(u_{\ell_1} : \mathscr{E}'_{\ell_1}(\delta, D, f_{\ell_0}(u_{\ell_0})). \\
\ldots \mathscr{E}'_{\ell_n}(\delta, D, f(\overline{u_{<\ell_n}})
\end{aligned}$$

where we wrote $f(\overline{u_{<\ell_n}})$ in place of

$$f_{\ell_0}(u_{\ell_0}), f_{\ell_1}(u_{\ell_1}), \ldots, f_{\ell_{n-1}}(u_{\ell_{n-1}})$$

as we will do going forward. In fact the family f_ℓ can be collected into an equivalence of type $\mathscr{E}'(\delta, D) \simeq \mathscr{E}(\delta, D)$ which also restricts to the $< \ell$ case.

Lemma A.10. *Let $\delta : \Delta$, and $h : \forall\kappa.\, H(\delta) \vdash D$ type and let $t : \forall\kappa.\, H\delta$.*

(a) *Elimination under a single clock allows us to produce a term of the type* $\mathsf{Elim}(\delta, D, t) \stackrel{def}{=} \mathscr{E}(\delta, D) \to D[t]$.
(b) *Clock elimination with constant Γ_ℓ parameter allows us to produce a term of the type* $\mathsf{Elim}^{\mathsf{const}}(\delta, D, t) \stackrel{def}{=} \mathscr{E}'(\delta, D) \to D[t]$

Proof. Case (a) is direct from typing of elimination under a single clock, case (b) follows by composing (a) with the equivalence f. □

In the following we will fix $D[t]$ to be $\forall\kappa.t\ [\kappa_0] = t\ [\kappa]$.

Lemma A.11. *For each ℓ we can type b_ℓ as shown:*

$$\begin{aligned}
\mathscr{T}_\ell(\delta) \stackrel{def}{=} \Pi(&\gamma : \Gamma_\ell[\delta]) \\
&(x : \forall\kappa.\, \Xi_\ell[\delta, \gamma] \to H\delta) \\
&(y : \Pi(\xi : \forall\kappa.\, \Xi_\ell[\delta, \gamma]).D[\lambda\kappa.x\ [\kappa](\xi\ [\kappa])]). \\
&(i : \Psi_\ell). \\
&D[\lambda\kappa.\, \mathsf{con}_\ell(\gamma, x\ [\kappa], i)] \\
&\quad [\varphi_\ell \mapsto \lambda\kappa.\, \lambda j.\, e_\ell[\gamma, \lambda\xi.\, y\ (\lambda_.\ \xi)\ [\kappa]\ j, i]]
\end{aligned}$$

$$\delta : \Delta \vdash b_\ell \stackrel{def}{=} \lambda\gamma\ x\ y\ i\ \kappa\ j.\, \mathsf{con}_\ell(\gamma, \lambda\xi.\, y\ (\lambda\_.\ \xi)\ [\kappa]\ j, i) : \mathscr{T}_\ell(\delta)$$

Proof. Follows directly by the typing rule for con_ℓ. □

Note that $\mathscr{T}_\ell(\delta)$ differs from $\mathscr{E}'_\ell(\delta, D, \overline{u_{<l}})$ only in the boundary of the final result. We bridge this gap with lemma A.12.

Lemma A.12. *The terms defined in lemma A.11 satisfy the boundary conditions of lemma A.10 (b) up to path equality, i.e., for each ℓ we have a term* lemD_ℓ *of type*

$$\lambda\kappa.\, \lambda j.\, e_\ell[\gamma, \lambda\xi.\, y\ (\lambda\_.\ \xi)\ [\kappa]\ j, i] = (\!|e_\ell|\!)^{\lambda\_.\,\delta}_{\overline{v_{<\ell}}, x \mapsto y}[\lambda\_.\ \gamma]$$

in the appropriate context, and where each v_ℓ is defined as

$$\begin{aligned}
v_\ell &: \mathscr{E}(\delta, D, \overline{v_{<\ell}}) \\
v_\ell &\stackrel{def}{=} f_\ell(g_\ell(b_\ell)) \\
g_\ell &: \mathscr{T}_\ell(\delta) \to \mathscr{E}'(\delta, D, \overline{v_{<l}}) \\
g_\ell(t) &\stackrel{def}{=} \lambda\gamma.\lambda x.\lambda y.(i.\mathsf{lemD}_\ell)^*(t\ \gamma\ x\ y)
\end{aligned}$$

Proof. To arrive at the desired conclusion we need a generalized version of the lemma which keeps track of how the proof in each case coheres with the earlier stages. We prove the following:

$$\begin{aligned}
\Omega_\ell \stackrel{\text{def}}{=} \delta &: \Delta, \gamma : \Gamma_\ell, x : \forall\kappa.\, \Xi_\ell \to \mathsf{H}\ \delta, \\
&y : \Pi(\xi : \forall\kappa.\, \Xi_\ell).D[\lambda\kappa.\, x\ [\kappa]\ (\xi\ [\kappa])], \\
&y_K : \Pi(\xi : \Xi_\ell).D[\lambda\kappa.\, x\ [\kappa]\ \xi], \\
&\tilde{y} : \Pi(\xi : \Xi_\ell).y_K\ \xi = y(\lambda_.\ \xi), \\
&i : \Psi_\ell, \varphi_\ell
\end{aligned}$$

$$\begin{aligned}
\Omega_\ell, \hat{\Gamma} \vdash\ & \mathsf{lemDg}_\ell(M) \\
&: (\lambda\kappa.\, \lambda j.\, M[\lambda\xi.\, y_K\ \xi\ [\kappa]\ j/x]) \\
&= (\!|M|\!)^{\lambda_.\,\delta}_{\overline{v_{<\ell}}, x \mapsto y, \hat{\gamma}}[\lambda_.\ \gamma/\gamma, \lambda_.\ \hat{\gamma}/\hat{\gamma}]
\end{aligned}$$

lemD_ℓ is then defined as

$$\mathsf{lemDg}_\ell(e_\ell)[\lambda\xi.\, y\ (\lambda\_.\xi)/y_K, \lambda\xi.\, \mathsf{refl}/\tilde{y}]$$

we will write σ for $[\lambda\xi.\, y_K\ \xi\ [\kappa]\ j/x]$ and τ for $[\lambda\_.\ \gamma/\gamma, \lambda\_.\ \hat{\gamma}/\hat{\gamma}]$. The extra y_K and $\tilde{y}$ parameters provide what to do for the case $M = x_j\ \overline{u}$ where we will apply $\tilde{y}$ to $\overline{u}$ to obtain the necessary equality between the applications of y_K and y. This allows us to derive that $\mathsf{lemDg}_\ell(-)$ commutes with substitution in the following way,

$$\mathsf{lemDg}_\ell(M[\bar{t}, \lambda\xi.\, M', \bar{r}]) \equiv \mathsf{lemDg}_\ell(M)[\bar{t}, S, R, R_K, \tilde{R}, \bar{r}]$$

where

$$\begin{aligned}
S &\stackrel{\text{def}}{=} \lambda\kappa.\lambda\xi.\, M'[x\ [\kappa]/x] \\
R &\stackrel{\text{def}}{=} \lambda\xi.\, (\!|M'|\!)^{\lambda_.\,\delta}_{\overline{v_{<\ell}}, x \mapsto y, (\hat{\gamma}, \xi)}\tau \\
R_K &\stackrel{\text{def}}{=} \lambda\xi.\lambda\kappa.\lambda j.\, M'\sigma \\
\tilde{R} &\stackrel{\text{def}}{=} \lambda\xi.\, \mathsf{lemDg}_\ell(M')
\end{aligned}$$

and moreover we have $\mathsf{lemDg}_{\ell'}(M) = \mathsf{lemDg}_\ell(M)$ for $\ell' < \ell$, whenever M only contains constructor nodes with labels smaller than ℓ'. We will use these properties for the constructor case of $\mathsf{lemDg}_\ell(-)$.

Finally, we need to show that $\mathsf{lemDg}_\ell(-)$ preserves judgemental equality in the sense that $M \equiv_b N$ implies

$$\mathsf{lemDg}_\ell(M) \equiv \mathsf{lemDg}_\ell(N)$$

This is needed to show that $\mathsf{lemDg}_\ell(-)$ is well-defined in the case of systems as in the proof of Lemma A.4, and to define $\mathsf{lemDg}_\ell(-)$ in the case of homogeneous compositions.

We induct on the structure of M. In case $M = x\ \overline{u}$, we need to build a path between $\lambda\kappa.\, \lambda j.\, (y_K\ (\overline{u}))\ [\kappa]\ j$ and

$$y(\lambda\kappa.\, \overline{u}[(\lambda\_.\ \gamma)[\overline{\kappa}]/\gamma, (\lambda\_.\ \hat{\gamma})[\overline{\kappa}]/\hat{\gamma}]).$$

The clock applications on the right hand side simplify, making the body of the lambda abstraction constant in κ. The left hand side η-contracts to $y_K\ \overline{u}$, so we can conclude by setting $\mathsf{lemDg}_\ell(x_j\ \overline{u})$ equal to $\tilde{y}\ \overline{u}$.

In the case $M = \mathsf{hcomp}^{j'}\ [\psi \mapsto M']\ M'_0$, we must build a path between $\lambda\kappa.\, \lambda j.\, \mathsf{hcomp}^{j'}\ [\psi \mapsto M'\sigma]\ M'_0\sigma$ and

$$\mathsf{comp}^{j'}_{D[\upsilon\tau\ j']}\ [\psi \mapsto (\!|M'|\!)^{\lambda\_.\,\delta}_{\overline{v_{<\ell}}, x \mapsto y, (\hat{\gamma}, j')}\tau]\ ((\!|M'_0|\!)^{\delta}_{\overline{v_{<\ell}}, x \mapsto y, \hat{\gamma}}\tau).$$

Let p be the path connecting

$$\lambda\kappa.\lambda j.\mathsf{hcomp}^{j'}\ [\psi \mapsto (\!|M'|\!)^{\lambda\_.\,\delta}_{\overline{v_{<\ell}},x\mapsto y,(\hat{y},j')}\tau\ [\kappa]\ j]\ ((\!|M'_0|\!)^{\delta}_{\overline{v_{<\ell}},x\mapsto y,\hat{y}}\tau\ [\kappa]\ j)$$

to the right hand side, obtained from the fact that both terms fill the same open box. We let q be the path connecting the left hand side to $p\,0$, obtained by combining $\mathsf{lemDg}_\ell(M')$ and $\mathsf{lemDg}_\ell(M'_0)$ with $\mathsf{hcomp}^{j'}$. Note that this is well-defined because $\mathsf{lemDg}_\ell(-)$ preserves judgemental equality. By transitivity we get $q \cdot p$ connecting the desired endpoints.

To prove that $\mathsf{lemDg}_\ell(-)$ preserves judgemental equality, we need that $\mathsf{lemDg}_\ell(M)$ when restricted by ψ is strictly equal to $\mathsf{lemDg}_\ell(M')$. Fortunately that is already true for q, while p is a constant path under those conditions, so by the right unit law we have path from $q \cdot p$ to $\mathsf{lemDg}_\ell(M')$. Using an hcomp with this latter path we define $\mathsf{lemDg}_\ell(M)$ so that it satifies the strict equality.

