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

Roy Dyckhoff[1]* and Christian Urban[2]

[1] University of St Andrews rd@dcs.st-and.ac.uk
[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 λ-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 β-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 β-reduction of the simply typed λ-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 β-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 cut_2 and cut_4 terms. *Contexts* Γ are finite functions from variables to formulae; $\Gamma, x : A$ 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

$$\begin{aligned} Ms &::= [] \mid (M :: Ms) \\ M &::= (x; Ms) \mid \lambda x.M \end{aligned}$$

Logical/Typing rules

$$\frac{\Gamma \Rightarrow M : A \quad \Gamma \xrightarrow[B]{---} Ms : C}{\Gamma \xrightarrow[A \supset B]{---} (M :: Ms) : C}\ S\supset \qquad \frac{}{\Gamma \xrightarrow[A]{---} [] : A}\ Ax$$

$$\frac{\Gamma, x : A \xrightarrow[A]{---} Ms : B}{\Gamma, x : A \Rightarrow (x; Ms) : B}\ Sel \qquad \frac{\Gamma, x : A \Rightarrow M : B}{\Gamma \Rightarrow \lambda x.M : A \supset B}\ R\supset$$

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

$$\begin{aligned} Ms &::= cut_1^A(Ms, Ms) \mid cut_2^A(M, x.Ms) \\ M &::= cut_3^A(M, Ms) \mid cut_4^A(M, x.M) \end{aligned}$$

Logical/Typing rules for Cuts

$$\frac{\Gamma \xrightarrow[B]{---} Ms : A \quad \Gamma \xrightarrow[A]{---} Ms' : C}{\Gamma \xrightarrow[B]{---} cut_1^A(Ms, Ms') : C}\ Cut_1 \qquad \frac{\Gamma \Rightarrow M : A \quad \Gamma, x : A \xrightarrow[B]{---} Ms : C}{\Gamma \xrightarrow[B]{---} cut_2^A(M, x.Ms) : C}\ Cut_2$$

$$\frac{\Gamma \Rightarrow M : A \quad \Gamma \xrightarrow[A]{---} Ms : B}{\Gamma \Rightarrow cut_3^A(M, Ms) : B}\ Cut_3 \qquad \frac{\Gamma \Rightarrow M : A \quad \Gamma, x : A \Rightarrow M' : B}{\Gamma \Rightarrow cut_4^A(M, x.M') : B}\ Cut_4$$

Rules for cut-reduction

Let ES denote the system of "explicit substitution" rules $1 \ldots 4$ on terms, and CC the system of "commuting cuts" rule 5 on terms:

1. (a) $cut_1^A([], Ms) \rightsquigarrow Ms$
   (b) $cut_1^A((M :: Ms), Ms') \rightsquigarrow (M :: cut_1^A(Ms, Ms'))$
2. (a) $cut_2^A(M, x.[]) \rightsquigarrow []$
   (b) $cut_2^A(M, x.(M' :: Ms)) \rightsquigarrow (cut_4^A(M, x.M') :: cut_2^A(M, x.Ms))$
3. (a) $cut_3^A((x; Ms), Ms') \rightsquigarrow (x; cut_1^A(Ms, Ms'))$
   (b) $cut_3^A(\lambda y.M, []) \rightsquigarrow \lambda y.M$
4. (a) $cut_4^A(M, x.(y; Ms)) \rightsquigarrow (y; cut_2^A(M, x.Ms)) \quad (y \neq x)$
   (b) $cut_4^A(M, x.(x; Ms)) \rightsquigarrow cut_3^A(M, cut_2^A(M, x.Ms))$
   (c) $cut_4^A(M, x.\lambda y.M') \rightsquigarrow \lambda y.cut_4^A(M, x.M')$
5. (a) $cut_1^A(cut_1^B(Ms, Ms'), Ms'') \rightsquigarrow cut_1^B(Ms, cut_1^A(Ms', Ms''))$
   (b) $cut_2^A(M, x.cut_1^B(Ms, Ms')) \rightsquigarrow cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms'))$
   (c) $cut_3^A(cut_3^B(M, Ms), Ms') \rightsquigarrow cut_3^B(M, cut_1^A(Ms, Ms'))$
   (d) $cut_4^A(M, x.cut_3^B(M', Ms)) \rightsquigarrow cut_3^B(cut_4^A(M, x.M'), cut_2^A(M, x.Ms))$

