The Clocks They Are Adjunctions
Denotational Semantics for Clocked Type Theory

Bassel Mannaa
Department of Computer Science, IT University of Copenhagen, Copenhagen, Denmark
basm@itu.dk

Rasmus Ejlers Møgelberg
Department of Computer Science, IT University of Copenhagen, Copenhagen, Denmark
mogel@itu.dk

Abstract

Clocked Type Theory (CloTT) is a type theory for guarded recursion useful for programming with coinductive types, allowing productivity to be encoded in types, and for reasoning about advanced programming language features using an abstract form of step-indexing. CloTT has previously been shown to enjoy a number of syntactic properties including strong normalisation, canonicity and decidability of type checking. In this paper we present a denotational semantics for CloTT useful, e.g., for studying future extensions of CloTT with constructions such as path types.

The main challenge for constructing this model is to model the notion of ticks used in CloTT for coinductive reasoning about coinductive types. We build on a category previously used to model guarded recursion, but in this category there is no object of ticks, so tick-assumptions in a context can not be modelled using standard tools. Instead we show how ticks can be modelled using adjoint functors, and how to model the tick constant using a semantic substitution.

2012 ACM Subject Classification Theory of computation → Type theory, Theory of computation → Categorical semantics

Keywords and phrases Guarded type theory, Coinduction, Presheaf model, Clocked type theory, Dependent adjunction

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.23

Funding This work was supported by DFF-Research Project 1 Grant no. 4002-00442, from The Danish Council for Independent Research for the Natural Sciences (FNU) and by a research grant (13156) from VILLUM FONDEN.

1 Introduction

In recent years a number of extensions of Martin-Löf type theory [14] have been proposed to enhance the expressiveness or usability of the type theory. The most famous of these is Homotopy Type Theory [18], but other directions include the related Cubical Type Theory [11], FreshMLTT [17], a type theory with name abstraction based on nominal sets, and Type Theory in Color [4] for internalising relational parametricity in type theory. Many of these extensions use denotational semantics to argue for consistency and to inspire constructions in the language.

This paper is part of a project to extend type theory with guarded recursion [16], a variant of recursion that uses a modal type operator $\triangleright$ (pronounced 'later') to preserve consistency of the logical reading of type theory. The type $\triangleright A$ should be read as classifying data of type A available one time step from now, and comes with a map $\mathsf{next} : A \to \triangleright A$

and a fixed point operator mapping a function $f : \triangleright A \to A$ to a fixed point for $f \circ \mathsf{next}$. This, in combination with *guarded recursive types*, i.e., types where the recursive variable is guarded by a $\triangleright$, e.g., $\mathsf{Str} \equiv \mathbb{N} \times \triangleright\mathsf{Str}$ gives a powerful type theory in which operational models of combinations of advanced programming language features such as higher-order store [7] and nondeterminism [8] can be modelled using an abstract form of step-indexing [1]. Combining this with a notion of clocks, indexing the $\triangleright$ operator with clock names, and universal quantification over clocks, one can encode coinduction using guarded recursion, allowing productivity [12] of coinductive definitions to be encoded in types [2].

The most recent type theory with guarded recursion is Clocked Type Theory (CloTT) [3], which introduces the notion of ticks on a clock. Ticks are evidence that time has passed and can be used to unpack elements of type $\triangleright A$ to elements of A. In fact, in CloTT, $\triangleright A$ is a special form of function type from ticks to A. The combination of ticks and clocks in CloTT can be used for coinductive reasoning about coinductive types, by encoding the *delayed substitutions* of [9].

Bahr et al [3] have shown that CloTT can be given a reduction semantics satisfying strong normalisation, confluence and canonicity. This establishes that productivity can indeed be encoded in types: For a closed term t of stream type, the n'th element can be computed in finite time. These syntactic results also imply soundness of the type theory. However, these results have only been established for a core type theory without, e.g., identity types, and the arguments can be difficult to extend to larger calculi. In particular, we are interested in extending CloTT with path types as in Guarded Cubical Type Theory [5]. Therefore a denotational model of CloTT can be useful, and this paper presents such a model.

The work presented here builds on a number of existing models for guarded recursion. The most basic such, modelling the single clock case, is the topos of trees model [7], in which a closed type is modelled as a family of sets X_n indexed by natural numbers n, together with restriction maps of the form $X_{n+1} \to X_n$ for every n. In other words, a type is a presheaf over the ordered natural numbers. In this model $\triangleright$ is modelled as $(\triangleright X)_0 = 1$ and $(\triangleright X)_{n+1} = X_n$ and guarded recursion reduces to natural number recursion. The guarded recursive type Str mentioned above can be modelled in the topos of trees as $\mathsf{Str}(n) = 1 \times \mathbb{N}^n$.

Bizjak and Møgelberg [10] recently extended this model to the case of many clocks, using a category $\mathsf{Set}^{\mathbb{T}}$ of covariant presheaves over a category $\mathbb{T}$ of time objects, i.e., pairs of a finite set X and a map $X \to \mathbb{N}$. In this model, universal quantification over clocks is modelled by constructing an object in the topos of trees and taking the limit of that. For example, taking the limits over the object Str gives the usual coinductive type of streams over natural numbers.

The main challenge when adapting the model of [10] to CloTT is to model ticks, which were not present in the language modelled in [10]. In particular, how does one model tick assumptions of the form $\alpha : \kappa$ in a context, when there appears to be no object of ticks in the model to be used as the denotation of the clock κ. In this paper we observe that these assumptions can be modelled using a left adjoint $\blacktriangleleft^\kappa$ to the functor $\blacktriangleright^\kappa$ used in [10] to model $\triangleright^\kappa$ the delay modality associated to the clock κ. Precisely we model context extension as $[\![\Gamma, \alpha : \kappa]\!] = \blacktriangleleft^\kappa [\![\Gamma]\!]$. To clarify what is needed to model ticks, we focus on a fragment of CloTT called the *tick calculus* capturing just the interaction of ticks with dependent types. We show that the tick calculus can be modelled soundly in a category with family [13] (a standard notion of model for dependent type theory), with an adjunction $\mathsf{L} \dashv \mathsf{R}$ of endofunctors on the underlying category, for which the right adjoint lifts to types and terms, and there is a natural transformation from L to the identity. This appears to be a general pattern seen also in the model of fresh name abstraction of FreshMLTT [17] and dependent path types

in cubical type theory [11]. Similarly challenging is how to model the special tick constant $\diamond$. Since there is no object of ticks, there is no element corresponding to $\diamond$ either. Still, we shall see that there exists a semantic substitution of $\diamond$ for a tick variable that can be used to model application of terms to $\diamond$.

The paper is organised as follows: The tick calculus and its model theory are introduced in Section 2. Section 3 introduces CloTT, omitting guarded recursive types and universes, which we leave for future work. Section 4 presents the basics of the model, in particular the presheaf category $\mathsf{Set}^{\mathbb{T}}$ and the adjunction $\blacktriangleleft^{\kappa} \dashv \blacktriangleright^{\kappa}$. The presence of ticks in contexts leads to a non-standard notion of substitutions, and we study the syntax and semantics of these in Section 5. Sections 6 and 7 extend the model with universal quantification over clocks and $\diamond$, respectively. Finally, Section 8 verifies the important clock irrelevance axiom, and Section 9 concludes and discusses future work.

2 A tick calculus

Before introducing CloTT we focus on a fragment to explain the notion of ticks and how to model these. To motivate ticks, consider the notion of applicative functor from functional programming [15]: a type former $\triangleright$ with maps $A \to \triangleright A$ and $\triangleright(A \to B) \to \triangleright A \to \triangleright B$ satisfying a number of equations that we shall not recall. These maps can be used for programming with the constructor $\triangleright$, but for reasoning in a dependent type theory, one needs an extension of these to dependent function types. For example, in guarded recursion one can prove a theorem X by constructing a map $\triangleright X \to X$ and taking its fixed point in X. If the theorem is that a property holds for all elements in a type of guarded streams satisfying $\mathsf{Str} \equiv \mathbb{N} \times \triangleright \mathsf{Str}$, then X will be of the form $\prod(xs : \mathsf{Str}).P$. To apply the (essentially coinductive) assumption of type $\triangleright \prod(xs : \mathsf{Str}).P$ to the tail of a stream, which has type $\triangleright \mathsf{Str}$ we need an extension of the applicative functor action.

What should the type of such an extension be? Given $a : \triangleright A$ and $f : \triangleright(\prod(x : A).B)$ the application of f to a should be something of the form $\triangleright B[??/x]$. If we think of $\triangleright$ as a delay, intuitively a is a value of type A delayed by one time, and the ?? should be the value delivered by a one time step from now. Ticks are evidence that time has passed, and they allow us to talk about values delivered in the future.

The *tick calculus* is the extension of dependent type theory with the following four rules

$$\frac{\Gamma \vdash}{\Gamma, \alpha{:}\mathrm{tick} \vdash} \qquad \frac{\Gamma, \alpha{:}\mathrm{tick} \vdash A}{\Gamma \vdash \triangleright(\alpha{:}\mathrm{tick})A}$$

$$\frac{\Gamma, \alpha{:}\mathrm{tick} \vdash t : A}{\Gamma \vdash \lambda(\alpha{:}\mathrm{tick})t : \triangleright(\alpha{:}\mathrm{tick})A} \qquad \frac{\Gamma \vdash t : \triangleright(\alpha{:}\mathrm{tick})A}{\Gamma, \beta{:}\mathrm{tick}, \Gamma' \vdash t\,[\beta] : A[\beta/\alpha]}$$

An assumption of the form α:tick in a context is an assumption that one time step has passed, and α is the evidence of this. Variables on the right-hand side of such an assumption should be thought of as arriving one time step later than those on the left. Ticks can be abstracted in terms and types, so that the type constructor $\triangleright$ now comes with evidence that time has passed that can be used in its scope. The type $\triangleright(\alpha{:}\mathrm{tick})A$ can be thought of as a form of dependent function type over ticks, which we abbreviate to $\triangleright A$ if α does not occur free in A. The elimination rule states that if a term t can be typed as $\triangleright(\alpha{:}\mathrm{tick})A$ before the arrival of tick β, t can be opened using β to give something of type $A[\beta/\alpha]$. Note that the causality restriction in the typing rule prevents a term like $\lambda x.\lambda(\alpha{:}\mathrm{tick}).x\,[\alpha]\,[\alpha] : \triangleright\triangleright A \to \triangleright A$ being well typed; a tick can only be used to unpack the same term once. The context Γ'

in the elimination rule ensures that typing rules are closed under weakening, also for ticks. Note that the clock object tick is not a type.

