Internal Effectful Forcing in System T

Martín H. Escardó ✉ 🏠 ⓘ
University of Birmingham, UK

Bruno da Rocha Paiva ✉ 🏠 ⓘ
University of Birmingham, UK

Vincent Rahli ✉ 🏠 ⓘ
University of Birmingham, UK

Ayberk Tosun ✉ 🏠 ⓘ
University of Birmingham, UK

───── **Abstract** ─────

The *effectful forcing* technique allows one to show that the denotation of a closed System T term of type $(\iota \Rightarrow \iota) \Rightarrow \iota$ in the set-theoretical model is a continuous function $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$. For this purpose, an alternative dialogue-tree semantics is defined and related to the set-theoretical semantics by a logical relation. In this paper, we apply effectful forcing to show that the dialogue tree of a System T term is itself System T-definable, using the Church encoding of trees.

1 Introduction

It is well known that the System T-definable functions $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ are continuous and that, moreover, their moduli of continuity are themselves System T-definable [24]. Effectful forcing [12] was generalised by Xu [27] to give an alternative proof of this fact. Effectful forcing gives a dialogue-tree semantics, and relates it to the set-theoretical semantics by a logical relation. In this paper, we strengthen this by showing that the dialogue trees are themselves System T-definable, using Church encoding. Dialogue trees are equivalent variants of Brouwer trees [10, 22]

From a constructive point of view, dialogue trees give more information than moduli of continuity. Given a dialogue tree, it is possible to derive a modulus of continuity, but the converse is only known to be possible in the presence of additional assumptions. For example, Ghani, Hancock and Pattison [15] show that if a function doesn't have a Brouwer tree, then it isn't continuous, using dependent choice, while Capretta and Uustalu [4] use the assumption of bar induction to show that every function with a *stable* modulus of continuity has a Brouwer tree. Our result does not assume dependent choice or bar induction, and, moreover, establishes the *System T-definability* of a dialogue tree of any closed term of type $(\iota \Rightarrow \iota) \Rightarrow \iota$.

Related work. Continuity is a key concept in mathematics, and in particular in constructive mathematics where it is often accepted that all real-valued functions on the unit interval are uniformly continuous. Prominent work in the area was done by Brouwer, who gave the first argument for the previous result, relying on his *continuity principle for numbers*, which states that all functions on the Baire space are continuous, as well as the Fan Theorem [23, 8]. Brouwer's continuity principle said that the values of a function $F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ on the Baire space could only rely on a finite amount of its inputs. More precisely, F was said to be

continuous if for all $\alpha : \mathbb{N} \to \mathbb{N}$, there existed some $n : \mathbb{N}$ such that for any inputs $\beta : \mathbb{N} \to \mathbb{N}$ which agreed on the first n entries with α we would have $F(\alpha) = F(\beta)$. When the value of n could be picked independently of α, then F was said to be uniformly continuous. Since then, more fine-grained ways of capturing continuity information have been devised, such as using trees to keep track of the specific entries of α that $F(\alpha)$ might depend on. These ideas were used by Kleene in his study of recursive functions of higher types [17], by Brouwer in his study of his bar theorem [23], and many others, including [13, 15, 14, 12, 4, 22, 3, 5, 2].

Starting with Troelstra's work [24, p.158] for N-HA$^\omega$, the definable functions of various other systems have been shown to be continuous. Among these we have: System T [12], MLTT [6, 7, 26], CTT [21], BTT [3], and TT$_{\mathcal{C}}^{\square}$ [5]. However, the existence of a modulus of continuity operator tends to be inconsistent with function extensionality as first shown for HA$_{NB}^\omega$ by Kreisel [18, p.154] and later for N-HA$^\omega$ by Troelstra [25, Thm.3.1(IIA)]. In the case of MLTT we don't even need function extensionality to get an inconsistency as shown by Escardó and Xu [11, 26]. Further related work is discussed in Sec. 7.

Main contributions.   (1) We strengthen the above work to show that not only the modulus of continuity of a System T-definable function is itself System T-definable, but also its dialogue tree is System T-definable, where we implemented trees in System T using Church encoding. (2) We prove the correctness of this translation with a logical relation. (3) We show how to compute moduli of continuity and uniform continuity internally, and prove the correctness of this construction. (4) The original presentation of effectful forcing [12] extends System T with an oracle, and here we show that this extension is not necessary and we can work directly with System T without oracles.

Organisation.   Sec. 2 recalls System T's syntax as well as its set semantics $[\![-]\!]_{\mathbf{Set}}$. Sec. 3 presents a simplified version of effectful forcing using the inductive dialogue translation $[\![-]\!]_{\mathcal{D}}$ and a modified version of the logical relation $\alpha \vDash x \approx y \in \sigma$. Sec. 4 presents the Church encodings of dialogue trees, the resulting internal dialogue translation $[\![-]\!]_{\mathcal{D}^{\mathsf{T}}}$ and the logical relation $x \sim y \in \sigma$ linking inductive dialogue trees and internal Church encoded dialogue trees. Finally, Secs. 5 and 6 explain how to compute moduli of continuity and uniform moduli of continuity inside System T from these dialogue trees. These translations and logical relations are summarised in the following diagram:

$$
\begin{array}{ccc}
& \mathsf{Term}_0^{\mathsf{T}}(\iota) & \\
{}^{[\![-]\!]_{\mathbf{Set}}}\swarrow & \downarrow {\scriptstyle [\![-]\!]_{\mathcal{D}}} & \searrow {}^{[\![-]\!]_{\mathcal{D}^{\mathsf{T}}}} \\
\mathbb{N} \xleftarrow[\alpha \vDash \cdot \approx \cdot \in \iota]{} & \mathcal{D}(\mathbb{N}) & \xleftarrow[\cdot \sim \cdot \in \iota]{} \mathsf{Term}_0^{\mathsf{T}}(\mathcal{D}_A^{\mathsf{T}}(\iota))
\end{array}
$$
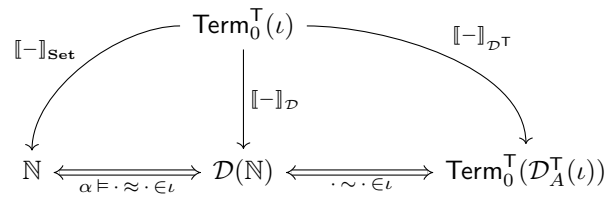
Metatheory.   Our metatheory is a spartan version of MLTT featuring $\prod$-types, $\sum$-types, inductive types, and a universe. All of our reasoning is constructive and we don't rely on function extensionality. We will fix some notation for the rest of the paper: we denote the type of natural numbers by $\mathbb{N}$, its zero element by 0, given a natural number n we write $1 + n$ for its successor, and we let $\mathsf{Natrec} : (\mathbb{N} \to X \to X) \to X \to \mathbb{N} \to \mathbb{N}$ be its recursor.

$$\boxed{\mathsf{Type}^\mathsf{T}} \qquad\qquad\qquad\qquad \boxed{\mathsf{Ctx}^\mathsf{T}}$$

$$\frac{}{\iota : \mathsf{Type}^\mathsf{T}} \qquad \frac{\sigma : \mathsf{Type}^\mathsf{T} \qquad \tau : \mathsf{Type}^\mathsf{T}}{\sigma \Rightarrow \tau : \mathsf{Type}^\mathsf{T}} \qquad \frac{}{\diamond : \mathsf{Ctx}^\mathsf{T}} \qquad \frac{\Gamma : \mathsf{Ctx}^\mathsf{T} \qquad (x : \sigma) \notin \Gamma}{\Gamma, (x : \sigma) : \mathsf{Ctx}^\mathsf{T}}$$

$$\boxed{\mathsf{Term}^\mathsf{T}(\Gamma, \sigma)} \qquad \frac{(x : \sigma) \in \Gamma}{x : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)} \qquad \frac{}{\mathtt{Zero} : \mathsf{Term}^\mathsf{T}(\Gamma, \iota)} \qquad \frac{t : \mathsf{Term}^\mathsf{T}(\Gamma, \iota)}{\mathtt{Succ}(t) : \mathsf{Term}^\mathsf{T}(\Gamma, \iota)}$$

$$\frac{t : \mathsf{Term}^\mathsf{T}(\Gamma, \iota \Rightarrow \sigma \Rightarrow \sigma) \qquad p : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma) \qquad q : \mathsf{Term}^\mathsf{T}(\Gamma, \iota)}{\mathtt{Rec}_\sigma(t, p, q) : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)} \qquad \frac{t : \mathsf{Term}^\mathsf{T}(\Gamma, x : \sigma, \tau)}{\lambda(x : \sigma).\, t : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma \Rightarrow \tau)}$$

$$\frac{t : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma \Rightarrow \tau) \qquad p : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)}{t(p) : \mathsf{Term}^\mathsf{T}(\Gamma, \tau)}$$

Figure 1 The syntax of intrinsically typed System T.

Formalisation. The results of this paper have been formalised in AGDA and we have tried to keep the presentation faithful to the formalisation [9]. The main difference is the use of variables for clarity, while the formalisation uses de Bruijn indices for System T.

2 System T

System T features three notions: contexts, types and terms. It has a single base type ι of natural numbers as well as a type of functions $\sigma \Rightarrow \tau$. Contexts are lists of paired variables and types. Terms are built up inductively from variables, a zero constant $\mathtt{Zero}$, a successor operation $\mathtt{Succ}$, a recursion operator $\mathtt{Rec}$, lambda abstraction and function application. More formally, we have the following definitions.

▶ **Definition 1** (🖱). *The types (1) of System T types* Type^T*; (2) of System T contexts* Ctx^T*; and (3) given a System T context* $\Gamma : \mathsf{Ctx}^\mathsf{T}$ *and a System T type* $\sigma : \mathsf{Type}^\mathsf{T}$*, of System T terms of type* σ *in context* Γ*, denoted* $\mathsf{Term}^\mathsf{T}(\Gamma, \sigma)$*, are defined in Fig. 1. We fix the shorthand* $\mathsf{Term}_0^\mathsf{T}(\sigma)$ *for* $\mathsf{Term}^\mathsf{T}(\diamond, \sigma)$*.*

Given contexts $\Gamma, \Sigma : \mathsf{Ctx}^\mathsf{T}$, a substitution from Γ to Σ is an assignment of terms $t : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)$ to each variable $(x : \sigma) \in \Delta$. Though we don't formally define it here, we assume we have a capture-free substitution of System T terms. Given a term $t : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)$ and substitution ρ from Δ to Γ, we denote the substituted term by $(t)\{\rho\} : \mathsf{Term}^\mathsf{T}(\Delta, \sigma)$.

