Strong normalisation of Herbelin’s explicit substitution calculus with substitution propagation

Roy Dyckhoff\textsuperscript{1*} and Christian Urban\textsuperscript{2}

\textsuperscript{1} University of St Andrews rd@dcs.st-and.ac.uk
\textsuperscript{2} University of Cambridge cu200@dpmms.cam.ac.uk

Abstract. Herbelin presented (at CSL’94) a simple sequent calculus for minimal implicational logic, extensible to full first-order intuitionistic logic, with a complete system of cut-reduction rules which is both confluent and strongly normalising. Some of the cut rules may be regarded as rules to construct explicit substitutions. He observed that the addition of a cut permutation rule, for propagation of such substitutions, breaks the proof of strong normalisation; the implicit conjecture is that the rule may be added without breaking strong normalisation. We prove this conjecture, thus showing how to model beta-reduction in his calculus (extended with rules to allow cut permutations).

1 Introduction

Herbelin gave in [8] a calculus for minimal implicational logic, using a notation for proof terms that, in contrast to the usual lambda-calculus notation for natural deduction, brings head variables to the surface. It is thus a sequent calculus, with the nice feature that its cut-free terms are in a natural 1-1 correspondence with the normal terms of the simply typed lambda-calculus. Other intuitionistic connectives can be added without difficulty.

The cut rules of the calculus are in part analogous to explicit substitution constructors [11] and certain auxiliary operators (e.g. list concatenation); the only exception is a rule that in some circumstances constructs an auxiliary term and in others constructs a term analogous to a beta-redex. Herbelin showed strong normalisation and confluence of a complete system of rules for eliminating cuts, observed that the addition of a further “cut permutation” rule, needed to allow explicit substitutions to propagate properly, as required for the simulation of beta-reduction of the simply typed lambda-calculus, would break the strong normalisation proof, thus raising the question of whether it also broke the strong normalisation result. In the present paper we answer this question; strong normalisation holds for the calculus with the addition of this rule (and of other cut permutation rules, to retain confluence).

Herbelin’s calculus can thus be seen in two ways:

1. In the cut-free case it is a natural basis [5] for automated proof search in logic programming, since it is a sequent calculus but free from the permutation problems of Gentzen’s calculus;
2. In the general case, when extended with the cut permutation rules, it can simulate beta-reduction and thus, with its strong proof-theoretic foundations, may be a natural basis for implementation of functional languages.

Our proof uses standard techniques, e.g. from [2]. That paper, in its use of recursive path ordering techniques, shows the use of higher-order rewriting to be unnecessary, by translating to a first-order system. In order not to obscure our argument, we omit the details of this translation; the conscientious reader is invited to fill in the missing details.

* Thanks are due to the second author’s family and the Dresden University of Technology for support of various kinds during a visit to Dresden covering the genesis of this paper.
2 Technical Background

We use a notation developed in [6], since it distinguishes the different kinds of cut term and in proofs emphasises the role of the *stoup* formula. The syntax of the calculus is as follows: formulae $A$ are as in implicational logic, there are (term-)variables $x, y, \ldots$, there are two kinds $Ms, M$ of proof-term and two kinds of sequent, one with and one without a *stoup* formula, which is written in the first case below the sequent arrow. $Ms$ is used in the stoup sequents; although the notation may suggest a list, not all terms $Ms$ are lists. Notions of “free” and “bound” and variable conventions are as usual, with variable binding not just in λ-terms but also in cut₂ and cut₄ terms. Contexts $Γ$ are finite functions from variables to formulae; $Γ, x : A$ indicates the extension, by the assignment of $A$ to $x$, of a context $Γ$ in which there is no assignment to $x$.

Syntax of cut-free terms

$$Ms ::= [\ ] \mid (M :: Ms)$$

$$M ::= (x; Ms) \mid λx. M$$

Logical/Typing rules

$$Γ ⊢ M : A \quad Γ \xrightarrow{b} Ms : C$$

$$Γ \xrightarrow{\lor} (M :: Ms) : C \quad S ⊔$$

$$Γ, x : A \xrightarrow{a} Ms : B$$

$$Γ, x : A \Rightarrow (x; Ms) : B \quad Sel$$

$$Γ, x : A ⇒ M : B$$

$$Γ ⇒ λx. M : A ⊔ B \quad R ⊔$$

Syntax of cut terms (type information is omitted in the type-free case)

$$Ms ::= cut^4_1(Ms, Ms) \mid cut^4_2(M, x. Ms)$$

$$M ::= cut^4_3(M, Ms) \mid cut^4_4(M, x. M)$$

Logical/Typing rules for Cuts

$$Γ \xrightarrow{a} Ms : A \quad Γ \xrightarrow{b} Ms' : C$$

$$Γ \xrightarrow{\lor} cut^4_1(Ms, Ms') : C \quad Cut_1$$

$$Γ ⇒ M : A \quad Γ, x : A \xrightarrow{b} Ms : C$$

$$Γ ⇒ cut^4_2(M, x. Ms) : C \quad Cut_2$$

$$Γ ⇒ M : A \quad Γ \xrightarrow{a} Ms : B$$

$$Γ ⇒ cut^4_3(M, Ms) : B \quad Cut_3$$

$$Γ ⇒ M : A \quad Γ, x : A ⇒ M' : B$$

$$Γ ⇒ cut^4_4(M, x. M') : B \quad Cut_4$$

Rules for cut-reduction

Let ES denote the system of “explicit substitution” rules 1...4 on terms, and CC the system of “commuting cuts” rule 5 on terms:

1. (a) $cut^4_1([\ ], Ms) \sim Ms$
   (b) $cut^4_1((M :: Ms), Ms') \sim (M :: cut^4_1(Ms, Ms'))$
2. (a) $cut^4_2(M, x.[\ ]) \sim [\ ]$
   (b) $cut^4_2(M, x.(M :: Ms)) \sim (cut^4_4(M, x. Ms) :: cut^4_2(M, x.Ms))$
3. (a) $cut^4_2((x; Ms), Ms') \sim (x; cut^4_2(Ms, Ms'))$
   (b) $cut^4_2(λy. M, [\ ]) \sim λy. M$
4. (a) $cut^4_2(M, x.(y; Ms)) \sim (y; cut^4_2(M, x. Ms)) \quad (y \neq x)$
   (b) $cut^4_2(M, x.(x; Ms)) \sim cut^4_2(M, cut^4_2(M, x. Ms))$
   (c) $cut^4_2(M, x.λy. M') \sim λy. cut^4_2(M, x. M')$
5. (a) $cut^4_2(cut^4_1(Ms, Ms'), Ms'') \sim cut^4_1(Ms, cut^4_2(Ms', Ms''))$
   (b) $cut^4_2(M, x.cut^4_1(Ms, Ms')) \sim cut^4_2(cut^4_2(M, x. Ms), cut^4_2(M, x. Ms'))$
   (c) $cut^4_2(cut^4_2(M, Ms), Ms') \sim cut^4_2(M, cut^4_2(Ms, Ms'))$
   (d) $cut^4_2(M, x.cut^4_2(Mp, Ms)) \sim cut^4_2(cut^4_2(M, x. Mp), cut^4_2(M, x. Ms))$
Strictly speaking, the rules 1 and 3 are not explicit substitution reduction rules but auxiliary rules. The last of all these rules, 5(d), is the one mentioned in the introduction as the proper propagation of an explicit substitution, e.g. inside a “beta-redex”. The other CC rules are added to ensure confluence once that rule is added. We allow rule applications inside terms, i.e. we in fact consider the contextual closures of all rules, as usual.

We need to consider one further rule, deliberately omitted from group 3:

\( B \quad \text{cut}^{B}_3(\lambda x.M, (M' :: M' :: M)) \sim \text{cut}^{B}_3(\text{cut}^{A}_4(M', x.M), M) \)

which generates a cut\(_4\)-instance; to simulate ordinary beta-reduction, this may need to propagate into its body \( M' \); the latter may be a cut\(_3\)-term, hence the utility of rule 5(d), and so on.

Herbelin [8] showed that a system of rules, essentially \( ES + B \) with different terminology, is complete, confluent and (for typed terms) strongly normalising. Note that the CC-rules are NOT required for completeness; without them, a strategy that (more or less) ignores cuts whose arguments are cuts is imposed.

