

A Gentzen-Style Monadic Translation of Gödel's System T

Chuangjie Xu 

Ludwig-Maximilians-Universität München, Germany

<http://cj-xu.github.io/>

cj-xu@outlook.com

Abstract

We introduce a syntactic translation of Gödel's System T parametrized by a weak notion of a monad, and prove a corresponding fundamental theorem of logical relation. Our translation structurally corresponds to Gentzen's negative translation of classical logic. By instantiating the monad and the logical relation, we reveal the well-known properties and structures of T-definable functionals including majorizability, continuity and bar recursion. Our development has been formalized in the Agda proof assistant.

2012 ACM Subject Classification Theory of computation → Program analysis; Theory of computation → Type theory; Theory of computation → Proof theory

Keywords and phrases monadic translation, Gödel's System T, logical relation, negative translation, majorizability, continuity, bar recursion, Agda

Digital Object Identifier 10.4230/LIPIcs.FSCD.2020.25

Supplementary Material Agda development: <https://github.com/cj-xu/GentzenTrans>

Funding *Chuangjie Xu*: Alexander von Humboldt Foundation and LMUexcellent initiative.

Acknowledgements The author is grateful to Martín Escardó, Hajime Ishihara, Nils Köpp, Paulo Oliva, Thomas Powell, Benno van den Berg, Helmut Schwichtenberg for many fruitful discussions and the anonymous reviewers for their valuable suggestions and comments.

1 Introduction

Via a syntactic translation of Gödel's System T, Oliva and Steila [17] construct functionals of bar recursion whose terminating functional is given by a closed term in System T. The author adapts their method to compute moduli of (uniform) continuity of functions $(\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ that are definable in System T [27]. Inspired by the generalizations of negative translations which replace double negation by an arbitrary nucleus [8, 12, 25], we introduce a monadic translation of System T into itself which unifies those in [17, 27]. This monadic translation structurally corresponds to Gentzen's negative translation.

Our translation is parametrized by a monad-like structure, which we call a nucleus, but without the restriction of satisfying the monad laws. We adopt the standard technique of logical relations to show the soundness of the translation in the sense that each term of T is related to its translation. Because the translation is parametrized by a nucleus, we have to assume that the logical relation holds for the nucleus. Such a soundness theorem is an instance of the *fundamental theorem of logical relation* [21] stating that if a logical relation holds for all constants then so does it for all terms.

Monadic translations have been widely used for assigning semantics to impure languages. Our goal is instead to reveal properties enjoyed by terms of T and to extract witnesses of these properties. For this purpose, the nuclei we work with are not extensions of T, but just simple structures given by types and terms of T, so that the translation remains in T and the extracted witnesses are terms of T. The Gentzen-style translation looks simpler



© Chuangjie Xu;

licensed under Creative Commons License CC-BY

5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020).

Editor: Zena M. Ariola; Article No. 25; pp. 25:1–25:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

than the other variants [19, 24]. But we demonstrate its power and elegance via its various applications including majorizability, (uniform) continuity and bar recursion. Of course these properties of T -definable functionals are well known [5, 7, 11, 13, 17, 19, 27]. The main contribution of the paper, however, is in obtaining these results in a single framework simply by choosing a suitable nucleus that satisfies the logical relation for the target property.

All the results in the paper are formalized in the Agda proof assistant [2], except the introductory section on negative translations of predicate logic. There are some differences in the Agda development. Firstly, it works with de Bruijn indices when representing the syntax of T to avoid handling variable names. Moreover, all the logical relations are defined between the Agda (or type-theoretic) interpretations of T -types. What we have proved in Agda is that the interpretation of any T -term is related to the one of its translation. In this way, we avoid dealing with the computation rules of T because they all hold judgmentally in the Agda interpretation. The Agda development is available at the author's GitHub page to which the link is given above the introduction.

1.1 Proof-theoretic translations

Recall that Gentzen's translation¹ simply places a double negation in front of atomic formulas, disjunctions and existential quantifiers [22]. One can replace double negation with a *nucleus*, that is, an endofunction j on formulas such that for any formulas A, B the following statements are provable:

$$A \rightarrow jA \quad (A \rightarrow jB) \rightarrow jA \rightarrow jB \quad (jA)[t/x] \leftrightarrow j(A[t/x]).$$

Nuclei are also known as *lax modalities* [1] and *strong monad* [8]. But in this paper we adopt the terminology and definition from [25] which brought the technical motivation to this work. Each nucleus determines a proof-theoretic translation of intuitionistic predicate logic IQL into itself, consisting of a formula translation $A \mapsto A_j^G$ defined as follows

$$\begin{aligned} (A \rightarrow B)_j^G &:= A_j^G \rightarrow B_j^G & P_j^G &:= jP \quad \text{for primitive } P \\ (A \wedge B)_j^G &:= A_j^G \wedge B_j^G & (A \vee B)_j^G &:= j(A_j^G \vee B_j^G) \\ (\forall x A)_j^G &:= \forall x A_j^G & (\exists x A)_j^G &:= j\exists x A_j^G \end{aligned}$$

and a soundness theorem stating that $\text{IQL} \vdash A$ implies $\text{IQL} \vdash A_j^G$. Working with different nuclei, one embed a logic system into another:

- if $jA = (A \rightarrow \perp) \rightarrow \perp$, then $\text{CQL} \vdash A$ implies $\text{MQL} \vdash A_j^G$;
- if $jA = (A \rightarrow R) \rightarrow R$ for some predicate variable R , then $\text{CQL} \vdash A$ implies $\text{IQL} \vdash A_j^G$;
- if $jA = A \vee \perp$, then $\text{IQL} \vdash A$ implies $\text{MQL} \vdash A_j^G$;

where CQL stands for classical predicate logic and MQL for minimal predicate logic. These results are well-known (see *e.g.* [12, 25]) and various instances of the translation have been applied in term extraction (see *e.g.* [8, 14])

Under the viewpoint of the proofs-as-programs correspondence, our translation of Gödel's System T presented in Section 2 is exactly a term/program version of the above proof-theoretic translation on minimal propositional logic.

¹ Nowadays it is known as the Gödel-Gentzen negative translation. Gödel's translation places a double negation also in front of the clause for implication, which makes it different from Gentzen's one in affine logic [3].

1.2 Gödel's System T

Recall that the term language of Gödel's System T can be given by the following grammar

$$\begin{aligned} \text{Type } \sigma, \tau &::= \mathbb{N} \mid \sigma \rightarrow \tau \\ \text{Term } t, u &::= x \mid \lambda x^\sigma. t \mid tu \mid 0 \mid \text{suc} \mid \text{rec}_\sigma \end{aligned}$$

where \mathbb{N} is the base type of natural numbers and $\sigma \rightarrow \tau$ the type of functions from σ to τ . A typing judgment takes the form $\Gamma \vdash t : \tau$, where Γ is a context (*i.e.* a list of distinct typed variables $x : \sigma$), t is a term and τ is a type. Here are the typing rules:

$$\begin{array}{c} \Gamma, x : \sigma \vdash x : \sigma \qquad \frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash \lambda x^\sigma. t : \sigma \rightarrow \tau} \qquad \frac{\Gamma \vdash t : \sigma \rightarrow \tau \quad \Gamma \vdash u : \sigma}{\Gamma \vdash tu : \tau} \\ \Gamma \vdash 0 : \mathbb{N} \qquad \Gamma \vdash \text{suc} : \mathbb{N} \rightarrow \mathbb{N} \qquad \Gamma \vdash \text{rec}_\sigma : \sigma \rightarrow (\mathbb{N} \rightarrow \sigma \rightarrow \sigma) \rightarrow \mathbb{N} \rightarrow \sigma \end{array}$$

We call $\Gamma \vdash t : \tau$ a *well-typed* term if it is derivable. We may omit the context Γ and simply write $t : \tau$ or t^τ if it is unambiguous. When mentioning terms of T in the paper, we refer to only the well-typed ones. We often omit superscript and subscript types if they can be easily inferred, and may write:

- $\lambda x_1 x_2 \cdots x_n. t$ instead of $\lambda x_1. \lambda x_2. \cdots \lambda x_n. t$,
- $f(a_1, a_2, \cdots, a_n)$ instead of $((f a_1) a_2) \cdots a_n$,
- $n + 1$ instead of $\text{suc } n$,
- τ^σ instead of $\sigma \rightarrow \tau$, and
- $f \circ g$ instead of $\lambda x. f(gx)$.

Using the primitive recursor, we can for instance define the function $\text{max} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ that returns the greater argument as follows:

$$\text{max} := \text{rec}_{\mathbb{N} \rightarrow \mathbb{N}}(\lambda n^\mathbb{N}. n, \lambda n^\mathbb{N} f^{\mathbb{N} \rightarrow \mathbb{N}}. \text{rec}_{\mathbb{N}}(\text{suc } n, \lambda m^\mathbb{N} g^{\mathbb{N} \rightarrow \mathbb{N}}. \text{suc}(f m))).$$

One can easily verify that the usual defining equations of max

$$\text{max}(0, n) = n \qquad \text{max}(m, 0) = m \qquad \text{max}(\text{suc } m, \text{suc } n) = \text{suc}(\text{max}(m, n))$$

hold using the computation rules of rec

$$\text{rec}_\sigma(a, f, 0) = a \qquad \text{rec}_\sigma(a, f, \text{suc } n) = f(n, \text{rec}_\sigma(a, f, n))$$

where $a : \sigma$ and $f : \mathbb{N} \rightarrow \sigma \rightarrow \sigma$. For the ease of understanding, we will use defining equations rather than T-terms involving rec in the paper.