At this point one would usually introduce the reduction rules associated with System T, or the corresponding notion of conversion of terms. As we will see later, the dialogue semantics of System T do not respect conversion of terms in general, hence we omit these rules. With that said though, we now introduce the set model [24] of System T, sometimes also called the functional model [1]. In this model we interpret ι as the metatheoretic natural numbers $\mathbb{N}$ and function types $\Rightarrow$ as the metatheoretic function space $\rightarrow$. A context Γ is then interpreted as the type of assignments from the context variables to elements of the translated types. A term $t : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)$ is interpreted as a metatheoretic function from the interpretation of Γ to the interpretation of σ.

$$\llbracket - \rrbracket_{\mathbf{Set}} : \mathsf{Type}^\mathsf{T} \to \mathsf{Type}$$
$$\llbracket \iota \rrbracket_{\mathbf{Set}} :\equiv \mathbb{N}$$
$$\llbracket \sigma \Rightarrow \tau \rrbracket_{\mathbf{Set}} :\equiv \llbracket \sigma \rrbracket_{\mathbf{Set}} \to \llbracket \tau \rrbracket_{\mathbf{Set}}$$

$$\llbracket - \rrbracket_{\mathbf{Set}} : \mathsf{Ctx}^\mathsf{T} \to \mathsf{Type}$$
$$\llbracket \Gamma \rrbracket_{\mathbf{Set}} :\equiv (x : \sigma) \in \Gamma \to \llbracket \sigma \rrbracket_{\mathbf{Set}}$$

$$\llbracket - \rrbracket_{\mathbf{Set}}^- : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma) \to \llbracket \Gamma \rrbracket_{\mathbf{Set}} \to \llbracket \sigma \rrbracket_{\mathbf{Set}}$$
$$\llbracket x \rrbracket_{\mathbf{Set}}^\gamma :\equiv \gamma(x)$$
$$\llbracket \mathtt{Zero} \rrbracket_{\mathbf{Set}}^\gamma :\equiv 0$$
$$\llbracket \mathtt{Succ}(t) \rrbracket_{\mathbf{Set}}^\gamma :\equiv 1 + \llbracket t \rrbracket_{\mathbf{Set}}^\gamma$$
$$\llbracket \mathsf{Rec}_\sigma(t_1, t_2, t_3) \rrbracket_{\mathbf{Set}}^\gamma :\equiv \mathsf{Natrec}(\llbracket t_1 \rrbracket_{\mathbf{Set}}^\gamma)(\llbracket t_2 \rrbracket_{\mathbf{Set}}^\gamma)(\llbracket t_3 \rrbracket_{\mathbf{Set}}^\gamma)$$
$$\llbracket \lambda(x : \sigma).\, t \rrbracket_{\mathbf{Set}}^\gamma :\equiv \lambda x' : \llbracket \sigma \rrbracket_{\mathbf{Set}}.\; \llbracket t \rrbracket_{\mathbf{Set}}^{\gamma, (x \mapsto x')}$$
$$\llbracket t_1(t_2) \rrbracket_{\mathbf{Set}}^\gamma :\equiv \llbracket t_1 \rrbracket_{\mathbf{Set}}^\gamma(\llbracket t_2 \rrbracket_{\mathbf{Set}}^\gamma)$$

Figure 2 Set model of System T.

▶ **Definition 2** (⟳). *The **set interpretation** of types, contexts and terms of System T is defined in Fig. 2. When defining the interpretation of terms we let γ range over $\llbracket \Gamma \rrbracket_{\mathbf{Set}}$. We fix the notation $\gamma, (x \mapsto x')$ to extend an assignment of $\llbracket \Gamma \rrbracket_{\mathbf{Set}}$ to an assignment of $\llbracket \Gamma, (x : \sigma) \rrbracket_{\mathbf{Set}}$.*

Each natural number can be encoded as a System T term in the standard manner.

▶ **Definition 3** (⟳). *Given a natural number n we may define a System T term $\underline{n} : \mathsf{Term}_0^\mathsf{T}(\iota)$ by induction on n according to the rules $\underline{0} :\equiv \mathtt{Zero}$ and $\underline{1 + n} :\equiv \mathtt{Succ}(\underline{n})$.*

▶ **Proposition 4** (⟳). *For all $n : \mathbb{N}$, we have $n = \llbracket \underline{n} \rrbracket_{\mathbf{Set}}$.*

3   Oracle-less Effectful Forcing

The original presentation of effectful forcing [12] works with System T extended with an oracle. Here we show that this extension is not necessary, and we can work directly with System T.

3.1   Dialogue Trees and Continuity

Effectful forcing starts by noting that a function $f : (I \to O) \to X$ can be thought of as an effectful term $t_f : X$ with access to an oracle $\alpha : I \to O$. The use of an oracle can be seen as an algebraic effect with the operation $\mathtt{question}_X\ (i : I)\ (k : O \to X) : X$, which prompts the oracle with question i and passes the oracle's answer to the continuation k eventually producing an element of X. Looking for the monad corresponding to oracle computations we find dialogue trees.

▶ **Definition 5** (⟳). *The inductive type $\mathsf{Dial}(I)(O)(X)$ of (I, O, X)-**dialogue trees** is defined by*

$$\frac{x : X}{\eta(x) : \mathsf{Dial}(I)(O)(X)} \qquad\qquad \frac{\phi : O \to \mathsf{Dial}(I)(O)(X) \qquad i : I}{\beta(\phi)(i) : \mathsf{Dial}(I)(O)(X)}$$

In the case where the oracle input and output are $\mathbb{N}$, we write $\mathcal{D}(X)$ for $\mathsf{Dial}(\mathbb{N})(\mathbb{N})(X)$.

Justifying this view, each (I, O, X)-dialogue tree encodes a function of type $(I \to O) \to X$.

▶ **Definition 6** (⟳). *The **dialogue** between a dialogue tree and an oracle is given by*

$\mathsf{dialogue} : \mathsf{Dial}(I)(O)(X) \to (I \to O) \to X$
$\mathsf{dialogue}(\eta(x))(\alpha) :\equiv x$
$\mathsf{dialogue}(\beta(\phi)(i))(\alpha) :\equiv \mathsf{dialogue}(\phi(\alpha(i)))(\alpha)$

▶ **Definition 7** (↻). *Given a function $f : (I \to O) \to X$ we say it is:*

- *dialogue continuous if $\exists\, d : \mathsf{Dial}(I)(O)(X), \forall\, \alpha : I \to O, f(\alpha) = \mathsf{dialogue}(d)(\alpha)$.*

When the oracle input is $I = \mathbb{N}$, so we have $f : (\mathbb{N} \to O) \to X$, we also say f is

- *continuous if $\forall\, \alpha : \mathbb{N} \to O, \exists\, n : \mathbb{N}, \forall\, \beta : \mathbb{N} \to O, \alpha =_n \beta \to f(\alpha) = f(\beta)$.*
- *uniformly continuous if $\exists\, n : \mathbb{N}, \forall\, \alpha, \beta : \mathbb{N} \to O, \alpha =_n \beta \to f(\alpha) = f(\beta)$.*

where we have used $\alpha =_n \beta$ to mean that α and β share the same initial segment of length n.

▶ Remark 8. Due to their inductive nature, any path along a dialogue tree must reach a leaf node in a finite number of steps and hence query the oracle a finite number of times. This means that given a dialogue tree $d : \mathsf{Dial}(\mathbb{N})(O)(X)$, the function $\mathsf{dialogue}(d) : (\mathbb{N} \to O) \to X$ must be continuous. Further assuming that the type O is finite, for example in the case of $\mathbb{B}$, then the dialogue tree d can even be fully searched for the largest query over all paths, in which case $\mathsf{dialogue}(d)$ will also be uniformly continuous.

3.2 A Dialogue Tree Translation of System T

As mentioned, dialogue trees form a monad and it is this structure that allows us to define the dialogue tree translation of System T.

▶ **Definition 9** (↻). *The **Kleisli extension of dialogue trees** is defined inductively by*

$$\mathsf{kleisli\text{-}ext} : (X \to \mathsf{Dial}(I)(O)(Y)) \to \mathsf{Dial}(I)(O)(X) \to \mathsf{Dial}(I)(O)(Y)$$

$$\mathsf{kleisli\text{-}ext}(f)(\eta(x)) :\equiv f(x)$$

$$\mathsf{kleisli\text{-}ext}(f)(\beta(\phi)(i)) :\equiv \beta(\lambda x.\ \mathsf{kleisli\text{-}ext}(f)(\phi(x)))(i)$$

The Kleisli extension of a function f will apply it at the leaves and graft in the resulting trees, leaving intermediate nodes unchanged. With this we may define the functorial action.

▶ **Definition 10** (↻). *The **functorial action of dialogue trees** is defined by*

$$\mathcal{D}\text{-}\mathsf{functor} : (X \to Y) \to \mathsf{Dial}(I)(O)(X) \to \mathsf{Dial}(I)(O)(Y)$$

$$\mathcal{D}\text{-}\mathsf{functor}(f) :\equiv \mathsf{kleisli\text{-}ext}(\eta \circ f)$$

To interpret the higher-order recursion in System T there are two possible approaches. In one, we choose to interpret System T types as algebras over the dialogue tree monad, as done in [22]. Alternatively, one may interpret System T types as metatheoretic types (akin to the set model) and use a generalised Kleisli extension to interpret higher-order recursion. The former gives a compositional semantics which is simpler to define, but as we are extending the formalisation of [12], we choose the latter. The following definition refers to the translation of System T types, which can be found in Fig. 3. One should see generalised Kleisli extension as a pointwise version of Kleisli extension.