A simpler (but less direct) proof of his SN theorem using the multiset path ordering theorem is given in [6]; the termination comes from the ordering of all the cut operators as greater than the non-cut operators, with \( \text{cut}^A > \text{cut}^B \) for \( A > B \) and with \( \text{cut}_4 = \text{cut}_2 > \text{cut}_3 = \text{cut}_1 \) for cut operators with the same type annotation.

The addition of any of the rules in 5 breaks both of these proofs, because of the switching of the types.

Note that there are no rules allowing permutation of \( \text{cut}^2 \) or \( \text{cut}^4 \) operators with \( \text{cut}^2 \) or \( \text{cut}^4 \) operators.

Routinely, “Subject Reduction” holds for all these reduction rules; thus, all the results (e.g. confluence) that we state or prove for untyped terms hold also for the typed terms.

3 Pure Terms

There are two obvious (and in fact equivalent) candidates for the class of terms that we will use to interpret lambda terms: those free of \((ES + CC)\)-redexes and those free of all instances (other than \( B \)-redexes) of cut: we choose the latter.

**Definition 1** \((ES + CC)\)-normality; purity.

1. A term is \((ES + CC)\)-normal iff it is free of all \( ES + CC \)-redexes;
2. A term is pure iff it is free of all instances (other than \( B \)-redexes) of cut.

By induction, \((ES + CC)\)-normal terms \( Ms \) are of the form \( [] \) or \( M :: Ms' \). Pure terms are \((ES + CC)\)-normal, since all \((ES + CC)\)-redexes are instances of cut; but the converse also holds:

**Proposition 2.** \((ES + CC)\)-normal terms \( Ms, M \) are pure.

**Proof.** The \( Ms \) case depends just on the \( M \) case. We proceed by induction and case analysis; we show that if an \((ES + CC)\)-normal term \( M \) is an instance of cut, then it is a redex; our inductive hypothesis is that all smaller \((ES + CC)\)-normal terms are pure. Consider the cases for \( M \) an instance of cut:

1. \( \text{cut}^3_3(\langle x :: Ms \rangle, Ms') \) is a 3(a)-redex;
2. \( \text{cut}^3(\lambda x.M', M) \) is a 3(b)-redex;
3. \( \text{cut}^3(\lambda x.M', (M'' :: Ms)) \) is a \( B \)-redex;
4. \( \text{cut}^3(\text{cut}^3_3(M', Ms), Ms') \) is a 5(c)-redex;
5. \( \text{cut}^3_3(\text{cut}^3_3(M', x.M''), Ms) \) is not allowed, since (by inductive hypothesis) \( \text{cut}_4(M', x.M'') \) is pure, contrary to its being a \( (non \ B\)-redex) \( \text{cut} \)-term;
6. \( \text{cut}^4(M', x.(z :: Ms)) \) is a 4(a) or 4(b)-redex;
7. \( \text{cut}^4(M', x.\lambda y.M'') \) is a 4(c)-redex;
8. \( \text{cut}^4(M', x.\text{cut}^3_3(M'', Ms)) \) is a 5(d)-redex;
9. \( \text{cut}_4(M', x.\text{cut}_4(M'', y.M''')) \) is not allowed, for the same reason as in (5) above.

It follows that the \((ES + CC)\)-normal/pure terms \( M \) are of the form \((y;Ms), \lambda x. M'\) or \(\text{cut}_3(\lambda y. M', (M'' :: Ms))\).

**Definition 3 (Implicit substitution; concatenation; generalised application).** For pure terms \(Ms, Ms', M, M'\), and for the variable \(x\), we define by simultaneous induction on the structure of \(Ms\) (resp. \(Ms\), resp. \(M\), resp. \(M'\))

1. The concatenation\(^1\) \(Ms \circ Ms'\) of \(Ms\) with \(Ms'\);
2. The implicit substitution \([M/x]Ms\) of \(M\) for \(x\) in \(Ms\);
3. The generalised application \([M]Ms\) of \(M\) to \(Ms\);
4. The implicit substitution \([M/x]M'\) of \(M\) for \(x\) in \(M'\).

as follows:

\[
\begin{align*}
\square @ Ms' &=_{df} Ms' \\
(M'' :: Ms'') @ Ms' &=_{df} (M'' :: (Ms'' @ Ms')) \\
[M/x]\square &=_{df} [] \\
[M/x](M'' :: Ms'') &=_{df} ([M/x]M'' :: [M/x]Ms'') \\
\{y;Ms''\} Ms &=_{df} (y; (Ms'' @ Ms)) \\
\{\lambda y. M''\}[] &=_{df} \lambda y. M'' \\
\{\lambda y. M''\}(M'' :: Ms'') &=_{df} \text{cut}_3(\lambda y. M'', (M'' :: Ms'')) \\
\{\text{cut}_3(\lambda y. M'', (M'' :: Ms''))\} Ms &=_{df} \text{cut}_3(\lambda y. M'', (M'' :: (Ms'' @ Ms))) \\
[M/x](y; Ms'') &=_{df} (y; [M/x]Ms') \quad (y \neq x) \\
[M/x](x; Ms'') &=_{df} [M][M/x]Ms'' \\
[M/x](\lambda y. M'') &=_{df} \lambda y. [M/x]M'' \\
[M/x]\text{cut}_3(\lambda y. M'', (M'' :: Ms'')) &=_{df} \text{cut}_3(\lambda y. [M/x]M'', ([M/x]M'' :: [M/x]Ms'')).
\end{align*}
\]

Note that the mutual induction is straightforward; first, concatenation is well-defined (as usual); second, generalised application has a definition depending only on concatenation; finally, the two forms of implicit substitution depend on simpler instances of themselves and of each other and on instances of generalised application.

### 4 Lambda Calculus

Our exposition of the lambda calculus uses approximately the notation of [10]. Lambda terms \(N\) and lists \(Ns\) of lambda terms are defined by the grammar

\[
\begin{align*}
N &::= (x.N) \mid (\lambda x. N) \mid ((\lambda x. N)N.N) \\
Ns &::= [] \mid \{N :: Ns\}
\end{align*}
\]

in which, if we omit the last production for \(N\), we get just the normal terms. Note that, for example, a term of the form \((\lambda x. N)N.Ns\) is NOT the term list \((\lambda x. N) :: (N' :: Ns))\; it is a term. The structure of this inductive definition is intended to make certain parts of the normalisation proof in [10] easy, but no definition of substitution is given in [10]; essentially, the translation to standard

\(^1\) This is just the usual concatenation of lists, included here for completeness.
notation is used, then the standard definition is used, then one translates back. We remedy this
minor oversight as follows:

We define implicit substitution \([N/x]Ns\) of \(N\) for \(x\) in \(Ns\) (and similarly for substitution in
\(N'\) as follows, by induction on the structure of \(Ns\) (resp. \(N'\)); we need an auxiliary definition of a
term \(\{N\}Ns\), by induction on the structure of \(N\) (and a subsidiary induction in one case on \(Ns\)),
to apply the term \(N\) to the list \(Ns\) of arguments in the usual way (this is not the construction of
a list, but of a term):

\[
\begin{align*}
[N/x][] & \equiv f [] \\
[N/x](N :: Ns) & \equiv f ([N/x]N' :: [N/x]Ns)
\end{align*}
\]

\[
\begin{align*}
\{(y Ns')\} Ns & \equiv f (y (Ns'@Ns)) \\
\{(\lambda y. N)\}[] & \equiv f (\lambda y. N) \\
\{(\lambda y. N)\}(N' :: Ns) & \equiv f ((\lambda y. N)N'Ns) \\
\{(\lambda y. N)'Ns\} Ns' & \equiv f ((\lambda y. N)'(Ns@Ns'))
\end{align*}
\]

\[
\begin{align*}
[N/x](y Ns) & \equiv f (y [N/x]Ns) \quad (y \neq x) \\
[N/x](x Ns) & \equiv f \{N\}[N/x]Ns \\
[N/x](\lambda y.N') & \equiv f (\lambda y.[N/x]N') \\
[N/x]((\lambda y. N')N''Ns) & \equiv f ((\lambda y.[N/x]N')(N''[N/x]Ns))
\end{align*}
\]

where \(\@\) is for the standard function that concatenates two lists.