2 A monadic translation of System T

Our syntactic translation of System T is parametrized by a nucleus, that is, a monad-like structure without the restriction of satisfying the monad laws.

► **Definition 1** (nuclei). A nucleus *relative to T* is a triple $(\text{JN}, \eta, \kappa)$ consisting of a type JN and two terms

$$\eta : \mathbb{N} \rightarrow \text{JN} \qquad \kappa : (\mathbb{N} \rightarrow \text{JN}) \rightarrow \text{JN} \rightarrow \text{JN}$$

of System T.

25:4 A Gentzen-Style Monadic Translation

Note that a nucleus is *not* an extension of \mathbb{T} , but instead a simple structure given by a type and two terms of \mathbb{T} . Therefore, our translation of any term of \mathbb{T} remains in \mathbb{T} rather than some monadic metatheory such as in [19, 24]. The simplest example is the identity nucleus where $\mathbb{J}\mathbb{N}$ is just \mathbb{N} and η, κ are the identity functions of suitable types. More examples are available in Section 3. Though the first component of a nucleus is just a type, we denote it as $\mathbb{J}\mathbb{N}$ because in the generalized notion of a nucleus discussed in Section 4 it will be a map \mathbb{J} on all the types of \mathbb{T} .

We are now ready to construct a syntactic translation of \mathbb{T} into itself:

► **Definition 2 (J-translation).** *Given a nucleus $(\mathbb{J}\mathbb{N}, \eta, \kappa)$, we assign to each type ρ of \mathbb{T} a type $\rho^{\mathbb{J}}$ as follows:*

$$\begin{aligned} \mathbb{N}^{\mathbb{J}} &:= \mathbb{J}\mathbb{N} \\ (\sigma \rightarrow \tau)^{\mathbb{J}} &:= \sigma^{\mathbb{J}} \rightarrow \tau^{\mathbb{J}}. \end{aligned}$$

Each term $\Gamma \vdash t : \rho$ is translated to a term $\Gamma^{\mathbb{J}} \vdash t^{\mathbb{J}} : \rho^{\mathbb{J}}$, where $\Gamma^{\mathbb{J}}$ is a new context assigning each $x : \sigma \in \Gamma$ to a fresh variable $x^{\mathbb{J}} : \sigma^{\mathbb{J}}$, and $t^{\mathbb{J}}$ is translated inductively as follows:

$$\begin{aligned} (x)^{\mathbb{J}} &:= x^{\mathbb{J}} & 0^{\mathbb{J}} &:= \eta 0 \\ (\lambda x.t)^{\mathbb{J}} &:= \lambda x^{\mathbb{J}}.t^{\mathbb{J}} & \text{succ}^{\mathbb{J}} &:= \kappa(\eta \circ \text{succ}) \\ (tu)^{\mathbb{J}} &:= t^{\mathbb{J}}u^{\mathbb{J}} & (\text{rec}_{\sigma})^{\mathbb{J}} &:= \lambda x^{\sigma^{\mathbb{J}}} f^{\mathbb{J}\mathbb{N} \rightarrow \sigma^{\mathbb{J}} \rightarrow \sigma^{\mathbb{J}}}. \text{ke}_{\sigma}(\text{rec}_{\sigma^{\mathbb{J}}}(x, f \circ \eta)) \end{aligned}$$

where $\text{ke}_{\sigma} : (\mathbb{N} \rightarrow \sigma^{\mathbb{J}}) \rightarrow \mathbb{J}\mathbb{N} \rightarrow \sigma^{\mathbb{J}}$ is an extension of κ defined inductively on σ :

$$\begin{aligned} \text{ke}_{\mathbb{N}} &:= \kappa \\ \text{ke}_{\sigma \rightarrow \tau} &:= \lambda g^{\mathbb{N} \rightarrow \sigma^{\mathbb{J}} \rightarrow \tau^{\mathbb{J}}} a^{\mathbb{J}\mathbb{N}} x^{\sigma^{\mathbb{J}}}. \text{ke}_{\tau}(\lambda n^{\mathbb{N}}. g(n, x), a). \end{aligned}$$

We often write \mathbb{J} to denote the nucleus $(\mathbb{J}\mathbb{N}, \eta, \kappa)$ and call the above the \mathbb{J} -translation of \mathbb{T} .

Thanks to the inductive translation of function types into function types, the translation of the simply-typed- λ -calculus fragment of \mathbb{T} is straightforward. There is no need of introducing a nonstandard, monadic notion of function application which plays an essential role in the other monadic translations [19, 24] as discussed in Section 4.

The more interesting part is the translation of the constants. Viewing η as a *unit* operator and κ as a *bind* operator in a monad \mathbb{J} on \mathbb{N} may reveal some intuition behind the translation of 0 and succ : It is natural to expect $n^{\mathbb{J}} = \eta n$ for each numeral $n := \text{succ}^n(0)$. This is indeed the case if the monad laws are satisfied, because $\kappa(\eta \circ -) : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{J}\mathbb{N} \rightarrow \mathbb{J}\mathbb{N}$ which is used to translate succ recovers exactly the “functoriality” of \mathbb{J} . It is also natural to expect $(\text{rec}_{\sigma})^{\mathbb{J}}$ to preserve the computation rules, *i.e.*

$$(\text{rec}_{\sigma})^{\mathbb{J}}(x, f, 0^{\mathbb{J}}) = x \quad (\text{rec}_{\sigma})^{\mathbb{J}}(x, f, (\text{succ } n)^{\mathbb{J}}) = f(n^{\mathbb{J}}, (\text{rec}_{\sigma})^{\mathbb{J}}(x, f, n^{\mathbb{J}})).$$

A promising candidate of such $(\text{rec}_{\sigma})^{\mathbb{J}}(x, f) : \mathbb{J}\mathbb{N} \rightarrow \sigma^{\mathbb{J}}$ is $\text{rec}_{\sigma^{\mathbb{J}}}(x, f \circ \eta) : \mathbb{N} \rightarrow \sigma^{\mathbb{J}}$. Hence, we extend κ to $\text{ke}_{\sigma} : (\mathbb{N} \rightarrow \sigma^{\mathbb{J}}) \rightarrow \mathbb{J}\mathbb{N} \rightarrow \sigma^{\mathbb{J}}$ to complete the translation of rec_{σ} .

We adopt the standard technique of logical relations to show that the above translation is sound in the sense that each term of \mathbb{T} is related to its translation². Because the translation is parametrized by a nucleus, we have to assume that the logical relation holds for the nucleus.

² We owe the idea of proving a unified theorem of logical relation to Thomas Powell.

► **Theorem 3** (Fundamental Theorem of Logical Relation). *Let $(\mathbb{J}\mathbb{N}, \eta, \kappa)$ be a nucleus. Given a binary relation $R_{\mathbb{N}} \subseteq \mathbb{N} \times \mathbb{J}\mathbb{N}$ between terms of \mathbb{T} , we extend it to $R_{\rho} \subseteq \rho \times \rho^{\mathbb{J}}$ for arbitrary type ρ of \mathbb{T} by defining*

$$f R_{\sigma \rightarrow \tau} g := \forall x^{\sigma} a^{\sigma^{\mathbb{J}}} (x R_{\sigma} a \rightarrow f x R_{\tau} g a).$$

If $R_{\mathbb{N}}$ satisfies

$$\forall n^{\mathbb{N}} (n R_{\mathbb{N}} \eta n) \quad \text{and} \quad \forall f^{\mathbb{N} \rightarrow \mathbb{N}} g^{\mathbb{N} \rightarrow \mathbb{J}\mathbb{N}} (\forall n^{\mathbb{N}} (f n R_{\mathbb{N}} g n) \rightarrow f R_{\mathbb{N} \rightarrow \mathbb{N}} \kappa g) \quad (\dagger)$$

then $t R_{\rho} t^{\mathbb{J}}$ for any closed term $t : \rho$ of \mathbb{T} .

Proof. We prove a more general statement that

for any term $\Gamma \vdash t : \rho$ of \mathbb{T} , if $\Gamma R \Gamma^{\mathbb{J}}$ then $t R_{\rho} t^{\mathbb{J}}$

where $(x_1 : \sigma_1, \dots, x_n : \sigma_n) R (x_1^{\mathbb{J}} : \sigma_1^{\mathbb{J}}, \dots, x_n^{\mathbb{J}} : \sigma_n^{\mathbb{J}})$ stands for $x_1 R_{\sigma_1} x_1^{\mathbb{J}} \wedge \dots \wedge x_n R_{\sigma_n} x_n^{\mathbb{J}}$, by structural induction over t .