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 $\quad cut_3^{A\supset B}(\lambda x.M, (M' :: Ms)) \rightsquigarrow cut_3^B(cut_4^A(M', x.M), Ms)$

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 $cut^A > cut^B$ for $A > B$ and with $cut_4 = cut_2 > cut_3 = cut_1$ 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 cut_2 or cut_4 operators with cut_2 or 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. $cut_3((x; Ms), Ms')$ is a 3(a)-redex;
2. $cut_3(\lambda x.M', [])$ is a 3(b)-redex;
3. $cut_3(\lambda x.M', (M'' :: Ms))$ is a B-redex;
4. $cut_3(cut_3(M', Ms), Ms')$ is a 5(c)-redex;
5. $cut_3(cut_4(M', x.M''), Ms)$ is not allowed, since (by inductive hypothesis) $cut_4(M', x.M'')$ is pure, contrary to its being a (non B-redex) *cut*-term;
6. $cut_4(M', x.(z; Ms))$ is a 4(a) or 4(b)-redex;
7. $cut_4(M', x.\lambda y.M'')$ is a 4(c)-redex;
8. $cut_4(M', x.cut_3(M'', Ms))$ is a 5(d)-redex;

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

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] $Ms@Ms'$ *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:

$$
\begin{array}{rcl}
[]@Ms' & =_{def} & Ms' \\
(M'' :: Ms'')@Ms' & =_{def} & (M'' :: (Ms''@Ms'))
\end{array}
$$

$$
\begin{array}{rcl}
[M/x][] & =_{def} & [] \\
[M/x](M'' :: Ms'') & =_{def} & ([M/x]M'' :: [M/x]Ms'')
\end{array}
$$

$$
\begin{array}{rcl}
\{(y; Ms'')\}Ms & =_{def} & (y; (Ms''@Ms)) \\
\{\lambda y.M''\}[] & =_{def} & \lambda y.M'' \\
\{\lambda y.M''\}(M''' :: Ms'') & =_{def} & cut_3(\lambda y.M'', (M''' :: Ms'')) \\
\{cut_3(\lambda y.M'', (M''' :: Ms''))\}Ms & =_{def} & cut_3(\lambda y.M'', (M''' :: (Ms''@Ms)))
\end{array}
$$

$$
\begin{array}{rcll}
[M/x](y; Ms'') & =_{def} & (y; [M/x]Ms'') & (y \neq x) \\
[M/x](x; Ms'') & =_{def} & \{M\}[M/x]Ms'' & \\
[M/x](\lambda y.M'') & =_{def} & \lambda y.[M/x]M'' & \\
[M/x]cut_3(\lambda y.M'', (M''' :: Ms'')) & =_{def} & cut_3(\lambda y.[M/x]M'', ([M/x]M''' :: [M/x]Ms'')). &
\end{array}
$$

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

$$
\begin{array}{rcl}
N & ::= & (x\ Ns) \mid (\lambda x.N) \mid ((\lambda x.N)NNs) \\
Ns & ::= & [] \mid (N :: Ns)
\end{array}
$$

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):

$$\begin{array}{rcl}
[N/x][] & =_{def} & [] \\
[N/x](N' :: Ns) & =_{def} & ([N/x]N' :: [N/x]Ns)
\end{array}$$

$$\begin{array}{rcl}
\{(y\ Ns')\}Ns & =_{def} & (y\ (Ns'@Ns)) \\
\{(\lambda y.N)\}[] & =_{def} & (\lambda y.N) \\
\{(\lambda y.N)\}(N' :: Ns) & =_{def} & ((\lambda y.N)N'Ns) \\
\{((\lambda y.N)N'Ns)\}Ns' & =_{def} & ((\lambda y.N)N'(Ns@Ns'))
\end{array}$$

$$\begin{array}{rcl}
[N/x](y\ Ns) & =_{def} & (y\ [N/x]Ns) \quad (y \neq x) \\
[N/x](x\ Ns) & =_{def} & \{N\}[N/x]Ns \\
[N/x](\lambda y.N') & =_{def} & (\lambda y.[N/x]N') \\
[N/x]((\lambda y.N')N''Ns) & =_{def} & ((\lambda y.[N/x]N')([N/x]N''[N/x]Ns))
\end{array}$$