The equality theory is likewise extended with the usual β and η rules:

$$\lambda(\alpha{:}\mathrm{tick})t\,[\beta] = t[\beta/\alpha] \qquad\qquad \lambda(\alpha{:}\mathrm{tick})(t\,[\alpha]) = t$$

As stated, the tick calculus should be understood as an extension of standard dependent type theory. In particular one can add dependent sums and function types with standard rules. Variables can be introduced from anywhere in the context, also past ticks.

We can now type the dependent applicative structure as

$$\begin{aligned}&\lambda(x{:}A)\lambda(\alpha{:}\mathrm{tick})x : A \to \triangleright A\\ &\lambda f\lambda y\lambda(\alpha{:}\mathrm{tick})f\,[\alpha](y\,[\alpha]) : \triangleright\left(\textstyle\prod(x:A).B\right) \to \prod(y:\triangleright A).\triangleright(\alpha{:}\mathrm{tick}).B[y\,[\alpha]/x]\end{aligned}$$

For a small example on how ticks in combination with the fixed point operator $\mathsf{dfix} : (\triangleright X \to X) \to \triangleright X$ can be used to reason about guarded recursive data, let $\mathsf{Str} \equiv \mathbb{N} \times \triangleright\mathsf{Str}$ be the type of guarded recursive streams mentioned above, and suppose $x{:}\mathsf{N} \vdash P(x)$ is a family to be thought of as a predicate on N (where $x :: xs$ is the pairing of x and xs). A lifting of P to streams would be another guarded recursive type $y{:}\mathsf{Str} \vdash \hat{P}(y)$ satisfying $\hat{P}(x :: xs) \equiv P(x) \times \triangleright(\alpha{:}\mathrm{tick})\hat{P}(xs\,[\alpha])$. If $p : \Pi(x{:}\mathsf{N})P(x)$ is a proof of P we would expect that also $\Pi(y{:}\mathsf{Str})\hat{P}(y)$ can be proved, and indeed this can be done as follows. Consider first

$$\begin{aligned}&f : \triangleright(\Pi(y{:}\mathsf{Str})\hat{P}(y)) \to \Pi(y{:}\mathsf{Str})\hat{P}(y)\\ &f\,q\,(x :: xs) \stackrel{\mathsf{def}}{=} p(x) :: \lambda(\alpha{:}\mathrm{tick})q\,[\alpha](xs\,[\alpha])\end{aligned}$$

Then $f(\mathsf{dfix}(f))$ has the desired type.

More generally, ticks can be used to encode [3] the *delayed substitutions* of [9], which have been used to reason coinductively about coinductive data. For more examples of reasoning using these see [9]. For reasons of space, we will not model general guarded recursive types in this paper, but see Section 4 for how to model the types used above.

2.1 Modelling ticks using adjunctions

We now describe a notion of model for the tick calculus. It is based on the notion of category with families (CwF) [13], which is a standard notion of model of dependent type theory. Recall that a CwF is a pair $(\mathscr{C}, T)$ such that $\mathscr{C}$ is a category with a distinguished terminal object and $T : \mathscr{C}^{\mathrm{op}} \to \mathsf{Fam}(\mathsf{Set})$ is a functor together with a comprehension map to be recalled below. The functor T associates to every object Γ in $\mathscr{C}$ a map $T(\Gamma) : \mathsf{Tm}(\Gamma) \to \mathsf{Ty}(\Gamma)$ and to every morphism $\gamma : \Delta \to \Gamma$ maps $\mathsf{Ty}(\gamma) : \mathsf{Ty}(\Gamma) \to \mathsf{Ty}(\Delta)$ and $\mathsf{Tm}(\gamma) : \mathsf{Tm}(\Gamma) \to \mathsf{Tm}(\Delta)$ such that $T(\Delta) \circ \mathsf{Tm}(\gamma) = \mathsf{Ty}(\gamma) \circ T(\Gamma)$. Following standard conventions, we write $\Gamma \vdash A$ to mean $A \in \mathsf{Ty}(\Gamma)$ and $\Gamma \vdash t : A$ to mean $t \in T(\Gamma)^{-1}(A)$, and we write $\Delta \vdash A[\gamma]$ for $\mathsf{Ty}(\gamma)(A)$ when $\Gamma \vdash A$, and likewise $\Delta \vdash t[\gamma] : A[\gamma]$ for $\mathsf{Tm}(\gamma)(t)$ when $\Gamma \vdash t : A$. We refer to the objects of $\mathscr{C}$ as contexts, morphisms as substitutions, elements of $\mathsf{Ty}(\Gamma)$ as types and elements of $\mathsf{Tm}(\Gamma)$ as terms.

Comprehension associates to each $\Gamma \vdash A$ a context $\Gamma.A$, a substitution $\mathsf{p}_A : \Gamma.A \to \Gamma$ and a term $\Gamma.A \vdash \mathsf{q}_A : A[\mathsf{p}_A]$, such that for every $\gamma : \Delta \to \Gamma$, and $\Delta \vdash t : A[\gamma]$ there exists a unique substitution $\langle\gamma, t\rangle : \Delta \to \Gamma.A$ such that $\mathsf{p}_A \circ \langle\gamma, t\rangle = \gamma$ and $\mathsf{q}_A[\langle\gamma, t\rangle] = t$.

To model the tick calculus we need an operation L modelling the extension of a context with a tick, plus an operation R modelling $\triangleright$. In the simply typed setting, R would be a right adjoint to context extension, but for dependent types this is not quite so, since these

operations work on different objects (contexts and types respectively). In the model we consider in this paper, the right adjoint does exist as an operation on contexts, but also extends to types and terms in the sense of the following definition.

▶ **Definition 1.** Let $(\mathscr{C}, T)$ be a CwF and let $\mathsf{R} : \mathscr{C} \to \mathscr{C}$ be a functor. An *extension of* R *to types and terms* is a pair of operations on types and term presented here in the form of rules

$$\frac{\Gamma \vdash A}{\mathsf{R}\Gamma \vdash \mathsf{R}A} \qquad \frac{\Gamma \vdash t : A}{\mathsf{R}\Gamma \vdash \mathsf{R}t : \mathsf{R}A}$$

commuting with substitutions in the sense that $(\mathsf{R}A)[\mathsf{R}\gamma] = \mathsf{R}(A[\gamma])$ and $(\mathsf{R}t)[\mathsf{R}\gamma] = \mathsf{R}(t[\gamma])$ hold for all substitutions γ, and commuting with comprehension in the sense that there exists an operation associating to each $\Gamma \vdash A$ a morphism $\zeta_{\Gamma,A} : \mathsf{R}\Gamma.\mathsf{R}A \to \mathsf{R}(\Gamma.A)$ in $\mathscr{C}$ inverse to $\langle \mathsf{R}\mathsf{p}_A, \mathsf{R}\mathsf{q}_A \rangle$. A *CwF with adjunction* is a pair of adjoint endofunctors $\mathsf{L} \dashv \mathsf{R} : \mathscr{C} \to \mathscr{C}$ with an extension of R to types and terms.

Given a CwF with adjunction, one can define an operation mapping types $\mathsf{L}\Gamma \vdash A$ to types $\Gamma \vdash \mathsf{R}_\Gamma A$ defined as $\mathsf{R}_\Gamma A = (\mathsf{R}A)[\eta]$ where η is the unit of the adjunction.

▶ **Lemma 2.** *There is a bijective correspondence between terms* $\mathsf{L}\Gamma \vdash a : A$ *and terms* $\Gamma \vdash b : \mathsf{R}_\Gamma A$ *for which we write* $\overline{(-)}$ *for both directions where* $\Gamma \vdash \overline{a} : \mathsf{R}_\Gamma A$ *is given by* $\overline{a} = (\mathsf{R}a)[\eta]$ *and* $\mathsf{L}\Gamma \vdash \overline{b} : A$ *is given by* $\overline{b} = \mathsf{q}_A[\epsilon \circ \mathsf{L}(\zeta_{\mathsf{L}\Gamma,A} \circ \langle \eta, b \rangle)]$. *Moreover, if* $\gamma : \Delta \to \Gamma$, $\mathsf{L}\Gamma \vdash a : A$ *and* $\Gamma \vdash b : \mathsf{R}_\Gamma A$ *then*

$$(\mathsf{R}_\Gamma A)[\gamma] = R_\Delta(A[\mathsf{L}\gamma]) \qquad \overline{a[\mathsf{L}\gamma]} = \overline{a}[\gamma] \qquad \overline{b[\gamma]} = \overline{b}[\mathsf{L}\gamma]$$

2.2 Interpretation

The notion of CwF with adjunction is almost sufficient for modelling the tick calculus, but to interpret tick weakening, we will assume given a natural transformation $\mathsf{p}_\mathsf{L} : \mathsf{L} \to \mathsf{id}_\mathscr{C}$. Defining

$$[\![\Gamma, \alpha : \mathrm{tick} \vdash]\!] = \mathsf{L}[\![\Gamma]\!]$$

p_L allows us to define a context projection $\mathsf{p}_{\Gamma'} : [\![\Gamma, \Gamma' \vdash]\!] \to [\![\Gamma \vdash]\!]$ by induction on Γ' using p_L in the case of tick variables. We can then define the rest of the interpretation as

$$[\![\Gamma, x : A, \Gamma' \vdash x : A]\!] = \mathsf{q}_A[\mathsf{p}_{\Gamma'}] \qquad [\![\Gamma \vdash \triangleright(\alpha{:}\mathrm{tick})A]\!] = \mathsf{R}_{[\![\Gamma]\!]}[\![A]\!]$$
$$[\![\Gamma \vdash \lambda(\alpha{:}\mathrm{tick})t]\!] = \overline{[\![t]\!]} \qquad [\![\Gamma, \alpha'{:}\mathrm{tick}, \Gamma' \vdash t\,[\alpha']]\!] = \overline{[\![t]\!]}[\mathsf{p}_{[\![\Gamma']\!]}]$$