- $t = x$. By the assumption $\Gamma R \Gamma^{\mathbb{J}}$.
- $t = \lambda x.u$. Assume $\Gamma R \Gamma^{\mathbb{J}}$ and $x R_{\sigma} x^{\mathbb{J}}$. We have $u R u^{\mathbb{J}}$ by induction hypothesis.
- $t = uv$. By induction hypothesis we have $u R_{\sigma \rightarrow \tau} u^{\mathbb{J}}$ and $v R_{\sigma} v^{\mathbb{J}}$. Then, by the definition of $R_{\sigma \rightarrow \tau}$, we have $uv R_{\tau} u^{\mathbb{J}} v^{\mathbb{J}}$.
- $t = 0$. By the assumption (\dagger) of η .
- $t = \text{suc}$. By the assumption (\dagger) of η , we have $\text{suc}(n) R_{\mathbb{N}} \eta(\text{suc}(n))$ for all $n : \mathbb{N}$. Then by the assumption (\dagger) of κ , we have $\text{suc} R_{\mathbb{N}} \kappa(\eta \circ \text{suc})$.
- $t = \text{rec}$. We prove $\text{rec} R \text{rec}^{\mathbb{J}}$ with the following claims:
 1. For any type σ of \mathbb{T} , the term ke_{σ} preserves the logical relation in the following sense:

$$\forall f^{\mathbb{N} \rightarrow \sigma} g^{\mathbb{N} \rightarrow \sigma^{\mathbb{J}}} (\forall n^{\mathbb{N}} (f n R_{\sigma} g n) \rightarrow f R_{\mathbb{N} \rightarrow \sigma} \text{ke}_{\sigma}(g)).$$

Proof. By induction on σ . ◀

2. For any x^{σ} and $y^{\sigma^{\mathbb{J}}}$ with $x R y$, and any $f^{\mathbb{N} \rightarrow \sigma \rightarrow \sigma}$ and $g^{\mathbb{J}\mathbb{N} \rightarrow \sigma^{\mathbb{J}} \rightarrow \sigma^{\mathbb{J}}}$ with $f R g$,

$$\forall n^{\mathbb{N}} (\text{rec}_{\sigma}(x, f, n) R_{\sigma} \text{rec}_{\sigma^{\mathbb{J}}}(y, g \circ \eta, n)).$$

Proof. By induction on n . ◀

We get a proof of $\text{rec} R \text{rec}^{\mathbb{J}}$ simply by applying (1) to (2). ◀

► **Remark 4.** The above proof can be carried out in the intuitionistic Heyting arithmetic in finite types HA^{ω} [23], with the theorem formulated as

if HA^{ω} proves (\dagger) then, for each closed term t of \mathbb{T} , HA^{ω} proves $t R t^{\mathbb{J}}$.

So are all the results in Section 3. Hence, the verification system here can be HA^{ω} . We leave it unspecified in the theorem for several reasons. Firstly, we hope to study other properties whose verification may require a stronger system as in [19]. Moreover, what we have proved in the Agda formalization is a version of the theorem for the Agda embedding of System \mathbb{T} , namely that the Agda interpretations of any \mathbb{T} -term and its translation are related. But that is only an implementation choice as explained in the introduction.

3 Applications of the monadic translation

We now apply the above framework to reveal various properties and structures of \mathbb{T} -definable functions including majorizability, (uniform) continuity and bar recursion. Each example consists of an algorithm to construct the desired structure given by the monadic translation, and a correctness proof of the algorithm given by the fundamental theorem of logical relation. For this, one only needs to choose a suitable nucleus that satisfies the logical relation for the target property.

3.1 Majorizability

Our first application is to recover Howard's majorizability proof of System \mathbb{T} [11]. Majorizability plays an important role in models of higher-order calculi and more recently in the proof mining program [14]. Howard's majorizability relation extends the usual ordering \leq on natural numbers to the one \triangleleft_ρ on functionals of arbitrary finite type ρ in the same way as in Theorem 3. Specifically \triangleleft_ρ is defined inductively on ρ as follows:

$$\begin{aligned} n \triangleleft_{\mathbb{N}} m &:= n \leq m \\ f \triangleleft_{\sigma \rightarrow \tau} g &:= \forall x^\sigma y^\tau (x \triangleleft_\sigma y \rightarrow fx \triangleleft_\tau gy). \end{aligned}$$

We say t is *majorized* by u if $t \triangleleft u$, and call u a *majorant* of t . Howard shows that each closed term of \mathbb{T} is majorized by some closed term of \mathbb{T} , which fits perfectly into our framework: Let us take $\mathbb{J}\mathbb{N} = \mathbb{N}$ and define $\eta : \mathbb{N} \rightarrow \mathbb{N}$ and $\kappa : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ by

$$\begin{aligned} \eta(n) &:= n & \kappa(g, 0) &:= g(0) \\ \kappa(g, n+1) &:= \max(\kappa(g, n), g(n+1)). \end{aligned}$$

The max function can be defined in \mathbb{T} using `rec` as shown in Section 1.2, and thus so is κ . Intuitively $\kappa(g, n)$ is the maximum of the values $g0, g1, \dots, gn$. Therefore, it satisfies the following property:

► **Lemma 5.** *For any $g : \mathbb{N} \rightarrow \mathbb{N}$, we have $gm \leq \kappa(g, n)$ whenever $m \leq n$.*

Proof. By induction on n . If $n = 0$, we are done because m has to be 0. If $m \leq n + 1$, we have two cases: (i) If $m = n + 1$, then $g(n + 1) \leq \kappa(g, n + 1)$ by definition. (ii) If $m \leq n$, then $gm \leq \kappa(g, n) \leq \kappa(g, n + 1)$ by induction hypothesis and definition. ◀

► **Corollary 6.** *Each closed term $t : \rho$ of \mathbb{T} is majorized by its translation t^J .*

Proof. We only need to check that the two conditions (†) are fulfilled. The first one holds because the ordering \leq is reflexive. For the second, let us assume $\forall n (fn \leq gn)$ and $n \leq m$. We have $gn \leq \kappa(g, m)$ by Lemma 5, and thus $fn \leq \kappa(g, m)$ by the transitivity of \leq . ◀

We draw the reader's attention to this simple example also because the nucleus defined above does *not* satisfy the monad laws: Here η is the identity function on \mathbb{N} . If the left-identity law $\kappa(f, \eta x) = fx$ holds, then κ has to be the identity function on $\mathbb{N}^{\mathbb{N}}$, which is not the case.

3.2 Lifting to higher-order functionals

In the previous example, we extend a relation on natural numbers to arbitrary finite types and then show that the resulting logical relation holds for all terms of \mathbb{T} . However, if one wants to prove a certain property P of functions $X \rightarrow \mathbb{N}$, the above syntactic method may

not work directly, because the property P may not be captured by the inductively defined logical relation. Our monadic translation can serve a preliminary step to solve the problem by lifting natural numbers to functions $X \rightarrow \mathbb{N}$ so that the desired property P becomes the base case of the logical relation.

Let X be a type of \mathbb{T} . Consider the nucleus $(\mathbb{J}\mathbb{N}, \eta, \kappa)$ with $\mathbb{J}\mathbb{N} = X \rightarrow \mathbb{N}$ and $\eta : \mathbb{N} \rightarrow \mathbb{J}\mathbb{N}$ and $\kappa : (\mathbb{N} \rightarrow \mathbb{J}\mathbb{N}) \rightarrow \mathbb{J}\mathbb{N} \rightarrow \mathbb{J}\mathbb{N}$ defined by

$$\eta(n) := \lambda x. n \quad \kappa(g, f) := \lambda x. g(fx, x).$$

Clearly η maps a natural number n to a constant function with value n . The intuition of $\kappa(g, f) : X \rightarrow \mathbb{N}$ is the following: Given an input x , we have an index fx to get a function $g(fx)$ from the sequence g . Then we apply it to the input x to get the final value.

Given $x : X$, we define a logical relation $R_\rho^x \subseteq \rho \times \rho^J$ inductively as in Theorem 3:

$$\begin{aligned} n R_{\mathbb{N}}^x f &:= n = fx \\ g R_{\sigma \rightarrow \tau}^x h &:= \forall y^\sigma, z^{\sigma^J} (y R_\sigma^x z \rightarrow gy R_\tau^x hz). \end{aligned}$$

Clearly the conditions (\dagger) hold; thus by Theorem 3 we have $t R_\rho^x t^J$ for any closed term t of \mathbb{T} . In particular, for any closed term $f : X \rightarrow \mathbb{N}$ of \mathbb{T} , we have

$$\forall \Omega^{X^J} (x R_X^x \Omega \rightarrow fx = f^J(\Omega, x)).$$

For some type X of \mathbb{T} , we may be able to construct a closed term $\Omega : X^J$ such that $x R_X^x \Omega$ for all $x : X$, by unfolding the statement $x R_X^x \Omega$. For example, if $X = \mathbb{N}^{\mathbb{N}}$, then $x R_X^x \Omega$ is unfolded to $\forall n^{\mathbb{N}}, f^{\mathbb{N}^{\mathbb{N} \rightarrow \mathbb{N}}} (n = fx \rightarrow xn = \Omega(f, x))$; we thus define $\Omega(f, x) := x(fx)$ as $fx = n$ by assumption and then have $x R_X^x \Omega$ by definition. Once we construct such a term $\Omega : X^J$, we have $f = f^J \Omega$ (up to pointwise equality). The term $\Omega : X^J$ which preserves the logical relation in the sense of $x R_X^x \Omega$ for all $x : X$ is known as a *generic element* [6, 7].

Given a property P of functions $X \rightarrow \mathbb{N}$, we define a predicate $Q_\rho \subseteq \rho^J$ on elements of the translated type ρ^J inductively on ρ :

$$\begin{aligned} Q_{\mathbb{N}}(f) &:= P(f) \\ Q_{\sigma \rightarrow \tau}(h) &:= \forall z^{\sigma^J} (Q_\sigma(z) \rightarrow Q_\tau(hz)). \end{aligned}$$

Note that Q is just an instance of the binary relation defined in Theorem 3. Once we prove the conditions (\dagger) for Q , *i.e.*

$$\forall n^{\mathbb{N}} Q_{\mathbb{N}}(\eta n) \quad \text{and} \quad \forall g^{\mathbb{N} \rightarrow X \rightarrow \mathbb{N}} (\forall n^{\mathbb{N}} Q_{\mathbb{N}}(gn) \rightarrow Q_{\mathbb{N} \rightarrow \mathbb{N}}(\kappa g))$$

we have $Q(t^J)$ for any closed term t of \mathbb{T} . If we prove also $Q_X(\Omega)$, then we have $P(f)$ for all closed terms $f : \mathbb{N} \rightarrow X$ of \mathbb{T} because $Q_{\mathbb{N}}(f^J \Omega)$ and $f = f^J \Omega$.

