Binding Structures

Raffi Sanna

What are Binding Structures?

Binding Structures are any syntactic constructs which introduce variables into the scopes of their subterms.

Examples

$$\lambda x.x$$

$$\forall z.P(z) \to Q(z)$$

$$\sum_{x \in A} x^2$$

The Untyped λ-Calculus

$$
\begin{aligned}
t :=\ & x \\
\mid\ & t_1\, t_2 \\
\mid\ & \lambda x.\, t
\end{aligned}
$$

Free and Bound Names

When a name appears as a subterm of a λ-term which uses the same name, then we say the name is **bound**. All non-bound names are **free**. A term without free variables is a **combinator**.

Free and Bound Names

$$\lambda x.(f\ (\lambda y.(g\ x)y))$$

Free and Bound Names

$$\lambda x.(f\ (\lambda y.(g\ x)y))$$

Free and Bound Names

$$\lambda x.(f\ (\lambda y.(g\ x)y))$$

Free and Bound Names

$$\lambda x.(f\ (\lambda y.(g\ x) y))$$

Free and Bound Names

$$\mathcal{FV}(t) = \begin{cases} \{x\} & \text{if } t = x \\ \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) & \text{if } t = t_1\ t_2 \\ \mathcal{FV}(t_1) - \{x\} & \text{if } t = \lambda x.t_1 \end{cases}$$

Free and Bound Names

$$\lambda x.((\lambda y.\,(x\,y))\;y)$$

Free and Bound Names

$$\lambda x.((\lambda y.(x\ y))\ y)$$

α-Equivalence

If you consistently rename a bound variable in a λ-term, it does not change the meaning of the term.

α-Equivalence

$$\lambda x.x$$
$$\equiv$$
$$\lambda y.y$$

α-Equivalence

$$\lambda x.f$$

$$\not\equiv$$

$$\lambda y.g$$

α-Equivalence

$$\lambda x.((\lambda y.(x\ y))\ (x\ y))$$

$$\stackrel{?}{\equiv}$$

$$\lambda x.((\lambda z.(x\ z))\ (x\ z))$$

α-Equivalence

$$\lambda x.((\lambda y.(x\ y))\ (x\ y))$$

$$\not\equiv$$

$$\lambda x.((\lambda z.(x\ z))\ (x\ z))$$

Capture-Avoiding Substitution

We write $t_1[x \leftarrow t_2]$ to mean the term t_1, but with all free instances of x replaced with t_2.

Capture-Avoiding Substitution

$$(x\ (\lambda y.(x\ (\lambda x.x))))[x \leftarrow R]$$

$$\equiv$$

$$(R\ (\lambda y.R\ (\lambda x.x)))$$

Capture-Avoiding Substitution

$$t_1[x \leftarrow t_2] = \begin{cases} t_2 & \text{if } t_1 = x \\ x_1 & \text{if } t_1 = x_1 \text{ and } x_1 \neq x \\ (t_3[x \leftarrow t_2]\ t_4[x \leftarrow t_2]) & \\ \quad \text{if } t_1 = (t_3\ t_4) & \\ \lambda x.t_3 & \\ \quad \text{if } t_1 = \lambda x.t_3 & \\ \lambda x_1.t_3[x \leftarrow t_2] & \\ \quad \text{if } t_1 = \lambda x_1.t_3 \text{ and } x_1 \notin \mathcal{FV}(t_2) & \end{cases}$$

Capture-Avoiding Substitution

$$(\lambda y.((\lambda x.(x\ z))\ x))[x \leftarrow R]$$

$$\equiv$$

?

Capture-Avoiding Substitution

$$(\lambda y.((\lambda x.(x\ z))\ x))[x \leftarrow R]$$

$$\equiv$$

$$(\lambda y.((\lambda x.(x\ z))\ R))$$

Edge Cases: Shadowing

$$\lambda x.\lambda x.x$$

Edge Cases: Shadowing

$$\lambda x.\lambda x.x$$

Edge Cases: Shadowing

If a name is bound twice, an instance of it is bound by the **closest** ancestor in the syntax tree.