▶ **Definition 11** (↻). *The **generalised Kleisli extension of dialogue trees** is defined by induction on System T types as follows*

$$\mathsf{Kleisli\text{-}ext}_\sigma : (X \to [\![\sigma]\!]_{\mathcal{D}}) \to \mathcal{D}(X) \to [\![\sigma]\!]_{\mathcal{D}}$$

$$\mathsf{Kleisli\text{-}ext}_\iota :\equiv \mathsf{kleisli\text{-}ext}$$

$$\mathsf{Kleisli\text{-}ext}_{\sigma_1 \Rightarrow \sigma_2} :\equiv \lambda f.\ \lambda d.\ \lambda s.\ \mathsf{Kleisli\text{-}ext}_{\sigma_2}(\lambda x.\ f(x)(s))(d)$$

For the dialogue interpretation, we will interpret the ground type ι as the type $\mathcal{D}(\mathbb{N})$ of dialogue trees over the natural numbers, and the function type $\Rightarrow$ is, as in the set model, interpreted by the metatheoretic function type $\to$.

$$\llbracket - \rrbracket_{\mathcal{D}} : \mathsf{Type}^{\mathsf{T}} \to \mathsf{Type}$$
$$\llbracket \iota \rrbracket_{\mathcal{D}} :\equiv \mathcal{D}(\mathbb{N})$$
$$\llbracket \sigma_0 \Rightarrow \sigma_1 \rrbracket_{\mathcal{D}} :\equiv \llbracket \sigma_0 \rrbracket_{\mathcal{D}} \to \llbracket \sigma_1 \rrbracket_{\mathcal{D}}$$

$$\llbracket - \rrbracket_{\mathcal{D}} : \mathsf{Ctx}^{\mathsf{T}} \to \mathsf{Type}$$
$$\llbracket \Gamma \rrbracket_{\mathcal{D}} :\equiv (x : \sigma) \in \Gamma \to \llbracket \sigma \rrbracket_{\mathcal{D}}$$

$$\llbracket - \rrbracket_{\mathcal{D}}^{-} : \mathsf{Term}^{\mathsf{T}}(\Gamma, \sigma) \to \llbracket \Gamma \rrbracket_{\mathcal{D}} \to \llbracket \sigma \rrbracket_{\mathcal{D}}$$
$$\llbracket x \rrbracket_{\mathcal{D}}^{\gamma} :\equiv \gamma(x)$$
$$\llbracket \mathtt{Zero} \rrbracket_{\mathcal{D}}^{\gamma} :\equiv \eta(0)$$
$$\llbracket \mathtt{Succ}(t) \rrbracket_{\mathcal{D}}^{\gamma} :\equiv \mathcal{D}\text{-functor}(1+)(\llbracket t \rrbracket_{\mathcal{D}}^{\gamma})$$
$$\llbracket \mathtt{Rec}_{\sigma}(t_1, t_2, t_3) \rrbracket_{\mathcal{D}}^{\gamma} :\equiv \mathsf{Kleisli\text{-}ext}_{\sigma}(\mathsf{Natrec}(\llbracket t_1 \rrbracket_{\mathcal{D}}^{\gamma} \circ \eta)(\llbracket t_2 \rrbracket_{\mathcal{D}}^{\gamma}))(\llbracket t_3 \rrbracket_{\mathcal{D}}^{\gamma})$$
$$\llbracket \lambda(x : \sigma).\ t \rrbracket_{\mathcal{D}}^{\gamma} :\equiv \lambda x' : \llbracket \sigma \rrbracket_{\mathcal{D}}.\ \llbracket t \rrbracket_{\mathcal{D}}^{\gamma,(x \mapsto x')}$$
$$\llbracket t_1(t_2) \rrbracket_{\mathcal{D}}^{\gamma} :\equiv \llbracket t_1 \rrbracket_{\mathcal{D}}^{\gamma}(\llbracket t_2 \rrbracket_{\mathcal{D}}^{\gamma})$$

Figure 3 Dialogue interpretation of System T.

▶ **Definition 12** (⟳). *The **dialogue interpretation** of types, contexts and terms of System T is defined in Fig. 3. When defining the interpretation of terms we let γ range over $\llbracket \Gamma \rrbracket_{\mathcal{D}}$. We fix the notation $\gamma, (x \mapsto x')$ to extend an assignment of $\llbracket \Gamma \rrbracket_{\mathcal{D}}$ to an assignment of $\llbracket \Gamma, (x : \sigma) \rrbracket_{\mathcal{D}}$.*

This translation differs from the one in [12] in two ways. The first is that we are no longer using a combinator version of System T. The inclusion of variables aids in internalizing metatheoretic functions, which we will do frequently in future sections, hence it is desirable even if it complicates the formalisation. Aside from this, we also do not extend System T with an oracle, which as we will note later, turns out to be unnecessary.

The final piece of the puzzle for effectful forcing is the existence of a generic sequence in this new interpretation. To compute dialogue trees we require the existence of a *generic sequence*, that is, we need $f : \mathcal{D}(\mathbb{N}) \to \mathcal{D}(\mathbb{N})$ such that for all $\alpha : \mathbb{N} \to \mathbb{N}$, the following commutes

$$
\begin{array}{ccc}
\mathcal{D}(\mathbb{N}) & \xrightarrow{\ f\ } & \mathcal{D}(\mathbb{N}) \\
{\scriptstyle \mathsf{dialogue}(-)(\alpha)}\downarrow & & \downarrow{\scriptstyle \mathsf{dialogue}(-)(\alpha)} \\
\mathbb{N} & \xrightarrow[\ \alpha\ ]{} & \mathbb{N}
\end{array}
$$

▶ **Definition 13** (⟳). *The **generic sequence** is defined as the following Kleisli extension:*

$$\mathsf{generic} : \mathcal{D}(\mathbb{N}) \to \mathcal{D}(\mathbb{N})$$
$$\mathsf{generic} :\equiv \mathsf{kleisli\text{-}ext}(\beta(\eta))$$

We can show this sequence to indeed satisfy the mentioned commuting diagram.

▶ **Definition 14** (⟳). *We define the **dialogue tree operator** by*

$$\mathsf{dialogue\text{-}tree} : \mathsf{Term}_0^{\mathsf{T}}((\iota \Rightarrow \iota) \Rightarrow \iota) \to \mathcal{D}(\mathbb{N})$$
$$\mathsf{dialogue\text{-}tree}(t) :\equiv \llbracket t \rrbracket_{\mathcal{D}}(\mathsf{generic})$$

In its original formulation, a new oracle term $\Omega : \mathsf{Term}_0^{\mathsf{T}}(\iota \Rightarrow \iota)$ was added to System T. Under the dialogue translation, this term was interpreted by $\mathsf{generic}$ and given $t : \mathsf{Term}_0^{\mathsf{T}}((\iota \Rightarrow \iota) \Rightarrow \iota)$ we could compute dialogue tree for t by taking the dialogue translation of $t(\Omega)$. This extension turns out to be unnecessary, using instead the above definition of $\mathsf{dialogue\text{-}tree}$ and modifying the logical relation used to prove correctness.

▶ **Definition 15** (↻). *Given a System T type σ, a sequence $\alpha : \mathbb{N} \to \mathbb{N}$, and elements $x : [\![\sigma]\!]_{\mathbf{Set}}$ and $y : [\![\sigma]\!]_{\mathcal{D}}$, we define the **effectful forcing logical relation** by induction on σ:*

$$\boxed{\alpha \vDash x \approx y \in \sigma} \qquad\qquad \frac{n = \mathsf{dialogue}(d)(\alpha)}{\alpha \vDash n \approx d \in \iota}$$

$$\frac{\forall x : [\![\sigma_1]\!]_{\mathbf{Set}},\, \forall y : [\![\sigma_1]\!]_{\mathcal{D}},\, (\alpha \vDash x \approx y \in \sigma_1) \to (\alpha \vDash f(x) \approx g(y) \in \sigma_2)}{\alpha \vDash f \approx g \in \sigma_1 \Rightarrow \sigma_2}$$

From this relation's fundamental lemma we can derive the correctness result from [12].

▶ **Theorem 16** (↻ Correctness of dialogue-tree). *For all sequences $\alpha : \mathbb{N} \to \mathbb{N}$ and closed terms $t : \mathsf{Term}_0^{\mathsf{T}}((\iota \Rightarrow \iota) \Rightarrow \iota)$ we have $[\![t]\!]_{\mathbf{Set}}(\alpha) = \mathsf{dialogue}(\mathsf{dialogue\text{-}tree}(t))(\alpha)$.*

This result shows that any System T-definable functionals $F : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ are dialogue continuous and hence continuous. Furthermore, by encoding $\mathbb{B}$ in System T, we also see that any functionals $F : (\mathbb{N} \to \mathbb{B}) \to \mathbb{N}$ definable in System T must be uniformly continuous.

As observed in [22, 3], this translation is not a model of System T as the dialogue interpretation of $\mathtt{Rec}$ does not satisfy the usual System T equations. For example, it does *not* validate the conversion $\mathtt{Rec}_\iota(f, x, \mathtt{Succ}(n)) \equiv f(n)(\mathtt{Rec}_\iota(f, x, n))$ under the context $(f : \iota \Rightarrow \iota \Rightarrow \iota), (x : \iota), (n : \iota)$ due to the existence of effectful terms. As a result, convertible terms may be assigned completely different dialogue trees, but this doesn't affect the correctness of our claims.

4 Internalising Dialogue Trees in System T

We now work on recreating the translation of Sec. 3.2 internally to System T using the Church encodings of dialogue trees. The main difference from the inductive case is that due to the use of Church encodings, we must now parameterise the translation by a System T type: that is, the motive for the elimination principle of the Church encoded dialogue trees. For the purpose of defining moduli of continuity it suffices to use the motive $A :\equiv (\iota \Rightarrow \iota) \Rightarrow \iota$. With this we could then define the dialogue tree following Sec. 4.1 and compute a modulus of continuity according to Sec. 5. The insufficiency of considering a single motive reveals itself only when proving the correctness of the Church encoded translation. For the correctness proof we require multiple motives, for example $A :\equiv \iota$ is used in the $\mathtt{Rec}$ case of Lem. 33.

4.1 Church-Encoded Trees in System T

For the rest of this subsection we fix a motive $A : \mathsf{Type}^{\mathsf{T}}$ and give the corresponding internal dialogue translation. We will also be slightly less general with the internal definitions of dialogue trees as we will only need to talk about the internalisation of $\mathcal{D}(\mathbb{N})$.