▶ **Proposition 3.** *The above interpretation of the tick calculus into a CwF with adjunction and tick weakening* p_L *is sound.*

3 Clocked Type Theory

Clocked Type Theory (CloTT) is an extension of the tick calculus with guarded recursion and multiple clocks. Rather than having a global notion of time as in the tick calculus, ticks are associated with clocks and clocks can be assumed and universally quantified. Judgements have a separate context of clock variables Δ, for example, the typing judgement has the form $\Gamma \vdash_\Delta t : A$, where Δ is a set of clock variables $\kappa_1, \ldots, \kappa_n$. The clock context can be thought of as a context of assumptions of the form $\kappa_1 : \mathrm{Clock}, \ldots, \kappa_n : \mathrm{Clock}$ that appear to the left of the assumptions of Γ, except that Clock is not a type. There are no operations for

Type formation rules

$$\frac{\Gamma, \alpha : \kappa \vdash_\Delta A \text{ type} \qquad \kappa \in \Delta}{\Gamma \vdash_\Delta \triangleright\,(\alpha : \kappa).A \text{ type}} \qquad \frac{\Gamma \vdash_{\Delta,\kappa} A \text{ type} \qquad \Gamma \vdash_\Delta}{\Gamma \vdash_\Delta \forall \kappa.A \text{ type}}$$

Typing rules

$$\frac{\Gamma \vdash_{\Delta,\kappa} t : A \qquad \Gamma \vdash_\Delta}{\Gamma \vdash_\Delta \Lambda\kappa.t : \forall\kappa.A} \qquad \frac{\Gamma \vdash_\Delta t : \forall\kappa.A \qquad \kappa' \in \Delta}{\Gamma \vdash_\Delta t[\kappa'] : A[\kappa'/\kappa]} \qquad \frac{\Gamma, \alpha : \kappa \vdash_\Delta t : A \qquad \kappa \in \Delta}{\Gamma \vdash_\Delta \lambda(\alpha : \kappa).t : \triangleright\,(\alpha : \kappa).A}$$

$$\frac{\Gamma \vdash_\Delta t : \triangleright\,(\alpha : \kappa).A \qquad \Gamma, \alpha' : \kappa, \Gamma' \vdash_\Delta}{\Gamma, \alpha' : \kappa, \Gamma' \vdash_\Delta t\,[\alpha'] : A\,[\alpha'/\alpha]} \qquad \frac{\Gamma \vdash_{\Delta,\kappa} t : \triangleright\,(\alpha : \kappa).A \qquad \Gamma \vdash_\Delta \qquad \kappa' \in \Delta}{\Gamma \vdash_\Delta (t\,[\kappa'/\kappa])\,[\diamond] : A[\kappa'/\kappa]\,[\diamond/\alpha]}$$

$$\frac{\Gamma \vdash_\Delta t : \triangleright^\kappa A \to A}{\Gamma \vdash_\Delta \mathsf{dfix}^\kappa\, t : \triangleright^\kappa A}$$

Judgemental equality

$$(\Lambda\kappa.t)[\kappa'] \equiv t[\kappa/\kappa'] \qquad (\Lambda\kappa.t[\kappa]) \equiv t \qquad (\lambda(\alpha' : \kappa).t)\,[\alpha] \equiv t\,[\alpha/\alpha']$$
$$\lambda(\alpha : \kappa).(t\,[\alpha]) \equiv t \qquad (\mathsf{dfix}^\kappa\, t)\,[\diamond] \equiv t\,(\mathsf{dfix}^\kappa\, t)$$

Figure 1 Selected typing and judgemental equality rules of Clocked Type Theory.

forming clocks, only clock variables. It is often convenient to have a single clock constant κ_0 and this can be added by working in a context of a single clock variable.

The rules for typing judgements and judgemental equality are given in Figure 1. These should be seen as an extension of a dependent type theory with dependent function and sum types, as well as extensional identity types. The rules for these are completely standard (ignoring the clock context), and thus are omitted from the figure. We write $\equiv$ for judgemental equality and $t =_A u$ for identity types. The model will also model the *identity reflection* rule

$$\frac{\Gamma \vdash_\Delta p : t =_A u}{\Gamma \vdash_\Delta t \equiv u : A}$$

of extensional type theory.

The guarded fixed point operator dfix is useful in combination with guarded recursive types. Suppose for example that we have a type of natural numbers $\mathbb{N}$ and a type of guarded recursive streams Str^κ satisfying $\mathsf{Str}^\kappa \equiv \mathbb{N} \times \triangleright^\kappa \mathsf{Str}^\kappa$. One can then use dfix for recursive programming with guarded streams, e.g., when defining a constant stream of zeros as $\mathsf{dfix}^\kappa(\lambda x.\,(0, x))$. The type of dfix ensures that only productive recursive definitions are typeable, e.g., $\mathsf{dfix}^\kappa(\lambda x.x)$ is not.

The tick constant $\diamond$ gives a way to execute a delayed computation t of type $\triangleright^\kappa A$ to compute a value of type A. In particular, if t is a fixed point, application to the tick constant unfolds the fixed point once. This explains the need to name ticks in CloTT: substitution of $\diamond$ for a tick variable α in a term allows for all fixed points applied to α in the term to be unfolded. In particular, the names of ticks are crucial for the strong normalisation result for CloTT in [3].

To ensure productivity, application of $\diamond$ must be restricted. In particular a term such as $\mathsf{dfix}^{\kappa}(\lambda x : \mathsf{Str}^{\kappa}.x\,[\diamond])$ should not be well typed. The typing rule for application to the tick constant ensures this by assuming that the clock κ associated to the delay is not free in the context of the term t. For example, the rule

$$\frac{\Gamma \vdash_{\Delta,\kappa} t : \triangleright\,(\alpha : \kappa).A \qquad \Gamma \vdash_{\Delta}}{\Gamma \vdash_{\Delta,\kappa} t\,[\diamond] : A\,[\diamond/\alpha]}$$

is admissible, which can be proved using weakening lemma for the clock variable context. This rule, however, is not closed under variable substitution, which is the motivation for the more general rule of Figure 1. The typing rule is a bit unusual, in that it involves substitution in the term in the conclusion. We shall see in Section 7.1 that this causes extra proof obligations for welldefinedness of the denotational semantics.

Universal quantification over clocks allow for coinductive types to be encoded using guarded recursive types [2]. For example $\mathsf{Str} \stackrel{\mathsf{def}}{=} \forall\kappa.\mathsf{Str}^{\kappa}$ is a coinductive type of streams. The head and tail maps $\mathsf{hd} : \mathsf{Str} \to \mathbb{N}$ and $\mathsf{tl} : \mathsf{Str} \to \mathsf{Str}$ can be defined as

$$\mathsf{hd}(xs) \stackrel{\mathsf{def}}{=} \pi_1(xs[\kappa_0]) \qquad\qquad \mathsf{tl}(xs) \stackrel{\mathsf{def}}{=} \Lambda\kappa.((\pi_2(xs[\kappa]))\,[\diamond])$$

using the clock constant κ_0.

Finally we recall the *clock irrelevance axiom*

$$\frac{\Gamma \vdash_{\Delta} t : \forall\kappa.A \qquad \Gamma \vdash_{\Delta} A\ \mathrm{type}}{\Gamma \vdash_{\Delta} \mathsf{cirr}^{\kappa} t : \forall\kappa'.\forall\kappa''.t[\kappa'] =_A t[\kappa'']} \tag{1}$$

crucial for correctness of the encoding of coinductive types [2]. Note that the hypothesis implies that κ is not free in A. This rule can be used to prove that $\forall\kappa.A$ is isomorphic to A if κ is not free in A. Likewise the *tick irrelevance axiom*

$$\frac{\Gamma \vdash_{\Delta} t : \triangleright^{\kappa} A}{\mathsf{tirr}^{\kappa} t : \triangleright\,(\alpha : \kappa).\triangleright\,(\alpha' : \kappa).t\,[\alpha] =_A t\,[\alpha']} \tag{2}$$

states that the identity of ticks is irrelevant for the equality theory, despite being crucial for the reduction semantics. Tick irrelevance implies fixed point unfolding

$$\frac{\Gamma \vdash_{\Delta,\kappa} f : \triangleright^{\kappa} A \to A \qquad \Gamma \vdash_{\Delta} \qquad \kappa' \in \Delta}{\Gamma \vdash_{\Delta} \mathsf{pfix}^{\kappa'} f[\kappa'/\kappa] : \triangleright\,(\alpha : \kappa).(\mathsf{dfix}^{\kappa'} f[\kappa'/\kappa])\,[\alpha] =_A (f(\mathsf{dfix}^{\kappa'} f))[\kappa'/\kappa]}$$

The type theory CloTT as defined in [3] also has guarded recursive types and a universe. We leave these for future work, see Section 9.

4 Presheaf semantics

The setting for the denotational semantics of CloTT is a category of covariant presheaves over a category $\mathbb{T}$ of time objects. This category has previously been used to give a model of GDTT [10].

We will assume given a countably infinite set CV of (semantic) clock variables, for which we use $\lambda, \lambda', \ldots$ to range over. A *time object* is a pair $(\Theta; \vartheta)$ where Θ is a finite subset of CV and $\vartheta : \Theta \to \mathbb{N}$ is a map giving the number of ticks left on each clock in Θ. We will write the finite sets Θ as lists writing e.g., Θ, λ for $\Theta \cup \{\lambda\}$ and $\vartheta[\lambda \mapsto n]$ for the extension of ϑ to Θ, λ, or indeed for the update of ϑ, if ϑ is already defined on λ. A morphism $(\Theta; \vartheta) \to (\Theta'; \vartheta')$ is a

function $\tau : \Theta \to \Theta'$ such that $\vartheta'\tau \leq \vartheta$ in the pointwise order. The inequality allows for time to pass in a morphism, but morphisms can also synchronise clocks in Θ by mapping them to the same clock in Θ', or introduce new clocks if τ is not surjective. Define GR to be the category $\mathsf{Set}^{\mathbb{T}}$ of covariant presheaves on $\mathbb{T}$. The topos of trees can be seen as a restriction of this where time objects always have a single clock.