Edge Cases: Shadowing

$$\lambda x.\lambda y.(f\ ((\lambda x.(g\ x))\ (\lambda z.(y\ x))))$$

Edge Cases: Shadowing

$$\lambda x.\lambda y.(f\ ((\lambda x.(g\ x))\ (\lambda z.(y\ x))))$$

Edge Cases: Free Substitution

$$(\lambda y.(x\ y))[x \leftarrow y]$$

$$\neq$$

$$(\lambda y.(y\ y))$$

Edge Cases: Free Substitution

$$(\lambda \alpha.(x\ \alpha))[x \leftarrow y]$$

$$\equiv$$

$$(\lambda \alpha.(y\ \alpha))$$

Edge Cases: Free Substitution

When the term being substituted contains a free variable, **any bound instances of the variable must be renamed**.

Edge Cases: Free Substitution

$$((\lambda x.(\lambda x.z))(\lambda y.z))[z \hookleftarrow x\ y]$$

$$\equiv$$

$$?$$

Edge Cases: Free Substitution

$$((\lambda x.(\lambda x.z))(\lambda y.z))[z \hookleftarrow x\ y]$$

$$\equiv$$

$$((\lambda \alpha.(\lambda \beta.(x\ y)))(\lambda \gamma.(x\ y)))$$

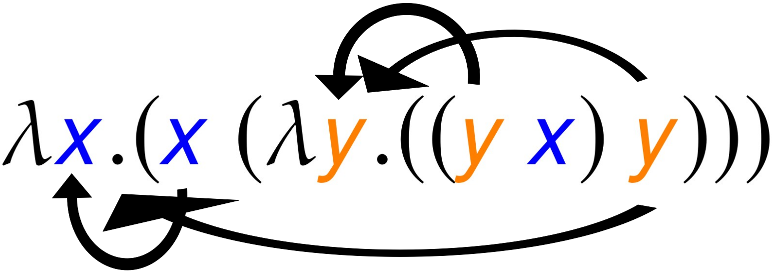
Implementing Bindings

- Manual α-Renaming
- De Brujin Indices
- Locally Nameless
- Higher-Order Abstract Syntax (HOAS)
- De Brujin Levels
- Nominal Logic
- Boxes Go Bananas
- Parameterized HOAS
- Scope Graphs
- …

De Brujin Indices

$$\lambda x.(x\ (\lambda y.((y\ x)\ y)))$$

De Brujin Indices



De Brujin Indices

$$\lambda.(0\ (\lambda.((0\ 1)\ 0)))$$

De Brujin Indices

The **De Brujin Index** of a variable reference is the number of λ-terms on the path from the reference to the binder.

De Brujin Indices

$$n \in \mathbb{N}$$

$$\begin{aligned} t := & \ n \\ & \mid t_1 \ t_2 \\ & \mid \lambda .\ t \end{aligned}$$

De Brujin Indices

$$\lambda x.(\lambda y.((x\ y)\ y))$$

De Brujin Indices

$$\lambda.(\lambda.((1\ 0)\ 0))$$

De Brujin Free Variables

$$\lambda y.((x\ y)\ y)$$

De Brujin Free Variables

$$\lambda.((1\ 0)\ 0)$$

De Brujin Free Variables

A De Brujin index is free if it is greater than the number of binders around it.