▶ **Definition 17** (↻). *Given $\sigma : \mathsf{Type}^{\mathsf{T}}$, we define **internal σ-dialogue trees** as:*

$$\mathcal{D}_A^{\mathsf{T}}(\sigma) : \mathsf{Type}^{\mathsf{T}}$$
$$\mathcal{D}_A^{\mathsf{T}}(\sigma) :\equiv (\sigma \Rightarrow A) \Rightarrow ((\iota \Rightarrow A) \Rightarrow \iota \Rightarrow A) \Rightarrow A$$

The constructors corresponding to η and β are the functions η_A^{T} and β_A^{T} defined as

$$\eta_A^{\mathsf{T}} : \mathsf{Term}_0^{\mathsf{T}}(\sigma \Rightarrow \mathcal{D}_A^{\mathsf{T}}(\sigma)) \qquad \beta_A^{\mathsf{T}} : \mathsf{Term}_0^{\mathsf{T}}((\iota \Rightarrow \mathcal{D}_A^{\mathsf{T}}(\sigma)) \Rightarrow \iota \Rightarrow \mathcal{D}_A^{\mathsf{T}}(\sigma))$$
$$\eta_A^{\mathsf{T}} :\equiv \lambda z.\, \lambda e.\, \lambda b.\, e(z) \qquad\qquad \beta_A^{\mathsf{T}} :\equiv \lambda \phi.\, \lambda x.\, \lambda e.\, \lambda b.\, \lambda y.\, \varphi(y)(e)(b)$$

$$\llbracket - \rrbracket_{\mathcal{D}^\mathsf{T}}^A : \mathsf{Type}^\mathsf{T} \to \mathsf{Type}^\mathsf{T} \qquad\qquad\qquad \llbracket - \rrbracket_{\mathcal{D}^\mathsf{T}}^A : \mathsf{Ctx}^\mathsf{T} \to \mathsf{Ctx}^\mathsf{T}$$

$$\llbracket \iota \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \mathcal{D}_A^\mathsf{T}(\iota) \qquad\qquad\qquad\qquad \llbracket \diamond \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \diamond$$

$$\llbracket \sigma_0 \Rightarrow \sigma_1 \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \llbracket \sigma_0 \rrbracket_{\mathcal{D}^\mathsf{T}}^A \Rightarrow \llbracket \sigma_1 \rrbracket_{\mathcal{D}^\mathsf{T}}^A \qquad\qquad \llbracket \Gamma, (x : \sigma) \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \llbracket \Gamma \rrbracket_{\mathcal{D}^\mathsf{T}}^A, (x : \llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$$

$$\llbracket - \rrbracket_{\mathcal{D}^\mathsf{T}}^A : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma) \to \mathsf{Term}^\mathsf{T}(\llbracket \Gamma \rrbracket_{\mathcal{D}^\mathsf{T}}^A, \llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$$

$$\llbracket x \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv x$$

$$\llbracket \mathtt{Zero} \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \eta_A^\mathsf{T}(\mathtt{Zero})$$

$$\llbracket \mathtt{Succ}(t) \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \mathcal{D}\text{-}\mathtt{functor}_A^\mathsf{T}(\mathtt{Succ})(\llbracket t \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$$

$$\llbracket \mathtt{Rec}_\sigma(t_1, t_2, t_3) \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \mathtt{Kleisli\text{-}ext}_{\sigma,A}^\mathsf{T}(\mathtt{Rec}_\sigma(\lambda x.\ \llbracket t_1 \rrbracket_{\mathcal{D}^\mathsf{T}}^A(\eta_A^\mathsf{T}(x)))(\llbracket t_2 \rrbracket_{\mathcal{D}^\mathsf{T}}^A))(\llbracket t_3 \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$$

$$\llbracket \lambda(x : \sigma).\ t \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \lambda(x : \llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A).\ \llbracket t \rrbracket_{\mathcal{D}^\mathsf{T}}^A$$

$$\llbracket t_1(t_2) \rrbracket_{\mathcal{D}^\mathsf{T}}^A :\equiv \llbracket t_1 \rrbracket_{\mathcal{D}^\mathsf{T}}^A(\llbracket t_2 \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$$

■ **Figure 4** Internal dialogue translation of System T with motive $A : \mathsf{Type}^\mathsf{T}$.

▶ **Definition 18** (⟳). *The **internal Kleisli extension** is the following closed System T term*

$$\mathtt{kleisli\text{-}ext}_A^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota)) \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota) \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota))$$

$$\mathtt{kleisli\text{-}ext}_A^\mathsf{T} :\equiv \lambda f.\ \lambda d.\ \lambda \eta'.\ \lambda \beta'.\ d(\lambda(x : \iota).\ f(x)(\eta')(\beta'))(\beta')$$

▶ **Definition 19** (⟳). *The **internal functor action** is the following closed System T term*

$$\mathcal{D}\text{-}\mathtt{functor}_A^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota) \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota))$$

$$\mathcal{D}\text{-}\mathtt{functor}_A^\mathsf{T} :\equiv \lambda f.\ \mathtt{kleisli\text{-}ext}_A^\mathsf{T}(\lambda x.\ \eta_A^\mathsf{T}(f(x)))$$

For the following definition we use the translation of types defined ahead in Fig. 4.

▶ **Definition 20** (⟳). *The **generalised internal Kleisli extension** is a family of closed System T terms indexed by $\sigma : \mathsf{Type}^\mathsf{T}$. These are defined by induction on the structure of σ.*

$$\mathtt{Kleisli\text{-}ext}_{\sigma,A}^\mathsf{T} : (\sigma : \mathsf{Type}^\mathsf{T}) \to \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A) \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota) \Rightarrow \llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$$

$$\mathtt{Kleisli\text{-}ext}_{\iota,A}^\mathsf{T} :\equiv \mathtt{kleisli\text{-}ext}_A^\mathsf{T}$$

$$\mathtt{Kleisli\text{-}ext}_{\sigma_1 \Rightarrow \sigma_2, A}^\mathsf{T} :\equiv \lambda f.\ \lambda d.\ \lambda s.\ \mathtt{Kleisli\text{-}ext}_{\sigma_2,A}^\mathsf{T}(\lambda x.\ f(x)(s))(d)$$

As made clear from the type signature, $\mathtt{Kleisli\text{-}ext}_{\sigma,A}^\mathsf{T}$ depends on the System T type σ, which is only available in the metatheory. During the internal translation we will always know what σ at which point we get a fixed System T term. With this we are now able to define the internal translation and the associated dialogue tree operators.

▶ **Definition 21** (⟳). *The **internal dialogue translation** of System T is defined in Fig. 4.*

▶ **Definition 22** (⟳). *The **internal generic sequence** is given by the term*

$$\mathtt{generic}_A^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_A^\mathsf{T}(\iota) \Rightarrow \mathcal{D}_A^\mathsf{T}(\iota))$$

$$\mathtt{generic}_A^\mathsf{T} :\equiv \mathtt{kleisli\text{-}ext}_A^\mathsf{T}(\beta_A^\mathsf{T}(\eta_A^\mathsf{T}))$$

▶ **Definition 23** (⟳). *The **internal dialogue tree operator** is the metatheoretic function*

$$\mathtt{dialogue\text{-}tree}_A^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota) \to \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_A^\mathsf{T}(\iota))$$

$$\mathtt{dialogue\text{-}tree}_A^\mathsf{T}(t) :\equiv \llbracket t \rrbracket_{\mathcal{D}^\mathsf{T}}^A(\mathtt{generic}_A^\mathsf{T})$$

As made explicit in the type signature, this operator lives in the metatheory but for any term $t : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota)$ gives another term $\mathtt{dialogue\text{-}tree}_A^\mathsf{T}(t) : \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_A^\mathsf{T}(\iota))$. Of course, if we could fully internalise this dialogue operator then we could further implement a moduli of continuity operator inside System T, which as we discussed, is impossible.

4.2 Avoiding Function Extensionality

Without functional extensionality in the metatheory, it is not provable that the metatheoretical equality and extensional equality of functions coincide. The correctness results we are interested, e.g. Thm. 37, are about equalities of natural numbers, so intuitively they should not rely crucially on function extensionality. It is not so simple however, for in the proofs of said results we will quickly encounter cases where we must reason about equality of higher-order functions. The solution is to use hereditarily extensional equality [24] instead of the metatheoretic or general extensional equality.

▶ **Definition 24** (⟳). *Given a System T type σ and elements $x, y : [\![\sigma]\!]_{\textbf{Set}}$, we define the* ***hereditarily extensionally equality*** *by induction on σ:*

$$\boxed{x \approx_\sigma y} \qquad \frac{n =_\mathbb{N} m}{n \approx_\iota m} \qquad \frac{\forall\, x, y : [\![\sigma_1]\!]_{\textbf{Set}}, \; x \approx_{\sigma_1} y \to f(x) \approx_{\sigma_2} g(y)}{f \approx_{\sigma_1 \Rightarrow \sigma_2} g}$$

When clear from context we omit the type annotation from the relation symbol. At ι, we can see that hereditarily extensional equality coincides with with equality of natural numbers, and at $\iota \Rightarrow \iota$ it coincides with extensional equality of functions $\mathbb{N} \to \mathbb{N}$. As we look at higher types it diverges from extensional equality of functions and in general we will not be able to prove it reflexive. Fortunately, all functions we need are provably well-behaved.

▶ **Lemma 25** (⟳). *For all System T types σ, the binary relation $\approx_\sigma$ is symmetric and transitive. Furthermore, if σ is of the shape $\sigma ::= \iota \mid \iota \Rightarrow \sigma$ then it is also reflexive.*

▶ **Lemma 26** (⟳). *For all closed $t : \mathsf{Term}_0^\mathsf{T}(\sigma)$, we have that $[\![t]\!]_{\textbf{Set}} \approx_\sigma [\![t]\!]_{\textbf{Set}}$.*

4.3 Correctness of the Syntactic Translation

For the correctness of the internal dialogue translation we expect it to agree with the more familiar inductive dialogue translation, meaning we must somehow equate inductive dialogue trees with their Church-encodings. We can already turn System T dialogue trees into metatheoretic Church-encoded trees with the set semantics $[\![-]\!]_{\textbf{Set}}$. For the inductive trees, it suffices to use the following encoding function, from which we define the correctness relation.