Termination of this definition is as in the previous section. The equivalence between this defini-
tion and the usual definition of substitution (with the usual notation) is a tedious exercise.

\(\beta\)-reduction is thus the reduction of a term by careful replacement of a subterm (the \(\beta\)-redex)
of the form \((\lambda x. N)N'Ns\) by the redct \([N'/x]N\) \(Ns\) in a single step.

5 Interpretation of Lambda Calculus

We now define (really trivial) bijective interpretations \((.)^*\) of \(\lambda\)-terms \(N\) as the pure terms of the
form \(M\) and \((.)^{**}\) of term-lists \(Ns\) as the pure terms of the form \(M\):

\[
\begin{align*}
(x Ns)^* & \equiv f (x; Ns^{**}) \\
(\lambda x.N)^* & \equiv f (\lambda x.N^*) \\
((\lambda x. N)N'Ns)^* & \equiv f cut_3(\lambda x. N^*, (N'^* :: Ns^{**})) \\
[[ ]]^* & \equiv f [] \\
(N :: Ns)^{**} & \equiv f (N^* :: Ns^{**})
\end{align*}
\]

**Proposition 4.** For \(\lambda\)-terms \(N, N'\) and term lists \(Ns, Ns'\), the following hold:

1. \((Ns@Ns')^{**} = Ns^{**}@Ns'^{**}\);  
2. \([N/x]Ns)^{**} = [N^*/x]Ns^{**};  
3. \((\{N\}Ns)^* = (N^*) Ns^{**};  
4. \([N/x]N'^* = [N^*/x]N'^*.

\(\blacksquare\)

**Proof.** Routine.
6 Strong Normalisation and Confluence of $ES + CC$

**Proposition 5.** The system $ES + CC$ is strongly normalising (SN).

*Proof. A lexicographic path order suffices, completely ignoring all the type information, with the $cut_2 = cut_1$ operators equal, and greater than $cut_1 = cut_3$, and with all cut operators greater than the non-cut operators. We rely throughout on the exposition of the lexicographic path order given in [1] rather than that in [2]. For an alternative proof using a polynomial interpretation, see Appendix A.*

**Proposition 6.** The system $ES + CC$ is confluent.

*Proof. By Proposition 5, it suffices to check the local confluence; for details see Appendix B.*

**Definition 7.** For a term $Ms$ or $M$, its purification is its $(ES + CC)$-normal form, written $\overline{Ms}$ (resp. $\overline{M}$).

**Lemma 8.** If $M \leadsto^*_ES + CC M'$, then $\overline{M} = \overline{M'}$. (Similarly for $Ms$.)

*Proof. Trivial.*

7 Simulation of $\beta$-reduction

**Proposition 9.** Let $M, M', Ms, Ms'$ be pure terms. Then

1. $cut_1(Ms, Ms') \leadsto^*_ES + CC Ms@Ms'$;
2. $cut_3(M, x, Ms) \leadsto^*_ES + CC [M/x]Ms$;
3. $cut_2(M, Ms) \leadsto^*_ES + CC \{M\}Ms$;
4. $cut_4(M, x, M') \leadsto^*_ES + CC [M/x]M'$.

*Proof. The first part is trivial; the third part is proved by induction; the remaining two parts are proved by a simultaneous induction on the heights of the LHS terms. For details see Appendix C.*

The above may be regarded as a weak normalisation result (for $ES + CC$), since we may use it to purify any innermost (non-$\beta$)-redex, repeating this operation until all such redexes are eliminated.

**Corollary 10.** For pure terms $M, M'$ and $Ms$,

$$cut_2(cut_4(M', x, M), Ms) \leadsto^*_ES + CC \{[M'/x]M\}Ms.$$

**Theorem 11.** If $N_1 \leadsto_\beta N_2$ in the $\lambda$-calculus, then $N_1^*$ reduces to $N_2^*$ by a $\beta$-reduction followed by 0 or more $ES + CC$ reductions.

*Proof. Consider first the case where $N_1$ is the $\beta$-redex, and so of the form $((\lambda x. N)N'Ns)$, for terms $N, N'$ and term list $Ns$. Thus, $N_2 = ([N'/x]N')Ns$. By Corollary 10, the reduct

$$cut_2(cut_4(N'^*, x, N^*), N^*s^*)$$

of the $\beta$-redex $N_1^* = cut_2(\lambda x. N^*, (N'^* :: Ns^*))$ is $(ES + CC)$-reducible to $\{[N'/x]N^*\}Ns^*$. By Proposition 4, this is just $N_2^*$. The general case, where the reduction is not at the root position of $N_1$, follows by induction on the structure of $N_1$.*

In other words, the system $(ES + CC + B)$ of rules acting on the untyped terms simulates $\beta$-reduction of the untyped $\lambda$-calculus; similarly for typed terms and the typed $\lambda$-calculus.
8 \textbf{β-reduction}

We may now define a rule $\beta$ on pure terms, typed or untyped, omitting the types in the latter case:

\[
\beta\quad \text{cut}_3^A \vdash B (\lambda x. M, M' :: Ms) \rightsquigarrow_{\beta} \text{cut}_3^B (\text{cut}_4^A (M', x.M), Ms)
\]

Note that, in the untyped case, the RHS of this is, by Corollary 10, just $([M'/x] M)Ms$. The correspondence between pure untyped terms and the terms of the untyped $\lambda$-calculus routinely extends to $\beta$-reduction.

Thus, a $\beta$-reduction is a single $B$-reduction followed by purification, i.e. zero or more (ES+CC)-reductions to normal form.

\textbf{Proposition 12.} The system, on pure typed terms, consisting just of the rule $\beta$ is SN.

\textit{Proof.} Use, for example, the proof, in different notation, in [10].

If we had a direct proof of Proposition 12, then we would have shown the strong normalisation of the simply-typed $\lambda$-calculus.

\textbf{Definition 13.} For any term $M$, we define $\|M\|$ to be the maximal length of all $\beta$-reduction sequences from $M$ if the latter is $\beta$-SN; otherwise we define $\|M\| = \infty$. (Similarly for $Ms$.)

\textbf{Corollary 14.} For every typed term $M$, we have $\|M\| < \infty$. (Similarly for $Ms$.)

\textit{Proof.} By Proposition 12 and König’s Lemma.

\textbf{Lemma 15.} For pure terms $M, Ms, M', M''$, $M'$, and $Ms', \text{we have}$

1. \( \{M\}Ms \rightsquigarrow_{\beta} \{M'\}Ms; \)
2. \( \{M\}Ms \rightsquigarrow_{\beta} \{M\}Ms'; \)
3. \( [M/x]Ms \rightsquigarrow_{\beta} [M'/x]Ms; \)
4. \( [M/x]Ms \rightsquigarrow_{\beta} [M/x]Ms'; \)
5. \( [M'/x]M'' \rightsquigarrow_{\beta} [M'/x]M''; \)
6. \( [M''/x]M \rightsquigarrow_{\beta} [M''/x]M'. \)

\textit{Proof.} These follow from consideration of the untyped $\lambda$-calculus: for example, substitution into a $\beta$-redex leaves it as a $\beta$-redex.

\textbf{Lemma 16.} For pure terms $M, M'$, $Ms$ with $M \rightsquigarrow_{\beta} M'$ we have $\{M\}Ms \rightsquigarrow_{\beta} \{M'\}Ms$.

\textit{Proof.} By induction on the definition of $\{M\}$. The essential idea is that any $\beta$-redex in $M$ is still a $\beta$-redex in $\{M\}Ms$, even though $M$ may well not be a subterm of $\{M\}Ms$.

9 \textbf{Adding $B$-reduction}

We will show that the addition of the $B$-rule upset neither confluence nor, provided we stick at least to (e.g.) typed terms, termination. A key ingredient in both of these proofs is the Projection Lemma, i.e. that a root $B$-reduction translates (under purification) to exactly one $\beta$-reduction.

\textbf{Lemma 17.} \( \text{cut}_3 (\lambda x. M, (M' :: Ms)) \rightsquigarrow_{\beta} \text{cut}_3 (\text{cut}_4 (M', x.M), Ms). \)

\textit{Proof.} The LHS = $\text{cut}_3 (\lambda x. M, (M' :: Ms))$, the latter (as a $B$-redex) being pure; this $B$-reduces to $\text{cut}_4 (\text{cut}_3 (M', x.M), Ms)$ and thus $\beta$-reduces to $\text{cut}_3 (\text{cut}_4 (M', x.M), Ms)$. By Lemma 8, this equals the RHS.