In case $M = \mathsf{con}_{\ell'}(\overline{t}, \lambda\xi.\, M', \overline{r})$, we need to build a path between $\lambda\kappa.\,\lambda j.\,\mathsf{con}_{\ell'}(\overline{t}, \lambda\xi.\, M'\sigma, \overline{r})$ and $v_{\ell'}(\lambda\_.\,\overline{t}, S, R, \overline{r})$ where $S = \lambda\kappa.\lambda\xi.\, M'[x\ [\kappa]/x]$ and $R = \lambda\xi.\,(\!|M'|\!)^{\lambda\_.\,\delta}_{\overline{v_{<\ell}},x\mapsto y,(\hat{y},\xi)}\tau$. We will work right to left by expanding the right hand side definition. By definition we have $v_{\ell'} = f_{\ell'}(g_{\ell'}(b_{\ell'}))$, so $v_{\ell'}(\lambda\_.\,\overline{t})$ is equal to $\mathsf{cirr}_{\Gamma_{\ell'}}(\lambda\_.\,\overline{t})^*(g_{\ell'}(b_{\ell'})(\overline{t}))$, so that by cirr–$\mathsf{coh}_{\Gamma_\ell}(\overline{t})$ the right hand side is equal to $g_{\ell'}(b_{\ell'})(\overline{t}, S, R, \overline{r})$, let us call this path p_1. Note that p_1 will be a constant path when $\varphi_{\ell'}[\overline{r}] = 1_{\mathbb{F}}$ as it will collapse to an equality between the boundaries of its endpoints, specified by their type. By definition we have $g_{\ell'}(b_{\ell'})(\overline{t}, S, R, \overline{r})$ equal to $((i_{\ell'}.\mathsf{lemD}_{\ell'}[\overline{t}, S, R])^*(b_{\ell'}(\overline{t}, S, R)))(\overline{r})$, which is defined as an hcomp and so by filling it is equal to the base of the composition $b_{\ell'}(\overline{t}, S, R, \overline{r})$, let us call this path p_2. Note that when $\varphi_{\ell'}[\overline{r}] = 1_{\mathbb{F}}$ we will have $p_2 = \lambda i_{\ell'}.\,\mathsf{lemD}_{\ell'}[\overline{t}, S, R]$. Finally $b_{\ell'}(\overline{t}, S, R, \overline{r})$ is equal to

$$\lambda\kappa.\lambda j.\,\mathsf{con}_{\ell'}(\overline{t}, \lambda\xi.\, R\,(\lambda\_.\,\xi)\ [\kappa]\ j, \overline{r}),$$

which is path equal to the left hand side by $\mathsf{lemDg}_\ell(M')$ and $\mathsf{con}_{\ell'}$ itself, we call the resulting path p_3. Note that when $\varphi_{\ell'}[\overline{r}] = 1_{\mathbb{F}}$ we will have p_3 built from $e_{\ell'}$ and $\mathsf{lemDg}_\ell(M')$ instead. By transitivity, $p_3 \cdot p_2 \cdot p_1$ forms a path between the left and right hand sides.

To preserve judgemental equality, when $\varphi_{\ell'}[\overline{r}] = 1_{\mathbb{F}}$, the path $\mathsf{lemDg}_\ell(M)$ must be equal to $\mathsf{lemDg}_\ell(e_{\ell'}[\overline{t}, \lambda\xi.\, M', \overline{r}])$ which in turn is equal to $\mathsf{lemDg}_{\ell'}(e_{\ell'})[\overline{t}, S, R, R_K, \tilde{R}, \overline{r}]$ by the commuting with substitution property, and where $R_K = \lambda\xi.\lambda\kappa.\lambda j.\, M'\sigma$ and $\tilde{R}$ is given by $\mathsf{lemDg}_\ell(M')$. To build the necessary path we first contract the singleton pair $(R_K, \tilde{R})$, so that the only non-trivial path in the composition is p_2, which as we noted matches $\mathsf{lemD}_{\ell'}$, i.e.

$$\mathsf{lemDg}_{\ell'}(e_{\ell'})[\overline{t}, S, R, R, \xi.\,\mathsf{refl}, \overline{r}].$$

Using this, we define $\mathsf{lemDg}_\ell(M)$ as an hcomp of the path just defined and $p_3 \cdot p_2 \cdot p_1$. □

Proof of Theorem 5.3. The v_ℓ terms from lemma A.12 collectively form a proof of $\mathscr{C}(\delta, D)$, so by lemma A.10 we conclude $\Pi(t : \forall\kappa.\mathsf{H}\,\delta).D[t] \equiv \Pi(t : \forall\kappa.\mathsf{H}\,\delta).\forall\kappa.t\ [\kappa_0] = t\ [\kappa]$, as required. □

A.5 Composition Structure for Higher Inductive Types (Sect. 5.1)

Following [22], for any HIT $\delta : \Delta \vdash \mathsf{H}\,\delta$ type we define its composition operation, comp, in terms of the trans and hcomp operations, resulting in the following judgemental equality rule

$$\mathsf{comp}^i_{\mathsf{H}\,\delta}\ [\varphi \mapsto u]\ u_0 \equiv \mathsf{hcomp}^i_{\mathsf{H}\,\delta[1/i]}\ [\varphi \mapsto v(i)]\ (\mathsf{trans}^i_{\mathsf{H}\,\delta}\ \varphi\ u_0)$$

where $v(i)$ is $\mathsf{trans}^j_{\mathsf{H}\,\delta[i\vee j/i]}\ (\varphi \vee i = 1)\ (u\ i)$.

Furthermore we include judgemental equalities for trans when applied to elements of the HIT built by homogeneous composition or by constructors. In Section 3.4 of [22], the authors describe how trans computes when applied to constructors specified by a signature of the form

$$\mathsf{c} : (\overline{x} : \overline{A}(\delta))(\overline{i} : \mathbb{I}) \to \mathsf{H}\,\delta[\varphi \mapsto e]$$

where $\overline{A}(\delta)$ is a telescope including both non-recursive and recursive arguments. They are able to treat both kinds of arguments uniformly by working in a variation of cubical type theory where trans and hcomp are the primitive operations for all types while comp is derived, so that they can use $\mathsf{trans}^i_{\overline{A}(\delta)}$ to transport all the arguments at once. We have kept comp as the primitive in our type theory, but we can reuse their description as long as we show how to transport the arguments for the constructors in our schema, i.e., provide a replacement for $\mathsf{trans}^i_{\overline{A}(\delta)}$.

Given a constructor declaration $(\Gamma, \overline{\Xi}, \Psi, \varphi, e)$ and $\delta : \mathbb{I} \to \Delta$ we have to define $\mathsf{trans}^i_{\Gamma[\delta\,i],\Theta_{\overline{\Xi}[\delta\,i],\mathsf{H}\,(\delta\,i)}}\ \psi\ (\overline{t}, \overline{a})$. The $\Gamma[\delta\,i]$ part of the telescope can be dealt with using comp, as shown in [22] by the definition of ctrans: a transport operation derived from composition. We are left with having to define $\mathsf{trans}^i_{\Theta_{\overline{\Xi}[\delta\,i,\tilde{t}[i]],\mathsf{H}\,\delta\,i}}\ \psi\ (\overline{t}, \overline{a})$ where $\tilde{t}$ connects $\overline{t}$ to the result of transporting it. It is then sufficient to show how to transport elements of $C(i) := \Xi_k[\delta\,i, \tilde{t}[i]] \to \mathsf{H}\,\delta\,i$ for each Ξ_k in $\overline{\Xi}$. Here we follow the recipe for transport in function types [31], like so

$$\mathsf{trans}^i_{C(i)}\ \psi\ a_k = \lambda\xi.\mathsf{trans}^i_{\mathsf{H}\,\delta\,i}\ \psi\ (a_k\ (\mathsf{ctrans}^i_{\Xi_k[\delta\,(1-i),\tilde{t}[1-i]]}\ \psi\ \xi))$$

where we make use of the fact that $a_k\ (-)$ is a subtree of $\mathsf{con}(\overline{t}, \overline{a}, \overline{i})$ so it is well-founded to recursively transport. On top of the above, the transport operation commutes with homogeneous composition, as described in Sect. 3.2 of [22].

A.6 Detailed version of Section 6

The standard models of both Cubical Type Theory and Clocked Type Theory are based on presheaf categories. In this section we recall these models and show how to model the combined CCTT in a presheaf category over the product of the categories used in the interpretations of Cubical and

Clocked Type Theory. One of the challenges in constructing the model is to equip the types of Clocked Type Theory (such as $\triangleright\,(\alpha : \kappa).A$) with composition structures as required to model types in Cubical Type Theory.

In this paper, following the convention of Mannaa et al. [36] (but breaking with the convention of Cohen et al. [21]) we will work with *covariant* presheaves. Recall that a covariant presheaf over a category $\mathbb{C}$ is a family of sets $X(c)$ indexed by objects of $\mathbb{C}$ together with a map mapping $f : c \to c'$ in $\mathbb{C}$ and $x \in X(c)$ to $f \cdot x \in X(c')$, respecting identities and composition:

$$\mathsf{id}_c \cdot x = x \qquad\qquad g \cdot (f \cdot x) = (gf) \cdot x \tag{15}$$

We write $\mathsf{PSh}(\mathbb{C})$ for the category of covariant presheaves on $\mathbb{C}$. We recall the notion of Category with Family (CwF) [26] a standard notion of model of dependent type theory.

Definition A.13. A category with family comprises:

1. A category $\mathbb{C}$. We use Γ, Δ to range over objects of $\mathbb{C}$
2. A functor $\mathsf{Fam} : \mathbb{C}^{\mathrm{op}} \to \mathsf{Set}$. Elements $A \in \mathsf{Fam}(\Gamma)$ are referred to as *families* over Γ. If $\sigma : \Delta \to \Gamma$ and $A \in \mathsf{Fam}(\Gamma)$ we write $A[\sigma]$ for $\mathsf{Fam}(\sigma)(A)$
3. A functor El associating to each Γ and each $A \in \mathsf{Fam}(\Gamma)$ a set $\mathsf{El}(A)$ of *elements* of A, and to each $\sigma : \Delta \to \Gamma$ a mapping $(-)[\sigma] : \mathsf{El}(A) \to \mathsf{El}(A[\sigma])$.
4. A *comprehension* operation mapping a family A over Γ to an object $\Gamma.A$ such that maps $\Delta \to \Gamma.A$ correspond bijectively to pair of maps $\sigma : \Delta \to \Gamma$ and elements $t \in A[\sigma]$, naturally in Δ.

We often refer to a CwF simply by the name of the underlying category $\mathbb{C}$. A CwF gives rise to a model of dependent type theory [29] in which contexts are modelled as objects of $\mathbb{C}$, types as families and terms as elements. Recall that any presheaf category is the underlying category of a CwF where families over an object Γ are indexed families of sets $X(c, \gamma)$ for $\gamma \in \Gamma(c)$ with maps $f \cdot (-) : X(c, \gamma) \to X(c', f \cdot \gamma)$ satisfying the equations (15), and elements are assignments mapping each $\gamma \in \Gamma(c)$ to $t(\gamma) \in X(c, \gamma)$ such that $t(f \cdot \gamma) = f \cdot t(\gamma)$.

We recall also the category $\int \Gamma$ of elements for a covariant presheaf Γ over a category $\mathbb{C}$. This has as objects pairs (c, γ) where $\gamma \in \Gamma(c)$ and morphisms from (c, γ) to (c', γ') morphisms $f : c \to c'$ such that $f \cdot \gamma = \gamma'$. Note that a family over Γ is precisely the same as a presheaf over $\int \Gamma$. We will also use the equivalence

$$\mathsf{PSh}(\mathbb{C})/\Gamma \simeq \mathsf{PSh}(\textstyle\int \Gamma) \tag{16}$$

which states that a slice of a presheaf category is itself a presheaf category.

