GENERALIZED ENCODINGS OF INDUCTIVE TYPES AND TOPICS IN CATEGORICAL LOGIC, HOMOTOPY TYPE THEORY, AND FOUNDATIONS.

SUDHIR MURTHY

Abstract. This proposal presents a generalization of encodings in Calculus of Construction (CoC) and expresses my related interests where Logic, Type Theory, and Category Theory converge.

The research topic aims to generalize well-established logical encodings of datatypes, using dependent sums, equality, and no other inductive types. These new encodings intend to satisfy corresponding universal properties, where the usual encodings fail. For example, to highlight this defect, while $\Pi x.(A \to x) \to (B \to x) \to x$ is logically equivalent to $A \vee B$, the encoding is not in bijection with the sum in the category of types.

I also suggest other topics I am interested in pursuing. Specifically, I wish to study the internal languages of toposes and higher groupoids with Homotopy Type Theory, with a formalistic philosophy of mathematics. I am especially curious about how model theory is generalized by categorical logic.

1. Introduction of First Topic

It is well-known that higher order intuitionistic logic, described by Calculus of Constructions (CoC), encodes data as Π-types [CH86]. The following table shows various equivalences in context $[A, B : \mathbf{Type}]$, $[s, t : A]$, and $[Q : A \to \mathbf{Prop}]$.

$A \wedge B$	$\Pi x : \mathbf{Prop}.\, (A \to B \to x) \to x$
$A \vee B$	$\Pi x : \mathbf{Prop}.\, (A \to x) \to (B \to x) \to x$
$\bot$	$\Pi x : \mathbf{Prop}.\, x$
$s = t$	$\Pi P : A \to \mathbf{Prop}.\, Ps \to Pt$
$\exists x : A.\, Qx$	$\Pi x : \mathbf{Prop}.\, (\Pi a : A, Qa \to x) \to x$
Bool	$\Pi x : \mathbf{Prop}.\, x \to x \to x$
Nat	$\Pi x : \mathbf{Prop}.\, (x \to x) \to (x \to x)$

If CoC were extended by the full suite of inductively defined logical connectives, then the encodings are provably equivalent. Even without inductive connectives, these encodings exhibit the same introduction and elimination rules. For propositions this suffices; however, for computationally-relevant types this is not enough. In addition, intro and elim rules should compose to identity. The goal of this research topic is to transform these encodings to preserve computational content.

2. Overview of Methods, Findings, and Expectations

To begin, I work within *Lean*'s type theory at a universe level u. Other foundations are discussed further in section 4. The key axioms utilized are:

- Function extensionality: for any types A, B and functions $f, g : A \to B$, if $\Pi a : A.\, fa = ga$ is inhabited then so is $f = g$.
- Proof irrelevance: for any proposition $P : \mathbf{Prop}$ and proof terms $p, q : P$, one can inhabit $p = q$.

The first axiom implies that inductively defined types satisfy corresponding universal properties. For example, the inductive product $A \times B : \mathbf{Type\ u}$ equipped with one constructor $\mathbf{mk} : A \to B \to A \times B$, is precisely the categorical product in a suitable category of types [1] denoted $\mathbb{C}$.

The goal is to find an "encoding" of any inductive type A; that is, to find a type B and bijection $f : B \to A$, so that B is built from no other inductive types apart from Σ and $=$. Based on preliminary attempts, I believe that "limit" types, such as the unit, are encodable in the form

$$\Sigma(\alpha : \Pi x : \mathbf{Type\ u}.\, Fx \to Gx).\, Nat\ F\ G\ \alpha.$$

These embody collections of natural transformations between functors $F, G : \mathbb{C} \to \mathbb{C}$. The first component collects one map in $Fx \to Gx$ per $x : \mathbf{Type\ u}$, and the second, proof this collection is indeed natural, i.e. naturality squares commute.

Similarly for "colimit" types, such as counit and sum, I expect encodings to embody dinatural transformations in the form

$$\Sigma(\alpha : \Pi x : \mathbf{Type\ u}.\, Sxx \to Txx).\, DiNat\ S\ T\ \alpha$$

for functors $S, T : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$.