All the remaining examples are about properties of \mathbb{T} -definable functions $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ which can be proved following the above steps. We instead enrich the ‘‘lifting’’ nucleus to reflect the computational content of the properties so that witnesses of the properties can be obtained as terms of \mathbb{T} directly via the translation.

3.3 Continuity

The next applications of our monadic translation are to recover the well-known results that every \mathbb{T} -definable function $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ is pointwise continuous and its restriction to any compact subspace is uniformly continuous [5].

There are various approaches to continuity: Kohlenbach [13] extracts a term from the extensionality proof via the Dialectica interpretation, and then uses the majorant of this term to construct a modulus of uniform continuity. Coquand and Jaber [6] extend type theory with a new constant for a generic element, decorate the operational semantics with forcing information, and then extract continuity information of a functional by applying it to the generic element. Escardó [7] also employs a generic element to compute continuity but in his model of dialogue trees, which is closely related to our syntactic approach as discussed in Remark 8. There are also various sheaf models [9, 10, 26] in which all functionals from the Baire space $\mathbb{N}^{\mathbb{N}}$ are continuous and those from the Cantor space $\mathbf{2}^{\mathbb{N}}$ are uniformly continuous. Powell [19] introduces a monadic translation for some call-by-value functional languages, one of whose instantiations tackles also continuity of T-definable functionals. His method corresponds to Kuroda’s negative translation as discussed in Section 4.3.

We enrich the “lifting” nucleus (Section 3.2) so that moduli of (uniform) continuity are obtained directly from the translation. For the sake of convenience, we extend System T with products. Such extension can be avoided by working with sequences of types and terms as in the literature of functional interpretations such as [15].

3.3.1 Translating products

We extend System T with product type $\sigma \times \tau$ and constants

$$\text{pair} : \sigma \rightarrow \tau \rightarrow \sigma \times \tau \quad \text{pr}_1 : \sigma \times \tau \rightarrow \sigma \quad \text{pr}_2 : \sigma \times \tau \rightarrow \tau$$

satisfying the usual computation rules. Similarly to Gentzen’s translation of conjunction, we translate product type component-wise, *i.e.* $(\sigma \times \tau)^J := \sigma^J \times \tau^J$. Then the above constants are translated into themselves but of the translated types, *e.g.* $\text{pr}_1^J := \text{pr}_1 : \sigma^J \times \tau^J \rightarrow \sigma^J$. Recall that the primitive recursor is translated using $\text{ke}_\sigma : (\mathbb{N} \rightarrow \sigma^J) \rightarrow \mathbb{J}\mathbb{N} \rightarrow \sigma^J$ which is defined inductively on σ . So we have to add the following case

$$\text{ke}_{\sigma \times \tau} := \lambda g^{\mathbb{N} \rightarrow \sigma^J \times \tau^J} a^{\mathbb{J}\mathbb{N}}. \text{pair} (\text{ke}_\sigma(\text{pr}_1 \circ g, a), \text{ke}_\tau(\text{pr}_2 \circ g, a))$$

into the definition of ke in order to complete the translation. For the fundamental theorem of logical relation, when extending a relation $R_{\mathbb{N}} \subseteq \mathbb{N} \times \mathbb{J}\mathbb{N}$ to $R_\rho \subseteq \rho \times \rho^J$, we add the following case for product type

$$u R_{\sigma \times \tau} v := (\text{pr}_1 u R_\sigma \text{pr}_1 v) \wedge (\text{pr}_2 u R_\tau \text{pr}_2 v).$$

and can easily show that the constants of product types are related to their translations. We often write $\langle a, b \rangle$ instead of $\text{pair}(a, b)$ for the sake of readability.

3.3.2 Pointwise continuity

Recall that a function $M : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ is a *modulus of continuity* of $f : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ if

$$\forall \alpha^{\mathbb{N}^{\mathbb{N}}} \beta^{\mathbb{N}^{\mathbb{N}}} (\alpha =_{M\alpha} \beta \rightarrow f\alpha = f\beta)$$

where $\alpha =_m \beta$ stands for $\forall i < m (\alpha i = \beta i)$. Our goal is to find a suitable nucleus J so that we can obtain such a functional M from the J-translation of f and then verify its correctness using the fundamental theorem of logical relation.

Let $\mathbb{J}\mathbb{N} = (\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}) \times (\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N})$. For $w : \mathbb{J}\mathbb{N}$ we write V_w to denote its first component and M_w the second, due to the intuition that M_w is a modulus of continuity of the value component V_w . Then we define $\eta : \mathbb{N} \rightarrow \mathbb{J}\mathbb{N}$ by

$$\eta(n) := \langle \lambda \alpha. n, \lambda \alpha. 0 \rangle$$

and $\kappa : (\mathbb{N} \rightarrow \mathbb{JN}) \rightarrow \mathbb{JN} \rightarrow \mathbb{JN}$ by

$$\kappa(g, w) := \langle \lambda\alpha. V_{g(V_w(\alpha))}(\alpha), \lambda\alpha. \max(M_{g(V_w(\alpha))}(\alpha), M_w(\alpha)) \rangle.$$

Note that the “value” components form a “lifting” nucleus in the sense of Section 3.2 so that natural numbers are lifted to functions $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$. And the “modulus” components will allow the translation to equip a continuity structure to the values. Reasonably $\eta(n)$ equips the constantly zero function as a modulus of continuity to the constant function $\lambda\alpha.n$ since its input is never accessed. As to $\kappa(g, w)$, its value at a point α has two possible moduli: one given by $g(V_w(\alpha))$ and the other by w ; thus the greater one is a modulus of continuity at α .

We work with a logical relation $R_\rho^\alpha \subseteq \rho \times \rho^J$ which is parametrized by $\alpha : \mathbb{N}^{\mathbb{N}}$. Specifically, its base case $R_{\mathbb{N}}^\alpha \subseteq \mathbb{N} \times \mathbb{JN}$ is defined by

$$n R_{\mathbb{N}}^\alpha w := n = V_w(\alpha) \wedge \forall\beta (\alpha =_{M_w(\alpha)} \beta \rightarrow V_w(\alpha) = V_w(\beta)).$$

The first component of $n R_{\mathbb{N}}^\alpha w$ states that the value of w at α is n , while the second explains exactly the intuition of the type \mathbb{JN} , namely that $M_w(\alpha)$ is a modulus of continuity of V_w at α . We leave the proof of (\dagger) to the reader. By Theorem 3, we have $t R_\rho^\alpha t^J$ for any $\alpha : \mathbb{N}^{\mathbb{N}}$ and for any closed term $t : \rho$ of \mathbb{T} . In particular, we have $f R_{\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}}^\alpha f^J$ for every closed term $f : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of \mathbb{T} .

The last step is to construct the generic element $\Omega : \mathbb{JN} \rightarrow \mathbb{JN}$ such that $\alpha R_{\mathbb{N}^{\mathbb{N}}}^\alpha \Omega$ for all $\alpha : \mathbb{N}^{\mathbb{N}}$. Once we unfold $\alpha R_{\mathbb{N}^{\mathbb{N}}}^\alpha \Omega$, we can see that, for any $w : \mathbb{JN}$, the value of $\Omega(w)$ has to be $\lambda\alpha. \alpha(V_w(\alpha))$ as discussed in Section 3.2. Then we also need to construct its modulus of continuity. There are two possible moduli at α : one is $V_w(\alpha) + 1$ because the modulus of continuity of $\lambda\alpha. \alpha n$ at α is $n + 1$, and the other is $M_w(\alpha)$. We just take the greater one and then end up with the following definition:

$$\Omega(w) := \langle \lambda\alpha. \alpha(V_w(\alpha)), \lambda\alpha. \max(V_w(\alpha) + 1, M_w(\alpha)) \rangle.$$

One may have noticed that the above is highly similar to the definition of κ . Indeed, we have $\Omega = \kappa(\lambda n. \langle \lambda\alpha. \alpha n, \lambda\alpha. n + 1 \rangle)$.

► **Theorem 7.** *Every closed term $f : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of \mathbb{T} has a modulus of continuity given by the term $M_{f^J(\Omega)}$.*

Proof. Because $f R_{\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}}^\alpha f^J$ and $\alpha R_{\mathbb{N}^{\mathbb{N}}}^\alpha \Omega$, we have $f\alpha R_{\mathbb{N}}^\alpha f^J(\Omega)$ for any $\alpha : \mathbb{N}^{\mathbb{N}}$, which implies (i) $f = V_{f^J(\Omega)}$ up to pointwise equality, and (ii) $M_{f^J(\Omega)}$ is a modulus of continuity of $V_{f^J(\Omega)}$. Therefore, $M_{f^J(\Omega)}$ is also a modulus of continuity of f . ◀

► **Remark 8.** The above development can be viewed as a syntactic (and simplified) version of Escardó’s approach via dialogue trees [7]. The algorithms to construct moduli of continuity in these two methods are exactly the same. On the other hand, though Powell works also with a monadic translation [19], his algorithm is different because he translates terms in the call-by-value manner. We will look into this in more detail in Section 4.3.