\textbf{Lemma 18 (Projection Lemma).} If $M \rightsquigarrow_{B} M'$ at the root position, then $M \rightsquigarrow_{\beta} M'$. 

Proof. Apart from the names of variables, this is just a restatement of Lemma 17. ■

Corollary 19. For terms $M, M', Ms$, we have (if the RHS $< \infty$)

$$\|\text{cut}_2(\lambda x. M, (M' :: Ms))\| > \|\text{cut}_2(\text{cut}_4(M', x. M), Ms)\|.$$

We also need to know that an arbitrary $B$-reduction translates, after purification, to zero or more $\beta$-reductions.

Proposition 20. The following hold:

1. If $Ms \sim_B Ms'$, then $\overline{Ms} \sim_B^* \overline{Ms'}$;
2. If $M \sim_B M'$, then $\overline{M} \sim_B^* \overline{M'}$.

Proof. By simultaneous inductions on the size of $Ms$ or $M$, and case analysis:

1. (a) $Ms = []$: trivial;
   (b) $Ms = (M :: Ms'')$: routine use of inductive hypothesis;
   (c) $Ms = \text{cut}_3(M'', Ms'')$: similar to the first two parts of case (2c) below.
   (d) $Ms = \text{cut}_3(M'', x. Ms'')$: similar to the case 2(d) below.
2. (a) $M = \lambda x. M''$: routine;
   (b) $M = (\lambda x. Ms'')$: routine;
   (c) $M = \text{cut}_3(M'', Ms'')$: there are three cases:
      i. the $\overline{B}$-reduction is of $M''$ to $M'''$: by inductive hypothesis, $\overline{M''} \sim_B^* \overline{M'''}$, whence, by Lemma 15 (1), $\overline{\{M''\}Ms''} \sim_B^* \overline{\{M''\}Ms'''}$. Now,
         $$\text{cut}_3(M'', Ms'') = \text{cut}_3(\overline{M''}, \overline{Ms''}) = \overline{\{M''\}Ms''}$$
         by Lemma 8 and Proposition 9 (3) respectively; and similarly
         $$\text{cut}_3(M'''', Ms''') = \text{cut}_3(\overline{M''''}, \overline{Ms''''}) = \overline{\{M'''\}Ms''''}$$
         whence $\overline{M} \sim_B^* \overline{M'}$.
      ii. the $\overline{B}$-reduction is of $Ms''''$ to $Ms'''$: similar, using Lemma 15 (2).
      iii. the $\overline{B}$-reduction is at the root of $M$: we use the Projection Lemma.
   (d) $M = \text{cut}_4(M'', x. M'''')$: there are two cases, dealt with as in (c), using Lemma 15 (5) and (6). ■

It is now easy to show that the system $ES + CC + B$ is confluent, using the confluence of $\beta$-reduction in the $\lambda$-calculus.

Theorem 21. The system $ES + CC + B$ is confluent.

Proof. Suppose that $M \sim_{ES+CC+B} M_1$ and $M \sim_{ES+CC+B} M_2$. Then, by Lemma 8 and Proposition 20, both $\overline{M} \sim_B^* \overline{M_1}$ and $\overline{M} \sim_B^* \overline{M_2}$, whence, by confluence of $\beta$-reduction in the $\lambda$-calculus, for some (pure) term $M^*$ we have $\overline{M_1} \sim_B^* M^*$ and $\overline{M_2} \sim_B^* M^*$. But then, both $M_1 \sim_{ES+CC+B}^* M^*$ and also $M_2 \sim_{ES+CC+B}^* M^*$. (Similarly for $Ms$.) ■

10 Strong Normalisation

Definition 22 (Bounded terms). A term $M$ (or $Ms$) is bounded iff for every subterm $M'$ or $Ms'$ thereof, $\| M' \| < \infty$ (resp. $\| Ms' \| < \infty$).

Proposition 23 (Boundedness).

1. Every typed term is bounded;
2. Every $(ES + CC + B)$-SN term is bounded;
3. For pure terms, $\beta$-SN is equivalent to “bounded”.
Proof. We consider the three parts in order:

1. Trivial, since every subterm of a typed term is typed and the purification of a typed term is typed, and (by Proposition 12) every pure typed term is β-SN.
2. Consider a subterm $M$ of an $(ES + CC + B)$-SN term; it also is $(ES + CC + B)$-SN and so is its purification $\overline{M}$. But then any infinite sequence of β-reductions from $\overline{M}$ can be seen, by Corollary 10, as a sequence of $(ES + CC + B)$-reductions, including infinitely many B-reductions. (Similarly for subterms $M$s.)
3. Pure β-SN terms are bounded, since, for every subterm $M$ of such a term, $M$ is pure and β-SN; so $\overline{M} = M$ is β-SN (and similarly for subterms $Ms$s). The converse is even more trivial. ■

Our aim now is to prove Theorem 31, i.e. the converse of part (2) of Proposition 23.

Lemma 24. For all terms $M$, $M'$, $Ms$, $Ms'$:

1. If $M \sim_{ES+CC} M'$ then $\|M\| = \|M'\|$
2. If $Ms \sim_{ES+CC} Ms'$ then $\|Ms\| = \|Ms'\|$
3. If $M \sim_{B} M'$ then $\|M\| \geq \|M'\|$
4. If $Ms \sim_{B} Ms'$ then $\|Ms\| \geq \|Ms'\|$.

Proof. The first part is trivial, since $\overline{M} = \overline{M'}$; the second part is similar. The other two parts use Proposition 20. ■

Definition 25 (Cosy occurrence; cosy embedding). An occurrence of a proper subterm in a term is cosy iff no cut$_2$ or cut$_4$ constructors intervene on the path to the root, apart from a possible occurrence at the subterm itself. A term is cosily embedded in a term iff it has a cosy occurrence in the latter term.

For example, in a term of the form cut$_1$(cut$_3$(M, x, Ms), cut$_3$(M, x, Ms')) the two cut$_3$ subterms are the only cosily embedded proper subterms; and in a term of the form cut$_4$(M, x, Ms), no proper subterms are cosily embedded.

Lemma 26. For all terms $M$, $Ms$, $Ms'$:

1. (a) $\| (M :: Ms) \| \geq \| M \|$
   (b) $\| (M :: Ms) \| \geq \| Ms \|$
2. $\| (x; Ms) \| = \| Ms \|$
3. $\| \lambda x. M \| = \| M \|$
4. (a) $\| \text{cut}_1(Ms, Ms') \| \geq \| Ms \|$
   (b) $\| \text{cut}_1(Ms, Ms') \| \geq \| Ms' \|$
5. $\| \text{cut}_2(M, Ms) \| \geq \| M \|$
6. $\| \text{cut}_2(M, Ms) \| \geq \| Ms \|$

Proof. 1. Because $(M :: Ms) = (\overline{M} :: Ms)$;
2. Because $(x; Ms) = (x; \overline{Ms})$;
3. Because $\lambda x. M = \lambda x. \overline{M}$;
4. Because, by Proposition 9 (1), $\text{cut}_1(Ms, Ms') = Ms \circ Ms'$;
5. Since $\text{cut}_3(M, Ms) = \text{cut}_3(M, Ms)$ and $\| M \| = \| \overline{M} \|$, we may, without loss of generality, assume that the terms $M$ and $Ms$ are pure; we now just appeal to Lemma 16.
6. Without loss of generality, for the same reason as in (5), $M$ and $Ms$ may be assumed to be pure. If $Ms = []$, then the result is trivial. Otherwise, we argue by induction on the size of $M$ and case analysis. Consider the possible forms of $M$:
   (a) $M = (x; Ms')$, whence by rule 3(a) it suffices to observe that $\| (x; Ms' \circ Ms) \| = \| Ms'' \| + \| Ms \|$
   (b) $M = \lambda x. M'$; so $\text{cut}_2(M, Ms)$ is pure and has $Ms$ as a sub-term, whence $\| \text{cut}_2(M, Ms) \| \geq \| Ms \|$. 