▶ **Definition 27** (⟳). *Given a type $A : \mathsf{Type}^\mathsf{T}$ we define the following encode function by induction*

$$\mathsf{encode}_A : \mathcal{D}(\mathbb{N}) \to [\![\mathcal{D}_A^\mathsf{T}(\iota)]\!]_{\textbf{Set}}$$
$$\mathsf{encode}_A(\eta(z)) :\equiv [\![\eta_A^\mathsf{T}]\!]_{\textbf{Set}}(z)$$
$$\mathsf{encode}_A(\beta(\phi)(x)) :\equiv [\![\beta_A^\mathsf{T}]\!]_{\textbf{Set}}(\mathsf{encode}_A \circ \phi)(x)$$

▶ **Definition 28** (⟳ Dialogue Correctness Logical Relation). *Given $\sigma : \mathsf{Type}^\mathsf{T}$, $x : [\![\sigma]\!]_\mathcal{D}$ and a family of closed terms $y : (A : \mathsf{Type}^\mathsf{T}) \to \mathsf{Term}_0^\mathsf{T}([\![\sigma]\!]_{\mathcal{D}^\mathsf{T}}^A)$, we define the* ***dialogue correctness logical relation*** *by induction on σ:*

$$\boxed{x \sim y \in \sigma} \qquad \frac{\forall\, A : \mathsf{Type}^\mathsf{T}, \; \mathsf{encode}_A(d) \approx_\iota t_A}{d \sim t \in \iota}$$

$$\frac{\forall\, x : [\![\sigma_1]\!]_\mathcal{D}, \; \forall\, y : (A : \mathsf{Type}^\mathsf{T}) \to \mathsf{Term}_0^\mathsf{T}([\![\sigma_2]\!]_{\mathcal{D}^\mathsf{T}}^A), \; x \sim y \in \sigma_1 \to f(x) \sim \{g_A(y_A)\}_{A : \mathsf{Type}^\mathsf{T}} \in \sigma_2}{f \sim g \in \sigma_1 \Rightarrow \sigma_2}$$

We extend this relation to act pointwise on contexts Γ and denote this extension $\gamma_1 \sim \gamma_2 \in \Gamma$.

For a number of the proofs it will be important that normalisation preserves this relation. For our purposes it suffices to compute in the Set model, once again avoiding conversion rules.

▶ **Lemma 29** (✐). *Fix two family of terms $t, s : (A : \mathsf{Type}^\mathsf{T}) \to \mathsf{Term}_0^\mathsf{T}(\llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$ and $x : \llbracket \sigma \rrbracket_\mathcal{D}$. If for all $A : \mathsf{Type}^\mathsf{T}$ we have $\llbracket t_A \rrbracket_\mathbf{Set} \approx_{\llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A} \llbracket s_A \rrbracket_\mathbf{Set}$ then $x \sim t \in \sigma$ implies $x \sim s \in \sigma$.*

Proof. We proceed by induction on the System T type σ. If σ is of the form ι then the result follows by symmetry and transitivity of $\approx$. If σ is a function type then we apply the inductive hypothesis pointwise.   ◀

With this we can now prove some lemmas about the monadic operations of internal dialogue trees. These will be used in the proof of the fundamental lemma, that is in Lem. 33.

▶ **Lemma 30** (✐). *Given a type $A : \mathsf{Type}^\mathsf{T}$, a dialogue tree $d : \llbracket \iota \rrbracket_\mathcal{D}$, and functions $f_1 : \mathbb{N} \to \llbracket \iota \rrbracket_\mathcal{D}$ and $f_2 : \mathbb{N} \to \llbracket \mathcal{D}_A^\mathsf{T}(\iota) \rrbracket_\mathbf{Set}$, if for all $n : \mathbb{N}$ we have $\mathsf{encode}_A(f_1(n)) \approx_{\mathcal{D}_A^\mathsf{T}(\iota)} f_2(n)$ then we have in addition $\mathsf{encode}_A(\mathsf{kleisli\text{-}ext}(f_1)(d)) \approx_{\mathcal{D}_A^\mathsf{T}(\iota)} \llbracket \mathtt{kleisli\text{-}ext}_A^\mathsf{T} \rrbracket_\mathbf{Set}(f_2)(\mathsf{encode}_A(d))$.*

Proof. We proceed by induction on the dialogue tree d. In the case that d is of the form $\eta(n)$, then the left-hand-side computes to $\mathsf{encode}_A(f_1(n))$ and the right side computes to $\llbracket f_2(\underline{n}) \rrbracket_\mathbf{Set}$, but these are equal by our assumption. In the case that d is of the form $\beta(\phi)(i)$, then we can apply the inductive hypothesis to each subtree $\phi(n)$.   ◀

▶ **Corollary 31** (✐). *Given a type $A : \mathsf{Type}^\mathsf{T}$, a dialogue tree $d : \mathcal{D}(\mathbb{N})$ and functions $g_1, g_2 : \mathbb{N} \to \mathbb{N}$, if $g_1 \approx g_2$ then $\mathsf{encode}_A(\mathcal{D}\text{-}\mathsf{functor}(g_2)(d)) \approx_{\mathcal{D}_A^\mathsf{T}(\iota)} \llbracket \mathcal{D}\text{-}\mathtt{functor}_A^\mathsf{T} \rrbracket_\mathbf{Set}(g_1)(\mathsf{encode})(d)A$.*

Proof. Instantiate Lem. 30 with $\eta \circ g_1$ and $\llbracket \eta_A^\mathsf{T} \rrbracket_\mathbf{Set} \circ g_2$.   ◀

▶ **Lemma 32** (✐ Kleisli Lemma). *Fix a function $f : \mathbb{N} \to \llbracket \sigma \rrbracket_\mathcal{D}$, a dialogue tree $n : \mathcal{D}(\mathbb{N})$, and families of terms $g : (A : \mathsf{Type}^\mathsf{T}) \to \mathsf{Term}_0^\mathsf{T}(\llbracket \sigma \rrbracket_{\mathcal{D}^\mathsf{T}}^A)$ and $m : (A : \mathsf{Type}^\mathsf{T}) \to \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_A^\mathsf{T}(\iota))$. If $n \sim m \in \iota$ and $\forall i : \mathbb{N}, f(i) \sim g(\underline{i}) \in \sigma$ then $\mathsf{Kleisli\text{-}ext}_\sigma(f)(n) \sim \mathtt{Kleisli\text{-}ext}_{\sigma,A}^\mathsf{T}(g)(m) \in \sigma$.*

Proof. We proceed by induction on the System T type σ. For the base case we must show for all $A : \mathsf{Type}^\mathsf{T}$ that $\llbracket \mathtt{kleisli\text{-}ext}_A^\mathsf{T}(g_A)(m_A) \rrbracket_\mathbf{Set} \approx \mathsf{encode}_A(\mathsf{kleisli\text{-}ext}(f)(n))$. By the assumption that $n \sim m \in \iota$, the left hand side equals $\llbracket \mathtt{kleisli\text{-}ext}_A^\mathsf{T} \rrbracket_\mathbf{Set}(\llbracket g_A \rrbracket_\mathbf{Set})(\mathsf{encode}_A(n))$ and by Lem. 30 this equals the right hand side. For the recursive step the inductive hypothesis suffices since generalised Kleisli extension at $\sigma_1 \Rightarrow \sigma_2$ is defined in terms of σ_2.   ◀

▶ **Lemma 33** (✐ Fundamental Lemma). *Given $t : \mathsf{Term}^\mathsf{T}(\Gamma, \sigma)$, an assignment $\gamma_1 : \llbracket \Gamma \rrbracket_\mathcal{D}$ and a substitution γ_2 from $\diamond$ to $\llbracket \Gamma \rrbracket_{\mathcal{D}^\mathsf{T}}$, if $\gamma_1 \sim \gamma_2 \in \Gamma$ then $\llbracket t \rrbracket_\mathcal{D}^{\gamma_1} \sim (\llbracket t \rrbracket_{\mathcal{D}^\mathsf{T}})\{\gamma_2\} \in \sigma$.*

Proof. We proceed by induction on t. We will often implicitly use Lem. 29 whenever we must show a term satisfies the logical relation after some normalisation occurs. The variable case follows from the assumption that $\gamma_1 \sim \gamma_2 \in \Gamma$. The application case follows by induction on both subterms and the lambda abstraction case follows by expanding the substitution and applying the inductive hypothesis to the lambda body. The $\mathtt{Zero}$ case amounts to showing that $\llbracket \eta_A^\mathsf{T}(\mathtt{Zero}) \rrbracket_\mathbf{Set} \approx \mathsf{encode}_A(\eta(0))$ which is seen to hold by expanding both definitions. The $\mathtt{Succ}(t_1)$ case follows by applying the inductive hypothesis to t_1 and using Cor. 31 to commute the encode_A to the outside. Finally, the $\mathtt{Rec}_\sigma(t_1, t_2, t_3)$ case follows from Lem. 32 which has two assumptions: for the first, we show for all n by induction that $\mathsf{Natrec}(\llbracket t_1 \rrbracket_\mathcal{D}^{\gamma_1})(\llbracket t_2 \rrbracket_\mathcal{D}^{\gamma_1})(n) \sim (\mathtt{Rec}(\llbracket t_1 \rrbracket_{\mathcal{D}^\mathsf{T}} \circ \eta_A^\mathsf{T}, \llbracket t_2 \rrbracket_{\mathcal{D}^\mathsf{T}}, \underline{n}))\{\gamma_2\} \in \sigma$ using the inductive hypothesis from the subterms t_1 and t_2. The second assumption holds by the inductive hypothesis from the subterm t_3.   ◀

▶ **Lemma 34** (⟳ Dialogue tree agreement). *Given $A : \mathsf{Type}^\mathsf{T}$ and $t : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota)$, we have $[\![\mathsf{dialogue\text{-}tree}_A^\mathsf{T}(t)]\!]_{\mathbf{Set}} \approx_{\mathcal{D}_A^\mathsf{T}(\iota)} \mathsf{encode}_A(\mathsf{dialogue\text{-}tree}(t))$.*

