Substitution in Lambda Calculus
Lecture Notes

Gert Smolka
Saarland University

December 13, 2015

We study the formalization of lambda calculus based on De Bruijn terms. The most interesting aspect is a system of substitution primitives and an accompanying equational theory providing for algebraic proofs. The equational theory can be presented as a confluent and terminating rewriting system providing for proof automation. We prove that parallel reduction is strongly substitutive, the key property needed for the proof of the Church-Rosser theorem.

1 The Problem

We consider De Bruijn terms

$$s, t \ ::= \ n \mid st \mid \lambda s \qquad (n : \mathbf{N})$$

and study the operation behind β-reduction:

$$(\lambda s)t \ \succ \ \beta st$$

As it turns out, there is a definition of β providing for elegant algebraic proofs of the relevant properties. The definition of β will be based on a cleverly designed system of substitution primitives due to Abadi, Cardelli, Curien, and Lévy [1].

Our starting point is the usual informal account of β-reduction using terms with bound variables:

$$(\lambda x.s)t \ \succ \ s^x_t$$

Recall that s^x_t is the term obtained from s by replacing all free occurrences of x with the term t. There is the proviso that the free variables of t must not be captured by binders in s. To make this work, bound variables in s may have to be renamed. Renaming is justified by the assumption that terms are equal if they are equal up to renaming of bound variables.

De Bruijn terms eliminate local variables and represent argument references as numbers called indices providing the distance to the binder (i.e., the number of lambdas to be skipped on the path to the root). Here are examples of De Bruijn terms and there named counterparts:

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

We now consider β-reduction:

$$(\lambda s)t \succ \beta st \quad\rightsquigarrow\quad (\lambda x.s)t \succ s^x_t$$

In the De Bruijn representation, the role of the bound variable x is taken by the index 0. The De Bruijn term βst may be obtained from s and t as follows:

1. In s, replace every free occurrence of the index 0 with the term t.
2. At every position where t replaces 0, raise the free indices in t by the number of lambdas above the position of the replacement.
3. Lower the free indices of s greater than 0 by 1 to account for the removal of the lambda above s.

For instance, the informal β-reduction

$$(\lambda y.\,x(\lambda z.y))(\lambda y.x) \succ (x(\lambda z.y))^y_{\lambda y.x} = x(\lambda zy.x)$$

should translate to the following De Bruijn β-reduction:

$$(\lambda(x{+}1)(\lambda 1))\;(\lambda(x+1)) \succ \beta\;((x{+}1)(\lambda 1))\;(\lambda(x+1)) = x(\lambda\lambda(x{+}2))$$

2 Substitutions and Instantiation

From the informal description of the operation βst it is clear that all free indices of s and t are affected by the operation, not just the index 0. We will therefore work with functions $\mathbf{N} \to \mathbf{Ter}$ mapping indices (i.e., numbers) to terms. We call such functions **substitutions**. The letters σ and τ will always denote substitutions. Here are two prominent substitutions:

$I := \lambda n.n$		**identity substitution**
$S := \lambda n.n+1$		**successor substitution**

Often, it is helpful to think of substitutions as infinite sequences:

$$\begin{aligned} I &\hat{=} (0, 1, 2, \dots) \\ S &\hat{=} (1, 2, 3, \dots) \\ \sigma &\hat{=} (\sigma 0, \sigma 1, \sigma 2, \dots) \end{aligned}$$

We define a **cons operation** that for a term and a substitution yields a substitution:

$$\begin{aligned} s.\sigma &:= \lambda n.\ \text{if } n{=}0 \text{ then } t \text{ else } \sigma(n-1) \\ &\hat{=} (s, \sigma 0, \sigma 1, \dots) \end{aligned}$$

We also need an **instantiation operation** $s[\sigma]$ for terms and substitutions, which replaces the free indices of s with the terms specified by σ, where the free indices of the inserted terms are raised by the number of lambdas above the insertion position. We characterize in the instantiation operation with three equations:

$$\begin{aligned} n[\sigma] &= \sigma n \\ (st)[\sigma] &= (s[\sigma])(t[\sigma]) \\ (\lambda s)[\sigma] &= \lambda(s[\Uparrow\sigma]) \end{aligned}$$

