From Debruijn to co-Debruijn using Category Theory

Everybody's Got To Be Somewhere[2]

Marius Weidner

Chair of Programming Languages, University of Freiburg
weidner@cs.uni-freiburg.de

Abstract. We explore the connection between De Bruijn indices used for variable representation in an intrinsically scoped syntax and the usage of co-De Bruijn representation. To understand the duality between De Bruijn and co-De Bruijn, we will look into the categorical concepts underpinning scopes, binders, and intrinsically scoped syntaxes in general. All definitions, examples, and concepts presented here are directly derived from Conor McBride's paper[2], with certain modifications and additional definitions and examples included. The original paper is written in Agda code but should have the same meaning as the on-paper math presented here.

Table of Contents

1 Scopes and Binders Categorically

Before we begin, it is important to fully understand the concept of variables, scopes, and binders. Subsequently, we will examine these concepts within the framework of category theory. Every categorical concept used will be defined and explained, thus any prerequisite knowledge beyond a solid understanding of set theory is not required.

Variables serve as placeholders for terms in a syntax and are bound by binding terms within a syntax. While variables could also appear free, i.e., not bound, we do not look at languages that allow this. In fact, our very goal is to ensure that terms in our syntaxes can only be constructed in such a way that variables can only be used when they are bound.

Scopes fundamentally hold information about all the variables present within a particular term. This information might simply entail the quantity of available variables or it could extend to include additional information such as the sort of a variable.

Binding terms introduce new variables within a subterm, thereby expanding the scope of the subterm by one or more variables.

1.1 The Category of Scopes

Definition 1. *A category consists of a set of objects and a set of morphisms. Morphisms are indexed by two objects: source and target. Given a Category $\mathcal{C}$, we denote its objects as $|\mathcal{C}|$, and its morphisms from specified source to target as $\mathcal{C}(S, T)$, where $S, T \in |\mathcal{C}|$.*

Interestingly, this definition of a category differs from other definitions in the sense that morphisms are only defined between concrete objects S and T. Morphisms do not exist independently of their source and target objects. This perspective is somewhat a constructivist view on categories, which fits, considering the original paper is formalized in Agda.

Definition 2. *For every set X, we define the category of scopes Δ_+^X with objects $\bar{x}, \bar{y}, \bar{s} \in X^*$ and morphisms $f, g \in \Delta_+^X(\bar{x}, \bar{y})$ inductively defined by the following inference rules in infix notation*[1]*:*

$$\frac{}{\varepsilon \sqsubseteq \varepsilon}\ \cdot \qquad \frac{\bar{x} \sqsubseteq \bar{y}}{\bar{x}x \sqsubseteq \bar{y}x}\ 1 \qquad \frac{\bar{x} \sqsubseteq \bar{y}}{\bar{x} \sqsubseteq \bar{y}y}\ 0$$

In the category of scopes, objects are scopes, while morphisms are *order-preserving embeddings*.

A scope is represented as a list of *sorts*, indicating the sort of variables present in the scope, with the length of the list indicating the number of variables in the scope.

Example 1. Consider constructing the language of second-order lambda calculus, where both expression and type variables exist. In such a scenario, we can represent the set of sorts as $X = \{e, t\}$, where e represents expression variables and t represents type variables.

A morphism in Δ_+^X exists only if we can embed one scope into another by adding variables into the source scope without altering the order of the original variables.

Example 2. Let $X = \top$, where $\top$ is the set with a single element $\star$. In the category Δ_+^T, objects correspond to *numbers* $n, p, q \in \top^* = \mathbb{N}$. In this scenario, objects are scopes with n variables, all of the same sort $\star$. Morphisms map from a smaller scope p to a larger scope q by selecting p variables in q and mapping all variables from p to them. Figure 1 illustrates the embedding of a scope with 3 variables into one with 5 variables in the required order-preserving manner.

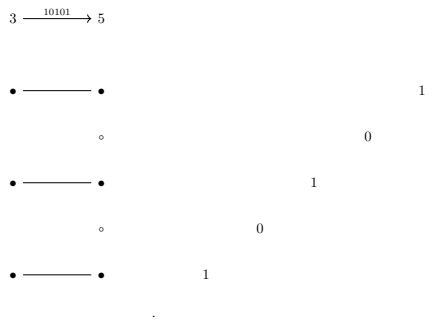

Fig. 1. Embedding a scope with 3 variables into a scope with 5

[1] In the construction of morphisms using inference rules, we usually omit the axiom rule $\cdot$.