3.3.3 Uniform continuity

The objective here is to, for each closed term $f : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of \mathbb{T} , construct a *modulus of uniform continuity* $M : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, *i.e.*

$$\forall\delta^{\mathbb{N}^{\mathbb{N}}} \alpha^{\mathbb{N}^{\mathbb{N}}} \beta^{\mathbb{N}^{\mathbb{N}}} (\alpha \leq^1 \delta \wedge \beta \leq^1 \delta \wedge \alpha =_{M\delta} \beta \rightarrow f\alpha = f\beta)$$

where $\alpha \leq^1 \beta$ stands for $\forall i(\alpha i \leq \beta i)$. The value $M\delta$ is called a modulus of uniform continuity of f on $\{\alpha : \mathbb{N}^{\mathbb{N}} \mid \alpha \leq^1 \delta\}$. The following fact of uniform continuity plays an important role in the construction:

25:10 A Gentzen-Style Monadic Translation

► **Lemma 9.** *If $f : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ is uniformly continuous on $\{\alpha : \mathbb{N}^{\mathbb{N}} \mid \alpha \leq^1 \delta\}$ with a modulus m , then it has a maximum image on $\{\alpha : \mathbb{N}^{\mathbb{N}} \mid \alpha \leq^1 \delta\}$.*

Proof. We compute the maximum image $\Theta(m, f, \delta)$ by induction on the modulus m :

$$\begin{aligned}\Theta(0, f, \delta) &:= f\delta \\ \Theta(m+1, f, \delta) &:= \Phi(\lambda i. \Theta(m, \lambda \alpha. f(i * \alpha), \delta \circ \text{succ}), \delta 0)\end{aligned}$$

where $i * \alpha$ is an infinite sequence with head i and tail α , and $\Phi : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ defined by

$$\begin{aligned}\Phi(\alpha, 0) &:= \alpha 0 \\ \Phi(\alpha, n+1) &:= \max(\Phi(\alpha, n), \alpha(n+1))\end{aligned}$$

i.e. $\Phi(\alpha, n)$ is the greatest αi for $i \leq n$ (note that Φ is the κ of the “majorizability” nucleus introduced in Section 3.1). In the base case of Θ , the modulus is 0 and thus f is constant with the value $f\delta$. To compute $\Theta(m+1, f, \delta)$, by induction hypothesis we have for each $i : \mathbb{N}$ the maximum image $\Theta(m, \lambda \alpha. f(i * \alpha), \delta \circ \text{succ})$ of the function $\lambda \alpha. f(i * \alpha)$ with a modulus m on the inputs bounded by $\delta \circ \text{succ}$. Because the inputs of f are bounded by δ , the greatest $\Theta(m, \lambda \alpha. f(i * \alpha), \delta \circ \text{succ})$ for $i < \delta 0$ is the maximum image of f , and we use Φ to find it. Note that both Φ and Θ can be defined in \mathbb{T} using rec . ◀

We are now ready to construct the nucleus. Let $\mathbb{JN} = (\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}) \times (\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N})$. The idea is exactly the same as in the previous treatment to pointwise continuity: For any $w : \mathbb{JN}$, its second component M_w is (expected to be) a modulus of uniform continuity of the first component V_w . Then we define $\eta : \mathbb{N} \rightarrow \mathbb{JN}$ by

$$\eta(n) := \langle \lambda \alpha. n, \lambda \alpha. 0 \rangle$$

and $\kappa : (\mathbb{N} \rightarrow \mathbb{JN}) \rightarrow \mathbb{JN} \rightarrow \mathbb{JN}$ by

$$\kappa(g, w) := \langle \lambda \alpha. V_{g(V_w(\alpha))}(\alpha), \lambda \delta. \max(\Phi(\lambda i. M_{g_i}(\delta), \Theta(M_w(\delta), V_w, \delta)), M_w(\delta)) \rangle$$

where Φ and Θ are defined in the proof of Lemma 9. Specifically, we construct a modulus of uniform continuity for the value of $\kappa(g, w)$ as follows: Given $\delta : \mathbb{N}^{\mathbb{N}}$, we have two possible moduli given by those of g and w at δ and thus we choose the larger one. Actually the only complication comes from the calculation of the modulus given by g . For each $i : \mathbb{N}$ we have a modulus $M_{g_i}(\delta)$. In the value of $\kappa(g, w)$, we apply g to $V_w(\alpha)$ for input α . Because V_w has a maximum image computed using Θ , we only need to find the greatest modulus $M_{g_i}(\delta)$ for i not greater than the maximum image of V_w , and we use Φ for this purpose.

Given $\delta : \mathbb{N}^{\mathbb{N}}$, the base case $R_{\mathbb{N}}^{\delta} \subseteq \mathbb{N} \times \mathbb{JN}$ of the logical relation is defined by

$$n R_{\mathbb{N}}^{\delta} w := n = V_w(\delta) \wedge \forall \alpha, \beta (\alpha \leq^1 \delta \wedge \beta \leq^1 \delta \wedge \alpha =_{M_w(\delta)} \beta \rightarrow V_w(\alpha) = V_w(\beta)).$$

In words, $n R_{\mathbb{N}}^{\delta} w$ means that n is the value of w at δ and that $M_w(\delta)$ is a modulus of uniform continuity of V_w on $\{\alpha : \mathbb{N}^{\mathbb{N}} \mid \alpha \leq^1 \delta\}$. It is routine to check that both η and κ preserve the logical relation in the sense of (\dagger) , which we again leave to the reader. By Theorem 3 we have $t R_{\rho}^{\delta} t^{\mathbb{J}}$ for any $\delta : \mathbb{N}^{\mathbb{N}}$ and for any closed term $t : \rho$ of \mathbb{T} . Moreover, the generic element $\Omega : \mathbb{JN} \rightarrow \mathbb{JN}$ defined by

$$\Omega := \kappa(\lambda n. \langle \lambda \alpha. \alpha n, \lambda \alpha. n+1 \rangle)$$

also preserves the logical relation in the sense of $\delta R_{\mathbb{N}^{\mathbb{N}}}^{\delta} \Omega$ for all $\delta : \mathbb{N}^{\mathbb{N}}$.

With a proof similar to the one of Theorem 7, we get the following result:

► **Theorem 10.** *Every closed term $f : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of \mathbb{T} has a modulus of uniform continuity given by the term $M_{f^{\mathbb{J}}}(\Omega)$.*

3.4 General bar recursion

To prove Schwichtenberg's theorem [20] that the System T definable functionals are closed under a rule-like version Spector's bar recursion of type levels 0 and 1, Oliva and Steila [17] introduce a notion of general bar recursion whose termination condition is given by decidable monotone predicates on finite sequences. As the last example, we recover their construction of general-bar-recursion functionals ([17, Definitions 3.1 & 3.3]) via an instantiation of our translation. For this, we need the following notations:

- We represent decidable predicates as functions $\mathbb{N}^* \rightarrow \mathbf{2}$, where \mathbb{N}^* is the type of finite sequences of natural numbers and $\mathbf{2} = \{0, 1\}$ is the type of booleans.
- For any $S : \mathbb{N}^* \rightarrow \mathbf{2}$ and $s : \mathbb{N}^*$, we write $S(s)$ instead of $S(s) = 1$.
- For any $s : \mathbb{N}^*$, we write $|s| : \mathbb{N}$ to denote its length and $\hat{s} : \mathbb{N}^{\mathbb{N}}$ the extension of s with infinitely many 0's.
- For any $s : \mathbb{N}^*$ and $n : \mathbb{N}$, we write $s * n : \mathbb{N}^*$ to denote appending n to s .
- For any $s : \mathbb{N}^*$ and $\alpha : \mathbb{N}^{\mathbb{N}}$, we write $s * \alpha : \mathbb{N}^{\mathbb{N}}$ to denote their concatenation.

Note that the treatment of \mathbb{N}^* and $\mathbf{2}$ is not essential. For instance, we can represent a finite sequence s by a pair $\langle \alpha, n \rangle$ and consider s as the prefix of the infinite sequence α of length n as in our Agda implementation. All the above operations on sequences are definable in T. We also need the following definitions:

- We call $\xi : (\mathbb{N}^* \rightarrow \sigma) \rightarrow (\mathbb{N}^* \rightarrow \sigma^{\mathbb{N}} \rightarrow \sigma) \rightarrow \mathbb{N}^* \rightarrow \sigma$ a functional of *general bar recursion* for $S : \mathbb{N}^* \rightarrow \mathbf{2}$ if $\mathcal{GBR}_S(\xi)$ holds where $\mathcal{GBR}_S(\xi)$ is defined by

$$\mathcal{GBR}_S(\xi) := \forall G^{\mathbb{N}^* \rightarrow \sigma} H^{\mathbb{N}^* \rightarrow \sigma^{\mathbb{N}} \rightarrow \sigma} S^{\mathbb{N}^*} \left\{ \begin{array}{l} S(s) \rightarrow \xi(G, H, s) = G(s) \\ \wedge \\ \neg S(s) \rightarrow \xi(G, H, s) = H(s, \lambda n^{\mathbb{N}}. \xi(G, H, s * n)) \end{array} \right\}.$$

- A predicate S is *monotone* if $S(s)$ implies $S(s * n)$ for all $s : \mathbb{N}^*$ and $n : \mathbb{N}$.
- For $Y : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, we say S *secures* Y if

$$\forall s^{\mathbb{N}^*} \left(S(s) \rightarrow \forall \alpha^{\mathbb{N}^{\mathbb{N}}} Y(s * \alpha) = Y(\hat{s}) \right).$$

Let $Y : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ be a closed term of T. Oliva and Steila show (i) for any S securing Y , from a functional of general bar recursion for S we can construct a functional of Spector's bar recursion for Y [17, Theorem 2.4], and (ii) we can construct a monotone predicate S that secures Y and a functional of general bar recursion for S [17, Theorem 3.4]. In this way, they give a new proof of Schwichtenberg's bar recursion closure theorem with an explicit construction of Spector's bar-recursion functionals.