The operation $\Uparrow\sigma$ modifies σ such that it does the right thing below the lambda: 0 is mapped to 0, and $n+1$ is mapped to σn, where the free indices in σn are raised by 1 to account for the enclosing lambda. Formally, we define $\Uparrow\sigma$ and an accompanying composition operation $\sigma \circ \tau$ based on the instantiation operation:

$$\begin{aligned} \Uparrow\sigma &:= 0.(\sigma \circ S) && \textbf{up} \\ \sigma \circ \tau &:= \lambda n.\ (\sigma n)[\tau] && \textbf{composition} \end{aligned}$$

It turns out that we have arrived at a recursive definition of the instantiation operation $s[\sigma]$. We argue termination of $s[\sigma]$ in two steps. We call a substitution σ a **renaming** if σn is an atomic term (i.e., an index) for every number n. It is easy to see that $s[\sigma]$ terminates if σ is a renaming. Hence we know that $(\sigma \circ \tau)n$ terminates if τ is a renaming. Thus $(\Uparrow\sigma)n$ terminates for every substitution σ. Hence $s[\sigma]$ terminates for all terms and all substitutions.

We can now define the β-operation:

$$\begin{aligned} \beta s t &:= s[t.I] \\ &\hat{=} s[t, 0, 1, \dots] \end{aligned}$$

Note that the free indices in s greater than 0 are in fact lowered by 1.

Exercise 1 (Eta Expansion) Convince yourself that the following facts are true:

$$\begin{aligned}
\beta s t &= s[t.I] & \Uparrow\sigma &= 0.(\sigma \circ S)\\
(st)[\sigma] &= (s[\sigma])(t[\sigma]) & 0.S &= I\\
(\lambda s)[\sigma] &= \lambda(s[\Uparrow\sigma]) & \sigma \circ I &= \sigma\\
0[s.\sigma] &= s & I \circ \sigma &= \sigma\\
s[I] &= s & (\sigma \circ \tau) \circ \mu &= \sigma \circ (\tau \circ \mu)\\
s[\sigma][\tau] &= s[\sigma \circ \tau] & (s.\sigma) \circ \tau &= s[\tau].(\sigma \circ \tau)\\
& & S \circ (s.\sigma) &= \sigma\\
& & 0[\sigma].(S \circ \sigma) &= \sigma
\end{aligned}$$

Figure 1: Basic equations

a) The η-expansion of a term t is $\lambda((t[S])0)$.

b) A term s is an η-redex if and only if there is a term t such that $s = \lambda((t[S])0)$.

Exercise 2 (One-step reduction for $\lambda\beta\eta$) Give an inductive definition of one-step reduction $s \succ t$ for $\lambda\beta\eta$. Formulate the rule for η-reduction without a side condition (i.e., do not use "0 not free in s").

3 Basic Equations

We have introduced four substitution operations: cons $s.\sigma$, instantiation $s[\sigma]$, up $\Uparrow\sigma$, and composition $\sigma \circ \rho$. The properties of these operations can be expressed with equations. It turns out that relatively few equations suffice to show all other equations. Figure 1 shows a carefully chosen set of equations we call **basic equations**. All basic equations are valid (Fact 6). All other equations formulated with the primitives used in Figure 1 can be shown valid by left-to-right rewriting with the basic equations [4]. It turns out that rewriting with basic equations is confluent and terminating [2]. Fact 7 states an interesting equation whose validity follows with left-to-right rewriting with the basic equations.

To establish the validity of the basic equations for substitutions, we assume that substitutions are **extensional**, that is, $\sigma = \tau$ whenever $\sigma n = \tau n$ for all n. The extensionality assumption simplifies our presentation but can be removed in principle.

We prove that the basic equations are valid. As it turns out, only the equations $s[I] = s$ and $s[\sigma][\tau] = s[\sigma \circ \tau]$ need inductive proofs, the other basic equations have straightforward validity proofs given the validity of these two equations.