Remark 1. Morphisms in Δ_+^X can be represented by *bit vectors* $\bar{b} \in \{0,1\}^*$ with one bit per variable of the target scope telling whether it has been mapped to or skipped by the source scope. We only need one bit of information, because the mapping is always sort preserving by definition.

Definition 3. *We define the infix operation of composition $f;g$ for two morphisms $f \in \Delta_+^X(\bar{x},\bar{y})$ and $g \in \Delta_+^X(\bar{y},\bar{z})$, resulting in a morphism $h \in \Delta_+^X(\bar{x},\bar{z})$, inductively on the inference rules of morphisms:*

$$\begin{aligned} \cdot \ ; \cdot \ &= \cdot \\ f1 \ ; \ g1 \ &= \ (f;g)1 \\ f0 \ ; \ g1 \ &= \ (f;g)0 \\ f \ \ ; \ g0 \ &= \ (f;g)0 \end{aligned}$$

Remark 2. The operation $f;g$ reads 'f than g' and applies f first and the result to g, in contrast to $f \circ g$ which reads 'f after g'. The operation $f;g$ reads 'f than g' and executes the morphism f first, and then applies g to the result. This contrasts with the composition $f \circ g$, which reads 'f after g', indicating the application of g followed by f.

Example 3. Let's reconsider the category $\Delta_+^\top$. Objects are represented by numbers, and morphisms can be encoded as bit vectors.

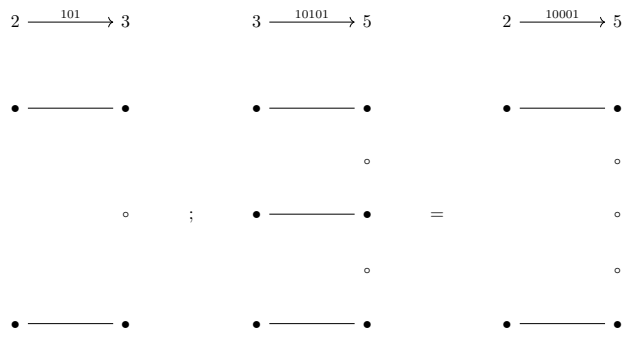

Fig. 2. Composition of two embeddings in the category $\Delta_+^\top$

Corollary 1. *Composition of morphisms in Δ_+^X is associative, i.e., $f;(g;h) = (f;g);h$ for $f \in \Delta_+^X(\bar{x},\bar{y})$, $g \in \Delta_+^X(\bar{y},\bar{z})$, and $h \in \Delta_+^X(\bar{x},\bar{s})$. The proof follows straightforward induction.*

Corollary 2. *For every object $\bar{x}$ in the category Δ_+^X, there exists an identity morphism $id_{\bar{x}} : \bar{x} \sqsubseteq \bar{x}$. The identity morphism can be constructed for every $\bar{x}$ using the axiom inference rule followed by $\|\bar{x}\|$ times applying the 1 rule. Additionally, the equalities $id_{\bar{x}};f = f$ and $g;id_{\bar{x}} = g$ hold for all $\bar{x}$ and morphisms $f \in \Delta_+^X(\bar{x},\bar{y})$ and $g \in \Delta_+^X(\bar{y},\bar{x})$, respectively. This can straightforwardly be proven through induction.*

Proposition 1. *The category of scopes Δ_+^X is a well-defined category for every set X. The composition operation $f;g$ is associative. Identity morphisms exist within Δ_+^X and act neutral in composition.*

This concludes the definition of the category of scopes Δ_+^X. Primarily, scopes serve as objects in this category and hold information regarding the sorts and quantities of variables present. Furthermore, morphisms in this category embed one scope into another in an order-preserving manner. Importantly, Δ_+^X is in fact a valid category, characterized by associative morphism composition and the existence of identity morphisms.

1.2 Intrinsically Scoped De Bruijn Syntax

Next, we look at the definition of a *intrinsically scoped* language that restricts the construction of terms such that variables can only be used if they were bound previously. The indices of terms in this syntax are objects in the category of scopes. Pointing to a variable is achieved by using morphisms in the category of scopes.

