De Bruijn's $C\lambda\xi\phi$, an early calculus of explicit substitutions

Pierre LESCANNE
Centre de Recherche en Informatique de Nancy (CNRS)
and INRIA-Lorraine
Campus Scientifique, BP 239,
F54506 Vandœuvre-lès-Nancy, France

email: Pierre.Lescanne@loria.fr

October 13, 1994

In a 1978 paper [6], N. G. de Bruijn presents the first calculus of explicit substitutions which he calls $C\lambda\xi\phi$. As his notations are somewhat difficult to read and slightly different of this today people are used to, we propose to describe his rules in contemporary notations borrowed from [1, 2, 4, 10, 9]. See [8, 7] for other notations. First let us remind the unfamiliar reader with de Bruijn's indices [5]. The first idea that comes in mind if one wants to avoid explicit naming of variables is to draw pictures. One replaces the variables by a dummy names a box $\square$ and one draws a physical link (a line) between the variable and its binders. In Figure 1 we have represented a few terms. This is exactly the approach proposed by N. Bourbaki [3]. De Bruijn follows the same idea, for him variables are natural numbers, the *indices*. The index of a variable is the number of λ one crosses before getting to the one that binds that variable. For instance in $\lambda x \lambda y \lambda z \cdot x$ the index of the only occurrence of x is $\underline{3}$ and in the notation of λ-terms with indices, x will be replace by $\underline{3}$. The indices allows us to associate directly a variable (an index) with its binder, therefore there is no need for the name of a variable next to each λ. Thus from a term a one creates an *abstraction* by adding just a λ on the front of a. For instance, $\lambda\underline{1}$ is equivalent to $\lambda x \cdot x$ in usual λ-calculus, $\lambda\underline{1}\lambda\underline{2}\underline{1}$ is equivalent to $\lambda x \cdot x\ \lambda y \cdot y\ x$ and $\lambda\lambda\lambda\underline{3}$ is equivalent to $\lambda x \lambda y \lambda z \cdot x$.

Substitution plays the main role in λ-calculus and is usually described as an external calculus, an epi-theory. The idea of explicit substitutions is to *internalize* the calculus of substitutions that is to include the substitution mechanism as first citizen inside the λ-calculus. For that, there are notations for substitutions and rules to manipulate the substitutions. In what follows we have, in fact, two presentations of the rules. One is close to de Bruijn with the genuine names for the rules. It is intended for the reader who wants to read de Bruijn's original work. The other is in a contemporary system of notations with contemporary names for rules whenever the same rule has been rediscovered since that time. The starting point is a weak form of the β-rule, i.e., a rule which replaces a term of the form $(\lambda a)b$ by an explicit substitution. The role of the other rules will be to get rid of the new created substitutions that is to

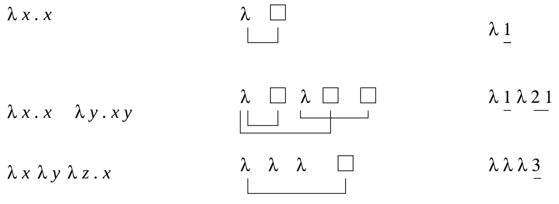

Figure 1: Bourbaki's style notations

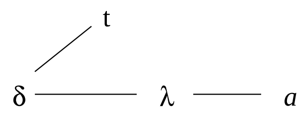

Figure 2: De Bruijn's schema for $a[t/]$

perform an actual substitution. According to de Bruijn, one of the advantages of such a calculus of explicit substitutions is to allow *small steps* in the calculus, leaving much flexibility in the order of evaluation of a term.