The category GR contains a special object of clocks, given by the first projection $\mathrm{Clk}(\Theta;\vartheta) = \Theta$. If Δ is a set, one can form the object Clk^Δ as $\mathrm{Clk}^\Delta(\Theta;\vartheta) = \Theta^\Delta$. Let $\mathbb{T}_\Delta$ be the category of elements of Clk^Δ, i.e., the objects are triples $(\Theta;\vartheta;f)$ where $(\Theta;\vartheta) \in \mathbb{T}$ and $f : \Delta \to \Theta$ and a morphism $\tau : (\Theta;\vartheta;f) \to (\Theta';\vartheta';f')$ is a morphism $\tau : (\Theta;\vartheta) \to (\Theta';\vartheta')$ such that $\tau \circ f = f'$. A clock context Δ will be interpreted as Clk^Δ and contexts, types and terms in clock context Δ will be modelled in the category $\mathsf{GR}[\Delta] \stackrel{\mathsf{def}}{=} \mathsf{Set}^{\mathbb{T}_\Delta}$ of covariant presheaves over $\mathbb{T}_\Delta$. If F is a covariant presheaf over $\mathsf{GR}[\Delta]$ and $\tau : (\Theta;\vartheta;f) \to (\Theta';\vartheta';f')$ and $x \in F(\Theta;\vartheta;f)$ we will write $\tau \cdot x$ for $F(\tau)(x) \in F(\Theta';\vartheta';f')$.

To describe the model of CloTT, we start by fixing a clock context Δ and modelling the fragment of CloTT excluding universal quantification over clocks and the tick constant $\diamond$. The resulting fragment is a version of the tick calculus with one notion of tick for each clock κ in Δ. To model this, we need the structure of a CwF with adjunction on $\mathsf{GR}[\Delta]$ for each κ in Δ. Recall that, like any presheaf category, $\mathsf{GR}[\Delta]$ can be equipped with the structure of a CwF where contexts are objects, types in context Γ are presheaves over the elements of Γ and terms are sections. Precisely, a type over Γ is a mapping associating a set $A(\gamma)$ to each $\gamma \in \Gamma(\Theta;\vartheta;f)$ and to each $\tau : (\Theta;\vartheta;f) \to (\Theta';\vartheta';f')$ a mapping $\tau \cdot (-) : A(\gamma) \to A(\tau \cdot \gamma)$ such that $\mathsf{id} \cdot x = x$ and $(\rho\tau) \cdot x = \rho \cdot (\tau \cdot x)$ for all x, τ and ρ. A term is a mapping associating to each γ an element $t(\gamma) \in A(\gamma)$ such that $t(\tau \cdot \gamma) = \tau \cdot t(\gamma)$. We often make the underlying $\mathbb{T}_\Delta$ object explicit writing $t_{(\Theta;\vartheta;f)}(\gamma)$.

As an example of a model of a type, recall the type of guarded streams satisfying $\mathsf{Str}^\kappa \equiv \mathbb{N} \times \triangleright\mathsf{Str}^\kappa$ from Section 3. This is a closed type in a clock context Δ (assuming $\kappa \in \Delta$), and so will be interpreted as a presheaf in $\mathsf{GR}[\Delta]$ defined as $[\![\mathsf{Str}^\kappa]\!](\Theta;\vartheta;f) = \mathbb{N}^{\vartheta(f(\kappa))+1} \times \{*\}$. We will assume that the products in this associate to the right, so that this is the type of tuples of the form $(n_{\vartheta(f(\kappa))}, (\dots, (n_0, *))\dots)$. This is needed to model the equality $\mathsf{Str}^\kappa \equiv \mathbb{N} \times \triangleright\mathsf{Str}^\kappa$, rather than just an isomorphism of types. Given a predicate $x{:}\mathsf{N} \vdash P$, we can lift it to a predicate $y : \mathsf{Str}^\kappa \vdash \hat{P}$ satisfying $\hat{P}(x :: xs) \equiv P(x) \times \triangleright(\alpha : \kappa).\hat{P}(xs\,[\alpha])$ as in Section 2, by defining

$$[\![\hat{P}]\!]_{(\Theta;\vartheta;f)}(n_{\vartheta(f(\kappa))}, (\dots, (n_0, *))\dots) = \{(x_{\vartheta(f(\kappa))}, (\dots, (x_0, *))\dots) \mid \forall i.x_i \in [\![P]\!]_{(\Theta;\vartheta;f)}(n_i)\}$$

It is a simple calculation (using the definitions below) that these interpretations model the type equalities mentioned above.

4.1 Adjunction structure on GR[Δ]

For the adjunction, recall that in the topos of trees the functor $\blacktriangleright$ is defined as $(\blacktriangleright F)(n+1) = Fn$ and $(\blacktriangleright F)(0) = \{*\}$. This has a left adjoint $\blacktriangleleft$ defined as $(\blacktriangleleft F)n = F(n+1)$. The right adjoint generalises in a straight forward way to the multiclock setting of CloTT: If F is in $\mathsf{GR}[\Delta]$, define

$$(\blacktriangleright^\kappa F)(\Theta;\vartheta;f) = \begin{cases} F(\Theta;\vartheta[f(\kappa)-];f) & \vartheta(f(\kappa)) > 0 \\ \{*\} & \text{otherwise} \end{cases}$$

where $\vartheta[f(\kappa)-](f(\kappa)) = \vartheta(f(\kappa)) - 1$ and $\vartheta[f(\kappa)-](\lambda) = \vartheta(\lambda)$ for $\lambda \neq f(\kappa)$. This is the same definition as used in the GDTT model of [10].

▶ **Lemma 4.** *The functor $\blacktriangleright^\kappa$ extends to types and terms.*

Proof. We just give the definitions. For $\gamma \in (\blacktriangleright^\kappa [\![\Gamma]\!])(\Theta;\vartheta;f)$ define

$$(\blacktriangleright^\kappa [\![A]\!])_{(\Theta;\vartheta;f)}(\gamma) = \begin{cases} \{*\} & \vartheta(f(\kappa)) = 0 \\ [\![A]\!]_{(\Theta;\vartheta[f(\kappa)-];f)}(\gamma) & \text{otherwise} \end{cases}$$

The case for terms is similar.

The isomorphism $\zeta_{\Gamma,A}$ is given by $\zeta_{\Gamma,A_{(\Theta;\vartheta;f)}} = \begin{cases} \langle *, * \rangle \mapsto * & \vartheta(f(\kappa)) = 0 \\ \mathsf{id} & \text{otherwise} \end{cases}$ ◀

At first sight it would seem that one can define a left adjoint to the above functor given by $(\blacktriangleleft^\kappa F)(\Theta;\vartheta;f) = F(\Theta;\vartheta[f(\kappa)+];f)$, where $\vartheta[f(\kappa)+]$ is defined similarly to $\vartheta[f(\kappa)-]$. Unfortunately, $\blacktriangleleft^\kappa F$ so described is not a presheaf because it has no well-defined action on maps since a map $\tau : (\Theta;\vartheta;f) \to (\Theta';\vartheta';f')$ does not necessarily induce a map $(\Theta;\vartheta[f(\kappa)+];f) \to (\Theta';\vartheta'[f'(\kappa)+];f')$: If $\tau(f(\kappa)) = \tau(\lambda)$ there is no guarantee that $\vartheta'[f'(\kappa)+](\tau(\lambda)) \leq \vartheta[f(\kappa)+](\lambda)$.

To get the correct description of the left adjoint consider the set $f^{-1}(f(\kappa)) \subseteq \Delta$ of syntactic clocks synchronised with κ by f. Given a morphism $\tau : (\Theta;\vartheta;f) \to (\Theta';\vartheta';f')$, more clocks can be synchronised with κ by f' than f, but never fewer. If we think of time as flowing in the direction of morphisms, the left adjoint must take into account all the possible ways that κ could have been synchronised with fewer syntactic clocks "in the past". Such a past is given by a subset $X \subset f^{-1}(f(\kappa))$ such that $\kappa \in X$.

▶ **Lemma 5.** *The functor $\blacktriangleright^\kappa$ has a left adjoint $\blacktriangleleft^\kappa$ given by*

$$\blacktriangleleft^\kappa F(\Theta;\vartheta;f) = \coprod_{\kappa \in X \subset f^{-1}(f(\kappa))} F(\Theta;\vartheta;f)[X,\kappa+]$$

where $(\Theta;\vartheta;f)[X,\kappa+] = (\Theta, \#_\Theta; \vartheta[\#_\Theta \mapsto \vartheta(f(\kappa))+1]; f[X \mapsto \#_\Theta])$ for $\#_\Theta$ a chosen clock name fresh for Θ.

Finally, the projection $\mathsf{p}_{\blacktriangleleft^\kappa} : \blacktriangleleft^\kappa \to \mathsf{id}$ maps an element (X,γ) in $\blacktriangleleft^\kappa F(\Theta;\vartheta;f)$ to $\chi \cdot \gamma$ where

$$\chi : (\Theta;\vartheta;f)[X,\kappa+] \to (\Theta;\vartheta;f)$$

is defined as $\chi(\#_\Theta) = f(\kappa)$ and $\chi(\lambda) = \lambda$ for $\lambda \in \Theta$. Collectively, Lemmas 4 and 5 together with the projection $\mathsf{p}_{\blacktriangleleft^\kappa}$ state that for each κ, $\mathsf{GR}[\Delta]$ carries the structure of a model of the tick calculus. This is enough to model the tick abstractions and applications of CloTT.