Fact 3 $\Uparrow I = I$.

Proof By extensionality it suffices to show $(\Uparrow I)n = In$. This follows by case analysis $n = 0$ or $n = n + 1$. ■

Fact 4 $s[I] = s$.

Proof By induction on s using Fact 3 for the abstraction case. ■

We now come to the validity proof for $s[\sigma][\tau] = s[\sigma \circ \tau]$. The amazingly tricky proof is summarized by the following fact.

Fact 5 Let ξ be a renaming.

1. $S \circ \Uparrow\tau \; = \; \tau \circ S$.
2. $\Uparrow\xi \circ \Uparrow\sigma \; = \; \Uparrow(\xi \circ \sigma)$.
3. $s[\xi][\tau] \; = \; s[\xi \circ \tau]$.
4. $\Uparrow\sigma \circ \Uparrow\xi \; = \; \Uparrow(\sigma \circ \xi)$.
5. $s[\sigma][\xi] \; = \; s[\sigma \circ \xi]$.
6. $\Uparrow\sigma \circ \Uparrow\tau \; = \; \Uparrow(\sigma \circ \tau)$.
7. $s[\sigma][\tau] \; = \; s[\sigma \circ \tau]$.

Proof 1. By extensionality.

2. By extensionality and case analysis on n.
3. By induction on s using (2) for the abstraction case.
4. By extensionality and case analysis on n. For $n = n + 1$, (3) reduces the claim to instance of (1).
5. By induction on s using (4) for the abstraction case.
6. By extensionality and case analysis on n. For $n = n + 1$, (3) and (5) reduce the claim to an instance of (1).
7. By induction on s using (6) for the abstraction case. ■

Fact 6 (Validity) All basic equations are valid.

Proof The proof of most of the equations is straightforward, some of them hold by definition. There are two hard equations, $s[I] = s$ and $s[\sigma][\tau] = s[\sigma \circ \tau]$, which require inductive proofs, and whose validity was established with Facts 4 and 5, respectively. There are two further non-obvious equations, $\sigma \circ I = \sigma$ and $(\sigma \circ \tau) \circ \mu = \sigma \circ (\tau \circ \mu)$. They follow with extensionality from the two hard equations. ■

Fact 7 (Distributivity) $(\beta st)[\sigma] \; = \; \beta(s[\Uparrow\sigma])(t[\sigma])$.

Proof Both sides normalize to $s[t[\sigma].\sigma]$ with the basic equations. ■

Remark The basic equations

$$S \circ (s.\sigma) \;=\; \sigma \qquad\qquad 0[\sigma].(S \circ \sigma) \;=\; \sigma$$

look surprising at first. They appear more natural when written as

$$\mathsf{tl}(s.\sigma) = \sigma \qquad\qquad \mathsf{hd}\,\sigma.\,\mathsf{tl}\,\sigma = \sigma$$

where the head and tail operations for substitutions are defined as follows:

$$\begin{aligned} \mathsf{hd}\,\sigma \;&:=\; \sigma 0 \\ \mathsf{tl}\,\sigma \;&:=\; \lambda n.\sigma(n+1) \;\hat{=}\; (\sigma 1, \sigma 2, \sigma 3, \ldots) \end{aligned}$$

We have $0[\sigma] = \mathsf{hd}\,\sigma$ and $S \circ \sigma = \mathsf{tl}\,\sigma$. So you may read $S \circ \sigma$ as $\mathsf{tl}\,\sigma$.

Exercise 8 Prove the following equations by normalizing with the basic equations:

a) $s[\sigma][S] = s[S][\Uparrow\sigma]$.

b) $s[\Uparrow\sigma][t.I] = s[t.\sigma]$.

c) $s[\Uparrow\sigma][t[\sigma].I] = s[t.I][\sigma]$.

Exercise 9 Define $S^0 := I$ and $S^{n+1} := S \circ S^n$. Prove $n = 0[S^n]$. Note that this means that all terms can be expressed with the primitives occurring in the basic equations. It also means that the basic equations constitute an instantiation algorithm.