Definition 4. *The indexed set of intrinsically scoped lambda calculus terms $Tm : \mathbb{N} \to Set$, where the index corresponds to the number of variables $n \in |\Delta_+^\top| = \mathbb{N}$ in scope, is defined inductively by the following inference rules*[2]*:*

$$\frac{1 \sqsubseteq n}{Tm\ n}\ \#\qquad \frac{Tm\ n \quad Tm\ n}{Tm\ n}\ \$\qquad \frac{Tm\ (n+1)}{Tm\ n}\ \lambda$$

The λ rule binds a new variable in its body, incrementing n by one within that body.

In contrast, the variable rule # points to a variable using a morphism in the scope category, precisely selecting one of the n bound variables. The referencing of a variable can be represented by a bit vector of length n with precisely a single 1 inside it. This is in turn equal to a number $m \in [0, n]$, indicating the position of the sole 1 in the bit vector. Numbers used as variables, indicating the amount of binders between the binding and the usage of a variable, are referred to as De Bruijn indices.

Example 4. Consider the $\mathbb{K}$ and $\mathbb{S}$ combinators expressed in lambda calculus, comparing their representations with variables as names and intrinsic De Bruijn notation:

$\mathbb{K} = \lambda x.\lambda y.x \qquad\qquad = \lambda\lambda\#1$

$\mathbb{S} = \lambda x.\lambda y.\lambda z.x\ z\ (y\ z) = \lambda\lambda\lambda\#2\ \#0\ (\#1\ \#0)$

Note that, for example, a De Bruijn term $\lambda\ \#1$ could not even be defined using the given inference rules.

[2] When constructing the \$ rule, we usually omit the dollar sign and, instead, express it by using a space between the two subterms.

2 From De Bruijn to co-De Bruijn

In the last section, all terms intrinsically possessed, by construction, knowledge of how many variables are in scope. Variable constructors then selected one out of all variables in scope. Now, our objective is to transition from De Bruijn representation to its dual, the co-De Bruijn representation. In the co-De Bruijn representation, instead of selecting the variable at the *latest* point in the syntax tree (i.e., the leaf node in the tree, the variable constructor), we determine the fate of a variable at the *earliest* possible moment. To understand this transition, we must delve deeper into the category of scopes.

2.1 The Slice Category of Scopes

Definition 5. *If $\mathcal{C}$ is a category and $S, T, O \in |\mathcal{C}|$, the slice category $\mathcal{C}/O$ has pairs (S, f) as objects, where $f \in \mathcal{C}(S, O)$. Morphisms in $(\mathcal{C}/O)((S, f), (T, g))$ are morphisms $h \in \mathcal{C}(S, T)$ such that $f = h; g$.*

The slice category of a given category has pairs of objects and morphisms as objects, where the morphisms must have the objects together with them in the pair as source and the object we slice through as their target. The morphisms that remain in the slice category map between these pairs but are simply a subset of the morphisms in the original category, thus composing as expected. We refer to the slice category $\Delta_+^X \setminus \bar{s}$ as the category of subscopes, with objects $\bar{b} \in |\Delta_+^X \setminus \bar{s}|$ and morphisms $h \in [\Delta_+^X \setminus \bar{s}](\bar{b}_1, \bar{b}_2)$. Given the scope $\bar{s}$ that we slice through, we now have objects that are pairs consisting of a subscope $\bar{x}$ of $\bar{s}$ and a mapping from $\bar{x}$ to $\bar{s}$.

Remark 3. Objects in $\Delta_+^X \setminus \bar{s}$ are *bit vectors* $\bar{b} \in \{0, 1\}^*$ with one bit per variable of slicing scope $\bar{s}$, telling whether it has been selected by the morphisms from the subscope.

Example 5. The morphism 0111 in $\Delta_+^T \setminus 5$ from 01110 to 11110 (or $(3, 01110)$ to $(4, 11110)$) embeds a scope with 3 variables into one with 4 by inserting an additional variable at the very beginning. Given that we sliced with object 5, both 01110 and 11110 are subscopes of a scope with 5 variables, selecting 3 and 4 variables from the scope with 5 variables. The morphism 0111 effectively embeds one subscope into the other.

