

Index lemmas for De Bruijn terms

type_theory debruijn design note

This note supports the debruijn design page. It fixes the definitions of shifting, index substitution and beta reduction implemented by `shift`, `substitute_bound` and `instantiate`, and proves the properties the page relies on: composition and commutation of shifts, safety of the downward shift in beta reduction, preservation of scope, and the substitution lemma.

1. Definitions

Terms are generated by $t ::= v \mid x \mid i \mid t \ u_1 \dots u_n \mid \lambda.t$ with values v , free names x and indices $i \in \mathbb{N}$. Values and free names are unaffected by every operation below; we omit their cases.

Definition. For $d \in \mathbb{Z}$ and $c \in \mathbb{N}$, the shift $\uparrow_c^d t$ is

$$\uparrow_c^d i = \begin{cases} i & \text{if } i < c \\ i + d & \text{if } i \geq c \end{cases}, \quad \uparrow_c^d (\lambda.t) = \lambda. \uparrow_{c+1}^d t, \quad \uparrow_c^d (t \ \bar{u}) = (\uparrow_c^d t) (\uparrow_c^d \bar{u}).$$

It is defined when no index becomes negative. The implementation carries the binder depth k instead of increasing c and tests $i \geq c + k$.

Definition. For $j \in \mathbb{N}$, the index substitution $[j \mapsto s]t$ is

$$[j \mapsto s]i = \begin{cases} s & \text{if } i = j \\ i & \text{otherwise} \end{cases}, \quad [j \mapsto s](\lambda.t) = \lambda.[j + 1 \mapsto \uparrow_0^1 s]t,$$

and homomorphic on applications. The implementation replaces, under k binders, the index $j + k$ by $\uparrow_0^k s$.

Definition. Beta reduction is $(\lambda.t) \ s \rightarrow \uparrow_0^{-1} ([0 \mapsto \uparrow_0^1 s]t)$. A term is *well scoped at depth n* if every index under k binders is smaller than $n + k$. We write $\text{fi}(t)$ for the set of free indices of t relative to its root: an index i under k binders contributes $i - k$ when $i \geq k$.

2. Shifts

Lemma 1 (identity and composition). $\uparrow_c^0 t = t$, and for $a, b \geq 0$, $\uparrow_c^a \uparrow_c^b t = \uparrow_c^{a+b} t$.

Proof. Induction on t . For an index i : if $i < c$ all three shifts leave it unchanged. If $i \geq c$ then $\uparrow_c^0 i = i$, and $i + b \geq c$ because $b \geq 0$, so $\uparrow_c^a \uparrow_c^b i = i + b + a = \uparrow_c^{a+b} i$. For $\lambda.t$ both sides move to cutoff $c + 1$ and the induction hypothesis applies; applications are immediate. $\square$

Lemma 2 (commutation). For $c \leq c'$ and $d, d' \geq 0$: $\uparrow_c^d \uparrow_{c'}^{d'} t = \uparrow_{c'+d}^{d'} \uparrow_c^d t$.

Proof. Induction on t ; the binder case increases c and c' together and keeps $c \leq c'$. For an index i there are three cases.

- $i < c$: both sides give i .
- $c \leq i < c'$: the left gives $\uparrow_c^d i = i + d$; the right gives $\uparrow_{c'+d}^{d'} (i + d) = i + d$ because $i + d < c' + d$.
- $i \geq c'$: the left gives $i + d' + d$; the right gives $i + d \geq c' + d$ and then $i + d + d'$.

$\square$

Lemma 3 (cancellation). $\uparrow_c^{-1} \uparrow_c^1 t = t$, and the left-hand side is always defined.

Proof. An index $i < c$ is untouched. An index $i \geq c$ becomes $i + 1 \geq c + 1 > c$ and then i , which is not negative. $\square$

3. Substitution

Lemma 4 (unfolding). $[j \mapsto s]t$ computed by the recursive definition equals the implementation's result: under k binders, $j + k$ is replaced by $\uparrow_0^k s$.

Proof. After passing k binders the recursive definition substitutes for index $j + k$ the term $\uparrow_0^1 \dots \uparrow_0^1 s$ (k shifts), which is $\uparrow_0^k s$ by Lemma 1. $\square$

Lemma 5 (free indices of a substitution). If t is well scoped at depth $n + 1$ and s at depth n , then $\text{fi}([0 \mapsto \uparrow_0^1 s]t) \subseteq \{1, \dots, n\}$.