The adjoint correspondent $\overline{\mathsf{p}_{\blacktriangleleft^\kappa}} : \mathsf{id} \to \blacktriangleright^\kappa$ to $\mathsf{p}_{\blacktriangleleft^\kappa}$ maps an element $\gamma \in F(\Theta;\vartheta;f)$ to its restriction in $F(\Theta;\vartheta[f(\kappa)-];f)$. This is the map referred to as next in [10]. Moreover, a simple calculation shows that the interpretation of $\triangleright^\kappa A$, i.e., $\triangleright(\alpha:\kappa).A$ for α not free in A, is the same as in [10], namely

$$[\![\Gamma \vdash_\Delta \triangleright^\kappa A \text{ type}]\!]_{(\Theta;\vartheta;f)}(\gamma) = [\![A]\!]_{(\Theta;\vartheta[f(\kappa)-];f)}(\gamma|_{(\Theta;\vartheta[f(\kappa)-];f)})$$

We can thus define the interpretation of dfix as in [10] by induction on $\vartheta(f(\kappa))$:

$$[\![\mathsf{dfix}\, t]\!]_{(\Theta;\vartheta;f)}(\gamma) = \begin{cases} * & \vartheta(f(\kappa)) = 0 \\ [\![t\ (\mathsf{dfix}\ t)]\!]_{(\Theta;\vartheta[f(\kappa)-];f)}(\gamma|_{(\Theta;\vartheta[f(\kappa)-];f)}) & \text{Otherwise} \end{cases}$$

Finally we note soundness of the tick irrelevance axiom (2).

▶ **Proposition 6.** *If $\Gamma \vdash_\Delta t : \triangleright^\kappa A$ then*

$$[\![\Gamma, \alpha:\kappa, \alpha':\kappa \vdash_\Delta t\,[\alpha] : A]\!] = [\![\Gamma, \alpha:\kappa, \alpha':\kappa \vdash_\Delta t\,[\alpha'] : A]\!]$$

5 Substitutions

Having described the interpretation of the fragment of CloTT that lives within a fixed clock context Δ it remains to describe the interpretation of the universal quantification over clocks and of the tick constant $\diamond$. Quantification over clocks can be seen as a dependent product over a type of clocks, and should therefore be modelled as a right adjoint to weakening in the clock context. Weakening is an example of a substitution and, as we shall see, the tick constant $\diamond$ will also be modelled using a form of substitution. We therefore first study substitutions, which are non-standard in CloTT because of the two contexts, and because of the unusual typing rules for ticks.

5.1 Syntactic substitutions

A syntactic substitution from $\Gamma \vdash_\Delta$ to $\Gamma' \vdash_{\Delta'}$ is a pair (ν, σ) of a substitution ν of clocks for clocks and a substitution σ of terms for variables and ticks for ticks variables. Substitutions are formed according to the following rules.

- If $\nu : \Delta' \to \Delta$ is a map of sets, then $(\nu, \cdot) : \Gamma \vdash_\Delta \to \cdot \vdash_{\Delta'}$
- If $(\nu, \sigma) : \Gamma \vdash_\Delta \to \Gamma' \vdash_{\Delta'}$ and $\Gamma \vdash_\Delta t : A\,(\nu, \sigma)$ then $(\nu, \sigma[x \mapsto t]) : \Gamma \vdash_\Delta \to \Gamma', x : A \vdash_{\Delta'}$
- If $(\nu, \sigma) : \Gamma \vdash_\Delta \to \Gamma' \vdash_{\Delta'}$ and $\Gamma, \alpha : \nu(\kappa), \Gamma'' \vdash_\Delta$ and $\Gamma', \beta : \kappa \vdash_{\Delta'}$ are welformed then $(\nu, \sigma[\beta \mapsto \alpha]) : \Gamma, \alpha : \nu(\kappa), \Gamma'' \vdash_\Delta \to \Gamma', \beta : \kappa \vdash_{\Delta'}$
- If $(\nu, \sigma) : \Gamma \vdash_\Delta \to \Gamma' \vdash_{\Delta'}$, $\kappa \notin \Delta'$ and $\kappa' \in \Delta$, then

$$(\nu[\kappa \mapsto \nu(\kappa')], \sigma[\alpha \mapsto \diamond]) : \Gamma \vdash_\Delta \to \Gamma', \alpha : \kappa \vdash_{\Delta', \kappa}$$

Here $A\,(\nu, \sigma)$ is the result of substituting A along (ν, σ) which is defined in the standard way.

5.2 Semantic substitutions

The clock substitution ν gives rise to a functor $\mathbb{T}_\Delta \to \mathbb{T}_{\Delta'}$ mapping an object $(\Theta; \vartheta; f)$ to $(\Theta; \vartheta; f\nu)$, and this induces a functor $\nu^* : \mathsf{GR}[\Delta'] \to \mathsf{GR}[\Delta]$ by $(\nu^* F)(\Theta; \vartheta; f) = F(\Theta; \vartheta; f\nu)$. This functor extends to a morphism of CwFs [13], in particular it maps a type A in context Γ in the CwF structure of $\mathsf{GR}[\Delta']$ to a type $\nu^* A$ in context $\nu^*\Gamma$ the CwF structure of $\mathsf{GR}[\Delta]$, and likewise for terms. For example, $(\nu^* A)(\gamma)$ for $\gamma \in \nu^*\Gamma(\Theta; \vartheta; f) = \Gamma(\Theta; \vartheta; f\nu)$ is defined as $A\gamma$. Moreover, this map commutes on the nose with comprehension and substitution. For example, if A is a type in context Γ in $\mathsf{GR}[\Delta']$ and $\gamma : \Gamma' \to \Gamma$, then $(\nu^* A)[\nu^*\gamma] = \nu^*(A[\gamma])$. Moreover, it commutes with $\blacktriangleright$ in the following sense.

▶ **Lemma 7.** *If $\nu : \Delta' \to \Delta$ and $\kappa \in \Delta'$ then $\nu^* \circ \blacktriangleright^\kappa = \blacktriangleright^{\nu(\kappa)} \circ \nu^*$.*

The interpretation of a substitution (ν, σ) is a morphism

$$[\![(\nu, \sigma)]\!] : [\![\Gamma \vdash_\Delta]\!] \to \nu^*[\![\Gamma \vdash'_{\Delta'}]\!]$$

in $\mathsf{GR}[\Delta]$, which we will define below. But first we state the substitution lemma, which must be proved by induction on terms and types simultaneously with the definition of the interpretation, as is standard for models of dependent type theory.

▶ **Lemma 8.** *Let $(\nu, \sigma) : \Gamma \vdash_\Delta \to \Gamma' \vdash_{\Delta'}$ be a substitution and let $\Gamma' \vdash_{\Delta'} J$ be a judgement of a wellformed type or a typing judgement. Then $[\![J\,(\nu, \sigma)]\!] = (\nu^*[\![J]\!])[[\![(\nu, \sigma)]\!]]$.*

The main difficulty for defining the interpretation of substitutions is that the operator $\blacktriangleleft^\kappa$ does not commute with clock substitutions in the sense that $\nu^*(\blacktriangleleft^\kappa\,\Gamma)$ is not necessarily equal to $\blacktriangleleft^{\nu(\kappa)}\,(\nu^*\Gamma)$. However, we can define a map in the relevant direction:

$$e^{\kappa,\nu}_\Gamma = \epsilon^{\nu(\kappa)}_{\nu^*(\blacktriangleleft^\kappa\Gamma)} \circ \blacktriangleleft^{\nu(\kappa)}\,\nu^*(\eta^\kappa_\Gamma) : \blacktriangleleft^{\nu(\kappa)}\,(\nu^*\Gamma) \to \nu^*(\blacktriangleleft^\kappa\,\Gamma)$$

where $\eta^\kappa_\Gamma : \Gamma \to \blacktriangleright^\kappa\blacktriangleleft^\kappa\,\Gamma$ is the unit of the adjunction and

$$\epsilon^{\nu(\kappa)}_{\nu^*(\blacktriangleleft^\kappa\Gamma)} : \blacktriangleleft^{\nu(\kappa)}\blacktriangleright^{\nu(\kappa)}\,\nu^*(\blacktriangleleft^\kappa\,\Gamma) \to \nu^*(\blacktriangleleft^\kappa\,\Gamma)$$

is the counit. The composition type checks since $\nu^*\,\blacktriangleright^\kappa = \blacktriangleright^{\nu(\kappa)}\,\nu^*$.

The map $e^{\kappa,\nu}$ has a simple description in the model: $e^{\kappa,\nu}_\Gamma$ maps an element (X,γ) in

$$\begin{aligned}
&\blacktriangleleft^{\nu(\kappa)}\,(\nu^*\Gamma)(\Theta;\vartheta;f)\\
&\qquad = \coprod_{\substack{\nu(k)\in X\\ f(X)=f(\nu(\kappa))}} (\nu^*\Gamma)(\Theta,\#_\Theta;\vartheta[\#_\Theta\mapsto\vartheta((f\circ\nu)(\kappa))+1];f[X\mapsto\#_\Theta])\\
&\qquad = \coprod_{\substack{\nu(k)\in X\\ f(X)=f(\nu(\kappa))}} \Gamma(\Theta,\#_\Theta;\vartheta[\#_\Theta\mapsto\vartheta((f\circ\nu)(\kappa))+1];f\circ\nu[\nu^{-1}X\mapsto\#_\Theta])
\end{aligned}$$

to $(\nu^{-1}X,\gamma)$ in the set $\nu^*(\blacktriangleleft^\kappa\,\Gamma)(\Theta;\vartheta;f)$ which equals

$$\blacktriangleleft^\kappa\,\Gamma(\Theta;\vartheta;f\circ\nu) = \coprod_{\substack{\kappa\in Y\\ f(\nu(Y))=f(\nu(\kappa))}} \Gamma(\Theta,\#_\Theta;\vartheta[\#_\Theta\mapsto\vartheta((f\circ\nu)(\kappa))+1];f\circ\nu[Y\mapsto\#_\Theta])$$

The interpretation of substitutions is defined as

$$\begin{aligned}
[\![\cdot]\!] &= x\mapsto\star\\
[\![(\nu,\sigma[x\mapsto t])]\!] &= \langle[\![(\nu,\sigma)]\!],[\![t]\!]\rangle\\
[\![(\nu,\sigma[\beta\mapsto\alpha])]\!] &= e^\kappa_{[\![\Gamma']\!]}\circ\blacktriangleleft^{\nu(\kappa)}\,[\![(\nu,\sigma)]\!]\circ \mathsf{p}_{\Gamma''}\\
[\![(\nu[\kappa\mapsto\nu(\kappa')],\sigma[\alpha\mapsto\diamond])]\!] &= \nu^*[\![([\kappa\mapsto\kappa'],[\alpha\mapsto\diamond])]\!]\circ[\![(\nu,\sigma)]\!]
\end{aligned}$$

where we have assumed the types as in the rules for forming substitutions and $\mathsf{p}_{\Gamma''}$ is the context projection defined as in Section 2.2. The last case uses $[\![([\kappa\mapsto\kappa'],[\alpha\mapsto\diamond])]\!]$ which will be defined in Section 7 below.