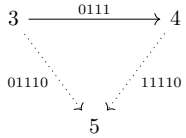

Fig. 3. In the diagram, objects in the slice category $\Delta_+^\top \setminus 5$ are represented by dotted arrows along with their source, while normal arrows are actual morphisms.

Definition 6. *Objects $T, S \in |\mathcal{C}|$ in category $\mathcal{C}$ possess a coproduct object $T+S$ if there exist morphisms $l \in \mathcal{C}(T, T+S)$ and $r \in \mathcal{C}(S, T+S)$ such that for every pair $f \in C(T, U)$ and $g \in \mathcal{C}(S, U)$, there exists a unique morphism $h \in \mathcal{C}(T+S, U)$ such that $f = l; h$ and $g = r; h$.*

Example 6. Coproduct in the category $\Delta_+^\top \setminus 5$. ... explain

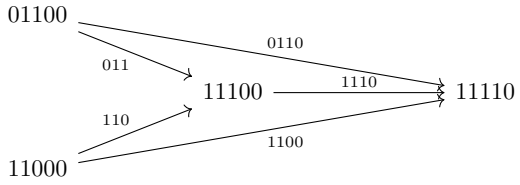

Fig. 4. Coproduct in the category $\Delta_+^\top \setminus 5$.

Remark 4. The coproduct $\bar{b}_1 + \bar{b}_2$ of two subscopes $\bar{b}_1, \bar{b}_2$ corresponds to the minimal subscope covering both $\bar{b}_1$ and $\bar{b}_2$. The coproduct $\bar{b}_1 + \bar{b}_2$ can be computed by pointwise disjunction of $\bar{b}_1$ and $\bar{b}_2$.

2.2 Sets Indexed by Scopes

Definition 7. *We define the category $\overline{Set}$ as the category of sets indexed by scopes $\bar{x} \in X^*$. The category has objects $T, S \in |\overline{Set}| = X^* \to Set = \bar{X}$. Morphisms $f \in \overline{Set}(T, S) = \{\bar{x} \in X^*\} \to T\ \bar{x} \to S\ \bar{x}$ are functions between sets implicitly indexed over the sets' indices.*

Example 7. The set of intrinsically scoped debruijn terms in Definition 4 is an object of $\overline{Set}$.

Definition 8. *Let $\_ \Uparrow \_ : \bar{X} \to \bar{X} = (T, \bar{x}) \mapsto (T(\bar{s}) \times \bar{s} \sqsubseteq \bar{x})$. We write $t \uparrow h$ for elements of $T \Uparrow \bar{x}$.*

Remark 5. The set $T \Uparrow \bar{x}$ packs an set $T \in \bar{X}$ indexed by $\bar{x} \in X^*$ applied to a subscope $\bar{s}$ of $\bar{x}$, together with a selection $\bar{b} \in |\Delta_+^X \setminus \bar{x}|$ of the variables of T.

2.3 The Notion of Relevant Pairs

Definition 9. *Let $Cov : \bar{x} \sqsubseteq \bar{s} \to \bar{y} \sqsubseteq \bar{s} \to Set$ be the set of* coverings *indexed by morphisms $\bar{b}_1$ and $\bar{b}_2$*

$$\frac{}{Cov\ \cdot\ \cdot}\ \cdot \quad \frac{Cov\ \bar{b}_1\ \bar{b}_2}{Cov\ \bar{b}_1 1\ \bar{b}_2}\ L \quad \frac{Cov\ \bar{b}_1\ \bar{b}_2}{Cov\ \bar{b}_1\ \bar{b}_2 1}\ R \quad \frac{Cov\ \bar{b}_1\ \bar{b}_2}{Cov\ \bar{b}_1 1\ \bar{b}_2 1}\ B$$

Remark 6. Coverings $Cov\ \bar{b}_1\ \bar{b}_2$ hold data about the coproduct of $\bar{b}_1$ and $\bar{b}_2$ as well as information about the original appearance of $\bar{b}_1$ and $\bar{b}_2$.