Exercise 10 Let $s = s[0[S].S]$. Convince yourself that the assumption says that 0 is not free in s. Prove $s[t.(S \circ \sigma)] = s[\sigma]$ by rewriting with the assumption and the basic equations.

Exercise 11 (Redundancy of Basic Equations) Derive the basic equations

$$s[I] = s \qquad\qquad s[\sigma][\tau] = s[\sigma \circ \tau] \qquad\qquad 0[\sigma].(S \circ \sigma) = \sigma$$

from the remaining basic equations. Note that left-to-right rewriting with the remaining equations does not suffice anymore. The trick is to use the basic equations

$$0[s.\sigma] \;=\; s \qquad\qquad (s.\sigma) \circ \tau \;=\; s[\tau].(\sigma \circ \tau)$$

from right-to-left.

Exercise 12 (Head and Tail) Consider the following equations:

$$\begin{aligned} \mathsf{hd}\,(s.\sigma) &= s & \mathsf{tl}\,(s.\sigma) &= \sigma & \mathsf{hd}\,\sigma.\mathsf{tl}\,\sigma &= \sigma \\ (st)[\sigma] &= (s[\sigma])(t[\sigma]) & \sigma \circ I &= \sigma & (\sigma \circ \tau) \circ \mu &= \sigma \circ (\tau \circ \mu) \\ (\lambda s)[\sigma] &= \lambda(s[\mathsf{hd}\,I.(\sigma \circ \mathsf{tl}\,I)]) & I \circ \sigma &= \sigma & (s.\sigma) \circ \tau &= s[\tau].(\sigma \circ \tau) \end{aligned}$$

a) Express 0, S, βst, and $\Uparrow\sigma$ with the primitives used in the equations.

b) Prove that the equations follow from the basic equations if $\mathsf{hd}\,\sigma$ and $\mathsf{tl}\,\sigma$ are seen as abbreviations for $0[\sigma]$ and $S \circ \sigma$.

c) Prove that the following equations follow from the above equations.

$$\begin{aligned} s[\sigma][\tau] &= s[\sigma \circ \tau] &\qquad (\mathsf{hd}\,\sigma)[\tau] &= \mathsf{hd}\,[\sigma \circ \tau] \\ s[I] &= s & \mathsf{tl}\,\sigma \circ \tau &= \mathsf{tl}\,(\sigma \circ \tau) \end{aligned}$$

d) Prove that the basic equations follow from the above equations if 0, S, βst, and $\Uparrow\sigma$ are seen as abbreviations.

This exercise is based on work of Tobias Tebbi and Steven Schäfer.

4 Strong Substitutivity of Parallel Reduction

To show the confluence of $\lambda\beta$, we need to show that the operation β is compatible with parallel reduction (Corollary 16). This is a special case of the fact that parallel reduction is strongly substitutive (Theorem 15). Given our setup for substitution, proving strong substitutivity of parallel reduction is routine. All we need to know about substitution are the basic equations and the distributivity law for β (Fact 7).

Recall the definition of **parallel reduction** $s \gg t$:

$$\frac{s \gg s' \qquad t \gg t'}{(\lambda s)t \gg \beta s't'} \qquad\qquad \frac{}{x \gg x} \qquad\qquad \frac{s \gg s' \qquad t \gg t'}{st \gg s't'} \qquad\qquad \frac{s \gg s'}{\lambda s \gg \lambda s'}$$

Fact 13 (Substitutivity) If $s \gg t$, then $s[\sigma] \gg t[\sigma]$.

Proof By induction on $s \gg t$ using Fact 7 for the beta rule. ∎

We extend the definition of parallel reduction to substitutions (pointwise):

$$\sigma \gg \tau \;:=\; \forall n.\, \sigma n \gg \tau n$$

Fact 14 (Compatibility) If $\sigma \gg \tau$, then $\Uparrow\sigma \gg \Uparrow\tau$.

