Pre-terms and de Bruijn Terms

Gert Smolka, Saarland University

November 21, 2019

The prototypical example for formulas with variable binders and local variables are the terms of the untyped lambda calculus. There are two complementary formalizations known as pre-terms and de Bruijn terms. While pre-terms represent local variables with specific names following the usual notation, de Bruijn terms eliminate argument names and express argument references with the distance to the binder. We verify translations between pre-terms and de Bruijn terms. The translations preserve term structure and free variables, and provide for a natural definition of alpha equivalence.

1 Pre-terms

We obtain **pre-terms** with the following inductive definition:

$$M, N : \mathsf{preterm} \ ::= \ x \mid MN \mid \lambda x.M \qquad (x \in \mathbb{N})$$

We speak of **variables**, **applications**, and **abstractions**. An abstraction $\lambda x.M$ describes a function taking an argument x. Within the **body** M of the abstraction x is a **bound variable** providing a reference to the argument of the function described. For instance, the pre-term $\lambda x.x$ describes the identity function that simply returns its argument.

Bound variables may be understood as local variables that are only visible within the abstraction in which they are introduced. We speak of the **scope** of a bound variable and say that λ is a **variable binder**. A variable x is called **free in a pre-term** M if it appears in M at a position that is not in the scope of a binder λx. A pre-term is **open** if some variable occurs free in it, and **closed** if no variable occurs free in it. For instance, $\lambda x.x$ is a closed pre-term and $\lambda x.(fx)y$ is an open pre-term with two free variables f and y.

We adopt common notational conventions for pre-terms:

$$\begin{aligned} MNN' &\rightsquigarrow (MN)N' \\ \lambda xy.M &\rightsquigarrow \lambda x.\lambda y.M \\ \lambda x.MN &\rightsquigarrow \lambda x.(MN) \end{aligned}$$

With this conventions we can drop most parentheses:

$$\lambda fxy.\, fxy \quad\rightsquigarrow\quad \lambda f.(\lambda x.(\lambda y.((fx)y)))$$

2 De Bruijn Terms

While named argument references are essential for human readers, they are a nuisance for formal purposes. Formally, we would like to think of pre-terms that are equal up to renaming of local variables (e.g., $\lambda xy.fxy$ and $\lambda yx.fyx$) as the same object. Pre-terms that are equal up to renaming of local variables are called *α-equivalent*.

It turns out that there is an elegant term representation eliminating named local variables. One speaks of de Bruijn terms honoring the inventor Nicolaas de Bruijn [1]. We obtain **de Bruijn terms** with the following inductive definition:

$$s, t, u : \mathsf{term} \;::=\; x \mid st \mid \lambda s \qquad (x \in \mathbb{N})$$

With de Bruijn's term representation binders do not introduce bound variables. Reference to a binder is established with an *index* x saying how many binders must be skipped on the path to the binder. Here are examples of de Bruijn terms representing closed pre-terms:

$$\begin{aligned} \lambda fxy.\, fxy &\rightsquigarrow \lambda\,\lambda\,\lambda\; 2\; 1\; 0 \\ \lambda f.\, f(\lambda x.\, fx(\lambda y.\, fxy)) &\rightsquigarrow \lambda\; 0\,(\lambda\; 1\; 0\,(\lambda\; 2\; 1\; 0)) \end{aligned}$$

Here is an example of a de Bruijn term with free variables:

$$(\lambda xy.fyx)\,x \quad\rightsquigarrow\quad (\lambda\,\lambda\;(f{+}2)\;0\;1)\,x$$

An index x that is below d binders refers to one of the binders if $x < d$, and to the free variable $x - d$ if $x \geq d$.

We define an inductive predicate $\mathsf{bound}\ s\ n$ that holds if all free variables in s are smaller than n:

$$\frac{x < n}{\mathsf{bound}\ x\ n} \qquad \frac{\mathsf{bound}\ s\ n \qquad \mathsf{bound}\ t\ n}{\mathsf{bound}\ (st)\ n} \qquad \frac{\mathsf{bound}\ s\ \mathsf{S}n}{\mathsf{bound}\ (\lambda s)\ n}$$

We can now define **closed de Bruijn terms** as terms that are bound by 0:

$$\mathsf{closed}\ s \;:=\; \mathsf{bound}\ s\ 0$$

$$
\begin{aligned}
&\mathsf{B} : \ \mathsf{preterm} \to \mathcal{L}(\mathsf{N}) \to \mathsf{term} \\
&\qquad \mathsf{B}\, x\, C \ := \ \mathsf{B}'\, C\, x \\
&\quad \mathsf{B}\, (MN)\, C \ := \ (\mathsf{B} M C)(\mathsf{B} N C) \\
&\mathsf{B}\, (\lambda x.M)\, C \ := \ \lambda(\mathsf{B} M (x :: C)) \\
\\
&\mathsf{B}' : \ \mathcal{L}(\mathsf{N}) \to \mathsf{N} \to \mathsf{term} \\
&\qquad \mathsf{B}'\, []\, x \ := \ x \\
&\mathsf{B}'\, (y :: C)\, x \ := \ \begin{cases} 0 & \text{if } y = x \\ \mathsf{S}\, (\mathsf{B}'\, C\, x) & \text{if } y \neq x \end{cases}
\end{aligned}
$$

