From De Bruijn to co-De Bruijn using Category Theory

Everybody's Got To Be Somewhere[2]

Marius Weidner

Chair of Programming Languages, University of Freiburg

February 3, 2024

Outline

The Category of Scopes: Δ_+^X

Definition

Let Δ_+^X be the category of scopes with

- Objects: $\bar{x}, \bar{y}, \bar{s} \in |\Delta_+^X| = X^*$ ad
- Morphisms: $f, g \in \Delta_+^X(\bar{x}, \bar{y})$ for $\bar{x}, \bar{y} \in X^*$ are inductively defined by inference rules

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

Remark

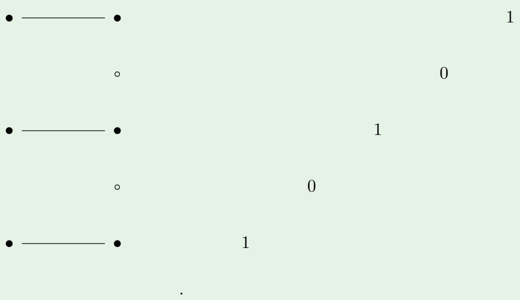
Morphisms in Δ_+^X can be represented by *bit vectors* $\bar{b} \in \{0,1\}^*$ with one bit per variable of the target scope telling whether it has been mapped to or skipped by the source scope.

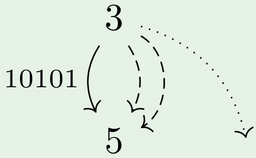
Objects & Morphisms in $\Delta_+^\top$

Example

Let $X = \top$, where $\top$ is the set with exactly one element $\langle\rangle$.
Then, objects in $\Delta_+^\top$ represent numbers.

$$3 \xrightarrow{10101} 5$$

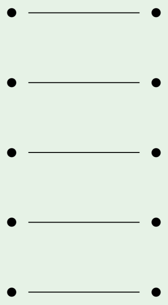




Identity and Composition in $\Delta_+^\top$

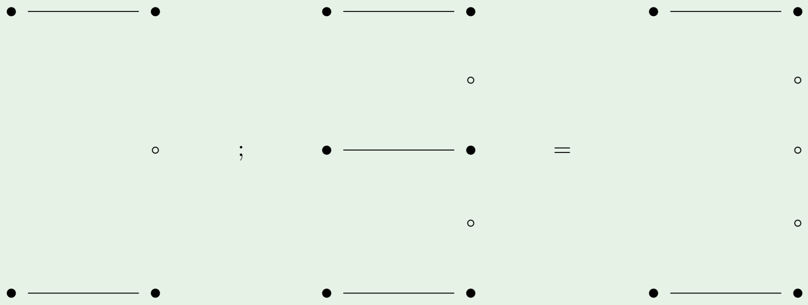
Example

$5 \xrightarrow{id} 5$



Example

$2 \xrightarrow{101} 3$; $3 \xrightarrow{10101} 5$ = $2 \xrightarrow{10001} 5$



Δ_+^X is in Fact a Category

Lemma

In Δ_+^X every object $\bar{x} \in X^*$ has an identity morphism, i.e. we can construct an identity morphism for every object $\bar{x}$ using the inference rules.

Proof.

$$\begin{aligned} id &: \bar{x} \sqsubseteq \bar{x} \\ id\ \varepsilon &= \cdot \\ id\ \bar{x}x &= id 1 \end{aligned}$$

□

Corollary

$$\begin{aligned} id - l &: id; f = f \\ id - r &: f; id = f \end{aligned}$$

Lemma

In Δ_+^X two morphisms $f : \bar{x} \sqsubseteq \bar{y}$ and $g : \bar{y} \sqsubseteq \bar{z}$ compose to a morphism $f;g : \bar{x} \sqsubseteq \bar{z}$, i.e. we can construct a morphism $f;g$ from f and g using the inference rules.