A.6.1 Modelling Cubical Type Theory.

The standard model of Cubical Type Theory [21] uses presheaves over the *category of cubes*. Here, since we use covariant presheaves, we will write $\mathscr{C}$ for the opposite of the category used by Cohen et al. [21]. So $\mathscr{C}$ has as objects finite sets I, J, K and as morphisms from I to J maps $f : I \to \mathsf{dM}(J)$ where $\mathsf{dM}(J)$ is the free de Morgan algebra on the set J. Composition is the standard Kleisli composition, using the fact that $\mathsf{dM}(-)$ is a monad. From the model construction we recall in particular the interval object $\mathbb{I}(I) \stackrel{\mathrm{def}}{=} \mathsf{dM}(I)$, representing the singleton set, and the face lattice $\mathbb{F}$, which at stage I is the free distributive lattice generated by elements $(i = 0)$ and $(i = 1)$ for each $i \in I$, and relations $(i = 0) \wedge (i = 1) = \bot_{\mathbb{F}}$.

This model construction can be extended to work for any category of the form $\mathsf{PSh}(\mathscr{C} \times \mathbb{D})$. To do so we rely on the Orton and Pitts' axiomatisation of models of CTT [35, 44]. Rather than recalling these axioms, we recall the sufficient conditions summarised by Coquand et al. [24] for a presheaf topos to satisfy these axioms. These conditions are as follows.

- The interval object $\mathbb{I}$ is connected, and a bounded distributive algebra structure with distinct 0 and 1 elements. Exponentiation by $\mathbb{I}$ has a right adjoint.
- The canonical map from $\mathbb{F}$ to the subobject classifier is a monomorphism, and the universal cofibration $\top : 1 \to \mathbb{F}$ is a levelwise decidable inclusion. Monomorphisms that can be described as pullbacks of the latter are referred to as *cofibrations*. The interval endpoint inclusions $0, 1 : 1 \to \mathbb{I}$ must be cofibrations, and cofibrations must be closed under finite union (finite disjunction), composition (dependent conjunction), and universal quantification over $\mathbb{I}$.

A *cubical model* is a presheaf category satisfying the axioms above. In $\mathsf{PSh}(\mathscr{C} \times \mathbb{D})$, defining both objects as constant on the $\mathbb{D}$ component, using the corresponding objects in $\mathsf{PSh}(\mathscr{C})$

$$\mathbb{I}(I, d) \stackrel{\mathrm{def}}{=} \mathbb{I}(I) \qquad\qquad \mathbb{F}(I, d) \stackrel{\mathrm{def}}{=} \mathbb{F}(I)$$

make $\mathsf{PSh}(\mathscr{C} \times \mathbb{D})$ a cubical model, as also observed by Coquand et al. [24].

Given such a model, Orton and Pitts [44] express the structure sufficient to model CTT using the internal language of the presheaf topos as an extensional type theory. We recall here some definitions that will be relevant later. We assume a $\omega + 1$ long hierarchy of Grothendieck universes, leading to a corresponding hierarchy of universes a la Russell U_i in the internal language, each classifying presheaves of the appropriate size.

Definition A.14. A *CCHM fibration* (A, α) over a type $\Gamma : \mathsf{U}_\omega$ is a family $A : \Gamma \to \mathsf{U}_\omega$ with a fibration structure $\alpha : \mathsf{isFib}\,\Gamma\,A$ where

$$\mathsf{isFib}\,\Gamma\,A \stackrel{\mathrm{def}}{=} (e : \{0, 1\})(p : \mathbb{I} \to \Gamma) \to \mathsf{Comp}\,e\,(A \circ p)$$

$$\begin{aligned}\mathsf{Comp}\,e\,A \stackrel{\mathrm{def}}{=} (\varphi : \mathbb{F})(u : [\varphi] \to (i : \mathbb{I}) \to A\,i)\\ \to \{u_0 : A\,e \mid \varphi \Rightarrow u\,e = u_0\}\\ \to \{u_1 : A\,\overline{e} \mid \varphi \Rightarrow u\,\overline{e} = u_1\}\end{aligned}$$

where $\overline{e}$ sends $0 : \mathbb{I}$ to 1 and vice versa.

Notice Comp 0 A closely matches the signature of the composition operation from CTT.

We can then build the following Category with Families of fibrant types, which we call Fib:

- A context Γ is a global element of U_ω.
- A family over Γ is a global CCHM fibration over Γ.
- An element t of a family (A, α) over Γ is a global element of $(\gamma : \Gamma) \to A\,\gamma$.

it then follows that Fib models the type formers of Cubical Type Theory.

Remark 2. *Cubical Type Theory as presented by Cohen et al. [21] includes specific judgemental equalities for the composition operator* comp^i_A *according to the shape of A, matching the behaviour of the given fibration structures in their model. The constructions by Orton and Pitts [44] do not necessarily produce fibration structures that satisfy the same equalities. One way to deal with this is to use alternative formulations of these rules [31, 54]. Since these equalities are mainly relevant for operational properties of CTT, we will ignore this issue in this paper.*

A.6.2 Modelling Clocked Cubical Type Theory. In the previous section we saw how to model Cubical Type Theory in any category of the form $\mathsf{PSh}(\mathscr{C} \times \mathbb{D})$. We now define the category $\mathscr{T}$ of *time objects* and extend the model to a model of CCTT in $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$. The objects of $\mathscr{T}$ are pairs $(\mathscr{E}; \delta)$, where $\mathscr{E}$ is a finite set (to be though of as a set of semantic clocks), and $\delta : \mathscr{E} \to \mathbb{N}$ is a map associating to each clock a finite amount of time steps that are left on that clock. A morphism $\sigma : (\mathscr{E}; \delta) \to (\mathscr{E}'; \delta')$ is a map $\sigma : \mathscr{E} \to \mathscr{E}'$ such that $\delta'\sigma \leq \delta$ in the pointwise order. This can be understood as a generalisation of the topos of trees model $\mathsf{PSh}(\omega^{\mathrm{op}})$ of guarded recursion [10], where the indexing category is restricted to objects where the first component $\mathscr{E}$ is a singleton. The category $\mathsf{PSh}(\mathscr{T})$ has previously been used to model type theories with guarded recursion and multiple clocks [12, 36], and (in a slight variation) Guarded Computational Type Theory [49].

The category $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$ has an object of clocks Clk defined as $\mathsf{Clk}(I, (\mathscr{E}; \delta)) = \mathscr{E}$. This can be used to model assumptions of the form κ : clock in contexts, and the types $\forall\kappa.A$ can be modelled as Π-types. To model $\triangleright$, recall that this is a Fitch-style modal type operator and that these can be modelled using dependent right adjoint types [19].

Definition A.15. Let $\mathbb{C}, \mathbb{D}$ be CwFs and let $\mathsf{L} : \mathbb{C} \to \mathbb{D}$ be a functor between the underlying categories. A dependent right adjoint to L is a mapping associating to a family A over $\mathsf{L}\Gamma$ a family $\mathsf{R}A$ over Γ and a bijective correspondence between elements of A and elements of $\mathsf{R}A$, both natural in Γ.

The naturality requirement for R means that if $\gamma : \Gamma' \to \Gamma$ and A is a family over $\mathsf{L}\Gamma$, then $(\mathsf{R}A)[\sigma] = \mathsf{R}(A[\mathsf{L}\sigma])$. Writing $\overline{(-)}$ for both directions of the bijective correspondence on elements, the naturality condition for this means that if t is an element of A over $\mathsf{L}\Gamma$, then $\overline{t}[\gamma] = \overline{t[\mathsf{L}\gamma]}$, and similarly for the opposite direction.

Given an endofunctor L on a CwF $\mathbb{C}$ as well as a dependent right adjoint R to L, one can model a Fitch-style modal operator by modelling extensions of contexts with ticks by L and the modal operator by R. In the case of Clocked Type Theory, the ticks, as well as the modal operators are indexed over an object Clk in the model: Semantically, the hypothesis $\Gamma \vdash \kappa$: clock of the context extension rule for $\Gamma, \alpha : \kappa \vdash$ corresponds to an element of the slice category over Clk. By (16) the slice category over Clk is itself a presheaf category and therefore carries a natural CwF structure. In fact, if $\chi : \Gamma \to \mathsf{Clk}$ is an object in the slice category, then families over χ in the slice category correspond bijectively to families in $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})$ over Γ. Exploiting this, Mannaa et al. [36] describe how to model ticks on clocks using a dependent right adjoint $\blacktriangleright$ to an endofunctor $\blacktriangleleft$ on the slice category over Clk. In this model $\Gamma, \alpha : \kappa \vdash$ is interpreted as the domain of $\blacktriangleleft(\llbracket\Gamma\rrbracket, \llbracket\kappa\rrbracket)$ and $\triangleright\,(\alpha : \kappa).A$ as the dependent right adjoint $\blacktriangleright$ applied to $\llbracket A\rrbracket$. The bijective correspondence on elements then models the bijective correspondence between terms $\Gamma, \alpha : \kappa \vdash t : A$ and terms $\Gamma \vdash u : \triangleright\,(\alpha : \kappa).A$ given by tick abstraction and application. Tick weakening, which syntactically corresponds to context projections $\Gamma, \alpha : \kappa \to \Gamma$ can be modelled using a natural transformation from $\blacktriangleleft$ to the identity.

A.6.3 Composition structure for dependent right adjoints. Before recalling the dependent right adjoint structure on the slice category $\mathsf{PSh}(\mathscr{C} \times \mathscr{T})/\mathsf{Clk}$ we now give general conditions ensuring that dependent right adjoint types on a cubical model carry composition structure. The conditions are all on the left adjoint functor.

Definition A.16. Let $\mathbb{C}$ be a cubical model with interval object $\mathbb{I}$ and let $\mathsf{L} : \mathbb{C} \to \mathbb{C}$ be a finite product preserving functor.

1. We say L preserves the interval if there is an isomorphism $\mathsf{L}(\mathbb{I}) \cong \mathbb{I}$ preserving endpoints, i.e., such that the following commutes for $e = 0, 1$.

$$\begin{array}{ccc} 1 & \xrightarrow{e} & \mathbb{I} \\ \downarrow\cong & & \downarrow\cong \\ \mathsf{L}1 & \xrightarrow{\mathsf{L}e} & \mathsf{L}\mathbb{I} \end{array}$$

2. We say L *preserves cofibrations* if $\mathsf{L}i : \mathsf{L}A \to \mathsf{L}B$ is a cofibration whenever $i : A \to B$ is, and if the diagram on the right below is a pullback whenever i is a cofibration and the diagram on the left is a pullback

$$\begin{array}{ccc} C & \xrightarrow{a} & A \\ \downarrow j & & \downarrow i \\ D & \xrightarrow{b} & B \end{array} \qquad \begin{array}{ccc} \mathsf{L}C & \xrightarrow{\mathsf{L}a} & \mathsf{L}A \\ \downarrow \mathsf{L}j & & \downarrow \mathsf{L}i \\ \mathsf{L}D & \xrightarrow{\mathsf{L}b} & \mathsf{L}B \end{array}$$

The condition of preserving cofibrations corresponds to giving an operation mapping cofibrations $\Gamma \vdash \varphi : \mathbb{F}$ on Γ to cofibrations $\mathsf{L}\Gamma \vdash \mathsf{L}_{\mathbb{F}}(\varphi) : \mathbb{F}$ such that $\mathsf{L}\Gamma.[\mathsf{L}_{\mathbb{F}}(\varphi)] \cong \mathsf{L}(\Gamma.[\varphi])$ as subobjects of $\mathsf{L}\Gamma$ and satisfying $\mathsf{L}_{\mathbb{F}}(\varphi[\sigma]) = \mathsf{L}_{\mathbb{F}}(\varphi)[\mathsf{L}\sigma]$. Here $[\varphi]$ is the family classified by φ.