Proof. It suffices to show $[\![t]\!]_\mathcal{D}(\mathsf{generic}) \sim \{[\![t]\!]_{\mathcal{D}^\mathsf{T}}^A(\mathsf{generic}_A^\mathsf{T})\}_{A:\mathsf{Type}^\mathsf{T}} \in \iota$. By Lem. 33 we know that $[\![t]\!]_\mathcal{D} \sim [\![t]\!]_{\mathcal{D}^\mathsf{T}} \in (\iota \Rightarrow \iota) \Rightarrow \iota$, leaving us to show $\mathsf{generic} \sim \mathsf{generic}_A^\mathsf{T} \in \iota \Rightarrow \iota$. This follows from Lem. 32 as both the generic sequences are defined with kleisli extension. ◀

▶ **Definition 35** (⟳). *We define the **internal dialogue operator** as the closed term*

$$\mathsf{dialogue}^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_{(\iota \Rightarrow \iota) \Rightarrow \iota}^\mathsf{T}(\iota) \Rightarrow (\iota \Rightarrow \iota) \Rightarrow \iota)$$
$$\mathsf{dialogue}^\mathsf{T} :\equiv \lambda d.\ d(\lambda z.\ \lambda\_.\ z)(\lambda \phi.\ \lambda x.\ \lambda \alpha.\ \phi(\alpha(x))(\alpha))$$

▶ **Lemma 36** (⟳). *Given a dialogue tree $d : \mathcal{D}(\mathbb{N})$ and sequence $\alpha : \mathbb{N} \to \mathbb{N}$ we have the following equality of natural numbers $\mathsf{dialogue}(d)(\alpha) = [\![\mathsf{dialogue}^\mathsf{T}]\!]_{\mathbf{Set}}(\mathsf{encode}_{(\iota \Rightarrow \iota) \Rightarrow \iota}(d))(\alpha)$.*

▶ **Theorem 37** (⟳ Correctness of $\mathsf{dialogue\text{-}tree}_A^\mathsf{T}$). *Given $t : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota)$ and a sequence $\alpha : \mathbb{N} \to \mathbb{N}$, we have $[\![t]\!]_{\mathbf{Set}}(\alpha) = [\![\mathsf{dialogue}^\mathsf{T}(\mathsf{dialogue\text{-}tree}_{(\iota \Rightarrow \iota) \Rightarrow \iota}^\mathsf{T}(t))]\!]_{\mathbf{Set}}(\alpha)$.*

Proof. By Lem. 34 we know that $[\![\mathsf{dialogue\text{-}tree}_A^\mathsf{T}(t)]\!]_{\mathbf{Set}} \approx \mathsf{encode}_A(\mathsf{dialogue\text{-}tree}(t))$. and by Lem. 26 we also have $[\![\mathsf{dialogue}^\mathsf{T}]\!]_{\mathbf{Set}} \approx [\![\mathsf{dialogue}^\mathsf{T}]\!]_{\mathbf{Set}}$. Composing these we get $[\![\mathsf{dialogue}^\mathsf{T}(\mathsf{dialogue\text{-}tree}_A^\mathsf{T}(t))]\!]_{\mathbf{Set}} = [\![\mathsf{dialogue}^\mathsf{T}]\!]_{\mathbf{Set}}(\mathsf{encode}_{(\iota \Rightarrow \iota) \Rightarrow \iota}(\mathsf{dialogue\text{-}tree}(t)))$, noting that we changed from $\approx$ to the metatheoretic equality as they coincide for natural numbers. Using Lem. 36 we can replace the right hand side by $\mathsf{dialogue}(\mathsf{dialogue\text{-}tree}(t))(\alpha)$ and chaining this with Thm. 16 we get $[\![t]\!]_{\mathbf{Set}}(\alpha)$ as needed. ◀

5 Computing Moduli of Continuity Internally

As we have previously explained, the fundamental idea of a dialogue tree is to encode information on how a System T term of type $(\iota \Rightarrow \iota) \Rightarrow \iota$ interacts with a given argument while computing a result. In Sec. 4, we presented the System T encodings of such dialogue trees. We now proceed to define two System T operators that compute *moduli of continuity* using the information contained in these internal trees:

1. One, presented in this section, that takes an $\mathbb{N}$-branching dialogue tree and computes the modulus of continuity of the function it encodes *at a given point* of the Baire space.
2. Another one, presented in Sec. 6, that takes a binary-branching dialogue tree and computes the *modulus of uniform continuity* of the function that it encodes.

We assume we have a max function on natural numbers as well as a corresponding System T term $\mathsf{max}^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}(\iota \Rightarrow \iota \Rightarrow \iota)$, the details of which can be found in our formalisation.

The main content of our modulus operator is given by a function that we call $\mathsf{max\text{-}q}$, which computes the maximum question occurring on a given path of a dialogue tree. We continue to follow our convention of implementing constructions involving dialogue trees in two forms: on external inductive type encodings, and on internal Church encodings. This is not only for the sake of clarity but also to enable the precise formulation of the lemmas that we need for our main result. Accordingly, we define the following two functions:

1. $\mathsf{max\text{-}q}$ on external inductive type encodings.
2. $\mathsf{max\text{-}q}^\mathsf{T}$ on internal Church encodings, implemented in System T.

▶ **Definition 38** (↻). *We define the **max question along a path** by induction on dialogue trees:*

$$\mathsf{max\text{-}q} : \mathcal{D}(\mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$
$$\mathsf{max\text{-}q}(\eta(n))(\alpha) :\equiv 0$$
$$\mathsf{max\text{-}q}(\beta(\phi)(n))(\alpha) :\equiv \mathsf{max}(n)(\mathsf{max\text{-}q}(\phi(\alpha(n)))(\alpha))$$

▶ **Definition 39** (↻). *We define the **internal max question along a path** function as*

$$\mathsf{max\text{-}q}^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_\iota^\mathsf{T}(\iota) \Rightarrow (\iota \Rightarrow \iota) \Rightarrow \iota)$$
$$\mathsf{max\text{-}q}^\mathsf{T} :\equiv \lambda d.\ \lambda\alpha.\ d(\lambda\_.\ \mathtt{Zero})(\lambda g.\ \lambda x.\ \mathtt{max}^\mathsf{T}(x)(g(\alpha(x))))$$

▶ **Lemma 40** (↻). *For all dialogue trees $d : \mathcal{D}(\mathbb{N})$ and sequences $\alpha : \mathbb{N} \to \mathbb{N}$, we have that* $\mathsf{max\text{-}q}(d)(\alpha) = [\![\mathsf{max\text{-}q}^\mathsf{T}]\!]_{\mathbf{Set}}(\mathsf{encode}_\iota(d))(\alpha)$.

We are now ready to define our modulus operators.

▶ **Definition 41** (↻). *We define the **external and internal modulus operators** as*

$$\mathsf{modulus} : \mathcal{D}(\mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \qquad \mathsf{modulus}^\mathsf{T} : \mathsf{Term}_0^\mathsf{T}(\mathcal{D}_\iota^\mathsf{T}(\iota) \Rightarrow (\iota \Rightarrow \iota) \Rightarrow \iota)$$
$$\mathsf{modulus}(d)(\alpha) :\equiv 1 + \mathsf{max\text{-}q}(d)(\alpha) \qquad \mathsf{modulus}^\mathsf{T} :\equiv \lambda d.\ \lambda\alpha.\ \mathtt{Succ}(\mathsf{max\text{-}q}^\mathsf{T}(d)(\alpha))$$

Before we proceed to prove the correctness of the internal modulus of continuity operator, we first formally define the notion of modulus of continuity.

▶ **Definition 42** (↻ Modulus of continuity). *Let $f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ be a function on the Baire space and let $\alpha : \mathbb{N} \to \mathbb{N}$ be a point. A natural number $m : \mathbb{N}$ is a **modulus of continuity** for f at α if the following holds: $\forall \beta : \mathbb{N} \to \mathbb{N}.\ \alpha =_m \beta \to f(\alpha) = f(\beta)$.*

We mentioned in Remark 8 that the computation encoded by any dialogue tree is continuous. The function $\mathsf{modulus}$, which we defined above, can be seen as the *computational content* witnessing this fact.

▶ **Lemma 43.** *Given any dialogue tree $d : \mathcal{D}(\mathbb{N})$ and any sequence $\alpha : \mathbb{N} \to \mathbb{N}$, $\mathsf{modulus}(d)(\alpha)$ is a modulus of continuity of the function $\mathsf{dialogue}(d) : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ at point α.*

In order to prove the correctness of $\mathsf{modulus}^\mathsf{T}$, we also need to know that the internal modulus operator agrees with the external modulus operator. In preparation for this, we prove the following lemma connecting the internal and external processes of extracting moduli of continuity from the respective encodings.

▶ **Lemma 44** (↻). *For every term $t : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota)$ and every point $\alpha : \mathbb{N} \to \mathbb{N}$ of the Baire space, we have $[\![\mathsf{max\text{-}q}^\mathsf{T}(\mathtt{dialogue\text{-}tree}_\iota^\mathsf{T}(t))]\!]_{\mathbf{Set}}(\alpha) = \mathsf{max\text{-}q}(\mathsf{dialogue\text{-}tree}(t))(\alpha)$.*

Proof. Fix a term $t : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota)$ and sequence $\alpha : \mathbb{N} \to \mathbb{N}$. We have the following

$$[\![\mathsf{max\text{-}q}^\mathsf{T}(\mathtt{dialogue\text{-}tree}_\iota^\mathsf{T}(t))]\!]_{\mathbf{Set}}(\alpha)$$
$$= [\![\mathsf{max\text{-}q}^\mathsf{T}]\!]_{\mathbf{Set}}([\![\mathtt{dialogue\text{-}tree}_\iota^\mathsf{T}(t)]\!]_{\mathbf{Set}})(\alpha) \qquad\qquad \text{(by Def. 2)}$$
$$= [\![\mathsf{max\text{-}q}^\mathsf{T}]\!]_{\mathbf{Set}}(\mathsf{encode}_\iota(\mathsf{dialogue\text{-}tree}(t)))(\alpha) \qquad\qquad \text{(by Lem. 34)}$$
$$= \mathsf{max\text{-}q}(\mathsf{dialogue\text{-}tree}(t))(\alpha) \qquad\qquad\qquad\qquad \text{(by Lem. 40)}$$

which completes the proof.   ◀

The key step in the above proof is the use of Lem. 34, which relies on the logical relation from Def. 28 and the hereditarily extensional equality relation from Def. 24. With Lem. 44 established, we may now proceed to prove our main result for functions on the Baire space: $\mathtt{modulus}^\mathsf{T}$ computes moduli of continuity for System T-definable functions on the Baire space.