Figure 1: Translating pre-terms into de Bruijn terms

3 Translating Pre-terms to de Bruijn Terms

When we translate pre-terms to de Bruijn terms, we delete the local names appearing with abstractions and replace bound variable occurrences with the distance to their binder:

$$(\lambda x y.\, f\, y\, x)\, x \quad \rightsquigarrow \quad (\lambda\, \lambda\, (f{+}2)\ 0\ 1)\, x$$

Moreover, free variable occurrences are raised by the number of surrounding binders so that they can be distinguished from bound occurrences.

Figure 1 defines a recursive function B translating pre-terms to de Bruijn terms. The function is given a stack of bound variables, which is used to determine whether a variable occurrence is free or bound. If the variable appears in the stack, the occurrence is bound and the first position in the stack is the distance to the binder. If the variable does not appear in the stack, the occurrence is free and the length of the stack is the increment needed to skip the surrounding binders.

With B we can formally define **α-equivalence of pre-terms**:

$$M \approx_\alpha N \ := \ (\mathsf{B}\, M\, [] = \mathsf{B}\, N\, [])$$

This is the most elegant definition of α-equivalence of pre-terms we know of. Defining α-equivalence without using de Bruijn terms is possible but more involved.

4 Translating de Bruijn Terms to Pre-terms

When we translate de Bruijn terms to pre-terms, we have freedom in choosing the local variables. Of course, we need to ensure that we get the same de Bruijn term

$$
\begin{aligned}
&\mathsf{P}:\ \mathsf{N} \to \mathsf{N} \to \mathsf{term} \to \mathsf{preterm}\\
&\mathsf{P}\ n\ d\ x\ :=\ \begin{cases} n + d - \mathsf{S}x & \text{if } d > x \quad \textit{(x bound)}\\ x - d & \text{if } d \le x \quad \textit{(x free)} \end{cases}\\
&\mathsf{P}\ n\ d\ (st)\ :=\ (\mathsf{P}\ n\ d\ s)\ (\mathsf{P}\ n\ d\ t)\\
&\mathsf{P}\ n\ d\ (\lambda s)\ :=\ \lambda(n + d).\ \mathsf{P}\ n\ (\mathsf{S}d)\ s
\end{aligned}
$$

Figure 2: Translating de Bruijn terms into pre-terms

when we translate back. For instance, the translation

$$(\lambda\ \lambda\ (f\ 2)\ 0\ 1)\ x \quad \rightsquigarrow \quad (\lambda x y.\, f\, y\, x)\ x$$

is fine if x and y are chosen such that the three numbers f, x, y are different.

Figure 2 defines a function P satisfying the following correctness statement:

$$\forall s n.\ \ \mathsf{bound}\ s\ n\ \to\ \mathsf{B}\ (\mathsf{P}\ n\ 0\ s)\ []\ =\ s$$

The function P is given a bound n on the free variables in s and labels the binders it encounters with the variables $n, \mathsf{S}n, \mathsf{S}(\mathsf{S}n), \ldots$. The second argument d of P counts the number of binders traversed. This way P can distinguish between local and global references and replace them with the correct variables.

As it comes to the proof of the correctness statement, we need to generalize the statement so that an induction on the de Bruijn term s can go through. To do so, we need the list $[n + d - 1, \ldots, n]$ of bound variables:

$$
\begin{aligned}
&\mathsf{L}:\ \mathsf{N} \to \mathsf{N} \to \mathcal{L}(\mathsf{N})\\
&\mathsf{L}\ 0\ n\ :=\ []\\
&\mathsf{L}\ (\mathsf{S}d)\ n\ :=\ (n + d) :: \mathsf{L}\ d\ n
\end{aligned}
$$

Lemma 1

1. $x < n\ \to\ B'\ (L\ d\ n)\ x = x + d.$
2. $n \le x < n + d\ \to\ B'\ (L\ d\ n)\ x = n + d - \mathsf{S}x.$

Proof By induction on d. ∎

Lemma 2 $\mathsf{bound}\ s\ n\ \to\ \mathsf{B}\ (\mathsf{P}\ n\ d\ s)\ (\mathsf{L}\ d\ n) = s.$

Proof By induction on s with d quantified using Lemma 1 for the variable case. ∎

Theorem 3 $\mathsf{bound}\ s\ n\ \to\ \mathsf{B}\ (\mathsf{P}\ n\ 0\ s)\ [] = s.$

Proof Immediate with Lemma 2. ∎

References

[1] Nicolaas Govert De Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. In *Indagationes Mathematicae (Proceedings)*, volume 75, pages 381–392. Elsevier, 1972.