The idea stems from generalizing the following example. The terminal type is the unit $\mathbf{1} \in \mathbb{C}$ inductively formed by one constructor $\mathbf{triv} : \mathbf{1}$. A first approximation, based on CoC encodings, is $\Pi x : \mathbf{Type\ u}.\, x \to x$ with obvious functions:

$$\begin{aligned} f &: \mathbf{1} \to \Pi x : \mathbf{Type\ u}.\, x \to x := \mathbf{1}.\mathbf{rec}\ \_\_\ \lambda x.\lambda t.t \\ g &: (\Pi x : \mathbf{Type\ u}.\, x \to x) \to \mathbf{1} := \lambda x.\mathbf{triv}. \end{aligned} \tag{2.1}$$

The recursor $\mathbf{1}.\mathbf{rec}$ is automatically generated when defining the inductive type $\mathbf{1}$. Its constructor enables us to define maps going into the type, whereas the recursor eliminates out (by pattern matching), e.g. f sends $\mathbf{triv} \mapsto \lambda x : \mathbf{Type\ u}.\, \mathbb{1}_x$.

While it is clear $g \circ f = \mathbb{1}$, it does not necessarily follow $f \circ g = \mathbb{1}$. To solve this problem, the idea is to liken elements of $\Pi x : \mathbf{Type\ u}.\, x \to x$ as natural transformations over the identity functor. At the moment, nothing ensures these are natural transformations. But by collecting pairs

$$\mathbf{Enc_1} := \Sigma(\alpha : \Pi x : \mathbf{Type\ u}.\, Id\ x \to Id\ x).\, Nat\ Id\ Id\ \alpha, \tag{2.2}$$

the second component guarantees the map $\alpha : \Pi x : \mathbf{Type\ u}.\, Id\ x \to Id\ x$ is natural. This data of commuting diagrams provides the "missing" computational content so that an explicit bijection with $\mathbf{1}$ may be constructed. This is carried out in the next section.

[1] Objects are elements of $\mathbf{Type\ u}$ and arrows are functions $f : A \to B$ identified upto provable equality. For example, functions $\lambda x : \mathbb{N}.\, x + 1$ and $\lambda n : \mathbb{N}.\, 1 + x$ are identified in this category because there is a proof they are equal from function extensionality (by induction on $\mathbb{N}$).

The blueprint does not succeed for colimits such as the sum. Consider types A, B : **Type u** and the sum $A + B$: **Type u** defined inductively by constructors **inl** : $A \to A + B$ and **inr** : $B \to A + B$. What is a suitable encoding? From CoC,

$$\Pi x : \textbf{Type u}.\, (A \to x) \to (B \to x) \to x$$

seems like a reasonable first guess. Obvious functions can be defined to and from $A + B$, but there is no way to prove they compose, both ways, to identity. So the next step is to view the type as a collection of natural transformations. One of the underlying functors seems to be $F := \mathrm{Hom}(\mathrm{Hom}(A, -), \mathrm{Hom}(B, -))$. However, it is impossible to "coherently" define an underlying *F.map*. For given any arrow $x \xrightarrow{f} y$, there is no way to define Ff in either direction

$$Ff : ((A \to x) \to (B \to x)) \rightleftarrows ((A \to y) \to (B \to y))$$

which fully utilizes the context. For instance, from $s : (A \to x) \to (B \to x)$, $t : A \to y$, and $b : B$, there is no application of terms involving all f, s, t, b to inhabit y. It seems there is some mixed variance at play which eludes functoriality.

A resolution is to lift "slippery" non-functorial maps of one parameter, into functorial ones with two parameters $S, T : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ and insist on dinaturality conditions. The second parameter accounts for the elusive, mixed variance. In the next section I consider the simplest example, the counit type, and determine what S and T should be.

This idea to employ dinatural maps in this setting, is not unique. Related research, from Paré and Román, encodes **Nat** as dinatural transformations [PR98]. They also motivate their work by generalizing the Coc encoding for **Nat**. Viewed through this lens, the reason for dinaturality is apparent: $\mathrm{Hom}(-, -)$ is not functorial (neither from $\mathbb{C} \to \mathbb{C}$ nor $\mathbb{C}^{\mathrm{op}} \to \mathbb{C}$) but it is dinatural (contravariant in the first component and covariant in the second).