Proof.

$$\begin{aligned} \_;\_ \quad &: \bar{x} \sqsubseteq \bar{y} \to \bar{y} \sqsubseteq \bar{z} \to \bar{x} \sqsubseteq \bar{z} \\ \cdot \ ; \cdot &= \cdot \\ f1 \ ; \ g1 &= (f;g)1 \\ f0 \ ; \ g1 &= (f;g)0 \\ f \ ; \ g0 &= (f;g)0 \end{aligned}$$

□

Corollary

assoc : $f; (g; h) = (f; g); h$

antisym : $(f : \bar{x} \sqsubseteq \bar{y}) \to (g : \bar{y} \sqsubseteq \bar{z}) \to \bar{x} = \bar{y} \wedge f = g = id\ \bar{x}$

Intrinsically Scoped De Bruijn Syntax via $\Delta_{+}^{\top}$

Definition

Let $Tm : \mathbb{N} \to Set$ be the set of lambda calculus terms inductively defined by

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

Example

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

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

The Slice Category of Subscopes: $\Delta_+^X \setminus \bar{s}$

Definition

Let $\Delta_+^X \setminus \bar{s}$ be the category of subscopes for a given $\bar{s} \in X^*$ with

- Objects: $\bar{b}, (\bar{x}, f) \in |\Delta_+^X \setminus \bar{s}| = (\bar{x} : X^* \times \Delta_+^X(\bar{x}, \bar{s}))$ and
- Morphisms: $h \in [\Delta_+^X \setminus \bar{s}]((\bar{x}, f), (\bar{y}, g))$ such that $f = h; g$

$$\begin{tikzcd}
\bar{x} \arrow[rr, "h"] \arrow[dr, dotted, "f"'] & & \bar{y} \arrow[dl, dotted, "g"] \\
& \bar{s} &
\end{tikzcd}$$

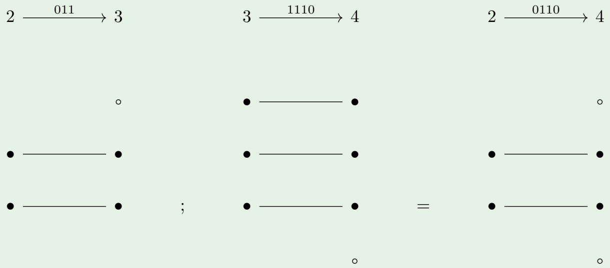
Remark

Objects in $\Delta_+^X \setminus \bar{s}$ can be represented by *bit vectors* $\bar{b} \in \{0, 1\}^*$ with one bit per variable of scope $\bar{s}$, telling whether it has been selected.

Objects & Morphisms in $\Delta_+^T \setminus 5$

Example

$$
\begin{tikzcd}
3 \arrow[rr, "0111"] \arrow[dr, dotted, "01110"'] & & 4 \arrow[dl, dotted, "11110"] \\
& 5 &
\end{tikzcd}
$$



Alternatively:

- $(3, 01110) \xrightarrow{0111} (4, 11110)$
- $01110 \xrightarrow{0111} 11110$

The Curious Case of Coproducts in $\Delta_+^X \setminus \bar{s}$

Theorem

Objects in the slice category $\bar{b}_1, \bar{b}_2 \in |\Delta_+^X \setminus \bar{s}|$ have a coproduct object $\bar{b}_1 + \bar{b}_2$, i.e. there exist morphisms $l \in [\Delta_+^X \setminus \bar{s}](\bar{b}_1, \bar{b}_1 + \bar{b}_2)$ and $r \in [\Delta_+^X \setminus \bar{s}](\bar{b}_2, \bar{b}_1 + \bar{b}_2)$ such that every pair $f \in [\Delta_+^X \setminus \bar{s}](\bar{b}_1, \bar{b}_3)$ and $g \in [\Delta_+^X \setminus \bar{s}](\bar{b}_2, \bar{b}_3)$ factor through a unqiue $h \in [\Delta_+^X \setminus \bar{s}](\bar{b}_1 + \bar{b}_2, \bar{b}_3)$ such that $f = l; h$ and $g = r; h$.

Example

$$\begin{tikzcd}
01100 \arrow[dr, "011"'] \arrow[drr, "0110"] & & \\
& 11100 \arrow[r, "1110"] & 11110 \\
11000 \arrow[ur, "110"] \arrow[urr, "1100"'] & &
\end{tikzcd}$$

Remark

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 pointswise disjunction of $\bar{b}_1$ and $\bar{b}_2$.