De Bruijn distinguishes two kinds of substitutions: substitutions that rename variables and substitutions that assign terms to variables. The substitutions of the first kind are associated with functions $\theta : \mathbb{N} \rightarrow \mathbb{N}$. Calculus of explicit substitution proposes a notation for representing those functions, and it is important to distinguish a function from its associated explicit substitution. The explicit substitution associated with a function θ will be written $\underline{\theta}$. Actually de Bruijn uses $\xi(n)$ for our $\underline{n}$ and $\phi(\theta)$ for our $\underline{\theta}$, hence the name $C\lambda\xi\phi$. Among those functions de Bruijn considers a function which he names θ_2 and which corresponds to:

$$\begin{array}{rccc} \theta_2 : & \underline{1} & \mapsto & \underline{2} \\ & \underline{2} & \mapsto & \underline{1} \\ & \underline{3} & \mapsto & \underline{3} \\ & & \vdots & \\ & \underline{n+2} & \mapsto & \underline{n+2} \\ & & \vdots & \end{array}$$

To include this substitution in contemporary notations, we propose to write $\underline{\theta_2}$ as $\updownarrow$ and to call it a *transposition*. The behavior of $\updownarrow$ can be described by its effect on indices as follows:

$$\begin{array}{lrcl} (LTheta_2) & \underline{1}[\updownarrow] & \rightarrow & \underline{2} \\ (RLTheta_2) & \underline{2}[\updownarrow] & \rightarrow & \underline{1} \\ (RRTheta_2) & \underline{n+2}[\updownarrow] & \rightarrow & \underline{n+2} \end{array}$$

De Bruijn's rules for $\underline{\theta}$ are

$$\begin{array}{lrcl} (A_1) & c[\underline{\theta}] & \rightarrow & c \\ (A_2) & \underline{n}[\underline{\theta}] & \rightarrow & \underline{\theta(n)} \\ (A_4) & (f\ a)[\underline{\theta}] & \rightarrow & f(a[\underline{\theta}]) \\ (A_6) & a[\underline{\theta}][\underline{\theta'}] & \rightarrow & a[\underline{\theta' \cdot \theta}] \\ (A_7) & (\lambda a)[\underline{\theta}] & \rightarrow & \lambda(a[\underline{L(\theta)}]) \\ (A_8) & (g\ a\ b)[\underline{\theta}] & \rightarrow & g\ (a[\underline{\theta}]\ b[\underline{\theta}]) \end{array}$$

where $L(\theta)(1) = 1$ and $L(\theta)(n+1) = \theta(n) + 1$, and $\theta' \cdot \theta(n) = \theta'(\theta(n))$. De Bruijn considers constant functions, e.g., c of arity 0, f of arity 1, and g of arity 2. (A_9) is a rule scheme which is just a generalization of (A_1), (A_4), and (A_8) to functions of arity $n = 3, \ldots$ Rules (A_3) and (A_5) are omitted purposely since they are not relevant here. Actually in (A_6) and (A_7), $\underline{\theta' \cdot \theta}$ and $\underline{L(\theta)}$ are defined directly on the underlying functions. L is just the *Lift* operation that is written $\Uparrow$ in contemporary notations and $\cdot$ is the composition written $\circ$ in contemporary notations. Notice that the composition introduced in rule (A_6) is not used elsewhere and is not necessary for a complete definition. In contemporary notations $\underline{\theta}$'s correspond to substitutions of the form $\Uparrow^i(\uparrow)$ and $\Uparrow^i(\updownarrow)$.

The second kind of substitutions are these of the form $t/$. De Bruijn writes $a[t/]$ as $\delta(t, \lambda(a))$ more precisely he uses the picture of Figure 2 which makes the rules somewhat difficult to retrieve. He gives the following set of rules.