3. Preliminary Results and Sketches of Proof

The purpose of this section is to demonstrate plausibility of success for the proposal, by specifically bijecting **1** with $\mathbf{Enc_1}$ given in (2.2). For functors $F, G : \mathbb{C} \to \mathbb{C}$ and $\alpha : \Pi x : \textbf{Type u}, Fx \to Gx$, *Nat* is defined by

$$Nat\ F\ G\ \alpha := \Pi x, y : \textbf{Type u}, \Pi f : x \to y, (Gf) \circ (\alpha x) = (\alpha y) \circ (Ff) \tag{3.1}$$

Intuitively this expresses each naturality square, for any $x \xrightarrow{f} y$, commutes.

$$\begin{array}{ccc} Gx & \xrightarrow{Gf} & Gy \\ {\scriptstyle \alpha x}\uparrow & & \uparrow{\scriptstyle \alpha y} \\ Fx & \xrightarrow[Ff]{} & Fy \end{array}$$

Lemma 3.1. *There exists a bijection between* **1** *and* $\boldsymbol{Enc_1}$.

Proof. The easy part is constructing function $f : \mathbf{1} \to \mathbf{Enc_1}$ and $g : \mathbf{Enc_1} \to \mathbf{1}$; similar to (2.1), we informally define

$$\begin{aligned} f &:= \mathbf{triv} \mapsto \langle \lambda x : \mathbf{Type\ u}.\ \mathbb{1}_x, \mathbf{pf} \rangle \\ g &:= \langle \alpha, h \rangle \mapsto \mathbf{triv} \end{aligned} \tag{3.2}$$

where **pf** is a proof the function $\lambda x : \mathbf{Type\ u}.\ \mathbb{1}_x$ is natural[2]. While $g \circ f = \mathbb{1}_\mathbf{1}$ is straightforward, the challenge is showing $f \circ g = \mathbb{1}_{\mathbf{Enc_1}}$.

The strategy is to apply function extensionality (many times) and proof irrelevance. The goal is reduced to a demonstration $\alpha\, X\, x = x$ from context $\langle \alpha, h \rangle : \mathbf{Enc_1}$, $X : \mathbf{Type\ u}$, and $x : X$. Since h encodes a proof α is natural, in particular the following square commutes.

$$\begin{array}{ccc} X & \xrightarrow{\lambda t.\, x} & X \\ {\scriptstyle \alpha X}\uparrow & & \uparrow{\scriptstyle \alpha X} \\ X & \xrightarrow[\lambda t.\, x]{} & X \end{array}$$

Feeding $x : X$ to the bottom left, and chasing both paths, completes the proof. □

The product can also be encoded in similar fashion (proof omitted).

Lemma 3.2. *For any types $A, B : \mathbf{Type\ u}$, there exists a bijection between $A \times B$ and $\boldsymbol{Enc}_{\mathbf{A}\times\mathbf{B}}$ where*

$$\boldsymbol{Enc}_{\mathbf{A}\times\mathbf{B}} := \Sigma\alpha : (A \to B \to x) \to x.\ \mathit{Nat}\ (\mathrm{Hom}(A, \mathrm{Hom}(B, -))\ \mathit{Id}\ \alpha.$$

Note: the functor $(\mathrm{Hom}(A, \mathrm{Hom}(B, -))$ *sends objects X to $A \to B \to X$ and maps $X \xrightarrow{f} Y$ to $\lambda s \lambda a \lambda b.\ f(sab)$.*

This result appears redundant; no additional encoding is required since, after all, the product of two types $A \times B$ is syntactic sugar for $\Sigma x : A.\, B$. However, as discussed in section 5, it may be possible to weaken the dependent sum to a non-inductive, CoC encoding. In that case, Lemma 3.2 is genuinely needed to encode a product.

Interestingly, Lemma 3.2 appears like an application of Yoneda lemma on the product. When uncurried, the type $A \to B \to x$ can be read $A \times B \to x$ and, hence, the connection.