(c) $M = \text{cut}_3(M', Ms'');$ so, by rule 5(c), $\text{cut}_3(M, Ms) = \text{cut}_3(M', Ms''@Ms)$ and, by inductive hypothesis, $\| \text{cut}_3(M', Ms''@Ms) \| \geq \| Ms'' @ Ms \| \geq \| Ms \|$. \hfill $\blacksquare$

**Corollary 27.** If a term $M$ is costly embedded in $M'$, then $\| M' \| \leq \| M' \|$, and similarly for other combinations such as $M$ costly embedded in $Ms$, etc.

**Proof.** By the lemma, using induction on the number of constructors between the subterm and the term. \hfill $\blacksquare$

On the other hand, whenever $\| M' \| > 0$, we have

$$\| M' \| \leq \| \text{cut}_2(M, x.[]) \| = 0$$

and

$$\| M' \| \leq \| \text{cut}_3(M, x.\lambda y.(y[])) \| = 0.$$

**Corollary 28.** We have the following:

1. For each of the $(ES + CC)$-reduction rules $L \leadsto R$, and for each non-variable subterm $S$ of $R$, and instantiation $\theta$ of the variables in the rule, we have $\| L^\theta \| \geq \| S^\theta \|$.
2. For the $B$-reduction rule $L \leadsto R$, and for each non-variable subterm $S$ of $R$, and instantiation $\theta$ of the variables in the rule, we have $\| L^\theta \| \geq \| S^\theta \|$, the inequality being strict if either side is finite.

**Proof.**
1. By Lemma 24 (1 or 2), we have $\| L^\theta \| = \| R^\theta \|$. It now suffices to note that, for each rule $L \leadsto R$, each non-variable proper subterm of $R$ is costly embedded.
2. By Lemma 18, assuming $\| L^\theta \| < \infty$, we have $\| L^\theta \| > \| R^\theta \|$; as before, the only non-variable proper subterm $S$ of $R$ is costly embedded. \hfill $\blacksquare$

This corollary is the crux of the present paper; it gives us information about the $(ES + CC + B)$-rules that will be exploited when we show that the rules are decreasing w.r.t. a suitably chosen ordering. Note that the above corollary tells us just about reductions at the root position; for reductions at non-root positions, it says nothing.

**Corollary 29.** Bounded terms are closed under $(ES + CC + B)$-reduction.

**Proof.** Let $M$ be bounded and let $R$ be a rule in $(ES + CC + B)$ such that $M \leadsto R M'$. Consider a subterm $M''$ of $M'$; we must show that $\| M'' \| < \infty$. Comparing the position of $M''$ in $M'$ to that of the reduct, we find three cases:

1. The reduct occurs as a subterm of $M''$; so we can pull $M''$ back to a subterm $M^*$ of $M$, with $M^* \leadsto M''$. Since $M$ is bounded, $\| M^* \| < \infty$. By Lemma 24, $\| M'' \| < \infty$.
2. The reduct has $M''$ as a proper subterm, and $M''$ is obtained by instantiation of a variable in the rule $R$; thus already $M''$ is a subterm of $M$, which is bounded, so $\| M'' \| < \infty$.
3. The reduct has $M''$ as a proper subterm, and $M''$ is obtained by instantiation of a non-variable subterm of the RHS of the rule $R$; by Corollary 28, $\| M'' \| \leq \| M^* \|$ for some term $M^*$ which is in fact a subterm of $M$, so $\| M'' \| < \infty$. \hfill $\blacksquare$

We now consider the bounded cut terms superfixed not, as before, with types but with, for each such term $M$, the natural number $\| M \|$ (and similarly for terms $Ms$). We again order the cut operators by $\text{cut}'' > \text{cut}'$ for $n > m$, use the sufixes $(4 = 2 > 3 = 1)$ as before for ordering cut operators with the same suffix, and order all cut operators as greater than all non-cut operators. There are now infinitely many operators; but for a given bounded term, with only finitely many cut sub-terms, we can compute an upper bound for all their superfixes; by Corollary 28, this bound suffices for all terms reachable from the term by any of the reduction rules, so we can w.l.o.g. assume that our signature is finite, as required for use of the fact that the lexicographic path ordering generated below is a simplification ordering and thus is well-founded.
From this ordering on this (finite) signature we generate the lexicographic path ordering 
\( \succ_{LPO} \) on all terms. As in [2], we can avoid the problem of the LPO techniques not being applicable
 to higher-order systems by translation into a terminating (but non-confluent) intermediate system where the bound variables are omitted. However in order not to obscure our argument, we
 will not give this translation, but rather apply the LPO techniques directly to our higher-order system.

**Proposition 30.** If \( M' \) is a bounded term and \( M' \succ_{EC+CC} M'' \), then \( M' \succ_{LPO} M'' \). (Similarly for \( Ms \).

**Proof.** It suffices to consider only reductions at root position, since \( \succ_{LPO} \) is closed under contextual closure. In fact, the previous proof (of Proposition 5) for \( (ES+CC) \) now works almost unchanged. We consider the rule \( B \) in order to illustrate the method; let

\[
M' = \text{cut}_{\lambda}^n(\lambda x. M, (M'' :: Ms)) \sim_B \text{cut}_{\lambda}^n(\text{cut}_{\lambda}^n(M'', x, M), Ms) = M''
\]

be an instance of rule \( B \). By Corollary 19, we have \( m > r \); similarly \( m > s \) by this and by
Corollary 28 (2). Then, since \( m > r \), we just need to compare \( M' \) with the two main subterms of
\( M'' \). That \( M' \succ_{LPO} \text{cut}_{\lambda}^n(M'', x, M) \) follows because \( m > s \) and \( M'' \), \( M \) are variables properly
occurring in \( M' \); that \( M' \succ_{LPO} Ms \) is trivial, the latter being a variable properly occurring in
\( M' \). It follows that \( M' \succ_{LPO} M'' \).

**Theorem 31.** Every bounded term is \( (ES + CC + B) \)-SN.

**Proof.** By the well-foundedness of \( \succ_{LPO} \) and Proposition 30.

**Corollary 32.** Every typed term is \( (ES + CC + B) \)-SN.

**Proof.** By Proposition 23 (1) and the above theorem.

(This answers Herbelin’s question.)

**Corollary 33.** The calculus of terms \( Ms, M \) with the \( (ES + CC + B) \)-reduction rules preserves
strong normalisation w.r.t. the calculus of pure terms under \( \beta \)-reduction.

## 11 Comments

It would be interesting to find a direct normalisation proof (in sequent calculus notation) for the
simply-typed \( \lambda \)-calculus and compare it with those of [10, 12].

Our cut-reductions 5(c), 5(d) for Herbelin’s explicit substitution calculus already appeared, in
different notation, as the rules \( \lambda 22 \) and \( \lambda 44 \) in Espírito Santo’s [7]; this paper *inter alia*

1. establishes a 1-1 correspondence, concerning both terms and reductions, between the lambda
calculus and his calculus \( \lambda_H \) of terms (similar to our calculus of pure terms);
2. shows that \( \lambda_H \) can be extended to a calculus \( \lambda_H^+ \) of terms which, in our notation, have no
instances of \( \text{cut}_1 \) and \( \text{cut}_2 \). These two operators are treated as defined functions; our category
of terms \( Ms \) can thus be replaced by that of term lists. This corresponds to a certain strategy
of reducing \( \text{cut}_1 \) and \( \text{cut}_2 \) terms by terms normal w.r.t. our rules 1 and 2; our rules 5(a) and
5(b) are then superfluous. Moreover, \( \text{cut}_3 \) and \( \text{cut}_4 \) terms are, following a reduction step, dealt
with immediately by auxiliary functions, defined by equations, rather than by use of explicit
operations.

However, the issue of whether \( B \)-reductions can be combined in an arbitrary fashion with \( (ES + CC) \)-reductions
is not addressed; this appears to us to be the main issue raised by Herbelin’s paper,
with its emphasis on explicit concatenations and explicit substitutions. We gratefully acknowledge
José Espírito Santo’s helpful comments illuminating the content of his paper.
Vestergaard and Wells [14] have considered explicit substitution calculi based on Gentzen’s L-systems, with de Bruijn indices rather than variable names and with “weak correspondences” with some known explicit substitution calculi.