De Brujin Free Variables

$$\lambda.\lambda.((1\ 0)\ 0)$$

De Brujin Free Variables

$$\lambda x.(z\ (\lambda y.(x\ z)))$$

De Brujin Free Variables

$$\lambda.(1\ (\lambda.(1\ 2)))$$

De Brujin Free Variables

$$\mathcal{FV}(\lambda.(1\ (\lambda.(1\ 2))))$$

De Brujin Free Variables

$$\mathcal{FV}(\lambda.\lambda.(1\ (\lambda.(1\ 2))))$$

De Brujin Free Variables

$$\mathcal{FV}(\lambda.(1\ (\lambda.(1\ 2))))$$
$$= \{0\}$$

De Brujin Free Variables

$$\mathcal{FV}(t) =$$

$$\begin{cases} \{n\} & \text{if } t = n \\ \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) & \text{if } t = t_1\ t_2 \\ \{n - 1 | n \in \mathcal{FV}(t_1), n \geq 1\} & \text{if } t = \lambda.t_1 \end{cases}$$

De Brujin Free Variables

$$\mathcal{FV}(\lambda.((\lambda.(2\ 3))\ 3))$$

$$=?$$

De Brujin Free Variables

$$\mathcal{FV}(\lambda.((\lambda.(2\ 3))\ 3)) = \{0, 1, 2\}$$

De Brujin Substitutions

$$(\lambda.((\lambda.(2\ 3))\ 3))[0 \leftarrow\!\dashv R]$$

$$\equiv$$

$$(\lambda.((\lambda.(R\ 3))\ 3))$$

De Brujin Substitutions

We write $t_1[n \leftarrow\!\!\shortmid\, t_2]$ to mean the term t_1, but with all free instances of n replaced with t_2.

De Brujin Substitutions

$$(\lambda.((\lambda.(0\ 2))\ 2))[1 \leftarrow R]$$

$$\equiv$$

$$?$$

De Brujin Substitutions

$$(\lambda.((\lambda.(0\ 2))\ 2))[1 \hookleftarrow R]$$

$$\equiv$$

$$(\lambda.((\lambda.(0\ 2))\ R))$$

De Brujin Edge Cases

Shadowing
Free Substitutions

De Brujin Edge Cases

~~Shadowing~~

Free Substitutions

De Brujin Free Substitutions

$$(\lambda y.(x\ y))[x \leftarrow y]$$

$$\not\equiv$$

$$(\lambda y.(y\ y))$$

De Brujin Free Substitutions

$$\lambda.\lambda.(\lambda.(1\ 0))[0 \mapsfrom 1]$$

$$\not\equiv$$

$$\lambda.\lambda.(\lambda.(1\ 0))$$

De Brujin Free Substitutions

$$\lambda.\lambda.(\lambda.(1\ 0))[0 \leftarrow 1]$$

$$\equiv$$

$$\lambda.\lambda.(\lambda.(2\ 0))$$

De Brujin Free Substitutions

When the term being substituted contains a free variable, all references to the free variable must be **incremented by the index being replaced**.

De Brujin Lifting

We write "$\uparrow^n_k t$" to mean "increment all the references with indices at least k by n".

De Brujin Lifting

$$\uparrow_k^n t = \begin{cases} n_1 & \text{if } t = n_1 \text{ and } n_1 < k \\ n_1 + n & \text{if } t = n_1 \text{ and } n_1 \geq k \\ \uparrow_k^n t_1 \;\; \uparrow_k^n t_2 & \text{if } t = t_1\ t_2 \\ \lambda.\uparrow_{k+1}^n t_1 & \text{if } t = \lambda t_1 \end{cases}$$

De Brujin Substitutions

$$t_1[n \hookleftarrow t_2] =$$

$$\begin{cases} \uparrow_0^n t_2 & \text{if } t = n \\ n_1 & \text{if } t = n_1 \neq n \\ t_1[n \hookleftarrow t_2]\ t_2[n \hookleftarrow t_2] & \text{if } t = t_1\ t_2 \\ \lambda.t_1[n+1 \hookleftarrow t_2] & \text{if } t = \lambda.t_1 \end{cases}$$

De Brujin Substitutions

$$(\lambda.((1\ (\lambda.3))\ 2))[1 \mapsfrom 0]$$

$$\equiv$$

$$?$$

De Brujin Substitutions

$$(\lambda.((1\ (\lambda.3))\ 2))[1 \hookleftarrow 0]$$

$$\equiv$$

$$(\lambda.((1\ (\lambda.2))\ 1))$$

De Brujin Shortcomings

If you can reason about De Brujin indices, you're clearly not human.