We firstly construct a nucleus for general bar recursion. Fix a type σ of T. Let $\mathbb{JN} = (\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}) \times (\mathbb{N}^* \rightarrow \mathbf{2}) \times ((\mathbb{N}^* \rightarrow \sigma) \rightarrow (\mathbb{N}^* \rightarrow \sigma^{\mathbb{N}} \rightarrow \sigma) \rightarrow \mathbb{N}^* \rightarrow \sigma)$. Given $w : \mathbb{JN}$, we write V_w, S_w, B_w to denote its three components. The intuition is that S_w is a monotone predicate securing V_w and B_w is a functional of general bar recursion for S_w . We define $\eta : \mathbb{N} \rightarrow \mathbb{JN}$ by

$$\eta(n) := \langle \lambda \alpha. n, \lambda s. 1, \lambda GH. G \rangle$$

and $\kappa : (\mathbb{N} \rightarrow \mathbb{JN}) \rightarrow \mathbb{JN} \rightarrow \mathbb{JN}$ by

$$\kappa(g, w) := \langle \lambda \alpha. V_{g(V_w \alpha)} \alpha, \lambda s. \min(S_w(s), S_{g(V_w \hat{s})}(s)), \lambda GH. B_w(\lambda s. B_{g(V_w \hat{s})}(G, H, s), H) \rangle$$

where $\min : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ returns the smaller argument. Lastly, we define the generic element $\Omega : \mathbb{JN} \rightarrow \mathbb{JN}$ by

$$\Omega := \kappa(\lambda n. \langle \lambda \alpha. \alpha n, \lambda s. \text{Le}(n, |s|), \Psi n \rangle)$$

25:12 A Gentzen-Style Monadic Translation

where $\text{Le} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbf{2}$ has value 1 iff its first argument is strictly smaller than the second, and $\Psi n : (\mathbb{N}^* \rightarrow \sigma) \rightarrow (\mathbb{N}^* \rightarrow \sigma^{\mathbb{N}} \rightarrow \sigma) \rightarrow \mathbb{N}^* \rightarrow \sigma$ is a T-definable functional of bar recursion for the constant function $Y = \lambda \alpha. n$ [17, Lemma 2.1], *i.e.*

$$\forall G^{\mathbb{N}^* \rightarrow \sigma} H^{\mathbb{N}^* \rightarrow \sigma^{\mathbb{N}} \rightarrow \sigma} s^{\mathbb{N}^*} \left\{ \begin{array}{l} n < |s| \rightarrow \Psi n(G, H, s) = G(s) \\ \wedge \\ n \geq |s| \rightarrow \Psi n(G, H, s) = H(s, \lambda m. \Psi n(G, H, s * m)) \end{array} \right\}.$$

For any $n : \mathbb{N}$, it is clear that $\lambda s. \text{Le}(n, |s|)$ is a monotone predicate that secures $\lambda \alpha. \alpha n$, and that Ψn is a functional of general bar recursion for $\lambda s. \text{Le}(n, |s|)$.

► **Theorem 11.** *For any closed term $Y : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of \mathbb{T} ,*

1. $S_{Y \wr \Omega}$ is a monotone predicate securing Y , and
2. $B_{Y \wr \Omega}$ is a functional of general bar recursion of $S_{Y \wr \Omega}$.

Proof. Given $\alpha : \mathbb{N}^{\mathbb{N}}$, we define the base case $R_{\mathbb{N}}^{\alpha} \subseteq \mathbb{N} \times \mathbb{J}\mathbb{N}$ of the logical relation by

$$n R_{\mathbb{N}}^{\alpha} w := n = V_w(\alpha) \wedge S_w \text{ is monotone} \wedge S_w \text{ secures } V_w \wedge \mathcal{GBR}_{S_w}(B_w).$$

To apply the fundamental theorem of logical relation, we need to check the conditions (†):

- It is trivial to prove $n R_{\mathbb{N}}^{\alpha} \eta n$ for all $n : \mathbb{N}$.
- As to κ , given $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{J}\mathbb{N}$ such that $f i R_{\mathbb{N}}^{\alpha} g i$ for all $i : \mathbb{N}$, our goal is to prove $f R_{\mathbb{N} \rightarrow \mathbb{N}}^{\alpha} \kappa g$. Let $n : \mathbb{N}$ and $w : \mathbb{J}\mathbb{N}$ with $n R_{\mathbb{N}}^{\alpha} w$ be given.
 - We have $f n = V_{g n}(\alpha) = V_{g(V_w(\alpha))}(\alpha) = V_{\kappa(g,w)}(\alpha)$ as in the previous examples.
 - Because S_w is monotone and so is $S_{g i}$ for all $i : \mathbb{N}$ by assumption, the predicate $S_{\kappa(g,w)}$ is also monotone.
 - If $S_{\kappa(g,w)}(s)$, then $S_w(s)$ and $S_{g(V_w \hat{s})}(s)$ by definition. Given $\alpha : \mathbb{N}^{\mathbb{N}}$, we have $V_w(s * \alpha) = V_w(\hat{s})$ because S_w secures V_w . Then we have $V_{\kappa(g,w)}(s * \alpha) = V_{g(V_w(s * \alpha))}(s * \alpha) = V_{g(V_w \hat{s})}(s * \alpha) = V_{g(V_w \hat{s})}(\hat{s}) = V_{\kappa(g,w)}(\hat{s})$ because $S_{g(V_w \hat{s})}$ secures $V_{g(V_w \hat{s})}$. Hence $S_{\kappa(g,w)}$ secures $V_{\kappa(g,w)}$.
 - Lastly we show that $B_{\kappa(g,w)}$ is a functional of general bar recursion for $S_{\kappa(g,w)}$. Let $G : \mathbb{N}^* \rightarrow \sigma$, $H : \mathbb{N}^* \rightarrow \sigma^{\mathbb{N}} \rightarrow \sigma$ and $s : \mathbb{N}^*$ be given. (1) If $S_{\kappa(g,w)}(s)$, then $S_w(s)$ and $S_{g(V_w \hat{s})}(s)$, and thus we have $B_{\kappa(g,w)}(G, H, s) = G(s)$. (2) If $\neg S_{\kappa(g,w)}(s)$, then we have two cases to check: (2.1) If $S_w(s)$, then $\neg S_{g(V_w \hat{s})}(s)$ by definition. It is not hard to show $B_{\kappa(g,w)}(G, H, s) = H(s, \lambda n. B_{\kappa(g,w)}(G, H, s * n))$. (2.2) If $\neg(S_w(s))$, then we always have $B_{\kappa(g,w)}(G, H, s) = H(s, \lambda n. B_{\kappa(g,w)}(G, H, s * n))$ no matter if $S_{g(V_w \hat{s})}(s)$ holds or not.

Hence, we have $f n R_{\mathbb{N}}^{\alpha} \kappa(g, w)$.

Given a closed term $Y : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of \mathbb{T} , for any $\alpha : \mathbb{N}^{\mathbb{N}}$ we have $Y R_{\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}}^{\alpha} Y^{\mathbb{J}}$ by Theorem 3. For any $n : \mathbb{N}$, we have $n R_{\mathbb{N}}^{\alpha} \langle \lambda \alpha. \alpha n, \lambda s. \text{Le}(n, |s|), \Psi n \rangle$ by definition; thus, $\alpha R_{\mathbb{N} \rightarrow \mathbb{N}}^{\alpha} \Omega$ holds by (†) for κ which we have just proved. Hence we have $Y \alpha R_{\mathbb{N}}^{\alpha} Y^{\mathbb{J}} \Omega$ for any $\alpha : \mathbb{N}^{\mathbb{N}}$. From this, we get (i) $Y = V_{Y \wr \Omega}$ up to pointwise equality, (ii) $S_{Y \wr \Omega}$ is a monotone predicate securing $V_{Y \wr \Omega}$ and thus also Y , and (iii) $B_{Y \wr \Omega}$ is a functional of general bar recursion for $S_{Y \wr \Omega}$. ◀

The above development is just a restructuring of the work of Oliva and Steila [17] that fits into our framework. But there are some small differences (or simplifications):

- [17] requires the predicate S to satisfy the bar condition $\forall \alpha^{\mathbb{N}^{\mathbb{N}}} \exists n^{\mathbb{N}} S(\bar{\alpha} n)$, where $\bar{\alpha} n : \mathbb{N}^*$ is the prefix of α of length n . As pointed out by Makoto Fujiwara in a personal discussion, this condition is not needed for the result. So we remove it in the above development.

- [17] assumes that the closed terms $Y : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ are of the form $\lambda\alpha.t$ where $\alpha : \mathbb{N}^{\mathbb{N}}$ is the only free variable in $t : \mathbb{N}$, and treats α as a special constant (for the generic element). Motivated by a version of Escardó's Agda development of [7], we avoid such extension by using the lifting nucleus that is introduced in Section 3.2.
- The syntactic translation of \mathbb{T} in [17, Definitions 3.1 & 3.3] contains only the construction of general-bar-recursion functionals while the one of monotone securing predicates is given in the proof of the main theorem [17, Theorem 3.4]. We combine both in the translation in order to split the constructions from the correctness proof.