Herbelin (private communication, March 2001) conjectured that our SN result for \((ES + CC + B)\) also follows from the SN result [9] for the \(\lambda\mu\)-calculus, mentioning however that a similar conjecture for the strong normalisability of Parigot’s \(\lambda\mu\)-calculi turned out to be mistaken. We have no opinion on this conjecture; in any case, it is good to have a more elementary proof.

An early version of this paper showed that “garbage reduction” rules (as in [2]) were admissible, as part of a (regrettably) faulty proof of Proposition 20. Such rules can be added as primitive rules without loss of confluence or termination.

The calculus of Herbelin also appears in the work of Cervesato and Pfenning [4], in the guise of a “spine calculus”; no theory of explicit substitutions therein appears to have been worked out, although some implementation of such substitutions is apparently in the Twelf implementation. We thank Iliano Cervesato for bringing this report to our attention.

There are of course issues in the explicit substitution world that are not addressed by the above, such as the questions of optimality, of sharing and of confluence on open terms. We make no claims about superiority of the system \(ES + CC + B\) over other explicit substitution calculi; we remark merely that it has an impeccable proof-theoretic pedigree.

In Appendix D we show that our calculus can simulate the \(\lambda\chi\)-calculus of [3] if we add another two simple rules (that can be added without losing confluence or termination). Issues arising with this simulation will be addressed in future work.

References

A Proof of Proposition 5

We define a function \( h : (M \cup M) \to \mathbb{N} \) as follows:

\[
\begin{align*}
    h([]) &= 1 \\
    h((M :: Ms)) &= h(M) + h(Ms) + 1 \\
    h(\text{cut}_1(M, Ms')) &= h(Ms') + 2 \cdot h(M) + 1 \\
    h(\text{cut}_2(M, x.Ms)) &= h(Ms) \cdot (3 \cdot h(M) + 1) \\
    h((x; Ms)) &= h(Ms) + 1 \\
    h(\lambda x.M) &= h(M) + 1 \\
    h(\text{cut}_3(M, Ms)) &= h(Ms) + 2 \cdot h(M) + 1 \\
    h(\text{cut}_4(M, x.Ms')) &= h(Ms') \cdot (3 \cdot h(M) + 1)
\end{align*}
\]

and observe that for every rule of ES + CC, \( h(L) > h(R) \).

B Proof of Proposition 6

Here we show systematically that each critical pair is joinable, considering pairs of rules \( R_1, R_2 \) where the LHS of \( R_1 \) unifies with a non-variable subterm of the LHS of \( R_2 \), in which case we say that \( R_1 \) overlaps with \( R_2 \).

1. Rule 1(a) overlaps with 5(a): thus,

\[
\text{cut}^A_1(\text{cut}^B_1([], Ms), Ms')
\]

reduces by 1(a) to

\[
\text{cut}^B_1(M, Ms');
\]

we can also reduce it by 5(a) to

\[
\text{cut}^B_1([], \text{cut}^A_1(Ms, Ms'))
\]

which reduces by 1(a) to

\[
\text{cut}^A_1(Ms, Ms').
\]

2. Rule 1(a) overlaps with 5(b): thus,

\[
\text{cut}^A_2(M, x.\text{cut}^B_1([], Ms))
\]

reduces by 1(a) to

\[
\text{cut}^A_2(M, x.Ms);
\]

we can also reduce it by 5(b) to

\[
\text{cut}^B_1(\text{cut}^A_2(M, x.[], \text{cut}^A_2(M, x.Ms))
\]

which reduces by 2(a) to

\[
\text{cut}^B_1([], \text{cut}^A_2(M, x.Ms))
\]

which reduces by 1(a) to

\[
\text{cut}^A_2(M, x.Ms).}
3. Rule 1(b) overlaps with 5(a): thus,

\[ \text{cut}_A^A(\text{cut}_1^B((M :: \text{Ms}), \text{Ms}')) \]

reduces by 1(b) to

\[ \text{cut}_A^A((M :: \text{cut}_1^B(\text{Ms}, \text{Ms}')), \text{Ms}''') \]

which reduces by 1(b) to

\[ (M :: \text{cut}_A^A(\text{cut}_1^B(\text{Ms}, \text{Ms}'), \text{Ms}''')) \]

which reduces by 5(a) to

\[ (M :: \text{cut}_A^B(\text{Ms}, \text{cut}_A^A(\text{Ms}', \text{Ms}''))) \]

we can also reduce it by 5(a) to

\[ \text{cut}_B^B((M :: \text{Ms}), \text{cut}_A^A(\text{Ms}', \text{Ms}'')) \]

which reduces by 1(b) to

\[ (M :: \text{cut}_B^B(\text{Ms}, \text{cut}_A^A(\text{Ms}', \text{Ms}''))) \].

4. Rule 1(b) overlaps with 5(b): thus,

\[ \text{cut}_A^A(\text{M}, x.\text{cut}_1^B((M' :: \text{Ms}), \text{Ms}')) \]

reduces by 1(b) to

\[ \text{cut}_A^A(\text{M}, x.(M' :: \text{cut}_1^B(\text{Ms}, \text{Ms}'))) \]

which reduces by 2(b) to

\[ \text{cut}_1^A(\text{M}, x.(M' :: \text{cut}_1^B(\text{Ms}, \text{Ms}'))) \]

which reduces by 5(b) to

\[ \text{cut}_1^A(\text{M}, x.M') :: \text{cut}_1^B(\text{cut}_A^A(\text{M}, x.Ms), \text{cut}_A^A(\text{M}, x.Ms')) \];

we can also reduce it by 5(b) to

\[ \text{cut}_1^B(\text{cut}_A^A(\text{M}, x.(M' :: \text{Ms})), \text{cut}_A^A(\text{M}, x.Ms')) \]

which reduces by 2(b) to

\[ \text{cut}_1^B(\text{cut}_A^A(\text{M}, x.M') :: \text{cut}_A^A(\text{M}, x.Ms), \text{cut}_A^A(\text{M}, x.Ms')) \]

which reduces by 1(b) to

\[ \text{cut}_1^B(\text{cut}_A^A(\text{M}, x.M') :: \text{cut}_A^B(\text{M}, x.Ms), \text{cut}_A^A(\text{M}, x.Ms')) \].

5. Rules 2(a), 2(b) have no overlaps.

6. Rule 3(a) overlaps with 5(c): thus,

\[ \text{cut}_A^A(\text{cut}_3^B((\text{x}; \text{Ms}), \text{Ms}'), \text{Ms}'') \]

reduces by 3(a) to

\[ \text{cut}_A^A((\text{x}; \text{cut}_3^B(\text{Ms}, \text{Ms}')), \text{Ms}'') \]

which reduces by 3(a) to

\[ (\text{x}; \text{cut}_A^A(\text{cut}_3^B(\text{Ms}, \text{Ms}'), \text{Ms}'')) \]

which reduces by 5(a) to

\[ (\text{x}; \text{cut}_A^B(\text{Ms}, \text{cut}_A^A(\text{Ms}', \text{Ms}''))) \];

we can also reduce it by 5(c) to

\[ \text{cut}_3^B((\text{x}; \text{Ms}), \text{cut}_A^A(\text{Ms}', \text{Ms}'')) \]