Proof Let $\sigma \gg \tau$. We show $(0.(\sigma \circ S))n \gg (0.(\tau \circ S))n$. For $n = 0$ the claim is trivial. For $n = n + 1$ it suffices to show $(\sigma \circ S)n \gg (\tau \circ S)n$. By definition of $\circ$ it suffices to show $(\sigma n)[S] \gg (\tau n)[S]$, which follows by Fact 13 and the assumption. ∎

Theorem 15 (Strong Substitutivity) If $s \gg t$ and $\sigma \gg \tau$, then $s[\sigma] \gg t[\tau]$.

Proof By induction on $s \gg t$ using Fact 14 for the lambda rule and Facts 7 and 14 for the beta rule. ∎

Corollary 16 If $s \gg s'$ and $t \gg t'$, then $\beta st \gg \beta s't'$.

5 Strong Substitutivity of Star Reduction

We use the opportunity and show that star reduction is strongly substitutive. The proof is routine and has much in common with the strong substitutivity for parallel reduction. The main fact needed for substitutions is once more Fact 7 (see Fact 18).

Recall the definition of **one step reduction $s \succ t$**:

$$\frac{}{(\lambda s)t \succ \beta st} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'} \qquad \frac{s \succ s'}{\lambda s \succ \lambda s'}$$

Fact 17 (Substitutivity) If $s \succ t$, then $s[\sigma] \succ t[\sigma]$.

Proof By induction on $s \succ t$ using Fact 7 for the beta rule. ∎

Fact 18 (Substitutivity) If $s \succ^* t$, then $s[\sigma] \succ^* t[\sigma]$.

Proof By induction on $s \succ^* t$ using Fact 17. ∎

We extend the definition of star reduction to substitutions (pointwise):

$$\sigma \succ^* \tau \;:=\; \forall n.\, \sigma n \succ^* \tau n$$

Fact 19 (Compatibility) If $\sigma \succ^* \tau$, then $\Uparrow\sigma \succ^* \Uparrow\tau$.

Proof Let $\sigma \succ^* \tau$. We show $(0.(\sigma \circ S))n \succ^* (0.(\tau \circ S))n$. For $n = 0$ the claim is trivial. For $n = n + 1$ it suffices to show $(\sigma \circ S)n \succ^* (\tau \circ S)n$. By definition of $\circ$ it suffices to show $(\sigma n)[S] \succ^* (\tau n)[S]$, which follows by Fact 18 and the assumption. ∎

Fact 20 (Compatibility) Let $s \succ^* s'$. Then $\lambda s \succ^* \lambda s'$.

Proof By induction on $s \succ^* s'$. ∎

Fact 21 (Compatibility) Let $s \succ^* s'$ and $t \succ^* t'$. Then $st \succ^* s't'$.

Proof By induction on $t \succ^* t'$ with nested induction on $s \succ^* s'$. ∎

Fact 22 (Substitutivity) If $\sigma \succ^* \tau$, then $s[\sigma] \succ^* s[\tau]$.

Proof By induction on s using Facts 19, 20 and 21. ∎

Theorem 23 (Strong Substitutivity) If $s \succ^* t$ and $\sigma \succ^* \tau$, then $s[\sigma] \succ^* t[\tau]$.

Proof Follows with transitivity of $\succ^*$ and Facts 18 and 22. ∎

6 Free Variables

All results so far have been obtained without using the notions of free variables and closed terms. There are different ways to define freeness and closedness. The following exercises formulate some ideas and ask for proofs. Be warned: Some of the proofs asked for may need extra lemmas not stated in the exercises.

Exercise 24 (Free Indices) We define the free indices of a term with an inductive predicate:

$$\frac{}{\mathsf{free}\ n\ n} \qquad \frac{\mathsf{free}\ n\ s}{\mathsf{free}\ n\ (st)} \qquad \frac{\mathsf{free}\ n\ t}{\mathsf{free}\ n\ (st)} \qquad \frac{\mathsf{free}\ (n+1)\ s}{\mathsf{free}\ n\ (\lambda s)}$$

Prove that $\neg\mathsf{free}\ n\ s$ if and only if $s[1.S] = s$.