The Category Set_X of Sets Indexed by Scope

Definition

Let Set_X be the category of sets indexed by scopes $\bar{x} \in X^*$ with

- Objects: $T, S \in |Set_X| = X^* \to Set = \bar{X}$ and
- Morphisms: $f_. \in Set_X(T, S) = (\bar{x} \in X^*) \to T(\bar{x}) \to S(\bar{x}) = T \dot{\to} S$

Definition

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 $T \Uparrow \bar{x}$.
We define $Ref : Set_X \dot{\to} Set_X$ to be the endofunctior induced by the mapping

- $Ref(T) = (\bar{x} \mapsto T \Uparrow \bar{x}) \in \bar{X}$ for objects and
- $Ref(f_.) = (t \uparrow h) \mapsto (f_.(t) \uparrow h) \in T \dot{\to} S$ for morphisms

Remark

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 $h \in |\Delta_+^X \setminus \bar{x}|$ of the variables of T.

Δ_+^X makes Ref a Monad!

Theorem

The endofunctor $Ref : Set_X \to Set_X$ gives rise to a monad with the two natural transformations

- $unit : Id(T) \xrightarrow{\cdot} Ref(T) = t \mapsto (t \uparrow id)$ *and*
- $mult : Ref(Ref(T)) \xrightarrow{\cdot} Ref(T) = ((t \uparrow h_1)h_2) \mapsto (t \uparrow h_1 ; h_2)$

Example

$$\begin{tikzcd}
Tm \arrow[r, dashed, "unit"] & \bar{x} \mapsto (Tm\ \bar{x} \uparrow \bar{x} \sqsubseteq \bar{x}) \arrow[dl, dashed, "Ref"'] \\
\bar{y} \mapsto ([\bar{x} \mapsto (Tm\ \bar{x} \uparrow \bar{x} \sqsubseteq \bar{x})]\bar{s} \uparrow \bar{s} \sqsubseteq \bar{y}) \arrow[r, dashed, "mult"] & \bar{y} \mapsto (Tm\ \bar{s} \uparrow \bar{s} \sqsubseteq \bar{y})
\end{tikzcd}$$

The Notion of Relevant Pairs

Definition

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 $\bar{b}_2$

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

Definition

Let the set of relevant pairs be defined as

- $\_ \times_R \_ : \bar{X} \to \bar{X} \to \bar{X} = (T, S, \bar{x}) \mapsto ((\_ \uparrow \bar{b}_1 : T \Uparrow \bar{x}) \times (\_ \uparrow \bar{b}_2 : S \Uparrow \bar{x}) \times Cov\ \bar{b}_1\ \bar{b}_2)$
- $\_ ,_R \_ : T \Uparrow \bar{x} \to S \Uparrow \bar{x} \to (T \times_R S) \Uparrow \bar{x}$
  $= ((t_1 \uparrow \bar{b}_1), (t_2 \uparrow \bar{b}_2)) \mapsto ((t_1 \uparrow \bar{b}_1'), (t_2 \uparrow \bar{b}_2'), \bar{b}_1 \oplus \bar{b}_2) \uparrow \bar{b}'$

Exploring Relevant Pairs

Remark

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

Example

Look at $\lambda x.\lambda y.\lambda z.z\ y = \lambda\lambda\lambda(\#0\ \#1)$.
The variable terms could also be represented as

- $z' : Tm \Uparrow 3 = \#0 \uparrow 001$
- $y' : Tm \Uparrow 3 = \#1 \uparrow 011$

And the application term could be a *relevant pair*

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

$$
\begin{tikzcd}
001 \arrow[dr, "01"'] \arrow[drr, "001"] & & \\
& \langle BR \arrow[r, "011"] & 111 \\
011 \arrow[ur, "11"] \arrow[urr, "011"'] & &
\end{tikzcd}
$$

Intrinsically Scoped co-De Bruijn Syntax

Definition

Let $Tm : \mathbb{N} \to Set$ be inductively defined:

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

Example

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

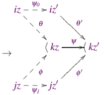
This is actually an Agda Paper

- All categorical concepts formalized in Agda
- Categorical concepts applied to formally reason over programming languages using theorem provers
- Suitable representations of objects and morphisms required
- Comes with a *universe of metasyntaxes with binding*


```
cop (θ o′) (ϕ o′)  = let ! ! ! ! tl , c , tr = cop θ ϕ in ! ! ! ! tl t-″  , c     , tr t-″
cop (θ o′) (ϕ os)  = let ! ! ! ! tl , c , tr = cop θ ϕ in ! ! ! ! tl t′s′ , c c′s , tr tsss
cop (θ os) (ϕ o′)  = let ! ! ! ! tl , c , tr = cop θ ϕ in ! ! ! ! tl tsss , c cs′ , tr t′s′
cop (θ os) (ϕ os)  = let ! ! ! ! tl , c , tr = cop θ ϕ in ! ! ! ! tl tsss , c css , tr tsss
cop    oz     oz   =                                      ! ! ! !   tzzz ,   czz ,   tzzz
```

The copU proof goes by induction on the triangles which share ψ' and inversion of the coproduct.

A further useful property of coproduct diagrams is that we can selectively refine them by a thinning into the covered scope.




```
subCop : (ψ : kz ⊑ kz′) → Cover ov θ′ ϕ′ →
         Σ _ λ iz → Σ _ λ jz → Σ (iz ⊑ kz) λ θ → Σ (jz ⊑ kz) λ ϕ →
         Σ (iz ⊑ iz′) λ ψ0 → Σ (jz ⊑ jz′) λ ψ1 → Cover ov θ ϕ
```

The implementation is a straightforward induction on the diagram.

The payoff from coproducts is the type of *relevant pairs* — the co-de-Bruijn touchstone:

```
record _×R_ (S T : K̅) (ijz : Bwd K) : Set where
  constructor pair
  field outl  : S ⇑ ijz;   outr : T ⇑ ijz
        cover : Cover tt (thinning outl) (thinning outr)
```

```
_,R_ : S ⇑ kz → T ⇑ kz → (S ×R T) ⇑ kz
(s ↑ θ) ,R (t ↑ ϕ) =
  let ! ψ , θ′ , ϕ′ , _ , c , _ = cop θ ϕ
  in  pair (s ↑ θ′) (t ↑ ϕ′) c ↑ ψ
```

The corresponding projections are readily definable.

```
outlR : (S ×R T) ⇑ kz → S ⇑ kz
outlR (pair s _ _ ↑ ψ) = thin⇑ ψ s
```

```
outrR : (S ×R T) ⇑ kz → T ⇑ kz
outrR (pair _ t _ ↑ ψ) = thin⇑ ψ t
```

7 Monoidal Structure of Order-Preserving Embeddings

Variable bindings extend scopes. The λ construct does just one ‘snoc’, but binding can be simultaneous, so the monoidal structure on Δ_+ induced by concatenation is what we need.

```
_++_ : Bwd K → Bwd K → Bwd K
kz ++ []       = kz
kz ++ (iz -, j) = (kz ++ iz) -, j
```

```
_++⊑_ : iz ⊑ jz → iz′ ⊑ jz′ → (iz ++ iz′) ⊑ (jz ++ jz′)
θ ++⊑    oz   = θ
θ ++⊑ (ϕ os) = (θ ++⊑ ϕ) os
θ ++⊑ (ϕ o′) = (θ ++⊑ ϕ) o′
```

Concatenation further extends to Coverings, allowing us to build them in chunks.

```
_++C_ : Cover ov θ ϕ → Cover ov θ′ ϕ′ → Cover ov (θ ++⊑ θ′) (ϕ ++⊑ ϕ′)
c ++C (d c′s)                = (c ++C d) c′s
c ++C (d cs′)                = (c ++C d) cs′
c ++C (_css {both = b} d)    = _css {both = b} (c ++C d)
c ++C czz                    = c
```

One way to build such a chunk is to observe that two scopes cover their concatenation.

Using co-De Bruijn is not hard, category theory is!

```
data Cov : (k l m : ℕ) → Set where
  · :                Cov 0       0       0
  L : Cov k l m → Cov (suc k) l       (suc m)
  R : Cov k l m → Cov k       (suc l) (suc m)
  B : Cov k l m → Cov (suc k) (suc l) (suc m)

data Term : ℕ → Set where
  #       : Term 1
  λ       : Term (suc n) → Term n
  _$[_]_ : Term k → Cov k l m → Term l → Term m

_ = λ (λ (λ ((# $[ L (R ·) ] #) $[ L (R (B ·)) ] (# $[ L (R ·) ] #))))
```

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.