As pointed out by an anonymous reviewer, this section unifies the results in Section 3.3 in the sense that moduli of (uniform) continuity can be defined from monotone securing bars and general-bar-recursion functionals: Let $Y : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ be a closed term of \mathbb{T} . The monotone predicate $S_{Y\downarrow\Omega}$ is a bar, *i.e.* $\forall\alpha^{\mathbb{N}^{\mathbb{N}}}\exists n^{\mathbb{N}}S_{Y\downarrow\Omega}(\bar{\alpha}n)$, as shown in [17, Theorem 3.4]. This together with the fact that it secures Y implies the pointwise continuity of Y . The witness of the fact that $S_{Y\downarrow\Omega}$ is a bar obtained via *e.g.* modified realizability [14] is a continuity modulus of Y . Our translation in Section 3.3 is just an explicit procedure to get these witnesses that are blurred in the proof of [17, Theorem 3.4]. On the other hand, the reviewer points out that we can construct a modulus $M : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ of uniform continuity of Y by

$$M(\delta) := B_{Y\downarrow\Omega}(G, H^\delta, \text{nil})$$

where $G(s) := 0$, $H^\delta(s, f) := 1 + \max\{fn \mid n \leq \delta(|s|)\}$ and $\text{nil} : \mathbb{N}^*$ is the empty sequence. The idea is that if s is a prefix of δ then $B_{Y\downarrow\Omega}(G, H^\delta, s)$ is a modulus of uniform continuity of the function $\lambda\alpha.Y(s * \alpha)$ on $\{\alpha : \mathbb{N}^{\mathbb{N}} \mid \alpha \leq^1 \delta\}$: If $S_{Y\downarrow\Omega}(s)$, we know $\lambda\alpha.Y(s * \alpha)$ is a constant function because $S_{Y\downarrow\Omega}$ secures Y . Then $B_{Y\downarrow\Omega}(G, H^\delta, s) = G(s) = 0$ is a proper modulus. If $\neg S_{Y\downarrow\Omega}(s)$, then the step functional H^δ finds the greatest value of the moduli of $\lambda\alpha.Y(s * n * \alpha)$ given by $B_{Y\downarrow\Omega}(G, H^\delta, s * n)$ for $n \leq \delta(|s|)$ and then adds 1 to get a modulus of $\lambda\alpha.Y(s * \alpha)$.

4 Generalization and variants of the monadic translation

We have developed a self-translation of System \mathbb{T} in the spirit of Gentzen's negative translation. We conclude the paper by generalizing it to translate sums and comparing it with another two monadic translations.

4.1 Translating sums

We generalize our Gentzen-style translation to translate also sums. Let us extend \mathbb{T} with sum type $\sigma + \tau$ and the following constants

$$\text{inj}_1 : \sigma \rightarrow \sigma + \tau \quad \text{inj}_2 : \tau \rightarrow \sigma + \tau \quad \text{case} : (\sigma \rightarrow \rho) \rightarrow (\tau \rightarrow \rho) \rightarrow \sigma + \tau \rightarrow \rho.$$

In his translation, Gentzen places a double negation in front of disjunctions (see Section 1.1). Following this inspiration, the sum type $\sigma + \tau$ should be translated into $J(\sigma^J + \tau^J)$. But the simple notion of a nucleus given by a type and two terms does not suffice. We have to work with the following more general notion: A nucleus (J, η, κ) consists of an endofunction J on types of (the extension of) \mathbb{T} , and terms

$$\eta : \sigma \rightarrow J\sigma \quad \kappa : (\sigma \rightarrow J\tau) \rightarrow J\sigma \rightarrow J\tau$$

25:14 A Gentzen-Style Monadic Translation

for any types σ, τ of (the extension of) \mathbb{T} . Then we add the following clauses to Definition 2 to complete the translation:

$$\begin{aligned} (\sigma + \tau)^J &:= J(\sigma^J + \tau^J) & \text{inj}_i^J &:= \eta \circ \text{inj}_i \quad \text{for } i \in \{1, 2\} \\ \text{ke}_{\sigma+\tau} &:= \kappa & \text{case}^J &:= \lambda f g. \text{ke}(\text{case}(f, g)). \end{aligned}$$

Generalizing the fundamental theorem of logical relation to cover sums is remained as a future task. Both natural numbers and sums, as instances of inductive types, are translated in a very similar way. Another future task is to generalize the translation to cover all (strictly positive) inductive and coinductive types.

4.2 The Kolmogorov-style monadic translation

Barthe and Uustalu’s call-by-name continuation passing style transformation [4] corresponds to Kolmogorov’s negative translation. By replacing the continuation monad with a nucleus, one obtains a Kolmogorov-style monadic translation which is studied in [24]. Kolmogorov places a double negation in front of every subformula. Similarly we place the nucleus in front of every subtype. Hence we have to work with the more general notion of a nucleus (J, η, κ) where J is an endofunction on types. Specifically, each type ρ of \mathbb{T} is translated to $J\langle\rho\rangle$ where $\langle\rho\rangle$ is defined inductively as follows

$$\begin{aligned} \langle\mathbb{N}\rangle &:= \mathbb{N} \\ \langle\sigma \square \tau\rangle &:= J\langle\sigma\rangle \square J\langle\tau\rangle \quad \text{for } \square \in \{\rightarrow, \times, +\}. \end{aligned}$$

Each term $t : \rho$ is translated to a term $\langle t \rangle : J\langle\rho\rangle$. In order to translate function application, we have to consider a monadic notion of application. Given $f : J(\sigma \rightarrow \mathcal{J}\tau)$ and $a : \sigma$, we “apply” f to a as

$$f \diamond a := \kappa(\lambda g^{\sigma \rightarrow \mathcal{J}\tau}. ga, f).$$

Then for any terms $t : \sigma \rightarrow \tau$ and $u : \sigma$, we define $\langle tu \rangle := \langle t \rangle \diamond \langle u \rangle$. The rest of the translation can be found in the appendix.

4.3 The Kuroda-style monadic translation

There is also a Kuroda-style monadic translation of System \mathbb{T} which has been studied by Powell in [18, 19], where each type ρ is translated to $J[\rho]$ with $[\rho]$ defined by

$$\begin{aligned} [\mathbb{N}] &:= \mathbb{N} & [\sigma \times \tau] &:= [\sigma] \times [\tau] \\ [\sigma \rightarrow \tau] &:= [\sigma] \rightarrow J[\tau] & [\sigma + \tau] &:= [\sigma] + [\tau]. \end{aligned}$$

Note that it actually corresponds to the variant of Kuroda’s negative translation where double negations are placed also in front of conclusions of implications (see [16, Section 6]). Here we need another notion of application for the term translation: Given $f : J(\sigma \rightarrow \mathcal{J}\tau)$ and $a : J\sigma$, we “apply” f to a as

$$f \bullet a := \kappa(\lambda g^{\sigma \rightarrow \mathcal{J}\tau}. \kappa(g, a), f).$$

The complete translation can be found in the appendix.

Powell makes use of the Kuroda-style translation to extract moduli of continuity [19, Section 5], similarly to our development in Section 3.3. However, our algorithms are different because the Kuroda-style translation is call-by-value while our Gentzen-style one is call-by-name. Consider the following example. Let $t = \lambda \alpha. \text{rec}(\alpha 0, \lambda nm. 0, 1) : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ which is a

constant function. If we apply the Kuroda-style translation to t with the continuity nucleus (similar to the one given in Section 3.3 but generalized to arbitrary types of T), then we get a modulus $\lambda\alpha.1$, because in the call-by-value strategy all the inputs of rec including $\alpha 0$ are evaluated. With the Gentzen-style translation, we get $\lambda\alpha.0$ because $\alpha 0$ is never evaluated in the call-by-name strategy.