Theorem A.17. *Let $\mathbb{C}$ be a cubical model, let $\mathsf{L} : \mathbb{C} \to \mathbb{C}$ be a functor preserving finite products, the interval and cofibrations, and let R be a dependent right adjoint to L. If a family A over $\mathsf{L}\Gamma$ carries a global composition structure, so does $\mathsf{R}A$ over Γ. Moreover, this assignment is natural in Γ.*

Note that this is a statement about *global* composition structures in the model. The theorem can not be proved in the internal logic of the topos, but can be proved in an extension of this using crisp type theory, similarly to the construction of universes for cubical type theory in crisp type theory [35]. The reason is that the proof uses the bijective correspondence of Definition A.15 which only applies to global terms.

To prove Theorem A.17, we need the following lemma giving an alternative description of the data of a composition structure. The lemma uses standard CwF notation writing $\mathsf{p} : \Delta.A \to \Delta$ for the projection out of a comprehension object, and (less standard) notation $\sigma.A : \Delta.A[\sigma] \to \Gamma.A$ if $\sigma : \Delta \to \Gamma$ for the functoriality of comprehension in the first component defined in the language of CwFs as $(\sigma\mathsf{p}, \mathsf{q})$.

Lemma A.18. *Let Γ be a global element of U_ω and let $A : \Gamma \to \mathsf{U}_\omega$. To give a composition structure on A corresponds to giving, for each global element Δ of U_ω and map $\sigma : \Delta \times \mathbb{I} \to \Gamma$, an assignment expressed as the rule*

$$\frac{\Delta \vdash \varphi : \mathbb{F} \qquad \Delta.\mathbb{I}.[\varphi[\mathsf{p}]] \vdash u : A[\sigma\mathsf{p}] \qquad \Delta \vdash u_e : A[\sigma \circ (\mathsf{id}, e)] \qquad \Delta.[\varphi] \vdash u_e[\mathsf{p}] = u[(\mathsf{id}_\Delta, e).[\varphi]]}{\Delta \vdash c_\sigma\, \varphi\, u\, u_e : A[\sigma \circ (\mathsf{id}, 1-e)]}$$

for each $e \in \{0, 1\}$, natural in Δ satisfying

$$\Delta.[\varphi] \vdash (c_\sigma\, \varphi\, u\, u_e)[\mathsf{p}] = u[(\mathsf{id}, 1-e).[\varphi]]$$

Proof. Given e, to give the part of isFib corresponding to e corresponds to giving $p : \mathbb{I} \to \Gamma \vdash c : \mathsf{Comp}\, e\, (A \circ p)$ in the model. This in turn corresponds to an assignment of morphisms $\tau : \Delta \to (\mathbb{I} \to \Gamma)$ to terms $\delta : \Delta \vdash c_\tau : \mathsf{Comp}\, e\, (A \circ \tau(\delta))$ natural in Δ. By uncurrying, the latter corresponds to an assignment mapping $\sigma : \Delta \times \mathbb{I} \to \Gamma$ to $\delta : \Delta \vdash c_\sigma : \mathsf{Comp}\, e\, (A \circ \sigma(\delta, -))$, also natural in Δ. By further uncurrying the arguments to the composition operator and using similar naturality arguments in each case, we arrive at the description in the lemma. □

Proof of Theorem A.17. By Lemma A.18 it suffices to give an assignment mapping $\sigma : \Delta \times \mathbb{I} \to \Gamma$ to a rule

$$\frac{\Delta \vdash \varphi : \mathbb{F} \qquad \Delta.\mathbb{I}.[\varphi[\mathsf{p}]] \vdash u : (\mathsf{R}A)[\sigma\mathsf{p}] \qquad \Delta \vdash u_e : (\mathsf{R}A)[\sigma \circ (\mathsf{id}, e)] \qquad \Delta.[\varphi] \vdash u_e[\mathsf{p}] = u[(\mathsf{id}_\Delta, e).[\varphi]]}{\Delta \vdash c^{\mathsf{R}A}_\sigma\, \varphi\, u\, u_e : (\mathsf{R}A)[\sigma \circ (\mathsf{id}, 1-e)]}$$

natural in Δ and satisfying the equality of Lemma A.18. Using $(\mathsf{R}A)[\sigma\mathsf{p}] = \mathsf{R}(A[\mathsf{L}(\sigma\mathsf{p})])$ the assumptions correspond to

$$\mathsf{L}(\Delta.\mathbb{I}.[\varphi[\mathsf{p}]]) \vdash \overline{u} : A[\mathsf{L}(\sigma\mathsf{p})] \qquad \mathsf{L}\Delta \vdash \overline{u_e} : A[\mathsf{L}(\sigma \circ (\mathsf{id}, e))]$$

satisfying

$$\mathsf{L}(\Delta.[\varphi]) \vdash \overline{u_e}[\mathsf{Lp}] = \overline{u}[\mathsf{L}((\mathsf{id}_\Delta, e).[\varphi])] \tag{17}$$

Since L preserves cofibrations, by the notation introduced after Definition A.16, $\mathsf{L}(\Delta.\mathbb{I}.[\varphi[\mathsf{p}]] \cong \mathsf{L}(\Delta.\mathbb{I}).[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])]$ as subobjects of $\mathsf{L}(\Delta.\mathbb{I})$. For simplicity we will leave this isomorphism implicit. Moreover, since L preserves finite products and the interval, there is an isomorphism $\xi_\Delta : \mathsf{L}\Delta.\mathbb{I} \cong \mathsf{L}(\Delta.\mathbb{I})$ natural in Δ such that

$$\begin{array}{ccc} & \mathsf{L}\Delta & \\ {\scriptstyle (\mathsf{id},e)}\swarrow & & \searrow{\scriptstyle \mathsf{L}(\mathsf{id},e)} \\ \mathsf{L}\Delta.\mathbb{I} & \xrightarrow{\xi_\Delta} & \mathsf{L}(\Delta.\mathbb{I}) \end{array} \tag{18}$$

Then, since $\mathsf{L}(\sigma\mathsf{p}) \circ \xi_\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])] = \mathsf{L}(\sigma) \circ \xi_\Delta \circ \mathsf{p}$

$$\mathsf{L}(\Delta).\mathbb{I}.[\mathsf{L}_{\mathbb{F}}(\varphi)[\mathsf{p}]] \vdash \overline{u}[\xi_\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])]] : A[\mathsf{L}(\sigma) \circ \xi_\Delta][\mathsf{p}]$$

and by (18)

$$\mathsf{L}\Delta \vdash \overline{u_e} : A[\mathsf{L}(\sigma) \circ \xi_\Delta][(\mathsf{id}, e)]$$

We can therefore apply the composition structure for A as in Lemma A.18 in the case of $L(\sigma) \circ \xi_\Delta : \mathsf{L}\Delta.\mathbb{I} \to \mathsf{L}\Gamma$, the cofibration $\mathsf{L}\Delta \vdash \mathsf{L}_{\mathbb{F}}(\varphi) : \mathbb{F}$, and the terms $\overline{u}[\xi_\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])]]$ and $\overline{u_e}$, if only we can prove that

$$\mathsf{L}\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi)] \vdash \overline{u_e}[\mathsf{p}] = \overline{u}[\xi_\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])]][(\mathsf{id}_{\mathsf{L}\Delta}, e).[\mathsf{L}_{\mathbb{F}}(\varphi)]]$$

Up to the isomorphism $\mathsf{L}\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi)] \cong \mathsf{L}(\Delta.[\varphi])$ the context projection p equals Lp, and so in context $\mathsf{L}(\Delta.[\varphi])$ the left hand side reduces to $\overline{u_e}[\mathsf{Lp}]$. The right hand side reduces using (18) to

$$\overline{u}[\mathsf{L}(\mathsf{id}_\Delta, e).\mathsf{L}_{\mathbb{F}}(\varphi)]$$

which up to the isomorphism $\mathsf{L}\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi)] \cong \mathsf{L}(\Delta.[\varphi])$ equals $\overline{u}[\mathsf{L}((\mathsf{id}_\Delta, e).[\varphi])]$, and so the required equality follows from (17). The composition structure for A therefore gives

$$\mathsf{L}\Delta \vdash c^A_{L(\sigma)\circ\xi_\Delta}\, \mathsf{L}_{\mathbb{F}}(\varphi)\, \overline{u}[\dots]\, \overline{u_e} : A[L(\sigma) \circ \xi_\Delta \circ (\mathsf{id}, 1-e)]$$

which, since $A[L(\sigma) \circ \xi_\Delta \circ (\mathsf{id}, 1-e)] = A[L(\sigma \circ (\mathsf{id}, 1-e))]$, corresponds to a term

$$\Delta \vdash c^{\mathsf{R}A}_\sigma\, \varphi\, u\, u_e : (\mathsf{R}A)[\sigma \circ (\mathsf{id}, 1-e)]$$

To show the equality

$$\Delta.\varphi \vdash (c^{\mathsf{R}A}_\sigma\, \varphi\, u\, u_e)[\mathsf{p}] = u[(\mathsf{id}, 1-e).[\varphi]]$$

is equivalent to showing

$$\mathsf{L}(\Delta.\varphi) \vdash \overline{(c^{\mathsf{R}A}_\sigma\, \varphi\, u\, u_e)[\mathsf{p}]} = \overline{u[(\mathsf{id}, 1-e).[\varphi]]}$$

which up to the isomorphism $\mathsf{L}(\Delta.\varphi) \cong \mathsf{L}\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi)]$ corresponds to showing that the term

$$\mathsf{L}\Delta.\mathsf{L}_{\mathbb{F}}(\varphi) \vdash (c^A_{L(\sigma)\circ\xi_\Delta}\, \mathsf{L}_{\mathbb{F}}(\varphi)\, \overline{u}[\xi_\Delta.[\mathsf{L}_{\mathbb{F}}(\varphi[\mathsf{p}])]]\, \overline{u_e})[\mathsf{p}] \tag{19}$$

equals

$$\mathsf{L}\Delta.\mathsf{L}_{\mathbb{F}}(\varphi) \vdash \overline{u}[\mathsf{L}((\mathsf{id}, 1-e)).[\mathsf{L}_{\mathbb{F}}(\varphi)]] \tag{20}$$

By the equality rule for the composition structure on A, (19) equals

$$\overline{u}[\xi_\Delta.[\mathsf{L}_\mathbb{F}(\varphi[\mathsf{p}])]][(\mathsf{id}, 1-e).[\mathsf{L}_\mathbb{F}(\varphi)]]$$

which equals (20) by (18). □

A.6.4 The dependent right adjoint.

We now specialise to the particular cubical model structure on $\mathsf{PSh}(\mathscr{C}\times\mathscr{T})$. The only assumption we will need for this is that $\mathbb{I}$ and $\mathbb{F}$ are included from $\mathsf{PSh}(\mathscr{C})$ as in the cubical model structures used by Coquand et al. [24].