The rest of this section is a sketch proof of the substitution lemma for the fragment of CloTT modelled so far, i.e., excluding quantification over clocks and $\diamond$. As we extend the interpretation we will also extend the proof of the substitution lemma.

The proof is by induction over judgements and is simultaneous with the definition of the interpretation of substitutions. The cases of standard dependent type theory (dependent functions and sums, identity types) can be essentially reduced to the standard proof of the substitution lemma for dependent type theory in presheaf models as follows (although the non-standard notion of substitution requires some new lemmas). Since $\mathsf{GR}[\Delta]$ is the object of presheaves over $\mathbb{T}_\Delta$, the category of elements of $[\![\Delta]\!] = \mathrm{Clk}^\Delta$, given a context $\Gamma\vdash_\Delta$ one can form the comprehension $[\![\Delta]\!].[\![\Gamma]\!]$ as an object of GR. Types and terms in context $[\![\Delta]\!].[\![\Gamma]\!]$ in the CwF structure associated to GR are then in bijective correspondence with those over $[\![\Gamma]\!]$ in the CwF structure of $\mathsf{GR}[\Delta]$. A substitution $(\nu,\sigma):\Gamma\vdash_\Delta\to\Gamma'\vdash_{\Delta'}$ induces a morphism $[\![\Delta]\!].[\![\Gamma]\!]\to[\![\Delta']\!].[\![\Gamma']\!]$ in GR and the substitution defined above corresponds to substitution along this map. Thus the interpretation of the standard type theoretic constructions are

the same as the standard ones in the presheaf model GR, and the corresponding cases of the substitution lemma can be proved similarly to the standard proof of the substitution lemma for presheaf models of type theory.

The cases corresponding to the constructions from the tick calculus follow from the following two equations.

$$\blacktriangleright^{\nu(\kappa)}\, e^{\kappa,\nu} \circ \eta^{\nu(\kappa)}_{\nu^*} = \nu^*(\eta^\kappa) \qquad : \nu^* \to \blacktriangleright^{\nu(\kappa)}\, \nu^* \blacktriangleleft^\kappa = \nu^* \blacktriangleright^\kappa \blacktriangleleft^\kappa \tag{3}$$

$$\nu^*(\epsilon^\kappa) \circ e^{\kappa,\nu}_{\blacktriangleright^\kappa} = \epsilon^{\nu(\kappa)}_{\nu^*} \qquad : \blacktriangleleft^{\nu(\kappa)}\, \nu^* \blacktriangleright^\kappa = \blacktriangleleft^{\nu(\kappa)} \blacktriangleright^{\nu(\kappa)}\, \nu^* \to \nu^* \tag{4}$$

For example, the case of $\triangleright\,(\alpha : \kappa).A$ can be proved as follows (writing $[\![\sigma]\!]$ for $[\![(\nu, \sigma)]\!]$)

$$\begin{aligned}
\nu^*[\![\Gamma' \vdash_{\Delta'} \triangleright\,(\alpha : \kappa).A]\!][\![\sigma]\!] &= \nu^* \left((\blacktriangleright^\kappa\, [\![\Gamma', \alpha : \kappa \vdash_{\Delta'} A]\!])\, [\eta^\kappa_{[\![\Gamma']\!]}] \right) [\![\sigma]\!] \\
&= \left(\blacktriangleright^{\nu(\kappa)}\, \nu^*[\![\Gamma', \alpha : \kappa \vdash_{\Delta'} A]\!] \right) [\nu^* \eta^\kappa_{[\![\Gamma']\!]}][\![\sigma]\!] \\
&= \left(\blacktriangleright^{\nu(\kappa)}\, \nu^*[\![\Gamma', \alpha : \kappa \vdash_{\Delta'} A]\!] \right) [\blacktriangleright^{\nu(\kappa)}\, e^\kappa \circ \eta^{\nu(\kappa)}_{\nu^*[\![\Gamma']\!]} \circ [\![\sigma]\!]] \\
&= \left(\blacktriangleright^{\nu(\kappa)}\, \nu^*[\![\Gamma', \alpha : \kappa \vdash_{\Delta'} A]\!] \right) [\blacktriangleright^{\nu(\kappa)}\, (e^\kappa \circ \blacktriangleleft^{\nu(\kappa)}\, [\![\sigma]\!]) \circ \eta^{\nu(\kappa)}_{\nu^*[\![\Gamma']\!]}] \\
&= \left(\blacktriangleright^{\nu(\kappa)}\, (\nu^*[\![\Gamma', \alpha : \kappa \vdash_{\Delta'} A]\!])\, [e^\kappa \circ \blacktriangleleft^{\nu(\kappa)}\, [\![\sigma]\!]] \right) [\eta^{\nu(\kappa)}_{\nu^*[\![\Gamma']\!]}] \\
&= \left(\blacktriangleright^{\nu(\kappa)}\, [\![\Gamma, \beta : \nu(\kappa) \vdash_\Delta A\, (\nu, \sigma[\alpha \mapsto \beta])]\!] \right) [\eta^{\nu(\kappa)}_{\nu^*[\![\Gamma']\!]}] \\
&= [\![\Gamma \vdash_\Delta \triangleright\,(\beta : \nu(\kappa)).(A\, (\nu, \sigma[\alpha \mapsto \beta]))]\!] \\
&= [\![\Gamma \vdash_\Delta (\triangleright\,(\alpha : \kappa).A)\, (\nu, \sigma)]\!]
\end{aligned}$$

6 Interpretation of clock quantification

Universal quantification over clocks should be modelled as a right adjoint to the semantic correspondent to clock weakening. Syntactically, clock weakening from context $\Gamma \vdash_\Delta$ to $\Gamma \vdash_{\Delta,\kappa}$ corresponds to the substitution (i, id_Γ), where i is the inclusion of Δ into Δ, κ. Recall from Section 5 that types and terms in context $[\![\Gamma \vdash_\Delta]\!]$ in the CwF structure of $\mathsf{GR}[\Delta]$ correspond to types and terms in context $[\![\Delta]\!].[\![\Gamma \vdash_\Delta]\!]$ in GR, and recall that $[\![\Delta]\!] = \mathrm{Clk}^\Delta$. By the substitution lemma, clock weakening from context $\Gamma \vdash_\Delta$ to $\Gamma \vdash_{\Delta,\kappa}$ corresponds to substitution along the composition

$$[\![\Delta, \kappa]\!].[\![\Gamma \vdash_{\Delta,\kappa}]\!] \xrightarrow{[\![\Delta,\kappa]\!].[\![(i,\mathsf{id}_\Gamma)]\!]} [\![\Delta, \kappa]\!].i^*[\![\Gamma \vdash_\Delta]\!] \to [\![\Delta]\!].[\![\Gamma \vdash_\Delta]\!]$$

All such substitutions have right adjoints, but to get a simple description of this (and to satisfy the substitution lemma), we will give a concrete description which can be briefly described as follows: To model $\forall\kappa.A$, open a fresh semantic clock $\#$, map κ to $\#$ and take the limit of $[\![A]\!]$ as $\#$ ranges over all natural numbers. To type this description we need the following lemma.

▶ **Lemma 9.** *Let $(\Theta; \vartheta; f)$ be an object of $\mathsf{GR}[\Delta]$, let $\#$ be fresh for Θ and let $\iota : \Theta \to \Theta, \#$ be the inclusion. The component of $[\![(i, \mathsf{id}_\Gamma)]\!]$ at $(\Theta, \#; \vartheta[\# \mapsto n]; f[\kappa \mapsto \#])$ is an isomorphism*

$$[\![\Gamma \vdash_{\Delta,\kappa}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])} \to i^*[\![\Gamma \vdash_\Delta]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])} = [\![\Gamma \vdash_\Delta]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];\iota f)}$$

If $\kappa' \in \Delta$, the inverse to $[\![(i, \mathsf{id}_\Gamma)]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])}$ is given by the component of

$$i^*([\![(\mathsf{id}_\Delta[\kappa \mapsto \kappa'], \mathsf{id}_\Gamma)]\!]) : i^*[\![\Gamma \vdash_\Delta]\!] \to i^*(\mathsf{id}_\Delta[\kappa \mapsto \kappa'])^*[\![\Gamma \vdash_{\Delta,\kappa}]\!] = [\![\Gamma \vdash_{\Delta,\kappa}]\!]$$

at $(\Theta, \#; \vartheta[\# \mapsto n]; f[\kappa \mapsto \#]))$.