A process is outlined for how the co-unit **0**, inductively defined as having no constructors, may be encoded. Its recursor automatically generates a map from **0** to any other type. Abbreviate this map by $\mathbf{exf} : \Pi x : \mathbf{Type\ u}.\ \mathbf{0} \to x$.

While the typical CoC encoding $\Pi x : \mathbf{Type\ u}.\, x$ does not look like a natural transformation, the weakening $\Pi x : \mathbf{Type\ u}.\, (x \to x) \to x$ does. And while $\mathrm{Hom}(-,-)$ is not functorial, the weakening $\mathrm{Hom}(-_1, -_2) : \mathbb{C}^{\mathrm{op}} \times \mathbb{C}$ is.

[2]This amounts to proving $f \circ \mathbb{1}_X = \mathbb{1}_Y \circ f$, for any arrow $X \xrightarrow{f} Y$.

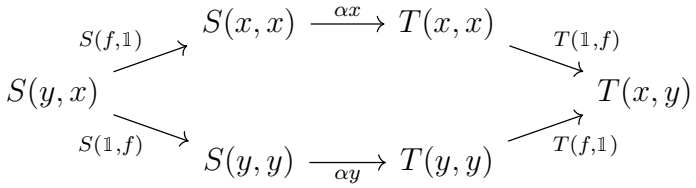
Functor $\mathrm{Hom}(-_1, -_2) : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ sends objects $X \times Y \mapsto X \to Y$ and arrows

$$\begin{array}{l} X_1 \xleftarrow{f} X_2 \\ Y_1 \xrightarrow[g]{} Y_2 \end{array} \quad \mapsto \quad X_1 \to Y_1 \xrightarrow{\lambda s \lambda x_2.\, g(s(f x_2))} X_2 \to Y_2.$$

Functor $Id(-_2) : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ sends objects $X \times Y \mapsto Y$ and arrows

$$\begin{array}{l} X_1 \xleftarrow{f} X_2 \\ Y_1 \xrightarrow[g]{} Y_2 \end{array} \quad \mapsto \quad Y_1 \xrightarrow{g} Y_2.$$

For functors $S, T : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ and $\alpha : \Pi x : \mathbf{Type\ u}.\, S(x,x) \to T(x,x)$, define $DiNat\ S\ T\ \alpha$: **Prop** as the proposition which expresses dinaturality conditions. Namely, that for every arrow $x \xrightarrow{f} y$, the diagrams commutes.



Conjecture 3.3. There exists a bijection between $\mathbf{0}$ and $\mathbf{Enc_0}$ where

$$\mathbf{Enc_0} := \Sigma(\alpha : \Pi x : \mathbf{Type\ u}.\, (x \to x) \to x.\, DiNat\ \mathrm{Hom}(-_1, -_2)\ Id(-_2)\ \alpha.$$

A valid criticism against $\mathbf{Enc_0}$ is that the simpler encoding $\Pi x : \mathbf{Type\ u}.\, x$ suffices. The recursor generated by $\mathbf{0}$ not only eliminates to all types at universe u, but to all types in all universes including **Prop**. So just by defining arrows $\mathbf{0} \rightleftarrows \Pi x : \mathbf{Type\ u}.\, x$, from virtue of inhabiting $\mathbf{0}$, the arrows vacuously compose to identity. However, the current encoding $\mathbf{Enc_0}$ does not require strong elimination principles. There is another important benefit: in absence of inductive type $\mathbf{0}$, I would like to show its encoding is the initial object in $\mathbb{C}$. Without $\mathbf{0}$ to strongly eliminate out of, the type $\Pi x : \mathbf{Type\ u}.\, x$ conveys too little.

4. Size Issues and Foundations

An important objective is to show that these encodings satisfy corresponding universal properties without reference to the inductive types they encode. For example, in absence of $\mathbf{0}$, I want the type $\mathbf{Enc_0}$ to still be initial; or in absence of $\mathbf{1}$, $\mathbf{Enc_1}$ terminal; or of $A \times B$, $\mathbf{Enc}_{A \times B}$ the product. However, there is a major setback. The types of these encodings do not live in universe u — they live in one universe higher.

This problem is not limited to *Lean*'s foundational type theory. While other foundations, such as *Coq* or *HoTT*, tend to differ in universe management, they do not overcome the core issue with these encodings. The culprit is quantification over all types at universe u; such a type lives in a universe higher.