We first recall the structure of the dependent right adjoint in details. Mannaa et al. [36] define this structure for $\mathsf{PSh}(\mathscr{T})$ but the constructions carry over directly to $\mathsf{PSh}(\mathscr{C}\times\mathscr{T})$. The structure arises as an extension of an endo-adjunction on the slice category as in the following lemma slightly generalised from Clouston et al. [19], to which we refer for details.

Lemma A.19. *Let $\mathbb{C}$ and $\mathbb{D}$ be CwFs and let $\mathsf{L}:\mathbb{C}\to\mathbb{D}$ be a functor between the underlying categories with a right adjoint R. Suppose R extends to families and elements as in the following data*

1. *An operation mapping families A over Γ in $\mathbb{D}$ to families $\mathsf{R}_{\mathsf{Fam}}(A)$ over $\mathsf{R}\Gamma$ satisfying $\mathsf{R}_{\mathsf{Fam}}(A[\gamma]) = (\mathsf{R}_{\mathsf{Fam}}(A))[R\gamma]$*
2. *An operation mapping elements t of A to elements $\mathsf{R}_{\mathsf{El}}(t)$ of $\mathsf{R}_{\mathsf{Fam}}(A)$ satisfying $\mathsf{R}_{\mathsf{El}}(t[\gamma]) = (\mathsf{R}_{\mathsf{El}}(t))[R\gamma]$.*

Then L has a dependent right adjoint mapping families A over $\mathsf{L}\Gamma$ to $\mathsf{R}A = (\mathsf{R}_{\mathsf{Fam}}A)[\eta]$ where $\eta:\Gamma\to\mathsf{RL}\Gamma$ is the unit of the adjunction.

The endoadjunction on the slice category is best described by using the equivalent description of the slice category as $\mathsf{PSh}(\int \mathrm{Clk})$. The right adjoint is the simplest to describe and is similar to the functor $\blacktriangleright$ on the topos-of-trees [10]:

$$\blacktriangleright\Gamma(I,(\mathscr{E};\delta),\lambda) = \begin{cases}\Gamma(I,(\mathscr{E};\delta[\lambda\mapsto n]),\lambda) & \text{if } \delta(\lambda)=n{+}1\\ 1 & \text{if } \delta(\lambda)=0\end{cases}$$

Here $\delta[\lambda\mapsto n](\lambda) = n$ and $\delta[\lambda\mapsto n](\lambda') = \delta(\lambda')$ for $\lambda'\neq\lambda$ and 1 in the second clause is a singleton set. This lifts to families and elements in the sense of Lemma A.19, for example, if A is a family over Γ then

$$\blacktriangleright_{\mathsf{Fam}}(A)(I,(\mathscr{E};\delta),\lambda)(\gamma) = \begin{cases}A(\gamma) & \text{if } \delta(\lambda)=n{+}1\\ 1 & \text{if } \delta(\lambda)=0\end{cases}$$

The left adjoint can be concretely described as $\blacktriangleleft\Gamma(I,(\mathscr{E};\delta),\lambda)$ having as elements pairs (σ,x) such that $(\mathsf{id}_I,\sigma):(I,(\mathscr{E}';\delta'),\lambda')\to(I,(\mathscr{E};\delta),\lambda)$ with $\delta'(\lambda')>\delta(\lambda)$ and $x\in\Gamma(I,(\mathscr{E}';\delta'),\lambda')$ considered up to the equivalence relation $\sim$ generated by $(\sigma\tau,x)\sim(\sigma,(\mathsf{id},\tau)\cdot x)$. We refer to Mannaa et al. [36] for the details of the adjunction structure as well as an abstract description of the left adjoint.

There is a natural transformation $\mathsf{p}_\blacktriangleleft:\blacktriangleleft\to\mathsf{id}$ defined as $\mathsf{p}_\blacktriangleleft(\sigma,x) = (\mathsf{id},\sigma)\cdot x$. This is used in our model to interpret tick-weakening.

Theorem A.20. *Let Γ be an object of $\mathsf{PSh}(\mathscr{C}\times\mathscr{T})$ and $\kappa:\Gamma\to\mathrm{Clk}$. Suppose A is a family over the domain of $\blacktriangleleft(\Gamma,\kappa)$ which carries a composition structure, then $\blacktriangleright A$ carries a composition structure as a family over Γ.*

Lemma A.21. *The category $\int\mathrm{Clk}$ for Clk considered an object in $\mathsf{PSh}(\mathscr{T})$ has coproducts.*

Proof. The coproduct of $((\mathscr{E};\delta),\lambda)$ and $((\mathscr{E}';\delta'),\lambda')$ is the object $((\mathscr{E}'';\delta''),\lambda'')$ where λ'' is fresh, $\mathscr{E}''$ is the disjoint union of $\mathscr{E}\setminus\{\lambda\}$ and $\mathscr{E}'\setminus\{\lambda'\}$ and $\{\lambda''\}$ and δ'' agrees with δ on $\mathscr{E}\setminus\{\lambda\}$, with δ' on $\mathscr{E}'\setminus\{\lambda'\}$ and maps λ'' to the minimum of $\delta(\lambda)$ and $\delta'(\lambda')$. □

Proof of Theorem A.20. Recall that families and terms in the CwF of the slice category $\mathsf{PSh}(\int\mathrm{Clk})$ over an object corresponding to an element $\kappa:\Gamma\to\mathrm{Clk}$ correspond bijectively to families and elements of the CwF of $\mathsf{PSh}(\mathscr{C}\times\mathscr{T})$ over Γ. Since this correspondence also respects the interpretation of the internal dependent type theory, it suffices to show that $\blacktriangleright A$ carries a composition structure as expressed on the slice category, when A does. By Theorem A.17 this reduces to showing that $\blacktriangleleft$ preserves finite products, the interval and cofibrations.

Each component $\blacktriangleleft 1(I,(\mathscr{E};\delta),\lambda)$ of $\blacktriangleleft 1$ is easily seen to be inhabited. If $(\sigma,\star)$ and $(\tau,\star)$ are two elements of $\blacktriangleleft 1(I,(\mathscr{E};\delta),\lambda)$, then both of these are related under the equivalence relation used in the definition of $\blacktriangleleft$ to $([\sigma,\tau],\star)$, where $[\sigma,\tau]$ is the copairing of σ and τ out of the coproduct of their domains, which exists by Lemma A.21. So $\blacktriangleleft$ preserves the terminal object.

Writing $[(\sigma,(x,y))]$ for the equivalence class represented by $(\sigma,(x,y))$, the map

$$\blacktriangleleft(A\times B)(I,(\mathscr{E};\delta),\lambda)\to(\blacktriangleleft A\times\blacktriangleleft B)(I,(\mathscr{E};\delta),\lambda)$$

maps $[(\sigma,(x,y))]$ to $([(\sigma,x)],[(\sigma,y)])$, and the inverse maps $([(\sigma,x)],[(\tau,y)])$ to $[([\sigma,\tau],(\mathsf{inl}(x),\mathsf{inr}(y)))]$.

For the interval, the map $\mathsf{p}_\blacktriangleleft:\blacktriangleleft\mathbb{I}\to\mathbb{I}$, which (as decribed in the main text) is defined as $\mathsf{p}_\blacktriangleleft(\sigma,x) = (\mathsf{id},\sigma)\cdot x$ has an inverse which at $(I,(\mathscr{E};\delta),\lambda)$ maps $x\in\mathbb{I}(I)$ to (σ,x) where $\sigma:(\mathscr{E};\delta')\to(\mathscr{E};\delta)$ is tracked by the identity and δ' agrees with δ everywhere except at λ where it is one higher. This clearly preserves endpoints.

For cofibrations, we first show that $\blacktriangleleft$ preserves pullbacks of cofibrations. Suppose that the diagram on the left below is a pullback with i and j cofibrations.

$$\begin{array}{ccc} C & \xrightarrow{a} & A\\ {\scriptstyle j}\downarrow & & \downarrow{\scriptstyle i}\\ D & \xrightarrow{b} & B\end{array}\qquad\qquad \begin{array}{ccc} \blacktriangleleft C & \xrightarrow{\blacktriangleleft a} & \blacktriangleleft A\\ {\scriptstyle \blacktriangleleft j}\downarrow & & \downarrow{\scriptstyle \blacktriangleleft i}\\ \blacktriangleleft D & \xrightarrow{\blacktriangleleft b} & \blacktriangleleft B\end{array}$$

We must show that also the diagram in the right is a pullback. Since $\mathbb{F}$ is constant in the time dimension, it follows, since i

is a fibration that any naturality square of the form

$$\begin{array}{ccc} A(I,(\mathscr{E};\delta),\lambda) & \xrightarrow{(\mathrm{id},\sigma)\cdot(-)} & A(I,(\mathscr{E}';\delta'),\lambda') \\ \downarrow i & & \downarrow i \\ B(I,(\mathscr{E};\delta),\lambda) & \xrightarrow{(\mathrm{id},\sigma)\cdot(-)} & B(I,(\mathscr{E}';\delta'),\lambda') \end{array}$$

is a pullback. From this it follows that the square on the right below is a pullback diagram.

$$\begin{array}{ccccc} \blacktriangleleft C & \xrightarrow{\blacktriangleleft a} & \blacktriangleleft A & \xrightarrow{\mathrm{p}_\blacktriangleleft} & A \\ \downarrow \blacktriangleleft j & & \downarrow \blacktriangleleft i & & \downarrow i \\ \blacktriangleleft D & \xrightarrow{\blacktriangleleft b} & \blacktriangleleft B & \xrightarrow{\mathrm{p}_\blacktriangleleft} & B \end{array}$$

By the pullback pasting diagram it therefore suffices to show that the outer diagram is a pullback, which follows again by the pullback pasting diagram applied to

$$\begin{array}{ccccc} \blacktriangleleft C & \xrightarrow{\mathrm{p}_\blacktriangleleft} & C & \xrightarrow{a} & A \\ \downarrow \blacktriangleleft j & & \downarrow j & & \downarrow i \\ \blacktriangleleft D & \xrightarrow{\mathrm{p}_\blacktriangleleft} & D & \xrightarrow{b} & B \end{array}$$

and naturality of $\mathrm{p}_\blacktriangleleft$.

Suppose now that $\chi_A : B \to \mathbb{F}$ classifies the cofibration $i : A \to B$. A similar argument to the one above for $\mathbb{I}$ shows that $\mathrm{p}_\blacktriangleleft : \blacktriangleleft\mathbb{F} \to \mathbb{F}$ is an isomorphism, which preserves truth. Then $\mathrm{p}_\blacktriangleleft \circ \blacktriangleleft(\chi_A)$ classifies $\blacktriangleleft i$, so also $\blacktriangleleft i$ is a cofibration. □

A.6.5 Interpreting ticks and tick application. A simple tick judgement $\Gamma \vdash \kappa : u \rightsquigarrow \Gamma'$ is interpreted as a map in $\mathsf{PSh}(\int \mathrm{Clk})$ from Γ to $\blacktriangleleft\Gamma'$. Here, for simplicity, we keep the clock κ implicit. An element t of the family $\blacktriangleright A$ over Γ' corresponds bijectively to an element $\bar{t}$ of A over $\blacktriangleleft\Gamma'$, which can then be reindexed along the tick u to interpret tick application. A forcing tick judgement $\Gamma \vdash (\kappa, u) \rightsquigarrow \Gamma'$ is interpreted as a map from Γ to $\blacktriangleleft(\Gamma'.\mathrm{Clk})$ in $\mathsf{PSh}(\int \mathrm{Clk})$, where the domain is considered an element of the slice category over Clk with map given by κ, and $\Gamma'.\mathrm{Clk}$ an object with map given by the second projection. The assumption of the rule for forcing tick application corresponds semantically to an element of a family $\blacktriangleright A$ over an object $\Gamma'.\mathrm{Clk}$, which corresponds bijectively to an element of A over $\blacktriangleleft(\Gamma'.\mathrm{Clk})$. Tick application is then interpreted by reindexing this element along the interpretation of the forcing tick.