Edwin Brady

De Brujin Shortcomings

$$\lambda.\lambda.(\lambda.(2\ 0))$$

De Brujin Shortcomings

$$\lambda.\lambda.(\lambda.(2\ 0))$$

Locally Nameless

Two bound variables are equal if they have the same binder, but two free variables are equal if they are spelled the same.

Locally Nameless

$$\lambda.(2\ 0) \equiv \lambda.(x\ 0)$$

Locally Nameless

$$
\begin{aligned}
n &\in \mathbb{N} \\
t &:= n \\
&\mid x \\
&\mid t_1\ t_2 \\
&\mid \lambda .\ t
\end{aligned}
$$

Locally Nameless

$$\lambda.(2\ (\lambda.(1\ 2)))$$

Locally Nameless

$$\lambda.(f\ (\lambda.(1\ g)))$$

Opening and Closing

$$\text{body}(\lambda.(f\ (\lambda.(1\ g))))$$

$$\equiv$$

?

Opening and Closing

$$\text{body}(\lambda.(f\ (\lambda.(1\ g))))$$

$$\neq$$

$$f\ (\lambda.(\underline{1}\ g))$$

Opening and Closing

$$\text{body}(\lambda.(f\ (\lambda.(1\ g))))$$

$$\equiv$$

$$f\ (\lambda.(x\ g))$$

Opening and Closing

When we want to reason about the body of a locally nameless λ-term, we must **open it** by assigning all references to a fresh name.

Opening and Closing

We write $\{n \rightarrow x\}t$ to mean the term t with all instances of the free index n replaced with the name x, where x is free in t.

Opening and Closing

$$\{n \longrightarrow x\}t = \begin{cases} x & \text{if } t = n \\ n_1 & \text{if } t = n_1 \neq n \\ x_1 & \text{if } t = x_1 \neq x \\ \{n \longrightarrow x\}t_1\ \{n \longrightarrow x\}t_2 & \text{if } t = t_1\ t_2 \\ \lambda.\{n+1 \longrightarrow x\}t_1 & \text{if } t = \lambda.t_1 \end{cases}$$

Opening and Closing

$$\{n \leftarrow x\}t =$$

$$\begin{cases} n & \text{if } t = x \\ x_1 & \text{if } t = x_1 \neq x \\ n_1 & \text{if } t = n_1 \\ \{n \leftarrow x\}t_1 \; \{n \leftarrow x\}t_2 & \text{if } t = t_1 \; t_2 \\ \lambda.\{n+1 \leftarrow x\}t_1 & \text{if } t = \lambda.t_1 \end{cases}$$

Opening and Closing

$$\{0 \rightarrow x\}(f\ (\lambda.(\underline{1}\ g)))$$

$$\equiv$$

$$f\ (\lambda.(x\ g))$$

Locally Nameless Substitution

$$t_1[x \hookleftarrow t_2] =$$

$$\begin{cases} t_2 & \text{if } t = x \\ x_1 & \text{if } t = x_1 \neq x \\ n_1 & \text{if } t = n_1 \\ t_1[x \hookleftarrow t_2]\ t_2[x \hookleftarrow t_2] & \text{if } t = t_1\ t_2 \\ \lambda.t_1[x+1 \hookleftarrow t_2] & \text{if } t = \lambda.t_1 \end{cases}$$

Locally Nameless Edge Cases

~~Shadowing~~

~~Free Substitutions~~

Comparison

	Renaming	De Brujin	LN
Names	✓	X	~
Shadowing	X	✓	✓
Free Sub.	X	X	✓
Structural	X	✓	✓
Inductive	✓	✓	X