$$
\begin{array}{llcl}
(\mathrm{B}) & (\lambda a)b & \rightarrow & a[b/] \\
(\mathrm{App}_1) & (a\ b)[c/] & \rightarrow & a[c/]\ b[c/] \\
(\mathrm{Lambda}_1) & (\lambda a)[b/] & \rightarrow & \lambda(a[\![\updownarrow]\!][b[\![\uparrow]\!]/]) \\
(\mathrm{FVar}) & \underline{1}[a/] & \rightarrow & a \\
(\mathrm{RVar}) & \underline{n+1}[a/] & \rightarrow & \underline{n} \\
(\mathrm{App}_2) & (a\ b)[\![s]\!] & \rightarrow & a[\![s]\!]\ b[\![s]\!] \\
(\mathrm{Lambda}_2) & (\lambda a)[\![s]\!] & \rightarrow & \lambda(a[\![\Uparrow(s)]\!]) \\
(\mathrm{FVarLift}) & \underline{1}[\![\Uparrow(s)]\!] & \rightarrow & \underline{1} \\
(\mathrm{RVarLift}) & \underline{n+1}[\![\Uparrow(s)]\!] & \rightarrow & \underline{n}[\![s]\!][\![\uparrow]\!] \\
(\mathrm{VarShift}) & \underline{n}[\![\uparrow]\!] & \rightarrow & \underline{n+1} \\
(\mathrm{LTheta}_2) & \underline{1}[\![\updownarrow]\!] & \rightarrow & \underline{2} \\
(\mathrm{RLTheta}_2) & \underline{2}[\![\updownarrow]\!] & \rightarrow & \underline{1} \\
(\mathrm{RRTheta}_2) & \underline{n+2}[\![\updownarrow]\!] & \rightarrow & \underline{n+2}
\end{array}
$$

Figure 3: The rewrite system $\lambda\xi\phi$

$$
\begin{array}{lrcl}
(B_1) & c[t/] & \rightarrow & c \\
(B_2) & \underline{n+1}[t/] & \rightarrow & \underline{n} \\
(B_3) & \underline{1}[t/] & \rightarrow & t \\
(B_4) & a[\uparrow][t/] & \rightarrow & a \\
(B_6) & (f\ a)[t/] & \rightarrow & f(a[t/]) \\
(B_7) & (\lambda a)[t/] & \rightarrow & \lambda(a[\updownarrow][t[\uparrow]/]) \\
(B_{10}) & g\ (a\ b)[t/] & \rightarrow & g(a[t/]\ b[t/])
\end{array}
$$

As above, (B_{11}) is a rule scheme which is just a generalization of (B_1), (B_6) and (B_{10}). Likewise, rules (B_5) and (B_8) are omitted purposely since they are not relevant here.

This system inspires us a calculus of explicit substitutions which we call $\lambda\xi\phi$ (Figure 3). Let us call $\mathbf{Terms}_{\xi\phi}$ the set of terms described by the grammar:

$$
\begin{array}{lcll}
\mathbf{Terms}_{\xi\phi} & a & ::= & \underline{n} \mid ab \mid \lambda a \mid a[\![s]\!] \mid a[t/] \\
\mathbf{Substitutions} & s & ::= & \Uparrow(s) \mid \uparrow \mid \updownarrow \\
\mathbf{Naturals} & n & ::= & n+1 \mid 1.
\end{array}
$$

We call $\xi\phi$ the system $\lambda\xi\phi\backslash(B)$, $\xi\phi$ can be shown to be strongly normalizing by using the lexicographic products $(<_\iota, <_{\kappa_1}, <_{\kappa_2})$. $<_\iota$ is defined by the interpretation $\iota : \mathbf{Terms}_{\xi\phi} \rightarrow \mathbf{Terms}_{\xi}$ where $\mathbf{Terms}_{\xi}$ is described by the grammar:

$$
\begin{array}{lcll}
\mathbf{Terms}_{\xi} & a & ::= & \underline{n} \mid ab \mid \lambda a \mid a[t/] \\
\mathbf{Naturals} & n & ::= & n+1 \mid 1.
\end{array}
$$

and ι is described as follows:

$$
\begin{array}{rcl}
\iota(\underline{n}) & = & \underline{1} \\
\iota(a\ b) & = & \iota(a)\ \iota(b) \\
\iota(\lambda a) & = & \lambda(\iota(a)) \\
\iota(a[\![s]\!]) & = & \iota(a) \\
\iota(a[b/]) & = & \iota(a)[\iota(b)/]
\end{array}
$$

$a <_\iota b$ if and only if $\iota(a) <_\xi \iota(b)$ where $<_\xi$ is a lexicographic path ordering described in $\mathbf{Terms}_\xi$ by the precedence that says that an abstraction is less than a closure and less than an application which could pictured by the following inequalities $\lambda < \_[\_/]$ and $\lambda < \_\ \_$.