A possible solution is to consider the type of "sets", an impredicative universe **Set** as used in *Coq*. However these encodings are dependent sums; even if the first component is a set and the second a proposition, is the collection of such pairs a set? To address this issue, a compelling foundations to work in is *HoTT* . Universe management in *HoTT* is radically different from *Lean* or *Coq*; there are no impredicative universe such as **Prop** or **Set**. Rather, at each universe there is a further hierarchy of truncation levels. Propositions are those types in the universe whose underlying equality is contractible — that is, they satisfy proof-irrelevance. Sets are those types in the universe whose underlying equality is propositional, etcetera. At least in *HoTT*, the condition being a set is not a typing judgement — it is a first order citizen which can be inhabited.

Interestingly, these same issues are not just type-theoretic; Shulman investigates set-theoretic challenges of category theory and finds similar problems with common set-theoretic foundations [Shu08]. For example, consider category theory in viewpoint of *NGB* — a simple theory of sets and classes. A natural transformation is defined as a family of component maps, indexed by objects in a given category. When the category is large and locally small, then a given natural transformation is a proper class. But there is no collection of all natural transformations in *NGB*, where it is impossible to collect proper classes.

While the challenges faced are serious, they reflect existing challenges in the foundations of mathematics. Investigating them is important and this research project may shed light on various design choices for foundations.

5. Generalization and Significance

Once encodings of various inductive types and their bijections have been formed, the project presents several directions for expansion.

(1) The first is generalizing out of *Lean*; for example in *HoTT*, since there is no universe of propositions, one might expect to see applications of $||\cdot||$ which converts a type to a proposition.
(2) One might weaken the necessity of inductively defined equality. In Lemmas 3.1 and 3.2, the proofs do use computational rules from inductive sums, but not from equalities. It seems possible to use CoC encodings of equality, which capture elimination principles to facilitate the same reasoning.
(3) Furthering this, the ultimate goal is to remove inductive types completely. Instead of dependent sums to capture pairs, one can use similar CoC encodings of existence. Then one can insist on an axiom of choice to eliminate out of weaker existential statements. This is why I have presented Lemma 3.2; with weaker existential quantifiers, an encoding for a product is needed.

I believe the significance of these generalizations is two-fold: (a) it may be possible to encode all other inductive data with just the dependent sum, and (b) it may offer new insight into universal properties for various inductive types.

The benefit of (a) is that metalogical analysis is easier with fewer inductive types. Just as the W-type, of well-founded trees, is a family of inductive types which characterizes them all, it would be novel to determine whether Σ is also such a family.

It might also be worthwhile to study how additional recursion schemes could be encoded by methods of this research. Just as regular inductive types correspond to initial algebras in category theory, what kind of categorical information do other recursion schemes exhibit? To help bridge that gap, this project could offer a type-theoretic interpretation.

The importance of (b) is highlighted by Paré and Romań's research where they obtain a new universal property classifying natural numbers [PR98]. Perhaps all inductive data is classifiable as some form of dinatural transformation. The encodings potentially gained from this research, could provide new universal properties for existing objects.

6. Related Interests

I wanted to take this space to introduce other concepts in Topos Theory and Logic, I am interested in further exploring. I believe the modality of "internalization", or provability from Gödel-Löb Logic, can be viewed as a monad on the category of $\mathbf{RCA_0}$-sentences (the theory of decidable, or recursive, sets). Additionally, categories of sentences, in general, and functors between them have practical benefits: they represent mechanical translations between sentences of a theory and their proofs. This gives rise to a "syntactic" notion of models and I am eager to explore how this concept relates to functorial semantics in categorical logic.

References

[CH86] Thierry Coquand and Gérard Huet. "The calculus of constructions". PhD thesis. INRIA, 1986.

[PR98] Robert Paré and Leopoldo Román. "Dinatural numbers". In: *Journal of Pure and Applied Algebra* 128.1 (1998), pp. 33–92.

[Shu08] Michael A Shulman. "Set theory for category theory". In: *arXiv preprint arXiv:0810.1279* (2008).