▶ **Theorem 45** (⟳ Correctness of $\mathtt{modulus}^\mathsf{T}$). *Let* $t : (\iota \Rightarrow \iota) \Rightarrow \iota$ *be a System T function on the Baire space. The result* $[\![\mathtt{modulus}^\mathsf{T}(t)]\!]_\mathbf{Set}$ *is a function giving a modulus of continuity for* $[\![t]\!]_\mathbf{Set} : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ *at each point of the Baire space.*

Proof. We know by Thm. 16 that t can be encoded by the inductive dialogue tree $\mathsf{dialogue\text{-}tree}(t)$, i.e. we have $[\![t]\!]_\mathbf{Set}(\alpha) = \mathsf{dialogue}(\mathsf{dialogue\text{-}tree}(t))(\alpha)$ for every $\alpha : \mathbb{N} \to \mathbb{N}$. We therefore know, by Lem. 43, that $\mathsf{modulus}$ gives moduli of continuity for $[\![t]\!]_\mathbf{Set}$, which is to say, we just have to show $[\![\mathtt{modulus}^\mathsf{T}(t)]\!]_\mathbf{Set}(\alpha) = \mathsf{modulus}(\mathsf{dialogue\text{-}tree}(t))(\alpha)$, for all $\alpha : \mathbb{N} \to \mathbb{N}$, and this follows from Lem. 44.   ◀

6 Computing Moduli of Uniform Continuity Internally

We now extend our work from the previous section as to define an operator computing moduli of *uniform continuity* of functions on the Cantor space. Our development here closely follows that of Sec. 5: we implement an analogue of the $\mathsf{max\text{-}q}$ function from Def. 38, which we call $\mathsf{max\text{-}q}_2$. This computes the maximum question in a $\mathbb{B}$-branching dialogue tree. Unlike the $\mathsf{max\text{-}q}$ operator, however, it performs this computation over the entire tree rather than a specific path, exploiting the finiteness of the branching in the context of the Cantor space.

Once again, we start by defining two operators:

1. $\mathsf{max\text{-}q}_2$ for external inductive type encodings.
2. $\mathsf{max\text{-}q}_2^\mathsf{T}$ for internal Church encodings.

System T does not include a type of Booleans, so we cannot directly talk about the points of the Cantor space in it i.e. functions of type $\alpha : \mathbb{N} \to \mathbb{B}$. To avoid extending System T with another ground type, we work with the embedding of the Cantor space into the Baire space, which we define below.

▶ **Definition 46** (⟳). *The embedding of the Cantor space into the Baire space is given by the function* $\mathsf{embed}_C : (\mathbb{N} \to \mathbb{B}) \to (\mathbb{N} \to \mathbb{N})$, *defined as* $\mathsf{embed}_C(\alpha)(i) = \mathsf{embed}_\mathbb{B}(\alpha(i))$, *where* $\mathsf{embed}_\mathbb{B}$ *denotes the function mapping* false *to* 0 *and* true *to* 1.

The fact that we are working with such an embedding of the Cantor space implies that we have to work with encodings of $\mathbb{B}$-branching dialogue trees into $\mathbb{N}$-branching ones. Accordingly, we define a pruning function that converts a $\mathbb{B}$-branching tree back into a $\mathbb{N}$-branching one.

▶ **Definition 47** (⟳). *We define a **pruning operation** by induction on dialogue trees by*

$\mathsf{prune} : \mathcal{D}(\mathbb{N}) \to \mathsf{Dial}(\mathbb{N})(\mathbb{B})(\mathbb{N})$

$\mathsf{prune}(\eta(n)) :\equiv \eta(n)$

$\mathsf{prune}(\beta(\phi)(n)) :\equiv \beta(\mathsf{prune} \circ \phi \circ \mathsf{embed}_\mathbb{B})(n)$

where the $\mathsf{embed}_\mathbb{B}$ *function maps* false *to* 0 *and* true *to* 1.

We now proceed to define the binary versions of the $\mathsf{max\text{-}q}$ functions.

▶ **Definition 48** (⟲). *We define the function* max-q$_2$ *as follows:*

max-q$_2$: $\mathsf{Dial}(\mathbb{N})(\mathbb{B})(\mathbb{N}) \to \mathbb{N}$
max-q$_2(\eta(n)) :\equiv 0$
max-q$_2(\beta(\phi)(n)) :\equiv \mathsf{max}(n)(\mathsf{max}(\mathsf{max\text{-}q}_2(\phi(0)))(\mathsf{max\text{-}q}_2(\phi(1))))$

▶ **Definition 49** (⟲). *We define the internal Church encoding version of* max-q$_2$ *as*

$\mathtt{max\text{-}q}_2^{\mathsf{T}}$: $\mathsf{Term}_0^{\mathsf{T}}(\mathcal{D}_\iota^{\mathsf{T}}(\iota) \Rightarrow \iota)$
$\mathtt{max\text{-}q}_2^{\mathsf{T}} :\equiv \lambda d.\ d(\lambda\_.\ \mathtt{Zero})(\lambda g.\ \lambda x.\ \mathtt{max}^{\mathsf{T}}(x)(\mathtt{max}^{\mathsf{T}}(g(\underline{0}))(g(\underline{1}))))$

▶ **Lemma 50** (⟲). *Given a tree* d : $\mathcal{D}(\mathbb{N})$, *we have* max-q$_2(\mathsf{prune}(d))$ $=$ $[\![\mathtt{max\text{-}q}_2^{\mathsf{T}}]\!]_{\mathbf{Set}}(\mathsf{encode}_\iota(d))$.

▶ **Definition 51** (⟲). *We define the **external and internal uniform modulus operators** as:*

$$
\begin{array}{ll}
\mathsf{modulus}_2 : \mathsf{Dial}(\mathbb{N})(\mathbb{B})(\mathbb{N}) \to \mathbb{N} & \mathtt{modulus}_2^{\mathsf{T}} : \mathsf{Term}_0^{\mathsf{T}}(\mathcal{D}_\iota^{\mathsf{T}}(\iota) \Rightarrow \iota) \\
\mathsf{modulus}_2(d) :\equiv 1 + \mathsf{max\text{-}q}_2(d) & \mathtt{modulus}_2^{\mathsf{T}} :\equiv \lambda d.\ \mathtt{Succ}(\mathtt{max\text{-}q}_2^{\mathsf{T}}(d))
\end{array}
$$

▶ **Definition 52** (⟲). *Given a function on the Cantor space* $f : (\mathbb{N} \to \mathbb{B}) \to \mathbb{N}$, *a natural number* $m : \mathbb{N}$ *is said to be a **modulus of uniform continuity** for* f *if the following holds:* $\forall \alpha, \beta : \mathbb{N} \to \mathbb{B}.\ \alpha =_m \beta \to f(\alpha) = f(\beta)$.

From Remark 8, we also know that the computation encoded by any $\mathbb{B}$-branching dialogue tree is *uniformly* continuous. The function modulus$_2$ above can be seen as the computational content of this fact.

▶ **Lemma 53** (⟲). *Given any dialogue tree* $d : \mathsf{Dial}(\mathbb{N})(\mathbb{N})(\mathbb{N})$, *the result* modulus$_2(\mathsf{prune}(d))$ *is a modulus of uniform continuity of the function* dialogue$(\mathsf{prune}(d)) : (\mathbb{N} \to \mathbb{B}) \to \mathbb{N}$.

Towards proving the correctness of the internal modulus of uniform continuity operator in Thm. 55, we now prove the following lemma, connecting the internal and external processes of extracting moduli of uniform continuity from the respective encodings of dialogue trees.

▶ **Lemma 54** (⟲). *For all terms* $t : \mathsf{Term}_0^{\mathsf{T}}((\iota \Rightarrow \iota) \Rightarrow \iota)$, *the external and internal uniform max questions agree, i.e.* $[\![\mathtt{max\text{-}q}_2^{\mathsf{T}}(\mathtt{dialogue\text{-}tree}_\iota^{\mathsf{T}}(t))]\!]_{\mathbf{Set}} = \mathsf{max\text{-}q}_2(\mathsf{prune}(\mathsf{dialogue\text{-}tree}(t)))$.

Proof. Let $t : \mathsf{Term}_0^{\mathsf{T}}((\iota \Rightarrow \iota) \Rightarrow \iota)$ be a System T term.

$[\![\mathtt{max\text{-}q}_2^{\mathsf{T}}(\mathtt{dialogue\text{-}tree}_\iota^{\mathsf{T}}(t))]\!]_{\mathbf{Set}}$

$\quad = [\![\mathtt{max\text{-}q}_2^{\mathsf{T}}]\!]_{\mathbf{Set}}([\![\mathtt{dialogue\text{-}tree}_\iota^{\mathsf{T}}(t)]\!]_{\mathbf{Set}})$ (by Def. 2)

$\quad = [\![\mathtt{max\text{-}q}_2^{\mathsf{T}}]\!]_{\mathbf{Set}}(\mathsf{encode}_\iota(\mathsf{dialogue\text{-}tree}(t)))$ (by Lem. 34)

$\quad = \mathsf{max\text{-}q}_2(\mathsf{prune}(\mathsf{dialogue\text{-}tree}(t)))$ (by Lem. 50)

◀

Similar to Lem. 44, the key step in the above proof is the use of Lem. 34, which relies on the logical relation from Def. 28. With Lem. 54 established, we can now proceed to prove our main result for functions on the Cantor space: $\mathtt{modulus}_2^{\mathsf{T}}$ computes moduli of uniform continuity for System T-definable functions on the Cantor space.