κ_1 and κ_2 are interpretations from $\mathbf{Terms}_{\xi\phi}$ to the set of elementary functions over $\mathbb{N}$. $\xi\phi$ is orthogonal, i.e., left-linear and without superposition, $\xi\phi$ is then confluent.

$$
\begin{array}{rcl@{\qquad}rcl}
\kappa_1(\underline{n}) & = & 2^{\kappa_1(n)} & \kappa_2(\underline{n}) & = & 2^{\kappa_2(n)_2} \\
\kappa_1(n+1) & = & \kappa_1(n)+1 & \kappa_2(n+1) & = & \kappa_2(n)+1 \\
\kappa_1(1) & = & 2 & & & \\
\kappa_1(ab) & = & \kappa_1(a)+\kappa_1(b)+1 & & & \\
\kappa_1(\lambda a) & = & \kappa_1(a)+1 & & & \\
\kappa_1(a\llbracket s\rrbracket\,) & = & \kappa_1(a)\kappa_1(s) & \kappa_2(a\llbracket s\rrbracket) & = & \kappa_2(a)\kappa_2(s) \\
\kappa_1(\Uparrow(s)) & = & \kappa_1(s) & \kappa_2(\Uparrow(s)) & = & 2\kappa_2(s) \\
\kappa_1(\uparrow) & = & 2 & \kappa_2(\uparrow) & = & 3 \\
\kappa_1(\updownarrow) & = & 2 & & & \\
\kappa_1(a/) & = & any & & &
\end{array}
$$

Figure 4: Interpretations for proving the termination of υ

There is two critical pairs between (B) and (App_1) on one side and between (B) and (App_2) on another side. Those critical pairs are:

$$
\begin{array}{rcl}
a[b/][c/] & = & a\llbracket\updownarrow\rrbracket[c\llbracket\updownarrow\rrbracket/][b[c/]/] \\
a[b/]\llbracket s\rrbracket & = & a\llbracket\Uparrow(s)\rrbracket[b\llbracket s\rrbracket/]
\end{array}
$$

Those critical pairs can be proved as inductive lemmas in $\mathbf{Terms}_{\xi\phi}/\overset{*}{\underset{\lambda\xi\phi}{\longleftrightarrow}}$, i.e., modulo the equality generated by $\lambda\xi\phi$ on $\mathbf{Terms}_{\xi\phi}$. Then it can be proved by classical methods that the rewriting relation $\overset{*}{\underset{\lambda\xi\phi}{\longrightarrow}}$ defined on $\mathbf{Terms}_{\xi\phi}$ and generated by $\lambda\xi\phi$ is confluent.

Acknowledgment: I would like to thank Professor N. G. de Bruijn for mentioning to me his report [6] and for interesting discussions.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. Technical Report 54, Digital Systems Research Center, February 1990. Preliminary version in *Proc. of the 17th POPL conference, Orlando (Fla., USA).*

[2] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[3] N. Bourbaki. *Éléments de mathématiques: Théories des ensembles*, volume 1. Hermann & C^{ie}, 1954.

[4] P.-L. Curien, Th. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. RR 1617, INRIA, Rocquencourt, February 1992.

[5] N. G. de Bruijn. Lambda calculus with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Proc. Koninkl. Nederl. Akademie van Wetenschappen*, 75(5):381–392, 1972.

[6] N. G. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. TH-Report 78-WSK-03, Technological University Eindhoven, Netherlands, Department of Mathematics, 1978.

[7] J. Field. On laziness and optimality in lambda interpreters: Tools for specification and analysis. In *Proceedings of the 17th Annual ACM Symposium on Principles Of Programming Languages, Orlando (Fla., USA)*, pages 1–15, San Fransisco, 1990. ACM.

[8] J. Field. *Incremental Reduction in the Lambda Calculus and Related Reduction Systems.* PhD thesis, Cornell U., November 1991.

[9] P. Lescanne and J. Rouyer-Degli. The calculus of explicit substitutions $\lambda\upsilon$. Technical Report RR-2222, INRIA-Lorraine, January 1994.

[10] A. Ríos. *Contributions à l'étude des λ-calculs avec des substitutions explicites.* Thèse de Doctorat d'Université, U. Paris VII, 1993.