which reduces by 3(a) to

\[ (\text{x}; \text{cut}_A^B(\text{Ms}, \text{cut}_A^A(\text{Ms}', \text{Ms}''))) \].
7. Rule 3(a) overlaps with 5(d). Consider two cases:
   (a) \[ \text{cut}^A_4(M, x.\text{cut}_B^P((x; Ms), Ms')) \]
   reduces by 3(a) to \[ \text{cut}^A_4(M, x.(x; \text{cut}_P^B(Ms, Ms'))) \]
   which reduces by 4(b) to \[ \text{cut}^A_4(M, \text{cut}_B^P(M, x.\text{cut}_P^B(Ms, Ms'))) \]
   which reduces by 5(b) to \[ \text{cut}^B_3(M, \text{cut}_P^B(\text{cut}_A^4(M, x.Ms), \text{cut}_A^4(M, x.Ms'))) \]
   we can also reduce it by 5(d) to \[ \text{cut}^P_3(\text{cut}_A^4(M, x.(x; Ms)), \text{cut}_A^4(M, x.Ms')) \]
   which reduces by 4(b) to \[ \text{cut}^B_3(\text{cut}_A^4(M, x.Ms), \text{cut}_A^4(M, x.Ms')) \]
   which reduces by 5(c) to \[ \text{cut}^A_4(M, \text{cut}_B^P(\text{cut}_A^4(M, x.Ms), \text{cut}_A^4(M, x.Ms'))) \].
   (b) Let \( y \neq x. \)
   \[ \text{cut}^A_4(M, x.\text{cut}_B^P((y; Ms), Ms')) \]
   reduces by 3(a) to \[ \text{cut}^A_4(M, x.(y; \text{cut}_P^B(Ms, Ms'))) \]
   which reduces by 4(a) to \( (y; \text{cut}^A_4(M, x.\text{cut}_P^B(Ms, Ms'))) \)
   which reduces by 5(b) to \( (y; \text{cut}_P^B(\text{cut}_A^4(M, x.Ms), \text{cut}_A^4(M, x.Ms'))) \);
   we can also reduce it by 5(d) to \[ \text{cut}^P_3(\text{cut}_A^4(M, x.(y; Ms)), \text{cut}_A^4(M, x.Ms')) \]
   which reduces by 4(a) to \[ \text{cut}^B_3((y; \text{cut}_A^4(M, x.Ms)), \text{cut}_A^4(M, x.Ms')) \]
   which reduces by 3(a) to \( (y; \text{cut}_B^P(\text{cut}_A^4(M, x.Ms), \text{cut}_A^4(M, x.Ms'))) \).

8. Rule 3(b) overlaps with 5(c): thus,
   \[ \text{cut}^A_3(\text{cut}_B^P(\lambda x.M, []), Ms) \]
   reduces by 3(b) to \[ \text{cut}^A_3(\lambda x.M, Ms); \]
   we can also reduce it by 5(c) to \[ \text{cut}^B_3(\lambda x.M, \text{cut}_A^4([], Ms)) \]
   which reduces by 1(a) to \[ \text{cut}^B_3(\lambda x.M, Ms). \]

Note the apparent change of type information; if we are ignoring types, this is no problem;
and if terms are typed, then, in this case, because of the [] argument, \( A = B \).
9. Rule 3(b) overlaps with 5(d): thus,

\[ \text{cut}_4^A(\lambda y.M, x.\text{cut}_3^B(\lambda z.M', [])) \]

reduces by 3(b) to

\[ \text{cut}_4^A(\lambda y.M, x.\lambda z.M') \]

which reduces by 4(c) to

\[ \lambda z.\text{cut}_4^A(\lambda y.M, x.M') \]

we can also reduce it by 5(d) to

\[ \text{cut}_B^2(\text{cut}_4^A(\lambda y.M, x.\lambda z.M'), \text{cut}_4^A(\lambda y.M, x.[])) \]

which reduces by 2(a) to

\[ \text{cut}_B^2(\text{cut}_4^A(\lambda y.M, x.\lambda z.M'), []) \]

which reduces by 4(c) to

\[ \text{cut}_B^2(\lambda z.\text{cut}_4^A(\lambda y.M, x.M'), []) \]

which reduces by 3(b) to

\[ \lambda z.\text{cut}_4^A(\lambda y.M, x.M') \].

10. Rules 4(a), 4(b), 4(c) and 4(d) have no overlaps.

11. Rule 5(a) overlaps with itself. Consider

\[ \text{cut}_1^4(\text{cut}_B^1(\text{cut}_C^1(Ms, Ms'), Ms''), Ms''') \]

which reduces by 5(a) at a non-root position to

\[ \text{cut}_1^4(\text{cut}_C^1(Ms, \text{cut}_B^1(Ms', Ms'')), Ms''') \]

which reduces by 5(a) again to

\[ \text{cut}_1^2(Ms, \text{cut}_1^4(\text{cut}_B^1(Ms', Ms''), Ms''')) \]

which reduces by 5(a) again to

\[ \text{cut}_1^2(Ms, \text{cut}_1^4(Ms', \text{cut}_1^4(Ms'', Ms'''))). \]

Reduction by 5(a) at the root position, however, produces

\[ \text{cut}_1^B(\text{cut}_1^C(Ms, Ms'), \text{cut}_1^A(Ms'', Ms''')) \]

which reduces by 5(a) to

\[ \text{cut}_1^C(Ms, \text{cut}_1^B(Ms', \text{cut}_1^A(Ms'', Ms'''))). \]

12. Rule 5(a) overlaps with 5(b). Consider

\[ \text{cut}_A^4(M, x.\text{cut}_1^B(\text{cut}_C^1(Ms, Ms'), Ms'')) \]

which reduces by 5(a) to

\[ \text{cut}_A^4(M, x.\text{cut}_1^C(Ms, \text{cut}_1^B(Ms', Ms'')))) \]

which reduces by 5(b) to

\[ \text{cut}_1^C(\text{cut}_A^4(M, x.Ms), \text{cut}_1^B(M, x.\text{cut}_1^B(Ms', Ms'')) \]

which reduces by 5(b) to
\[ \text{cut}_1^B(\text{cut}_4^B(M, x.Ms), \text{cut}_4^B(\text{cut}_4^B(M, x.Ms'), \text{cut}_4^B(M, x.Ms''))}; \]
we can also reduce it by 5(b) to
\[ \text{cut}_1^B(\text{cut}_4^B(M, x.\text{cut}_1^C(Ms, Ms')), \text{cut}_4^B(M, x.Ms'')) \]
which reduces by 5(b) to
\[ \text{cut}_1^B(\text{cut}_4^B(M, x.Ms), \text{cut}_4^B(M, x.Ms'), \text{cut}_4^B(M, x.Ms'')) \]
which reduces by 5(a) to
\[ \text{cut}_1^C(\text{cut}_4^B(M, x.Ms), \text{cut}_4^B(M, x.Ms'), \text{cut}_4^B(M, x.Ms''))]. \]

13. Rule 5(b) has no (further) overlaps.

14. Rule 5(c) overlaps with itself. Consider
\[ \text{cut}_3^A(\text{cut}_5^B(M, Ms), Ms') \]
which reduces by 5(c) at a non-root position to
\[ \text{cut}_3^A(\text{cut}_5^B(M, \text{cut}_1^B(Ms, Ms'))), Ms') \]
which reduces by 5(c) to
\[ \text{cut}_5^C(M, \text{cut}_1^A(\text{cut}_1^B(Ms, Ms'), Ms'')) \]
which reduces by 5(a) to
\[ \text{cut}_5^C(M, \text{cut}_1^B(Ms, \text{cut}_1^A(Ms', Ms''))}. \]
But also, it reduces by 5(c) at the root position to
\[ \text{cut}_5^B(\text{cut}_5^C(M, Ms), \text{cut}_1^A(Ms', Ms'')) \]
and again by 5(c) to
\[ \text{cut}_5^C(M, \text{cut}_1^B(Ms, \text{cut}_1^A(Ms', Ms''))}. \]

15. Rule 5(c) overlaps with 5(d):
\[ \text{cut}_4^A(M, x.\text{cut}_5^B(M'), Ms'), \text{cut}_5^B(M, Ms') \]
reduces by 5(c) to
\[ \text{cut}_4^A(M, x.\text{cut}_5^C(M', Ms')) \]
which reduces by 5(d) to
\[ \text{cut}_5^C(M, x.\text{cut}_1^A(M', \text{cut}_1^B(Ms, Ms'))} \]
which reduces by 5(b) to
\[ \text{cut}_5^C(\text{cut}_4^B(M, x.M'), \text{cut}_1^B(M, x.Ms'), \text{cut}_4^B(M, x.Ms''))]; \]
we can also reduce it by 5(d) to
\[ \text{cut}_5^B(\text{cut}_4^B(M, x.\text{cut}_5^C(M', Ms)), \text{cut}_4^B(M, x.Ms')) \]
which reduces by 5(d) to
\[ \text{cut}_5^B(\text{cut}_5^C(\text{cut}_4^B(M, x.M'), \text{cut}_4^B(M, x.Ms'), \text{cut}_4^B(M, x.Ms''))} \]
which reduces by 5(c) to
\[ \text{cut}_5^C(\text{cut}_4^B(M, x.M'), \text{cut}_5^B(\text{cut}_4^B(M, x.Ms'), \text{cut}_4^B(M, x.Ms''))}. \]

16. Rule 5(d) has no (further) overlaps.
C Proof of Proposition 9

Note that in each case the term on the RHS is a pure term; by definition the functions @, [], and ., all produce pure terms from pure arguments. The proposition follows from the following cases formulated as lemmata.