Proof. We prove the stronger statement for every $k \geq 0$: if t is well scoped at depth $k + n + 1$, then every free index of $u_k = [k \mapsto \uparrow_0^{k+1} s]t$ lies in $\{0, \dots, k - 1\} \cup \{k + 1, \dots, k + n\}$. The case $k = 0$ is the lemma (using Lemma 4).

Induction on t . If $t = i$ with $i = k$, then $u_k = \uparrow_0^{k+1} s$, whose free indices are $j + k + 1$ for $j \in \text{fi}(s) \subseteq \{0, \dots, n - 1\}$, so they lie in $\{k + 1, \dots, k + n\}$. If $i \neq k$, then $u_k = i$ with $i < k + n + 1$ and $i \neq k$, which is in the stated set. Applications are immediate. For $t = \lambda.t'$, $u_k = \lambda.u'_{k+1}$ where t' is well scoped at depth $(k + 1) + n + 1$ and, by Lemma 1, $\uparrow_0^1 \uparrow_0^{k+1} s = \uparrow_0^{k+2} s$, so the induction hypothesis gives free indices of u'_{k+1} in $\{0, \dots, k\} \cup \{k + 2, \dots, k + 1 + n\}$. Removing the binder removes 0 and subtracts one: $\{0, \dots, k - 1\} \cup \{k + 1, \dots, k + n\}$. $\square$

Theorem 6 (safe instantiation, preservation of scope). If $\lambda.t$ and s are well scoped at depth n , then $\uparrow_0^{-1} ([0 \mapsto \uparrow_0^1 s]t)$ is defined and well scoped at depth n . Consequently `instantiate` and `reduce_once` never return a `ScopeError` on well-scoped input, and every reduct of a well-scoped term is well scoped.

Proof. By Lemma 5 every free index of the substituted body lies in $\{1, \dots, n\}$. The shift by -1 at cutoff 0 only touches free indices, mapping them into $\{0, \dots, n - 1\}$; bound indices stay below their cutoff. Hence no index becomes negative and the result is well scoped at depth n . A beta step in a context replaces a well-scoped subterm by a well-scoped subterm at the same depth, so scope is preserved globally. $\square$

Lemma 7 (substitution lemma). For $i < j$ and terms s, r : $[j \mapsto r]([i \mapsto s]t) = [i \mapsto [j \mapsto r]s]([j \mapsto r]t)$ whenever $i \notin \text{fi}(r)$.

Proof. Induction on t . For the index i : the left gives $[j \mapsto r]s$; the right leaves i by $[j \mapsto r]$ (as $i \neq j$) and then gives $[j \mapsto r]s$. For the index j : the left leaves j and gives r ; the right gives r and then $[i \mapsto \dots]r = r$ because $i \notin \text{fi}(r)$. Other indices are untouched on both sides. Under a binder both indices increase by one and both replacements are shifted by $\uparrow_0^1$; the side condition is preserved because $\text{fi}(\uparrow_0^1 r) = \text{fi}(r) + 1$, and $[j + 1 \mapsto \uparrow_0^1 r](\uparrow_0^1 s) = \uparrow_0^1 ([j \mapsto r]s)$ by the shift-substitution commutation, which is proved the same way as Lemma 2. $\square$

4. Relation to named syntax

Let $\ulcorner t \urcorner$ be the translation `from_named`, and $\ulcorner b \urcorner_x$ the translation of b in which x is the innermost binder.

Theorem 8 (translation commutes with beta). $\ulcorner b[x := a] \urcorner = \uparrow_0^{-1} ([0 \mapsto \uparrow_0^1 \ulcorner a \urcorner] \ulcorner b \urcorner_x)$, where $b[x := a]$ is capture-avoiding named substitution.

Proof. By induction on b , generalised to k inner binders between the occurrence and x , as in Lemma 5. An occurrence of x under k inner binders has index k in $\ulcorner b \urcorner_x$ and receives $\uparrow_0^{k+1} \ulcorner a \urcorner$, which after the final downward shift is $\uparrow_0^k \ulcorner a \urcorner$: the translation of a placed under those k binders, as the named substitution places a under the same binders (renamed if needed, which the translation does not see). Another bound variable keeps its distance to its binder; after removing the binder of x , variables bound outside it are one binder closer, which is the downward shift. Free names are unchanged. $\square$