To define the interpretation of ticks, say that family A over an object Γ is *timeless* if the map

$$(\blacktriangleleft(\mathrm{p}), \mathrm{q}[\mathrm{p}_\blacktriangleleft]) : \blacktriangleleft(\Gamma.A) \to (\blacktriangleleft\Gamma).A[\mathrm{p}_\blacktriangleleft]$$

is an isomorphism.

Lemma A.22. *The families* $\mathbb{I}$ *and* Clk*, as well as any cofibration, are timeless. As a consequence*

$$\blacktriangleleft([\![\Gamma, \mathsf{TL}(\Gamma')]\!]) \cong \blacktriangleleft([\![\Gamma]\!]).[\![\mathsf{TL}(\Gamma')]\!]$$

in $\mathsf{PSh}(\int \mathrm{Clk})$.

Proof. As noted in the proof of Theorem A.20, cofibrations A over Γ are constant in the time dimension in the sense that for any $\sigma : ((\mathscr{E};\delta),\lambda) \to ((\mathscr{E}';\delta'),\lambda')$ the map

$$(\mathrm{id},\sigma)\cdot(-) : A(\gamma) \to A((\mathrm{id},\sigma)\cdot\gamma)$$

is an isomorphism. This means that the inverse to the map of the lemma can be defined to map the element $((\sigma,\gamma),a) \in ((\blacktriangleleft\Gamma).A[\mathrm{p}_\blacktriangleleft])(I,(\mathscr{E};\delta),\lambda)$ to $[(\sigma,(\gamma,a'))] \in \blacktriangleleft(\Gamma.A)(I,(\mathscr{E};\delta),\lambda)$ where a' is the unique element such that $(\mathrm{id}_I,\sigma)\cdot a' = a$. A similar argument applies in the case of $\mathbb{I}$.

In the case of Clk, first note that

$$\blacktriangleleft(\Gamma.\mathrm{Clk}) = \blacktriangleleft(\Gamma \times \mathrm{Clk}) \cong \blacktriangleleft\Gamma \times \blacktriangleleft\mathrm{Clk}$$

since, as we saw in the proof of Theorem A.20, $\blacktriangleleft$ preserves finite products. It thus suffices to show that $\mathrm{Clk} \cong \blacktriangleleft\mathrm{Clk}$. In order to prove this, we first prove that any element in $\blacktriangleleft(\mathrm{Clk})(I,(\mathscr{E};\delta),\lambda)$ can be represented on the form (id,κ), for $\mathrm{id} : ((\mathscr{E};\delta+1),\lambda) \to ((\mathscr{E};\delta),\lambda)$ and $\kappa \in \mathscr{E}$. Here $\delta+1$ is the composition of δ with the successor function. This is true, because an element represented by (σ,κ) where $\sigma : ((\mathscr{E}';\delta'),\lambda') \to ((\mathscr{E};\delta),\lambda)$ and $\kappa \in \mathscr{E}'$ is equivalent to the element given by (σ,κ) where σ now is considered a map from $((\mathscr{E}';\delta'+1),\lambda')$ to $((\mathscr{E};\delta),\lambda)$. This element is in turn equivalent to $(\mathrm{id},\sigma(\kappa))$ as required. As a consequence, we can define an inverse to $\mathrm{p}_\blacktriangleleft$ to map $\kappa \in \mathrm{Clk}((\mathscr{E};\delta),\lambda)$ to $[(\mathrm{id},\kappa)]$.

The last statement of the lemma now follows by induction on Γ'. □

For $\Gamma \vdash \kappa : u \rightsquigarrow \Gamma'$, let n be the number of ticks on clock κ in $\Gamma \setminus \Gamma'$. Then the interpretation of u factors through $\blacktriangleleft^n\Gamma'$, and a tick assumption is interpreted as the appropriate projection $\blacktriangleleft^n \to \blacktriangleleft$. Likewise if $\Gamma \vdash (\kappa,u) \rightsquigarrow \Gamma'$ let n be the number of ticks on κ in $\Gamma \setminus \Gamma'$, then the interpretation of u factors through $\blacktriangleleft^n(\Gamma'.\mathrm{Clk})$ with tick variables interpreted as projections. The construction tirr is interpreted using the following lemma.

Lemma A.23. *For any* $n \geq 1$ *there exists a unique natural transformation* $\alpha : \blacktriangleleft^n \to \blacktriangleleft$ *such that* $\mathrm{p}_\blacktriangleleft \circ \alpha = \mathrm{p}^n_\blacktriangleleft$*, and a unique* $\beta : \blacktriangleleft^n(-.\mathrm{Clk}) \to \blacktriangleleft(-.\mathrm{Clk})$ *commuting with the projection to* $-.\mathrm{Clk}$*.*

In particular, this means that tirr is interpreted using constant paths in our model. Before proving Lemma A.23 we first give an alternative description of $\blacktriangleleft^n$ similar to the explicit description for $\blacktriangleleft$ itself.

Lemma A.24. *Let* $n \geq 0$*. Up to isomorphism, the functor* $\blacktriangleleft^n$ *maps* Γ *to the presheaf given at* $(I,(\mathscr{E};\delta),\lambda)$ *by equivalence classes of pairs* (σ,x) *such that* $(\mathrm{id}_I,\sigma) : (I,(\mathscr{E}';\delta'),\lambda') \to (I,(\mathscr{E};\delta),\lambda)$ *with* $\delta'(\lambda') \geq \delta(\lambda)+n$ *and* $x \in \Gamma(I,(\mathscr{E}';\delta'),\lambda')$ *considered up to the equivalence relation* $\sim$ *generated by* $(\sigma\tau,x) \sim (\sigma,(\mathrm{id},\tau)\cdot x)$*. The projection* $\mathrm{p}_\blacktriangleleft : \blacktriangleleft^{n+1} \to \blacktriangleleft^n$ *maps* $[(\sigma,x)]$ *to* $[(\sigma,x)]$*.*

We omit the routine proof of this, which is by induction on n.

Proof of Lemma A.23. Suppose $\alpha : \blacktriangleleft^n \to \blacktriangleleft$ is such that $\mathrm{p}_{\blacktriangleleft} \circ \alpha = \mathrm{p}^n_{\blacktriangleleft}$. We will show that $\alpha([(\sigma, x)]) = ([(\sigma, x)])$ for any $[(\sigma, x)] \in (\blacktriangleleft^n \Gamma)(I, (\mathscr{E}; \delta), \lambda)$.

First consider the case of a representable object $\Gamma = y(I, (\mathscr{E}'; \delta'), \lambda')$, and an element of the form

$$([(\sigma, \mathrm{id})]) \in \blacktriangleleft^n(y(I, (\mathscr{E}'; \delta'), \lambda'))(I, (\mathscr{E}; \delta), \lambda).$$

That is, $(\mathrm{id}_I, \sigma) : (I, (\mathscr{E}'; \delta'), \lambda') \to (I, (\mathscr{E}; \delta), \lambda)$ and $\delta'(\lambda') \geq \delta(\lambda) + n$ and id is the identity on $(I, (\mathscr{E}'; \delta'), \lambda')$ which is an element of $y(I, (\mathscr{E}'; \delta'), \lambda')$. Then $\alpha([(\sigma, \mathrm{id})])$ must be on the form $([(\tau, \rho)])$, and since $\mathrm{p}_{\blacktriangleleft}([(\tau, \rho)]) = (\mathrm{id}_I, \tau) \circ \rho$ and $\mathrm{p}^n_{\blacktriangleleft}([(\sigma, \mathrm{id})]) = (\mathrm{id}_I, \sigma)$, the assumption implies $(\mathrm{id}_I, \tau) \circ \rho = (\mathrm{id}_I, \sigma)$, and in particular that ρ is on the form (id_I, ρ'). Therefore

$$\begin{aligned}\alpha([(\sigma, \mathrm{id})]) &= [(\tau, (\mathrm{id}_I, \rho'))] = [(\tau, (\mathrm{id}_I, \rho') \cdot \mathrm{id})] \\ &= [(\tau\rho', \mathrm{id})] = [(\sigma, \mathrm{id})]\end{aligned}$$

Now, consider a general $[(\sigma, x)] \in (\blacktriangleleft^n \Gamma)(I, (\mathscr{E}; \delta), \lambda)$. That is, $(\mathrm{id}, \sigma) : (I, (\mathscr{E}'; \delta'), \lambda') \to (I, (\mathscr{E}; \delta), \lambda)$ and $\delta'(\lambda') \geq \delta(\lambda) + n$ and $x \in \Gamma(I, (\mathscr{E}'; \delta'), \lambda')$. Let $\gamma : y(I, (\mathscr{E}'; \delta'), \lambda') \to \Gamma$ be the element corresponding to x by the yoneda lemma, i.e., $\gamma\,(f, \sigma) = (f, \sigma) \cdot x$. Then

$$\begin{aligned}\alpha([(\sigma, x)]) &= \alpha([(\sigma, \gamma(\mathrm{id}))]) \\ &= \alpha(\blacktriangleleft^n(\gamma)[(\sigma, \mathrm{id})]) \\ &= \blacktriangleleft(\gamma)(\alpha([(\sigma, \mathrm{id})])) \\ &= \blacktriangleleft(\gamma)([(\sigma, \mathrm{id})]) \\ &= [(\sigma, x)]\end{aligned}$$

For the second statement of the theorem, recall that [36, Proposition 7.1] states that $\mathrm{p}_{\blacktriangleleft} : \blacktriangleleft(\Gamma.\mathrm{Clk}) \to \Gamma.\mathrm{Clk}$ is an isomorphism. Therefore, also the projection $\mathrm{p}^n_{\blacktriangleleft} : \blacktriangleleft^n(\Gamma.\mathrm{Clk}) \to \Gamma.\mathrm{Clk}$ is an isomorphism, from which the statement follows. □

A.6.6 Semantics for HITs. We provide semantics for any HIT definable in the schema with a direct generalization of the method employed in [22]. To each HIT signature given we associate a notion of algebra. An initial such algebra is then constructed and shown to support the necessary structure to model the HIT. For this section we denote objects of the time category by $x, x', \ldots$ to avoid notational clashes.

Reflecting the syntax of higher inductive types, we will work with homogeneous composition and transport separately. The semantic counterpoint to hcomp is the notion of a fibrancy structure. A fibrancy structure for a type global type A is an operation taking a face $\varphi : \mathbb{F}$, an element $u_0 : A$ and a partial path $p : A^{\mathbb{I}}$ defined on the extent φ and equal to u_0 at the endpoint $0 : \mathbb{I}$. It then produces an element $h_A(\varphi, u_0, u) : A$ which is equal to $u(1)$ on extent φ.