Definition 10. *Let the set of relevant pairs be defined as* $\_ \times_R \_ : \bar{X} \to \bar{X} \to \bar{X} = (T, S, \bar{x}) \mapsto ((\_ \uparrow \bar{b}_1 : T \Uparrow \bar{x}) \times (\_ \uparrow \bar{b}_2 : S \Uparrow \bar{x}) \times Cov\ \bar{b}_1\ \bar{b}_2)$ *+ were elements are denoted as* $\_,_R \_ : T \Uparrow \bar{x} \to S \Uparrow \bar{x} \to (T \times_R S) \Uparrow \bar{x} = ((t_1 \uparrow \bar{b}_1), (t_2 \uparrow \bar{b}_2)) \mapsto ((t_1 \uparrow \bar{b}_1'), (t_2 \uparrow \bar{b}_2'), \bar{b}_1 \oplus \bar{b}_2) \uparrow \bar{b}'$

Example 8. Look at the term $\lambda x.\lambda y.\lambda z.z\ y = \lambda\lambda\lambda(\#0\ \#1)$ in De Bruijn notation. The variable terms could also be written as $z' : Tm \Uparrow 3 = \#0 \uparrow 001$ and $y' : Tm \Uparrow 3 = \#1 \uparrow 011$. And the application term could be a *relevant pair*

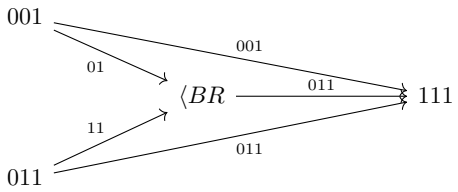

Fig. 5. Covering in the category $\Delta_+^\top \setminus 3$.

$z',_R y' : (Tm \times_R Tm) \Uparrow 3 = (\#0 \uparrow 01, \#1 \uparrow 11, BR) \uparrow 011$

2.4 Intrinsically Scoped co-De Bruijn Syntax

Definition 11. *Let* $Tm : \mathbb{N} \to Set$ *be inductively defined:*

$$\frac{}{Tm\ 1}\ \#\quad \frac{(Tm \times_R Tm)\ n}{Tm\ n}\ \$\quad \frac{Tm\ (n+1)}{Tm\ n}\ \lambda$$

Example 9. We again consider the $\mathbb{K}$ and $\mathbb{S}$ combinators. For convenience the cover computed by the relevant pair next to the comma.

$\mathbb{K} = \lambda x.\lambda y.x \qquad\qquad = \lightning$
$\mathbb{S} = \lambda x.\lambda y.\lambda z.x\ z\ (y\ z) = \lambda\ \lambda\ \lambda($
$\quad (((\# \uparrow 10),_{[LR]} (\# \uparrow 01)) \uparrow 101)\ ,_{[LRB]}$
$\quad (((\# \uparrow 10),_{[LR]} (\# \uparrow 01)) \uparrow 011)$
$)$

The $\mathbb{K}$ combinator is not definable with our co-De Bruijn syntax because unused variables are not allowed to exist. We could extend the definition with a rule λ' that just takes a $Tm\ n$ as subterm, not increasing n and forgetting about the variable.

The paper actually abstracts entirely from specific binding rules, allowing the binding rules to freely choose how many variables they wish to bind, ranging

from 0 to n. Furthermore, it includes a translation from lambda calculus in De Bruijn representation to co-De Bruijn representation, using the fact that $\_ \Uparrow \_$ is, in fact, a monad.

The paper then proceeds to construct a universe of metasyntaxes with binding, essentially a syntax with rules that enable the construction of any syntax. The most interesting chapter of the paper is the definition of *hereditary* substitution on this metasyntax. This substitution effectively traverses the tree only when necessary (i.e., if the substitution contains variables known to be present in the subterm) and performs weakening without traversing the tree at all, again utilizing our monad over sets indexed by scopes. Additionally, the substitution ensures termination because it is indexed (and substitution inductively proceeds over) the *active* scope, which refers to variables that still require substitution.

References

[1] Conor McBride. *Cats and types: Best friends?* Aug. 2021. URL: https://www.youtube.com/watch?v=05IJ3YL8p0s.

[2] Conor McBride. "Everybody's Got To Be Somewhere". In: *Electronic Proceedings in Theoretical Computer Science* 275 (July 2018), pp. 53–69. ISSN: 2075-2180. DOI: 10.4204/eptcs.275.6. URL: http://dx.doi.org/10.4204/EPTCS.275.6.