▶ **Theorem 55** (🗘 Correctness of $\mathtt{modulus}_2^\mathsf{T}$). *Let* $t : \mathsf{Term}_0^\mathsf{T}((\iota \Rightarrow \iota) \Rightarrow \iota)$ *be a System T function on the Baire space. The result* $[\![\mathtt{modulus}_2^\mathsf{T}(t)]\!]_{\mathbf{Set}}$ *is a modulus of uniform continuity of the function* $[\![t]\!]_{\mathbf{Set}} \circ \mathsf{embed}_C$.

Proof. Let $t : (\iota \Rightarrow \iota) \Rightarrow \iota$ be a System T term. We know by Thm. 16 that t can be encoded by the inductive dialogue tree $\mathsf{dialogue\text{-}tree}(t)$. We therefore know that, as given by Lem. 53, $\mathsf{modulus}_2$ gives a modulus of uniform continuity for $[\![t]\!]_{\mathbf{Set}}$, which is to say that we just have to show $[\![\mathtt{max\text{-}q}_2^\mathsf{T}(\mathtt{dialogue\text{-}tree}^\mathsf{T}(t)]\!]_{\mathbf{Set}} = \mathsf{max\text{-}q}_2(\mathsf{prune}(\mathsf{dialogue\text{-}tree}(t)))$ which is given by Lem. 54. ◀

7 Discussion and conclusion

This paper relies on and extends the work on effectful forcing by Escardó [12]. We present the first constructive internalisation of the dialogue trees featured in this technique. In addition to constructing such trees in System T, we define the operators that compute moduli of continuity from dialogue trees inside of System T.

Further related work. Putting dialogue trees aside, other internalisations of effectful forcing have been explored before. In [16] the authors provide a framework for internalising the effectful forcing technique for System T. It abstracts away from dialogue trees to a postulated type with enough structure to carry out the analogous translation. Due to System T's lack of polymorphism, there is a disconnection between Church encoded dialogue trees and inductive dialogue trees. Namely, the type of the former is too big and will contain terms that do not behave at all like dialogue trees. As a result it seems impossible to apply this framework to the case of Churh encoded dialogue trees, requiring instead our more direct approach. Similarly, in [27], with a nucleus as a parameter, a general translation from System T into itself is developed. Assuming some suitable conditions on the parameters of the translation, it is then proved sound through a logical relation With different instantiations of this translation the author is able to derive System T-definable moduli of continuity, as well as System T-definable bar recursion functionals. The present paper strengthens this continuity result by showing that the dialogue tree of a System T function is itself System T-definable.

In the literature one can also find different extensions of Escardó's effectful forcing. In [22] it is extended to prove that System T validates the realizable bar thesis, i.e., Brouwer's bar thesis such that the bar is System T-definable, which is equivalent to the dialogue continuity. In [3] this technique is further generalised to handle dependent type theory, and to then show that all functions on the Baire space of the dependent theory BTT [20] are continuous by building external dialogue trees. The authors rely on a call-by-name interpretation, which as opposed to [12, 22], gives rise to a dialogue-based model of the theory.

In [5] the authors use a different technique to prove the continuity of the $\mathrm{TT}_C^\square$ functions on the Baire and Cantor space. This technique consists of deriving Brouwer trees internally to the theory using effects. $\mathrm{TT}_C^\square$ is an effectful and extensional variant of MLTT, which therefore allows using effects to build such trees internally to the theory, using computations introduced in [19]. While the techniques used in [12, 22, 3] build dialogue trees by induction on terms, such inductive constructions are not internalizable without reflection mechanisms, which are not supported by $\mathrm{TT}_C^\square$. Therefore, [5] relies instead on classical logic to prove finiteness of the computed trees and termination of the internal program.

The above line of work relies on the dialogue-based notion of continuity defined in Def. 7 to derive the continuity of functions on the Baire space and the uniform continuity of functions on the Cantor space.

Open questions and further directions.   It would interesting to determine whether the assumptions of bar induction and *stable* moduli of continuity used by Capretta and Uustalu [4] to prove the equivalence of continuity with the existence of Brouwer trees are strictly necessary or not. Finally, we envisage some possible applications of our result, such as characterising the System T-definable functions $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ in terms of the heights of the trees measured by ordinal notations. For instance, it is natural to conjecture that the heights of such trees are bounded by the ordinal ϵ_0.

References

1 Jeremy Avigad and Solomon Feferman. Gödel's functional ("dialectica") interpretation. In *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 337–405. Elsevier, 1998. `doi:10.1016/S0049-237X(98)80020-7`.

2 Martin Baillon. *Continuity in Type Theory. (Continuité en théorie des types)*. PhD thesis, University of Nantes, France, 2023. URL: `https://tel.archives-ouvertes.fr/tel-04617881`.

3 Martin Baillon, Assia Mahboubi, and Pierre-Marie Pédrot. Gardening with the pythia A model of continuity in a dependent setting. In Florin Manea and Alex Simpson, editors, *CSL*, volume 216 of *LIPIcs*, pages 5:1–5:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.CSL.2022.5`.

4 Venanzio Capretta and Tarmo Uustalu. A coalgebraic view of bar recursion and bar induction. In Bart Jacobs and Christof Löding, editors, *FOSSACS*, volume 9634 of *LNCS*, pages 91–106. Springer, 2016. `doi:10.1007/978-3-662-49630-5_6`.

5 Liron Cohen, Bruno da Rocha Paiva, Vincent Rahli, and Ayberk Tosun. Inductive continuity via brouwer trees. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPICS.MFCS.2023.37`.

6 Thierry Coquand and Guilhem Jaber. A note on forcing and type theory. *Fundam. Inform.*, 100(1-4):43–52, 2010. `doi:10.3233/FI-2010-262`.

7 Thierry Coquand and Guilhem Jaber. A computational interpretation of forcing in type theory. In *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 203–213. Springer, 2012. `doi:10.1007/978-94-007-4435-6_10`.

8 Michael A. E. Dummett. *Elements of Intuitionism*. Clarendon Press, second edition, 2000.

9 Martín Escardó, Bruno da Rocha Paiva, Vincent Rahli, and Ayberk Tosun. Internal effectful forcing in Agda. Source code: `https://github.com/martinescardo/TypeTopology/blob/master/source/EffectfulForcing/Internal/PaperIndex.lagda`, and HTML rendering: `https://cs.bham.ac.uk/~mhe/InternalEffectfulForcing/EffectfulForcing.Internal.index.html`.

10 Martín H. Escardó and Paulo Oliva. Dialogue to Brouwer, 2017. URL: `https://www.cs.bham.ac.uk/~mhe/TypeTopology/EffectfulForcing.MFPSAndVariations.Dialogue-to-Brouwer.html`.

11 Martín H. Escardó and Chuangjie Xu. The inconsistency of a Brouwerian continuity principle with the Curry-Howard interpretation. In *TLCA*, pages 153–164, 2015. `doi:10.4230/LIPIcs.TLCA.2015.153`.

12 Martín H. Escardó. Continuity of Gödel's System T definable functionals via effectful forcing. In *MFPS*, volume 298, pages 119–141. Elsevier B.V, 2013. `doi:10.1016/j.entcs.2013.09.010`.

13 Neil Ghani, Peter G. Hancock, and Dirk Pattinson. Continuous functions on final coalgebras. In Neil Ghani and John Power, editors, *CMCS*, volume 164 of *Electronic Notes in Theoretical Computer Science*, pages 141–155. Elsevier, 2006. `doi:10.1016/j.entcs.2006.06.009`.

14   Neil Ghani, Peter G. Hancock, and Dirk Pattinson. Continuous functions on final coalgebras. In Samson Abramsky, Michael W. Mislove, and Catuscia Palamidessi, editors, *MFPS*, volume 249 of *Electronic Notes in Theoretical Computer Science*, pages 3–18. Elsevier, 2009. `doi:10.1016/j.entcs.2009.07.081`.

15   Neil Ghani, Peter G. Hancock, and Dirk Pattinson. Representations of stream processors using nested fixed points. *Log. Methods Comput. Sci.*, 5(3), 2009. URL: `http://arxiv.org/abs/0905.4813`.

16   Tatsuji Kawai. Representing definable functions of HA$^\omega$ by neighbourhood functions. *Annals of Pure and Applied Logic*, 170(8):891–909, 2019. `doi:10.1016/j.apal.2019.04.011`.

17   S.C. Kleene. Recursive functionals and quantifiers of finite types revisited i. In J.E. Fenstad, R.O. Gandy, and G.E. Sacks, editors, *Generalized Recursion Theory II*, volume 94 of *Studies in Logic and the Foundations of Mathematics*, pages 185–222. Elsevier, 1978. `doi:10.1016/S0049-237X(08)70933-9`.

18   Georg Kreisel. On weak completeness of intuitionistic predicate logic. *J. Symb. Log.*, 27(2):139–158, 1962. `doi:10.2307/2964110`.

19   John Longley. When is a functional program not a functional program? In *ICFP*, pages 1–7, 1999. `doi:10.1145/317636.317775`.

20   Pierre-Marie Pédrot and Nicolas Tabareau. An effectful way to eliminate addiction to dependence. In *LICS*, pages 1–12, 2017. `doi:10.1109/LICS.2017.8005113`.

21   Vincent Rahli and Mark Bickford. A nominal exploration of intuitionism. In *CPP*, pages 130–141, 2016. Extended version: `http://www.nuprl.org/html/Nuprl2Coq/continuity-long.pdf`. `doi:10.1145/2854065.2854077`.

22   Jonathan Sterling. Higher order functions and Brouwer's thesis. *J. Funct. Program.*, 31:e11, 2021. `doi:10.1017/S0956796821000095`.

23   Anne S. Troelstra and Dirk van Dalen. *Constructivism in Mathematics An Introduction*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1988.

24   A.S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. New York, Springer, 1973.

25   A.S. Troelstra. A note on non-extensional operations in connection with continuity and recursiveness. *Indagationes Mathematicae*, 39(5):455–462, 1977. `doi:10.1016/1385-7258(77)90060-9`.

26   Chuangjie Xu. *A continuous computational interpretation of type theories*. PhD thesis, University of Birmingham, UK, 2015. URL: `http://etheses.bham.ac.uk/5967/`.

27   Chuangjie Xu. A Gentzen-Style Monadic Translation of Gödel's System T. In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*, volume 167 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSCD.2020.25`.