Consider a list of constructors $\Delta \vdash \mathscr{K}$ constrs, consisting of constructors $(\ell_i, (\Gamma_i; \overline{\Xi_i}; \Psi_i; \varphi_i; e_i)) \in \mathscr{K}$ for each $1 \leq i \leq n$, and let $\Delta' \vdash \delta : [\![\Delta]\!]$. A $\mathscr{K}, \delta$-algebra is a presheaf $A \in \mathrm{PSh}(\mathscr{C} \times \mathscr{T})/\Delta'$ with a fibrancy structure h_A, and for each $(\ell_i, (\Gamma_i; \overline{\Xi_i}; \Psi_i; \varphi_i; e_i))$ semantic constructors c^A_i. A semantic constructor is a map $c_i : \Pi(\gamma : [\![\Gamma_i]\!](\delta)).\overline{([\![\Xi_i]\!](\delta, \gamma) \to A)} \to [\![\Psi]\!] \to A$. Given such c^A_i's, we can interpret the boundary conditions given by the constructors, and the semantic constructors are subject to the boundary coherence conditions generated by this interpretation.

As in the syntax, we define the semantic boundary interpretation $\|-\|$ by induction over the grammar for these terms. We denote the list of constructors in $\mathscr{K}$ before the i'th constructor by $\mathscr{K}_{<i}$, and the list of semantic constructors for these by $C_{<i}$. The boundary coherence condition states that $c^A_i(\gamma, \theta, \bar{i}) = \|M_{i,j}\|_{C_{<i}}(\gamma, \theta, \bar{i})$ when $\varphi_{i,j}(\bar{i})$, which is the direct interpretation of $[\varphi_{i,0}\ \|M_{i,0}\|, \ldots, \varphi_{i,n}\ \|M_{i,n}\|]$. We define the boundary interpretation for the remaining grammar as follows:

$$\begin{aligned}\|x_i(\overline{u}(\gamma, \theta, i))\|_{C_{<i}} &= [\![\theta_i]\!]_{\mathrm{id}, [\![\overline{u}]\!]} \\ \|\mathrm{con}_j(\bar{t}, \overline{\lambda\xi.N}, \bar{i})\|_{C_{<i}} &= c^A_j([\![\bar{t}]\!], \overline{\lambda\xi.\|N\|_{C_{<i}}}, [\![\bar{i}]\!]) \\ \|\mathrm{hcomp}[\varphi \mapsto u]\ u_0\|_{C_{<i}} &= h_A(\varphi, \|u_0\|_{C_{<i}}, \|u\|_{C_{<i}})\end{aligned}$$

A morphism of $\mathscr{K}, \delta$-algebras $f : A \to B$ is a natural transformation preserving all $\mathscr{K}, \delta$-algebra structure up to equality. For constructors this means that $f(c^A_i(\gamma, \theta, r)) = c^B_i(\gamma, f \circ \theta, r)$.

We will now define a presheaf H^{pre}, and carve out the initial $\mathscr{K}, \delta$-algebra as a subpresheaf $H \subset H^{pre}$ following the construction exemplified in [22]. Let $\rho \in [\![\Delta]\!](I, x)$. The presheaf H^{pre} consists of formal constructor elements, and formal solutions to composition problems. Concretely the presheaf $H^{pre}(I, x, \rho)$ will contain elements of two forms: Firstly formal constructors $con_i(\gamma, \theta, r)$, where $\gamma \in [\![\Gamma_i]\!](I, x, \rho)$, $r \in [\![\Psi]\!](I, x, \rho)$, $[\![\varphi_i]\!](I, r) \neq 1_{\mathbb{F}}$, and θ is a family indexed by $f : (I, x) \to (J, x')$ and an element $\xi \in [\![\Xi_{i,j}]\!](J, x', f \cdot \rho, f \cdot \gamma)$ of elements of $H^{pre}(J, x', f \cdot \rho, f \cdot \gamma)$. Secondly hcomp elements of the form $\mathrm{hcomp}[\psi \mapsto u]\, u_0$ where $\psi \neq 1 \in \mathbb{F}(I), u_0 \in H^{pre}(I, x, \rho)$ and u is a family indexed by pairs of a map $f : I \to J$ in $\mathscr{C}$, such that $f \cdot \varphi = 1_{\mathbb{F}}$, and $r \in \mathbb{I}(J)$ of elements $u_{f,r} \in H^{pre}(J, x, (f, \mathrm{id}) \cdot \rho)$. To define the action of H^{pre} on morphisms, we first need to give interpretations of boundary terms. While this is not formally the same case as for general algebras, the interpretation follows the same structure and so we leave out the definition here. We denote this boundary interpretation in H^{pre} again by $\|-\|$, owing to the fact that when the definition is complete, the two will coincide. Let $g : (I, x) \to (J, x')$. We define the action on elements on H^{pre} as follows:

$$H^{pre}(g)(con_i(\overline{t},\overline{a},\overline{r})) :=$$
$$\begin{cases} \|M(g\cdot\overline{t}, g.\overline{a}, g\cdot\overline{r})\|_C & \text{if } \varphi_i(g\cdot\overline{r}) \\ con_i(g\cdot\overline{t}, g.\overline{a}, g\cdot\overline{r}) & \text{else} \end{cases}$$
$$H^{pre}(g)(\mathsf{hcomp}[\psi\mapsto u]u_0) :=$$
$$\begin{cases} u_{g,1} & \text{if } \psi(g\cdot\overline{r}) \\ \mathsf{hcomp}[g\cdot\psi\mapsto g.u](g\cdot u_0) & \text{else} \end{cases}$$

Here $g.\overline{a}$ is given by $(g.\overline{a})_{i,f,\xi} = a_{i,fg,\xi}$, $g.u$ is u similarly reindexed by the cubical component of g and C is list of constructors con_i on which the boundary interpretation is based.

Now we carve out those elements where the families taken as input are natural. This means that we include elements $con_i(\gamma,\theta,r)$ where each $\theta_{f,\xi}$ is in H and $\theta_{gf,g\cdot\xi} = g\cdot\theta_{f,\xi}$. Additionally, we include those $\mathsf{hcomp}[\psi\mapsto u]\,u_0$ elements where u_0 and each $u_{f,r}$ is in H, $u_{gf,(g,\mathrm{id})\cdot r} = (g,\mathrm{id})\cdot u_{f,r}$ and $(f,\mathrm{id})\cdot u_0 = u_{f,0}$ for suitable maps f such that $f\cdot\psi = 1_{\mathbb{F}}$.

Lemma A.25. *The presheaf H is the initial $\mathscr{K},\delta$-algebra for each $\delta:\Delta$. As a consequence, it has a unique map to any cubical type A with the structure of a $\mathscr{K},\delta$-algebra.*

Proof. Let A be a $\mathscr{K},\delta$-algebra with fibrancy structure h_A and constructors semantic c_i. We are tasked with producing a natural transformation $e : H\to A$, and we define $e(u)$ by induction on the structure of u: In the case where $u = con_i(\overline{t},\overline{a},\overline{r})$, we define $e(u) := c_i(\overline{t},\overline{e(a)},\overline{r})$, and in the case $u = \mathsf{hcomp}[\varphi\mapsto u]u_0$ we define $e(u) := h_A(\varphi, e(u), e(u_0))$. In both cases, $e(x)$ is e applied levelwise to the family x. While making this definition, we have to verify naturality at each stage, which follows in each case from the conditions on the c_i's or the computational behaviour of the fibrancy structures. □

Lemma A.26. *The presheaf H carries a composition structure.*

The proof of this lemma uses the fact that Δ, Γ_i and $\Xi_{i,j}$ all carry transport structures. Given those we define a transport structure on H by initiality, following the syntactic description of the action of trans on constructors in Section A.5. Combining this with the homogeneous composition structure we conclude the proof by Lemma 5 of [22].

Proposition A.27. *The presheaf H supports the dependent elimination principle of H.*

This follows from Lemma A.25 as in [22].

In addition to Lemma A.26 and Proposition A.27, we need to verify that H supports the formation and introduction rules, as well as the judgemental equalities These verifications are routine.

A.6.7 Verifying the principle of induction under clocks.

The principle of induction under clocks is verified in the model by the following theorem:

Theorem A.28. *Consider a HIT $\Delta\vdash H$ type, a type family $\Gamma,\forall\overline{\kappa}.H(\delta\,[\overline{\kappa}])\vdash D$ type and a term $\Gamma\vdash\delta:\forall\overline{\kappa}.\Delta$. Suppose that we semantically have an elimination list, i.e., for each i we have a term $u_i : [\![D[\lambda\overline{\kappa}.\mathrm{con}_i(\gamma\,[\overline{\kappa}],\overline{x\,[\overline{\kappa}]},\psi)]]\!]$ over the context*

$$\left[\!\!\left[\begin{array}{l}\Gamma,\gamma:\forall\overline{\kappa}.\Gamma_i[\delta\,[\overline{\kappa}]],\overline{x}:\overline{\forall\overline{\kappa}.\Xi_i[\delta\,[\overline{\kappa}],\gamma\,[\overline{\kappa}]]\to H(\delta\,[\overline{\kappa}])},\\ \overline{y}:\overline{(\forall\overline{\kappa}.\Xi_i[\delta\,[\overline{\kappa}],\gamma\,[\overline{\kappa}]])\to D[\lambda\overline{\kappa}.\lambda\xi.x\,[\overline{\kappa}]\,\xi\,[\overline{\kappa}]]},\psi:\Psi_i\vdash\end{array}\right]\!\!\right].$$

Then we can construct a term of $[\![D]\!]$ over $[\![\Gamma.\forall\overline{\kappa}.H(\delta\,[\overline{\kappa}])]\!]$.

Proof. The semantics of clock quantification of A is given by the semantic Π over the clock object, $\Pi\mathrm{Clk}.[\![A]\!]$. It is shown in [12] this is equivalent at I,x,ρ to the limit

$$\lim_n [\![A]\!](I, x+\lambda_n, \iota\cdot\rho, \lambda)$$

with structure maps given by the map $\mathrm{tick}^\lambda : (I, x+\lambda_{n+1})\to(I,x+\lambda_n)$. Here $(\mathscr{E},f)+\lambda_n = (\mathscr{E}\cup\{\lambda\}, f[\lambda\mapsto n])$. The map tick^λ is given by the identity maps, so it can be viewed as letting a single time step pass on the clock λ. Nominally such a family depends on the choice of fresh clock λ, but of course this dependence goes away since we have isomorphisms $x+\lambda_n\cong x+\lambda'_n$ in the time category. Hence we will suppress the change of λ, as long as these are fresh meaning that the environment variable is of the form $\iota\cdot\rho$.

The action of morphisms on HIT's in the model can only change the constructor type by acting on the cube component. As a consequence of this, the structure of elements in $[\![\forall\kappa.\mathsf{H}\,(\delta\,[\kappa])]\!]$ are determined at each level by their structure at 0.

It is sufficient to work with $d : [\![\forall\kappa.\Delta]\!](I,x)$, with the proof for general Γ following by pulling back along the term $\Gamma\vdash d:\forall\kappa.\Delta$. We start by making explicit the data of the assumed terms u_i. The input to a u_i is as follows:

(a) We have a family $t = (t_n)\in[\![\forall\kappa.\Gamma_i\,[\kappa]]\!](I,x,d)$.
(b) For each j we have a family

$$a = (a^j_n)\in[\![\forall\kappa.\Xi_{i,j}[\delta\,[\kappa],\gamma\,[\kappa]]\to\mathsf{H}\,(\delta\,[\kappa])]\!](I,x,d,t).$$

This can be further unfolded as follows: We have for each n, $f:(I,x+\lambda_n)\to(J,x')$ and

$$z\in[\![\Xi_{i,j}[\delta\,[\kappa],\gamma\,[\kappa]]]\!](J,x',f\cdot d_n, f\cdot t_n, f(\lambda))$$

an element $a^j_{n,f,z}\in[\![\mathsf{H}\,(\delta\,[\kappa])]\!](J,x',f\cdot d_n,f(\lambda))$.