We can now define the interpretation of universal quantification over clocks:

$$\llbracket \Gamma \vdash_\Delta \forall\kappa.A \text{ type}\rrbracket_{(\Theta;\vartheta;f)}(\gamma) \stackrel{\text{def}}{=}$$
$$\{(\omega_n) \in \prod_{n\in\mathbb{N}} \llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])}(\llbracket(i,\mathsf{id}_\Gamma)\rrbracket^{-1}(\iota\cdot\gamma)) \mid \forall n.\omega_n = (\omega_{n+1})|_n\}$$

Here, by assumption $\gamma \in \llbracket \Gamma \vdash_\Delta \rrbracket_{(\Theta;\vartheta;f)}(\gamma)$ and so

$$\iota\cdot\gamma \in \llbracket \Gamma \vdash_\Delta \rrbracket_{(\Theta,\#;\vartheta[\#\mapsto n];\iota f)} = i^*\llbracket \Gamma \vdash_\Delta \rrbracket_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])}$$

which means that $\llbracket(i,\mathsf{id}_\Gamma)\rrbracket^{-1}(\iota\cdot\gamma) \in \llbracket \Gamma \vdash_{\Delta,\kappa} \rrbracket_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])}$ and thus the type of each ω_n is a welldefined set. In the condition for the families, $(\omega_{n+1})|_n$ is the restriction of ω_{n+1} along the map $(\Theta,\#;\vartheta[\#\mapsto n+1];f[\kappa\mapsto\#]) \to (\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])$ given by the identity on $\Theta,\#$. For the interpretation of terms define

$$\llbracket \Gamma \vdash_\Delta \Lambda\kappa.t : \forall\kappa.A\rrbracket_{(\Theta;\vartheta;f)} = (\llbracket \Gamma \vdash_{\Delta,\kappa} t : A\rrbracket_{(\Theta,\#;\vartheta[\#\mapsto n];\iota f)}(\llbracket(i,\mathsf{id}_\Gamma)\rrbracket^{-1}(\iota\cdot\gamma)))_n$$
$$\llbracket \Gamma \vdash_\Delta t[\kappa'] : A[\kappa'/\kappa]\rrbracket_{(\Theta;\vartheta;f)} = [\#\mapsto f(\kappa')]\cdot(\llbracket \Gamma \vdash_\Delta t : \forall\kappa.A\rrbracket_{(\Theta;\vartheta;f)}(\gamma))_{\vartheta(f(\kappa'))}$$

To see that the latter type checks, note first that

$$\llbracket \Gamma \vdash_\Delta t : \forall\kappa.A\rrbracket_{(\Theta;\vartheta;f)}(\gamma))_{\vartheta(f(\kappa'))} \in \llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket_{(\Theta,\#;\vartheta[\#\mapsto\vartheta(f(\kappa'))];f[\kappa\mapsto\#])}(\llbracket(i,\mathsf{id}_\Gamma)\rrbracket^{-1}(\iota\cdot\gamma))$$

and since $[\#\mapsto f(\kappa')] : (\Theta,\#;\vartheta[\#\mapsto\vartheta(f(\kappa'))];f[\kappa\mapsto\#]) \to (\Theta;\vartheta;f[\kappa\mapsto f(\kappa')])$, the right hand side of the definition is in

$$\begin{aligned}
&\llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket_{(\Theta;\vartheta;f[\kappa\mapsto f(\kappa')])}([\#\mapsto f(\kappa')]\cdot\llbracket(i,\mathsf{id}_\Gamma)\rrbracket^{-1}(\iota\cdot\gamma))\\
=&(\mathsf{id}_\Delta[\kappa\mapsto\kappa'])^*(\llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket)_{(\Theta;\vartheta;f)}([\#\mapsto f(\kappa')]\cdot\llbracket(i,\mathsf{id}_\Gamma)\rrbracket^{-1}(\iota\cdot\gamma))\\
=&(\mathsf{id}_\Delta[\kappa\mapsto\kappa'])^*(\llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket)_{(\Theta;\vartheta;f)}([\#\mapsto f(\kappa')]\cdot\llbracket(\mathsf{id}_\Delta[\kappa\mapsto\kappa'],\mathsf{id}_\Gamma)\rrbracket(\iota\cdot\gamma))\\
=&(\mathsf{id}_\Delta[\kappa\mapsto\kappa'])^*(\llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket)_{(\Theta;\vartheta;f)}(\llbracket(\mathsf{id}_\Delta[\kappa\mapsto\kappa'],\mathsf{id}_\Gamma)\rrbracket([\#\mapsto f(\kappa')]\cdot\iota\cdot\gamma))\\
=&(\mathsf{id}_\Delta[\kappa\mapsto\kappa'])^*(\llbracket \Gamma \vdash_{\Delta,\kappa} A \text{ type}\rrbracket)_{(\Theta;\vartheta;f)}(\llbracket(\mathsf{id}_\Delta[\kappa\mapsto\kappa'],\mathsf{id}_\Gamma)\rrbracket(\gamma))\\
=&\llbracket \Gamma \vdash_\Delta A[\kappa'/\kappa] \text{ type}\rrbracket_{(\Theta;\vartheta;f)}(\gamma)
\end{aligned}$$

where the last equality is by the substitution lemma.

▶ **Lemma 10.** *The β and η rules for universal quantification over clocks are sound for the interpretation.*

7 Interpretation of $\diamond$

To interpret the rule for the tick constant $\diamond$, we define a substitution

$$\llbracket([\kappa\mapsto\kappa'],[\alpha\mapsto\diamond])\rrbracket : \llbracket \Gamma \vdash_\Delta \rrbracket \to [\kappa\mapsto\kappa']^*\llbracket \Gamma,\alpha:\kappa \vdash_{\Delta,\kappa} \rrbracket$$

for every context $\Gamma \vdash_\Delta$ with $\kappa \notin \Delta$. The interpretation of application to $\diamond$ can then be defined as

$$\llbracket \Gamma \vdash_\Delta t[\kappa'/\kappa]\,[\diamond] : A[\kappa'/\kappa][\diamond/\alpha]\rrbracket = ([\kappa\mapsto\kappa']^*\llbracket \Gamma,\alpha:\kappa \vdash_{\Delta,\kappa} t : A\rrbracket)[\llbracket([\kappa\mapsto\kappa'],[\alpha\mapsto\diamond])\rrbracket]$$

which has type

$$([\kappa\mapsto\kappa']^*\llbracket \Gamma,\alpha:\kappa \vdash_{\Delta,\kappa} A \text{ type}\rrbracket)[\llbracket([\kappa\mapsto\kappa'],[\alpha\mapsto\diamond])\rrbracket]$$

which equals the required $[\![\Gamma, \alpha : \kappa \vdash_{\Delta,\kappa} A[\kappa'/\kappa][\diamond/\alpha] \text{ type}]\!]$ by the substitution lemma.

Suppose $\gamma \in [\![\Gamma \vdash_\Delta]\!]_{(\Theta;\vartheta;f)}$. We define

$$[\![([\kappa \mapsto \kappa'], [\alpha \mapsto \diamond])]\!](\gamma) \stackrel{\mathsf{def}}{=} \Big(\{\kappa\}, [\![(i, \mathsf{id}_\Gamma)]\!]^{-1}(\iota \cdot \gamma))\Big)$$

where $\iota : (\Theta; \vartheta; f) \to (\Theta, \#; \vartheta[\# \mapsto n+1]; f)$ is the inclusion for $n = \vartheta(f(\kappa'))$. We must show that this defines an element of $[\kappa \mapsto \kappa']^*[\![\Gamma, \alpha : \kappa \vdash_{\Delta,\kappa}]\!]_{(\Theta;\vartheta;f)}$. To see this, note first that

$$\iota \cdot \gamma \in [\![\Gamma \vdash_\Delta]\!]_{(\Theta,\#;\vartheta[\#\mapsto n+1];f)} = i^*[\![\Gamma \vdash_\Delta]\!]_{(\Theta,\#;\vartheta[\#\mapsto n+1];f[\kappa\mapsto\#])}$$

so by Lemma 9,

$$[\![(i, \mathsf{id}_\Gamma)]\!]^{-1}(\iota \cdot \gamma) \in [\![\Gamma \vdash_{\Delta,\kappa}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n+1];f[\kappa\mapsto\#])}$$

and

$$\begin{aligned}\Big(\{\kappa\}, [\![(i, \mathsf{id}_\Gamma)]\!]^{-1}(\iota \cdot \gamma))\Big) &\in \coprod_{\kappa \in X \subset f^{-1}(f(\kappa))} [\![\Gamma \vdash_{\Delta,\kappa}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n+1];f[\kappa\mapsto f(\kappa')][X\mapsto\#])} \\ &= [\![\Gamma, \alpha : \kappa \vdash_{\Delta,\kappa}]\!]_{(\Theta;\vartheta;f[\kappa\mapsto f(\kappa')])} \\ &= [\kappa \mapsto \kappa']^*[\![\Gamma, \alpha : \kappa \vdash_{\Delta,\kappa}]\!]_{(\Theta;\vartheta;f)}\end{aligned}$$

We note that this satisfies the equality for $\diamond$ in Figure 1.

▶ **Lemma 11.** *The interpretations of* $(\mathsf{dfix}^{\kappa'}\, t)\,[\diamond]$ *and* $t\,(\mathsf{dfix}^{\kappa'}\, t)$ *are equal.*

7.1 Welldefinedness

As mentioned in Section 3 the unusual typing rule for $t\,[\diamond]$ introduces a problem of welldefinedness of the interpretation: If t is a term, a proof of the typing judgement $\Gamma \vdash_\Delta t\,[\diamond] : A[\diamond/\alpha]$ consists of a term s such that $s[\kappa'/\kappa] = t$, a type B such that $B[\kappa'/\kappa] = A$ and a proof of a typing judgement $\Gamma \vdash_{\Delta,\kappa} s : B$. In general there may be different possible choices of s and B, but the next lemma states that the interpretation of the term $t\,[\diamond]$ is independent of this choice. This means that the interpretation of a welltyped term is a welldefined object, independent of the choice of typing derivation.