Exercise 25 (Closed Terms) We define an inductive predicate $\mathsf{dclosed}\ n\,s$ on numbers and terms:

$$\frac{n < d}{\mathsf{dclosed}\ d\ n} \qquad \frac{\mathsf{dclosed}\ d\ s \qquad \mathsf{dclosed}\ d\ t}{\mathsf{dclosed}\ d\ (st)} \qquad \frac{\mathsf{dclosed}\ (d+1)\ s}{\mathsf{dclosed}\ d\ (\lambda s)}$$

Show that the following statements are equivalent:

1. $\mathsf{dclosed}\ 0\,s$
2. $\forall n.\ \neg\mathsf{free}\ n\,s$
3. $\forall \sigma.\ s[\sigma] = s$
4. $s[S] = s$

Exercise 26 (Naive Substitution) Consider the naive substitution operation s^0_t:

$$\begin{aligned} n^k_u &:= \text{if } n = k \text{ then } u \text{ else } n \\ (st)^k_u &:= s^k_u\, t^k_u \\ (\lambda s)^k_u &:= \lambda(s^{k+1}_u) \end{aligned}$$

Let $(\lambda s)t$ be closed. Prove $\beta st = s^0_t$.

7 Realization in Coq

Coq's structural recursion does not admit a direct recursive definition of the instantiation operation. The problem can be solved by defining instantiation in two steps: First a specialized version for renamings and then a general version using the specialized version. This impairs some overhead on the proofs of Facts 4 and 5. The

Coq library Autosubst [5] can generate all substitution operations and the proofs of the accompanying basic equations automatically, given an inductive definition of terms where binders are marked. Autosubst comes with a tactic *asimpl*, which automatically proves equations formed with the primitives used in the basic equations.

Autosubst assumes functional extensionality for substitutions, as we did in this presentation. It is possible to prove strong substitutivity of parallel and star reduction without the extensionality assumption. To do so, one uses equivalence of substitutions rather than equality and proves that all operations on substitutions are invariant under equivalence.

8 Notes

The nameless representation of the λ-calculus was invented by De Bruijn [3] for the computer implementation of a proof checker for the mathematical language Automath. De Bruijn [3] gives the recursive definition of the instantiation operation given in these notes. He uses informal notation and does not use explicit operations for cons and composition. De Bruijn [3] argues that the nameless representation of the λ-calculus is advantageous for the proof of the Church-Rosser theorem.

The proof assistant Coq is implemented with a De Bruijn representation of terms.

The system of substitution primitives used in these notes and the accompanying equational theory are from the theory of explicit substitutions [1, 2], which developed in the 1990's. Curien, Hardin, and Levy [2] study a confluent and terminating rewriting system (the σ_{SP}*-calculus*) for expressions

$$\begin{aligned} M, N &::= 0 \mid MN \mid \lambda M \mid M[A] \mid X \\ A, B &::= I \mid S \mid M.A \mid A \circ B \mid \xi \end{aligned}$$

describing terms and substitutions. The letters X and ξ represent variables ranging over terms and substitutions, respectively. Our basic equations are the rewriting rules of the σ_{SP}-calculus, up to minor details. Schäfer, Smolka, and Tebbi [4] show that the σ_{SP}-calculus decides all valid equations $M = N$ and $A = B$.

The idea to use the substitution primitives and the basic equations for a formal development of the λ-calculus seems to appear first in the work of Autosubst [5].

References

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *J. Funct. Program.*, 1(4):375–416, 1991.

[2] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *J. ACM*, 43(2):362–397, 1996.

[3] Nicolaas Govert de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)*, 34:381–392, 1972.

[4] Steven Schäfer, Gert Smolka, and Tobias Tebbi. Completeness and decidability of de bruijn substitution algebra in Coq. In *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 67–73. ACM, 2015.

[5] Steven Schäfer, Tobias Tebbi, and Gert Smolka. Autosubst: Reasoning with De Bruijn terms and parallel substitutions. In Xingyuan Zhang and Christian Urban, editors, *Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015*, LNAI. Springer-Verlag, Aug 2015.