(c) The next family is indexed by j as above, a morphism $g:(I,x)\to(J,x')$ and a family

$$z := (z_n)\in[\![\forall\kappa.\Xi_{i,j}[\delta\,[\kappa],\gamma\,[\kappa]]]\!](J,x',g\cdot d,g\cdot t).$$

Note here that this means we have a family $h := (h_n)\in[\![\forall\kappa.\mathsf{H}\,(\delta\,[\kappa])]\!](J,x',g\cdot d)$ given by $h_n := a^j_{n,g+\lambda_n,z_n}$. We then have an element $b^j_{n,z}\in[\![D]\!](J,x',g\cdot d,h)$, and we denote the family by b.

(d) An element $r\in\Psi_i(I)$.

When given this, we have an element

$$(\mathrm{con}_i(t_n,a_n,r))_n\in[\![\forall\kappa.\mathsf{H}\,(\delta\,[\kappa])]\!](I,x,d)$$

and we assume that we can inhabit D over this element, i.e., that we have

$$u_i(t, a, b, r) \in [\![D]\!](I, x, d, (\mathsf{con}_i(t_n, a_n, r))_n).$$

We need to inhabit $[\![D]\!]$ at an arbitrary element

$$h \in [\![\forall \kappa.\mathsf{H}\,(\delta\,[\kappa])]\!](I, x, d).$$

Observe that the structure of the family h is completely determined by the structure of h_0. This holds because the family is compatible with the tick maps, and these are trivial in the cube component which means in particular that they cannot change the outer constructor of an element of the HIT. This means that we only have to inhabit it at families either of the form $(\mathsf{con}_i(t_n, a_n, r))$ or of the form $(\mathsf{hcomp}[\varphi \mapsto u^n]\, u_0^n)$. We can do this by induction on the structure, since it is the same for each component of the family, and can therefore assume that we can inhabit $[\![D]\!]$ at structurally simpler families. Explicitly, the induction is done on the structure of the 0 component in the family, so that the inductive hypothesis says that whenever we have a family (h'_n) such that h'_0 is structurally simpler than h_0, we can inhabit $[\![D]\!]$ over this family. We denote the element over a family h by $\alpha(h)$.

We consider first the case where we are given a family

$$(\mathsf{con}_i(t_n, a_n, r))_n \in [\![\forall \kappa.\mathsf{H}\,(\delta\,[\kappa])]\!](I, x, d).$$

Note that (t_n) is forced to be compatible with tick maps and that r is constant. This means that we have $(t_n)_n \in [\![\forall \kappa.\Gamma_i[\delta\,[\kappa]]]\!](I, x, d)$ and $r \in \Psi(I)$, and we have the data of (a) and (d). The implied typing of a_n^j is a family over $f : (I, x{+}\lambda_n) \to (J, x')$ and $z \in [\![\Xi_{i,j}[\delta\,[\kappa], \gamma\,[\kappa]]]\!](J, x', f{\cdot}d_n, f{\cdot}t_n, f(\lambda))$ of elements $a^j_{n,f,z} \in [\![\mathsf{H}\,(\delta\,[\kappa])]\!](J, x', f \cdot d_n, f(\lambda))$. This is exactly the typing of (b), so we need only to construct the family in (c). We need this to be a family over $g : (I, x) \to (J, x')$ and $z := (z_n) \in [\![\forall \kappa.\Xi_{i,j}[\delta\,[\kappa], \gamma\,[\kappa]]]\!](J, x', g \cdot d, g \cdot t)$. Given this, we again define the family

$$h := (h_n) \in [\![\forall \kappa.\mathsf{H}\,(\delta\,[\kappa])]\!](J, x', g \cdot d)$$

by $h_n := a^j_{n,g+\lambda_n,z_n}$. Note that this family is compatible with ticks, and hence we obtain $\alpha((h_n))$ $in [\![D]\!](J, x', g \cdot d, h)$ by inductive hypothesis, since h_0 is given by a component of the structurally simpler a. We define

$$\alpha((\mathsf{con}_i(t_n, a_n, r))_n) = u_i(t, a, \alpha((h_n)), r).$$

Consider now the hcomp case. Since $[\![D]\!]$ is a type, it carries a composition structure. The strategy for inhabiting $[\![D]\!]$ will be to construct an appropriate composition problem and apply the inductive hypothesis. Consider a family $h = (\mathsf{hcomp}[\varphi \mapsto u^n]\, u_0^n) \in [\![\forall \kappa.\mathsf{H}\,(\delta\,[\kappa])]\!](I, x, d)$. Here the implicit typing is as follows: For each n, we have for each $f : I \to J$ such that $f \cdot \varphi = \top$ and $s \in \Psi(J)$ an element $u^n_{f,s} \in [\![\mathsf{H}\,(\delta\,[\kappa])]\!](J, x + \lambda_n, f \cdot d_n, \lambda)$ and an element $u_0^n \in [\![\mathsf{H}\,(\delta\,[\kappa])]\!](J, x + \lambda_n, d_n, \lambda)$. The u_0^n family has the right shape for us to employ our induction hypothesis to inhabit $[\![D]\!]$ over it. Because the shape of f and s does not depend on the time component, we also have for each f a family in n given by $(u^n_{f,s})$. All these families have 0 component structurally simpler, hence we can inhabit $[\![D]\!]$ over each of them by $\alpha((u_0^n))$ and $\alpha((u^n_{f,i}))$. We will write $\alpha((u^n))$ for the family given at f, i by $\alpha((u^n_{f,i}))$. We can then define $\alpha(h) = \mathsf{comp}_{D[v]}[\varphi \mapsto \alpha((u^n))]\, \alpha((u_0^n))$ where $v = (\mathsf{hfill}_{[\![\mathsf{H}\,\delta\,[\kappa]]\!]}[\varphi \mapsto u^n]\, u_0^n)$.

It must be verified that α is natural in morphisms of the category while defining it. Both because we need α to be natural for the desired conclusion and because the recursive calls used to define α require it. Let $f : (I, x) \to (J, x')$. Naturality means that $f \cdot \alpha(h) = \alpha(f \cdot h)$, which unfolds as follows:

- Consider a family $h = (\mathsf{con}_i(t_n, a_n, r))$. By definition $\alpha(h) = u_i(t, a, b, r)$ where $b_{g,z} = \alpha((a^j_{n,g+\lambda_n,z_n}))$. If f does not trigger a boundary condition for con_i we have

$$f \cdot \alpha(h) = u_i(f \cdot t, f.a, f.b, f \cdot r) \text{ and}$$

$$(f \cdot h)_n = \mathsf{con}_i((f + \lambda_n) \cdot t_n, (f + \lambda_n).a_n, (f + \lambda_n) \cdot r).$$

  Recall here that $g.a_n$ is defined to be the family a_n is given by composing the map index with g and acting on the input index. Applying α to this yields $u_i(((f + \lambda_n) \cdot t_n), ((f{+}\lambda_n).a_n), b', f \cdot r)$ where b' is defined from $((f + \lambda_n).a_n)$ as above, i.e.,

$$b'_{g,z} = \alpha((((f + \lambda_n).a)_{n,g+\lambda_n,z_n})).$$

  By definition of the morphism action on a limit we have $((f{+}\lambda_n){\cdot}t_n) = f{\cdot}(t_n)$ and $((f{+}\lambda_n).a_n) = f.(a_n)$. The action of f and $f{+}\lambda_n$ are the same on r, since they have the same cubical component. It remains to argue that $f.b$ coincides with the b' defined from the family $f.a$, which already a part of the naturality of the u_i's. We check this componentwise for the family, so we have to show that $(f.b)_{g,z} = b'_{g,z}$. The left hand side is by definition equal to $b_{gf,f\cdot z} = \alpha((a^j_{n,gf+\lambda_n,(f+\lambda_n)\cdot z_n}))$, while the right hand side is equal to $\alpha((((f{+}\lambda_n).a)^j_{g+\lambda_n,z_n}))$. Unfolding the right hand side we get $\alpha(((a)^j_{(g+\lambda_n)(f+\lambda_n),(f+\lambda_n)\cdot z_n}))$, which is the same as the left hand side since $(g{+}\lambda_n)(f{+}\lambda_n) = gf + \lambda_n$.
  If f triggers a boundary condition of con_i we have by assumption that

$$f \cdot \alpha(h) = f \cdot u_i(t, a, b, r) = [\![\langle\!|e|\!\rangle^{\delta}_{\mathscr{E}, \overline{x} \mapsto \overline{y}}]\!](f \cdot t, f \cdot a, f \cdot b, f \cdot r)$$

  where e is the boundary term of con_i. On the other hand, the structure of H means that $\alpha(f{\cdot}h) = \alpha((\|e\|((f{+}\lambda_n){\cdot}t_n, (f{+}\lambda_n).a_n, (f{+}\lambda_n){\cdot}r))$ where $\|e\|$ is the boundary evaluation for $\mathsf{H}\,(\delta\,[\kappa])$. Equality of these two can be shown by induction on the structure of e.
- Consider a family $h = (\mathsf{hcomp}[\varphi \mapsto u^n]\, u_0^n)$. By definition $\alpha(h) = \mathsf{comp}_{D[v]}[\varphi \mapsto \alpha((u^n))]\, \alpha((u_0^n))$. If f

does not make φ true we have to verify that

$f \cdot \mathsf{comp}_{D[v]}\,[\varphi \mapsto \alpha((u^n))]\;\alpha((u_0^n)) = \alpha(f \cdot h)$.

We unfold the right hand side:

$$\begin{aligned}
&\alpha(f \cdot h) = \\
&\quad \alpha((\mathsf{hcomp}[f \cdot \varphi \mapsto (f + \lambda_n).u^n]\;(f + \lambda_n) \cdot u_0^n)) \\
&= \mathsf{comp}_{D[f \cdot v]}[f \cdot \varphi \mapsto \alpha(((f + \lambda_n).u^n))] \\
&\qquad\qquad\qquad\qquad \alpha(((f + \lambda_n) \cdot u_0^n)) \\
&= \mathsf{comp}_{D[f \cdot v]}[f \cdot \varphi \mapsto f.\alpha((u^n))]\;f \cdot \alpha((u_0^n)) \\
&= f \cdot \mathsf{comp}_{D[v]}[\varphi \mapsto \alpha((u^n))]\;\alpha((u_0^n))
\end{aligned}$$

The second to last step above follows by the inductive hypothesis that α is natural on simpler families, and the last line is exactly the left hand side of the original equation.

If f makes φ true, both composition in D and hcomp reduce to appropriate components of the input, i.e., we have $\alpha(f \cdot h) = \alpha((u^n_{(f+\lambda),1}))$ and $f \cdot \alpha(h) = \alpha((u^n))_{f,1}$ for the right and left hand sides of the equality we are trying to show. These agree by definition of $\alpha((u^n))_{f,1}$.

This means that α assembles into a term of $[\![D]\!]$ over $[\![\Gamma.\forall\kappa.\mathsf{H}\,(\delta\,[\kappa])]\!]$ as desired. It is moreover clear from the definition of α that it will validate the computation rules of the induction principle, since the value on constructor families was given directly by the appropriate u_i and the value on hcomp families was given by the appropriate composition in D. □