▶ **Proposition 12.** *If* $\Gamma \vdash_{\Delta,\kappa} s : B$ *and* $\Gamma \vdash_{\Delta,\kappa} u : C$ *are such that* $\Gamma \vdash_\Delta s[\kappa'/\kappa] : B[\kappa'/\kappa]$ *is equal to* $\Gamma \vdash_\Delta u[\kappa'/\kappa] : C[\kappa'/\kappa]$ *then*

$$[\kappa \mapsto \kappa']^*[\![s\,[\alpha]]\!][[\![([\kappa \mapsto \kappa'], [\alpha \mapsto \diamond])]\!]] = [\kappa \mapsto \kappa']^*[\![u\,[\alpha]]\!][[\![([\kappa \mapsto \kappa'], [\alpha \mapsto \diamond])]\!]]$$

Proof. The assumption implies that also

$$\Gamma, \alpha : \kappa' \vdash_\Delta (s\,[\alpha])[\kappa'/\kappa] : B[\kappa'/\kappa] \qquad \text{and} \qquad \Gamma, \alpha : \kappa' \vdash_\Delta (u\,[\alpha])[\kappa'/\kappa] : B[\kappa'/\kappa]$$

are equal, and so by the substitution lemma

$$([\kappa \mapsto \kappa'][\![s\,[\alpha]]\!])[[\![(i, \mathsf{id}_\Gamma[\alpha \mapsto \alpha])]\!]] = ([\kappa \mapsto \kappa'][\![u\,[\alpha]]\!])[[\![(i, \mathsf{id}_\Gamma[\alpha \mapsto \alpha])]\!]]$$

Now, if $\gamma \in [\![\Gamma \vdash_\Delta]\!]_{(\Theta;\vartheta;f)}$ then

$$\begin{aligned}
&[\kappa \mapsto \kappa']^*[\![s\,[\alpha]]\!][\![([\kappa \mapsto \kappa'], [\alpha \mapsto \diamond])]\!](\gamma) \\
&\qquad = [\kappa \mapsto \kappa']^*[\![s\,[\alpha]]\!]\left(\{\kappa\}, [\![(i, \mathsf{id}_\Gamma)]\!]^{-1}(\iota \cdot \gamma)\right) \\
&\qquad = [\kappa \mapsto \kappa']^*[\![s\,[\alpha]]\!]\,(\{\kappa\}, [\![([\kappa \mapsto \kappa'], \mathsf{id}_\Gamma)]\!](\iota \cdot \gamma)) \\
&\qquad = [\kappa \mapsto \kappa']^*[\![s\,[\alpha]]\!][\![([\kappa \mapsto \kappa'], \mathsf{id}_\Gamma[\alpha \mapsto \alpha])]\!]\,(\{\kappa\}, (\iota \cdot \gamma)) \\
&\qquad = [\kappa \mapsto \kappa']^*[\![u\,[\alpha]]\!][\![([\kappa \mapsto \kappa'], \mathsf{id}_\Gamma[\alpha \mapsto \alpha])]\!]\,(\{\kappa\}, (\iota \cdot \gamma)) \\
&\qquad = [\kappa \mapsto \kappa']^*[\![u\,[\alpha]]\!][\![([\kappa \mapsto \kappa'], [\alpha \mapsto \diamond])]\!](\gamma)
\end{aligned}$$ ◀

8 Clock irrelevance

We now show how to model the clock irrelevance axiom (1). For this it suffices to show that if $\Gamma \vdash_\Delta t : \forall\kappa.A$, $\Gamma \vdash_\Delta A$ type and $\kappa', \kappa'' \in \Delta$ then

$$[\![\Gamma \vdash_\Delta t[\kappa'] : A]\!] = [\![\Gamma \vdash_\Delta t[\kappa''] : A]\!]$$

Recall that for $\gamma \in [\![\Gamma \vdash_\Delta]\!]_{(\Theta;\vartheta;f)}$

$$[\![\Gamma \vdash_\Delta t[\kappa'] : A]\!]_{(\Theta;\vartheta;f)}(\gamma) = [\# \mapsto f(\kappa')] \cdot ([\![\Gamma \vdash_\Delta t : \forall\kappa.A]\!]_{(\Theta;\vartheta;f)}(\gamma))_n$$

where $n = \vartheta(f(\kappa'))$. Here the element $([\![\Gamma \vdash_\Delta t : \forall\kappa.A]\!]_{(\Theta;\vartheta;f)}(\gamma))_n$ lives in

$$\begin{aligned}
&[\![\Gamma \vdash_{\Delta,\kappa} A \text{ type}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])}([\![(i, \mathsf{id}_\Gamma)]\!]^{-1}(\iota \cdot \gamma)) \\
&= i^*([\![\Gamma \vdash_\Delta A \text{ type}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f[\kappa\mapsto\#])}([\![(i, \mathsf{id}_\Gamma)]\!]([\![(i, \mathsf{id}_\Gamma)]\!]^{-1}(\iota \cdot \gamma))) \\
&= [\![\Gamma \vdash_\Delta A \text{ type}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f)}(\iota \cdot \gamma)
\end{aligned}$$

Clock irrelevance will follow from the following lemma.

▶ **Lemma 13.** *Suppose* $\Gamma \vdash_\Delta A$ type *and* $\gamma \in [\![\Gamma \vdash_\Delta]\!]_{(\Theta;\vartheta;f)}$. *The map*

$$\iota \cdot (-) : [\![\Gamma \vdash_\Delta A \text{ type}]\!]_{(\Theta;\vartheta;f)} \to [\![\Gamma \vdash_\Delta A \text{ type}]\!]_{(\Theta,\#;\vartheta[\#\mapsto n];f)}(\iota \cdot \gamma)$$

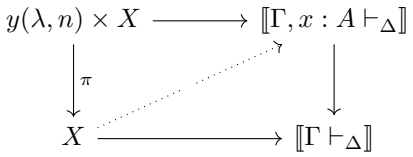
is an isomorphism.

In particular, there is an element x such that $\iota \cdot x = ([\![\Gamma \vdash_\Delta t : \forall\kappa.A]\!]_{(\Theta;\vartheta;f)}(\gamma))_n$ and so

$$[\![\Gamma \vdash_\Delta t[\kappa'] : A]\!]_{(\Theta;\vartheta;f)}(\gamma) = [\# \mapsto f(\kappa')] \cdot \iota \cdot x = x$$

Likewise $[\![\Gamma \vdash_\Delta t[\kappa''] : A]\!]_{(\Theta;\vartheta;f)}(\gamma) = x$ proving clock irrelevance.

Lemma 13 can be proved by induction on A using the techniques of [10]. In particular, [10] proves that the statement of the lemma is equivalent to the statement that the context projection map $[\![\Gamma, x : A \vdash_\Delta]\!] \to [\![\Gamma \vdash_\Delta]\!]$ (considered as a morphism in GR) is *orthogonal* to all objects of the form $y(\lambda, n)$ where y is the yoneda embedding. Here, orthogonality means that for all squares as below, there exists a unique filling diagonal:



where π is the second projection. In particular, this implies that the condition is closed under Π- and Σ-types, and substitutions, see [10] for details. This proof can be easily extended to the cases of $\triangleright(\alpha : \kappa).A$ and $\forall\kappa.A$.

9 Conclusion and future work

We have constructed a model of CloTT modelling ticks on clocks using an adjunction where the right adjoint extends to types and terms. The description of the left adjoint $\blacktriangleleft^\kappa$ is fairly heavy to work with, but by abstracting away the required properties needed to model the tick calculus, the model can be described in reasonable space.

Future work includes extending the model to universes, which we expect to be easy using the universes constructed in [10]. As noted there the axiom of clock irrelevance forces universes to be indexed by syntactic clock contexts. Fortunately, we can model a notion of universe polymorphism in the clock context: inclusions of clock contexts induce inclusion of universes, and these commute with type constructions on the nose. These results, however, must be adapted to CloTT, in particular the code for the $\triangleright$ type constructor should be extended to the tick abstracting generalisation used in CloTT. We expect this to be a simple adaptation. Using universes, guarded recursive types can be encoded [6], indeed these can also be added as primitive, given that many of them exist in the model [10].

Our motivation for constructing this model is to study extensions of CloTT. In particular, we would like to extend CloTT with path types as in [5]. This requires an adaptation of the model to the cubical setting, using $\mathbb{T}$ indexed families of cubical sets [11] rather than just sets.

Acknowledgements. We thank Patrick Bahr for useful discussions.

References

1 A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst*, 23(5):657–683, 2001.

2 R. Atkey and C. McBride. Productive coprogramming with guarded recursion. In *ICFP*, pages 197–208. ACM, 2013.

3 P. Bahr, H. B. Grathwohl, and R. E. Møgelberg. The clocks are ticking: No more delays! In *LICS*, pages 1–12. IEEE, 2017.

4 J.-P. Bernardy, T. Coquand, and G. Moulin. A presheaf model of parametric type theory. *Electronic Notes in Theoretical Computer Science*, 319:67–82, 2015.

5 L. Birkedal, A. Bizjak, R. Clouston, H. B. Grathwohl, B. Spitters, and A. Vezzosi. Guarded cubical type theory: Path equality for guarded recursion. In *CSL*, 2016.

6 L. Birkedal and R. E. Møgelberg. Intensional type theory with guarded recursive types qua fixed points on universes. In *LICS*, pages 213–222. IEEE, 2013.

7 L. Birkedal, R.E. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *LMCS*, 8(4), 2012.

8 A. Bizjak, L. Birkedal, and M. Miculan. A model of countable nondeterminism in guarded type theory. In *RTA-TLCA*, pages 108–123, 2014.

9 A. Bizjak, H. B. Grathwohl, R. Clouston, R. E. Møgelberg, and L. Birkedal. Guarded dependent type theory with coinductive types. In *FOSSACS*, 2016.

10 A. Bizjak and R. Møgelberg. Denotational semantics for guarded dependent type theory. *arXiv*, 2017. `arXiv:1802.03744`.

11 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.TYPES.2015.5`.

12 T. Coquand. Infinite objects in type theory. In *International Workshop on Types for Proofs and Programs*, pages 62–78. Springer, 1993.

13 P. Dybjer. Internal type theory. In *International Workshop on Types for Proofs and Programs*, pages 120–134. Springer, 1995.

14 P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Napoli, 1984.

15 C. McBride and R. Paterson. Applicative programming with effects. *Journal of functional programming*, 18(1):1–13, 2008.

16 H. Nakano. A modality for recursion. In *LICS*, pages 255–266, 2000.

17 A. M. Pitts, J. Matthiesen, and J. Derikx. A dependent type theory with abstractable names. *Electronic Notes in Theoretical Computer Science*, 312:19–50, 2015.

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