**Lemma 34.** For pure terms $Ms, Ms'$,

$$\text{cut}_1(Ms, Ms') \sim_{E_{S+CC}} Ms@Ms'.$$

*Proof.* Routine, by induction on the length of $Ms$ which, being pure, is already known to be a list.

**Lemma 35.** For pure terms $M, Ms$,

$$\text{cut}_2(M, Ms) \sim_{E_{S+CC}} \{M\}Ms.$$

*Proof.* By induction on the height of the term $M$ and case analysis:

1. When $M$ is of the form $(y; Ms')$ the LHS is

$$\text{cut}_2(y; Ms', Ms)$$

which reduces by 3(a) to $(y; \text{cut}_1(Ms', Ms))$, which reduces by Lemma 34 to $(y; Ms'@Ms)$, which is just $\{(y; Ms')\}Ms$.

2. When $M$ is of the form $\lambda x.M'$, there are two cases:
   (a) When $Ms = []$; this is easy.
   (b) When $Ms = (M'' :: Ms')$, the LHS of (3) is $\text{cut}_2(\lambda x.M', (M'' :: Ms'))$ which is (by definition) the same term as $\{\lambda x.M'\}Ms$ without any reduction.

3. When $M$ is of the form $\text{cut}_2(\lambda y.M', M'' :: Ms')$, the LHS of (3) is

$$\text{cut}_2(\text{cut}_2(\lambda y.M', M'' :: Ms'), Ms)$$

which reduces by 5(c) to

$$\text{cut}_2(\lambda y.M', \text{cut}_1(M'' :: Ms', Ms))$$

which reduces by 1(b) to

$$\text{cut}_2(\lambda y.M', (M'' :: \text{cut}_1(Ms', Ms))).$$

which by Lemma 34 reduces to

$$\text{cut}_2(\lambda y.M', M'' :: (Ms'@Ms)).$$

which is just $\{M\}Ms$.

**Lemma 36.** For pure terms $M, Ms, M'$,

1. $\text{cut}_3(M, x.Ms) \sim_{E_{S+CC}} [M/x]Ms$;

2. $\text{cut}_4(M, x.M') \sim_{E_{S+CC}} [M/x]M'$.

*Proof.* By simultaneous induction on the heights of the terms and case analysis. We consider the cases systematically:

1. (a) When $Ms$ is $[]$, the LHS reduces by 2(a) to $[]$, which is just $[M/x][]$.
   (b) When $Ms$ is $(M' :: Ms')$, the LHS reduces by 2(b) to $\text{cut}_4(M, x.M') :: \text{cut}_3(M, x.Ms')$.

The first part of this reduces, by inductive hypothesis, to $[M/x]M'$; the second reduces, by inductive hypothesis, to $[M/x]Ms'$; their combination by $::$ is just $[M/x](M' :: Ms')$, i.e. $[M/x]Ms$. 

2. (a) When $M'$ is of the form $(y; M's)$ (for $y \neq x$), the LHS reduces by 4(a) to

$$(x; cut_3(M, x. Ms'))$$

which by inductive hypothesis reduces to

$$(y; [M/x] Ms')$$

which is just

$[M/x](y; Ms').$ (b) When $M'$ is of the form $(x; Ms')$, the LHS reduces by 4(b) to

$cut_3(M, cut_3(M, x. Ms'))$

which by inductive hypothesis reduces to

$cut_3(M, [M/x] Ms')$

which by Lemma 35 reduces to

$$\{ M \} [M/x] Ms'$$

which is just

$$[M/x](x; Ms').$$

(c) When $M'$ is of the form $\lambda y. M''$, this is easy.

(d) When $M'$ is of the form $cut_3(\lambda y. M''$, $(M'' :: Ms')$, the LHS of (4) is

$cut_3(M, x. cut_3(\lambda y. M'', (M'' :: Ms'))$)

which reduces by 5(d) to

$cut_3(cut_4(M, x. \lambda y. M''), cut_3(M, x. (M'' :: Ms')))$

which reduces by 4(c) and 2(b) to

$cut_3(\lambda y. cut_4(M, x. M''), (cut_4(M, x. M'' :: cut_3(M, x. Ms')))$

which by inductive hypothesis (three times) reduces to


which is just

$$[M/x]cut_3(\lambda y. M'', (M'' :: Ms')).$$
D Simulation of Lambda-x

In this section we show that the $\lambda x$-calculus of Bloo and Rose [3] can be simulated by the reduction system $ES + CC + B_i$ if we add the following reduction rules

\[
\begin{align*}
\mathcal{cut}_3^x(M_s, []) & \sim M_s \\
\mathcal{cut}_3^x(M, []) & \sim M
\end{align*}
\]

These rules are harmless with respect to confluence and strong normalisation.

The terms of $\lambda x$ are given by the following grammar:

\[N ::= x \mid ((\lambda x.N) \mid NN \mid N(x := N)).\]

In $\lambda x$ the beta-reduction

\[
\beta \quad (\lambda x.N)N' \sim \beta \quad [N'/x]N
\]

is replaced by the reduction

\[
\beta \quad (\lambda x.N)N' \sim \beta \quad N(x := N')
\]

where the reduct contains the constructor for explicit substitutions. The following reduction rules apply to this term constructor.

\[
\begin{align*}
x1 & \quad x(x := N) \sim x1 \quad N \\
x2 & \quad y(x := N) \sim x2 \quad y \\
x3 & \quad (\lambda y.N')(x := N) \sim x3 \quad \lambda y. N'(x := N) \\
x4 & \quad (N'N')(x := N) \sim x4 \quad N'(x := N)N'(x := N)
\end{align*}
\]

We translate $\lambda x$-terms into Herbelin’s calculus as follows:

\[
\begin{align*}
(x)^* & \equiv_{df} (x; []) \\
(\lambda x.N)^* & \equiv_{df} \lambda x.N* \\
(NN')^* & \equiv_{df} \mathcal{cut}_3(N^*, N^*: []) \\
(N(x := N))^* & \equiv_{df} \mathcal{cut}_3(N^*, x.N^*)
\end{align*}
\]

The simulation is then as follows:

1. Rule $x1$ is mapped onto the reduction sequence

\[
\mathcal{cut}_3^x(\lambda x.N^*, N^*: []) \sim \mathcal{cut}_3^x(\mathcal{cut}_3(N^*, x.N^*), [] \sim \mathcal{cut}_3(N^*, x.N^*)
\]

2. Rule $x2$ is mapped onto the reduction sequence

\[
\mathcal{cut}_3^x(\lambda x.N^*, x.(x; [[]]) \sim \mathcal{cut}_3^x(\lambda x(N^*, x.N^*), [[]] \sim \mathcal{cut}_3(N^*, [], []) \sim N^*
\]

3. Rule $x3$ is mapped onto the reduction sequence

\[
\mathcal{cut}_3^x(N^*, x.(y; [[]]) \sim (y; \mathcal{cut}_3(N^*, x.[[]]) \sim (y; []
\]

4. Rule $x3$ is mapped onto the reduction sequence

\[
\mathcal{cut}_3^x(N^*, x.\lambda y.N^*) \sim \lambda y.\mathcal{cut}_3^x(N^*, x.N^*)
\]

5. Rule $x4$ is mapped onto the reduction sequence

\[
\begin{align*}
\mathcal{cut}_3^x(N^*, x.\mathcal{cut}_3(N^*, N^*: [[]]) \sim \mathcal{cut}_3^x(\mathcal{cut}_3(N^*, x.N^*), \mathcal{cut}_3(N^*, x.N^*: [[]]) \\
& \sim \mathcal{cut}_3^x(\mathcal{cut}_3(N^*, x.N^*), \mathcal{cut}_3(N^*, x.N^*: [[]]) \sim \mathcal{cut}_3^x(N^*, [x]) \\
& \sim \mathcal{cut}_3^x(N^*, x.N^*) \mathcal{cut}_3^x(N^*, x.N^*: [[]])
\end{align*}
\]