References

- 1 Peter Aczel. The Russell–Prawitz modality. *Mathematical Structures in Computer Science*, 11(4):541–554, 2001. doi:10.1017/S0960129501003309.
- 2 Agda community. The Agda Wiki. URL: <https://wiki.portal.chalmers.se/agda/pmwiki.php>.
- 3 Rob Arthan and Paulo Oliva. On affine logic and łukasiewicz logic, 2014. arXiv:1404.0570 [cs.LG].
- 4 Gilles Barthe and Tarmo Uustalu. CPS translating inductive and coinductive types. In *2002 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'02)*, volume 37 of *SIGPLAN Notices*, pages 131–142. ACM Press, New York, 2002. doi:10.1145/503032.503043.
- 5 Michael J. Beeson. *Foundations of Constructive Mathematics*. Springer, 1985. doi:10.1007/978-3-642-68952-9.
- 6 Thierry Coquand and Guilhem Jaber. A note on forcing and type theory. *Fundamenta Informaticae*, 100(1-4):43–52, 2010. doi:10.3233/FI-2010-262.
- 7 Martín H. Escardó. Continuity of Gödel’s system T functionals via effectful forcing. In *Proceedings of the Twenty-ninth Conference on the Mathematical Foundations of Programming Semantics (MFPS'2013)*, volume 298 of *Electronic Notes in Theoretical Computer Science*, pages 119–141. Elsevier, 2013. doi:10.1016/j.entcs.2013.09.010.
- 8 Martín H. Escardó and Paulo Oliva. The Peirce translation. *Annals of Pure and Applied Logic*, 163(6):681–692, 2012. doi:10.1016/j.apal.2011.11.002.
- 9 Martín H. Escardó and Chuangjie Xu. A constructive manifestation of the Kleene–Kreisel continuous functionals. *Annals of Pure and Applied Logic*, 167(9):770–793, 2016. Fourth Workshop on Formal Topology (4WFTop). doi:10.1016/j.apal.2016.04.011.
- 10 Michael P. Fourman. Notions of choice sequence. In *The L. E. J. Brouwer Centenary Symposium*, volume 110 of *Studies in Logic and the Foundations of Mathematics*, pages 91–105. Elsevier, 1982. doi:10.1016/S0049-237X(09)70125-9.
- 11 William A. Howard. Hereditarily majorizable functionals of finite type. In *Metamathematical investigation of intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*, pages 454–461. Springer, Berlin, Heidelberg, 1973. doi:10.1007/BFb0066739.
- 12 Hajime Ishihara. A note on the Gödel–Gentzen translation. *Mathematical Logic Quarterly*, 46(1):135–137, 2000. doi:10.1002/(SICI)1521-3870(200001)46:1<135::AID-MALQ135>3.0.CO;2-R.
- 13 Ulrich Kohlenbach. Pointwise hereditary majorization and some applications. *Archive for Mathematical Logic*, 31(4):227–241, 1992. doi:10.1007/BF01794980.
- 14 Ulrich Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Springer Monographs in Mathematics. Springer, 2008. doi:10.1007/978-3-540-77533-1.
- 15 Paulo Oliva. Unifying functional interpretations. *Notre Dame Journal of Formal Logic*, 47(2):263–290, 2006. doi:10.1305/ndjfl/1153858651.
- 16 Paulo Oliva and Gilda Ferreira. On the relation between various negative translations. In *Logic, Construction, Computation*, volume 3 of *Mathematical Logic Series*, pages 227–258. Ontos-Verlag, 2012. doi:10.1515/9783110324921.
- 17 Paulo Oliva and Silvia Steila. A direct proof of Schwichtenberg’s bar recursion closure theorem. *The Journal of Symbolic Logic*, 83(1):70–83, 2018. doi:10.1017/jsl.2017.33.

- 18 Thomas Powell. A functional interpretation with state. In *Proceedings of the Thirty third Annual IEEE Symposium on Logic in Computer Science (LICS 2018)*, pages 839–848. IEEE Computer Society Press, July 2018. doi:10.1145/3209108.3209134.
- 19 Thomas Powell. A unifying framework for continuity and complexity in higher types, 2019. arXiv:1906.10719 [cs.LG].
- 20 Helmut Schwichtenberg. On bar recursion of types 0 and 1. *The Journal of Symbolic Logic*, 44(3):325–329, 1979. doi:10.2307/2273126.
- 21 Richard Statman. Logical relations and the typed lambda-calculus. *Information and Control*, 65(2/3):85–97, 1985. doi:10.1016/S0019-9958(85)80001-2.
- 22 Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2nd edition, 2000.
- 23 Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in mathematics, Vol. II*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988.
- 24 Tarmo Uustalu. Monad translating inductive and coinductive types. In *Types for Proofs and Programs (TYPES 2002)*, volume 2646 of *Lecture Notes in Computer Science*, pages 299–315. Springer, 2002. doi:10.1007/3-540-39185-1_17.
- 25 Benno van den Berg. A Kuroda-style j-translation. *Archive for Mathematical Logic*, 58(5-6):627–634, 2019. doi:10.1007/s00153-018-0656-x.
- 26 Gerrit van der Hoeven and Ieke Moerdijk. Sheaf models for choice sequences. *Annals of Pure and Applied Logic*, 27(1):63–107, 1984. doi:10.1016/0168-0072(84)90035-6.
- 27 Chuangjie Xu. A syntactic approach to continuity of T-definable functionals. *Logical Methods in Computer Science*, 16(1):22:1–22:11, 2020. URL: <https://lmcs.episciences.org/6130>.

A The Kolmogorov- and Kuroda-style monadic translations of T

We present the Kolmogorov- and Kuroda-style monadic translations of System T extended with products and sums, parametrized by a nucleus (J, η, κ) , where J is an endofunction on types of T and $\eta : \sigma \rightarrow J\sigma$ and $\kappa : (\sigma \rightarrow J\tau) \rightarrow J\sigma \rightarrow J\tau$ are terms of T. Recall the following notions of monadic application that will be needed in the translations:

- Given $f : J(\sigma \rightarrow J\tau)$ and $a : \sigma$, we define

$$f \diamond a := \kappa(\lambda g^{\sigma \rightarrow J\tau}. ga, f).$$

- Given $f : J(\sigma \rightarrow J\tau)$ and $a : J\sigma$, we define

$$f \bullet a := \kappa(\lambda g^{\sigma \rightarrow J\tau}. \kappa(g, a), f).$$

► **Definition 12** (Kolmogorov-style monadic translation). *We assign to each type ρ a type $J\langle\rho\rangle$ where $\langle\rho\rangle$ is defined as follows*

$$\begin{aligned} \langle\mathbb{N}\rangle &:= \mathbb{N} \\ \langle\sigma \square \tau\rangle &:= J\langle\sigma\rangle \square J\langle\tau\rangle \quad \text{for } \square \in \{\rightarrow, \times, +\}. \end{aligned}$$

Each term $\Gamma \vdash t : \rho$ is translated to a term $\langle\Gamma\rangle \vdash \langle t \rangle : J\langle\rho\rangle$, where $\langle\Gamma\rangle$ is a new context assigning each $x : \sigma \in \Gamma$ to a fresh variable $\hat{x} : J\langle\sigma\rangle$, and $\langle t \rangle$ is defined inductively as follows:

$$\begin{aligned} \langle x \rangle &:= \hat{x} & \langle 0 \rangle &:= \eta(0) \\ \langle \lambda x.t \rangle &:= \eta(\lambda \hat{x}. \langle t \rangle) & \langle \text{succ} \rangle &:= \eta(\kappa(\eta \circ \text{succ})) \\ \langle tu \rangle &:= \langle t \rangle \diamond \langle u \rangle & \langle \text{rec} \rangle &:= \eta(\kappa(\lambda a. \eta(\kappa(\lambda f. \eta(\kappa(\text{rec}^\diamond(a, f))))))) \\ \langle \text{pair} \rangle &:= \eta(\lambda a. \eta(\lambda b. \eta(\text{pair}(a, b)))) & \langle \text{inj}_i \rangle &:= \eta(\eta \circ \text{inj}_i) \\ \langle \text{pr}_i \rangle &:= \eta(\kappa(\text{pr}_i)) & \langle \text{case} \rangle &:= \eta(\kappa(\lambda f. \eta(\kappa(\lambda g. \eta(\kappa(\text{case}(f, g))))))) \end{aligned}$$

where $\text{rec}^\diamond : \sigma \rightarrow (\mathbb{N} \rightarrow \mathbb{J}(\mathbb{J}\sigma \rightarrow \mathbb{J}\sigma)) \rightarrow \mathbb{N} \rightarrow \mathbb{J}\sigma$ is defined by

$$\begin{aligned}\text{rec}^\diamond(a, f, 0) &:= \eta(a) \\ \text{rec}^\diamond(a, f, n+1) &:= f(\eta n) \diamond \text{rec}^\diamond(a, f, n).\end{aligned}$$

► **Definition 13** (Kuroda-style monadic translation). We assign to each type ρ a type $\mathbb{J}[\rho]$ where $[\rho]$ is defined as follows

$$\begin{aligned}[\mathbb{N}] &:= \mathbb{N} & [\sigma \times \tau] &:= [\sigma] \times [\tau] \\ [\sigma \rightarrow \tau] &:= [\sigma] \rightarrow \mathbb{J}[\tau] & [\sigma + \tau] &:= [\sigma] + [\tau].\end{aligned}$$

Each term $\Gamma \vdash t : \rho$ is translated to a term $[\Gamma] \vdash [t] : \mathbb{J}[\rho]$, where $[\Gamma]$ is a new context assigning each $x : \sigma \in \Gamma$ to a fresh variable $\bar{x} : [\sigma]$, and $[t]$ is defined inductively as follows:

$$\begin{aligned}[x] &:= \eta(\bar{x}) & [0] &:= \eta(0) \\ [\lambda x.t] &:= \eta(\lambda \bar{x}.[t]) & [\text{suc}] &:= \eta(\eta \circ \text{suc}) \\ [tu] &:= [t] \bullet [u] & [\text{rec}] &:= \eta(\lambda a.\eta(\lambda f.\eta(\text{rec}^\bullet(a, f)))) \\ [\text{pair}] &:= \eta(\lambda \alpha.\eta(\lambda b.\eta(\text{pair}(a, b)))) & [\text{inj}_i] &:= \eta(\eta \circ \text{inj}_i) \\ [\text{pr}_i] &:= \eta(\eta \circ \text{pr}_i) & [\text{case}] &:= \eta(\lambda f.\eta(\lambda g.\eta(\text{case}(f, g))))\end{aligned}$$

where $\text{rec}^\bullet : \sigma \rightarrow (\mathbb{N} \rightarrow \mathbb{J}(\sigma \rightarrow \mathbb{J}\sigma)) \rightarrow \mathbb{N} \rightarrow \mathbb{J}\sigma$ is defined by

$$\begin{aligned}\text{rec}^\bullet(a, f, 0) &:= \eta(a) \\ \text{rec}^\bullet(a, f, n+1) &:= fn \bullet \text{rec}^\bullet(a, f, n).\end{aligned}$$