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

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

β-reduction is thus the reduction of a term by careful replacement of a subterm (the *β-redex*) of the form $((\lambda x.N)N'Ns)$ by the *reduct* $\{[N'/x]N\}Ns$ in a single step.

5 Interpretation of Lambda Calculus

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

$$\begin{array}{rcl}
(x\ Ns)^\bullet & =_{def} & (x; Ns^{\bullet\bullet}) \\
(\lambda x.N)^\bullet & =_{def} & (\lambda x.N^\bullet) \\
((\lambda x.N)N'Ns)^\bullet & =_{def} & cut_3(\lambda x.N^\bullet, (N'^\bullet :: Ns^{\bullet\bullet}))
\end{array}$$

$$\begin{array}{rcl}
[]^{\bullet\bullet} & =_{def} & [] \\
(N :: Ns)^{\bullet\bullet} & =_{def} & (N^\bullet :: Ns^{\bullet\bullet})
\end{array}$$

Proposition 4. *For λ-terms N, N' and term lists Ns, Ns', the following hold:*

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

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 $cut_2 = cut_4$ 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 $M \rightsquigarrow^*_{ES+CC} M'$, then $\overline{M} = \overline{M'}$. (Similarly for Ms.)*

Proof. Trivial. ■

7 Simulation of β-reduction

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

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

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-B)-redex, repeating this operation until all such redexes are eliminated.

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

$$cut_3(cut_4(M', x.M), Ms) \rightsquigarrow^*_{ES+CC} \{[M'/x]M\}Ms.$$ ■

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

Proof. Consider first the case where N_1 is the β-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

$$cut_3(cut_4(N'^\bullet, x.N^\bullet), Ns^{\bullet\bullet})$$

of the B-redex $N_1^\bullet = cut_3(\lambda x.N^\bullet, (N'^\bullet :: Ns^{\bullet\bullet}))$ is $(ES + CC)$-reducible to $\{[N'^\bullet/x]N^\bullet\}Ns^{\bullet\bullet}$. By Proposition 4, this is just $N_2^\bullet$. 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* β-reduction of the untyped λ-calculus; similarly for typed terms and the typed λ-calculus.

8 β-reduction

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

$\beta \qquad cut_3^{A\supset B}(\lambda x.M, M' :: Ms) \rightsquigarrow_\beta \overline{cut_3^B(cut_4^A(M', x.M), Ms)}$

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 λ-calculus routinely extends to β-reduction.

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

Proposition 12. *The system, on pure typed terms, consisting just of the rule β is SN.*

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 λ-calculus.

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

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

Proof. By Proposition 12 and König's Lemma. ■

Lemma 15. *For pure terms M, Ms, M', M'', Ms', with $M \rightsquigarrow_\beta^* M'$ and $Ms \rightsquigarrow_\beta^* Ms'$, 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. $[M''/x]M \rightsquigarrow_\beta^* [M''/x]M'$.

Proof. These follow from consideration of the untyped λ-calculus: for example, substitution into a β-redex leaves it as a β-redex. ■

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

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

9 Adding B-reduction

We will show that the addition of the B-rule upsets 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 β-reduction.

Lemma 17. $\overline{cut_3(\lambda x.M, (M' :: Ms))} \rightsquigarrow_\beta \overline{cut_3(cut_4(M', x.M), Ms)}$.

Proof. The $LHS = cut_3(\lambda x.\overline{M}, (\overline{M'} :: \overline{Ms}))$, the latter (as a B-redex) being pure; this B-reduces to $cut_3(cut_4(\overline{M'}, x.\overline{M}), \overline{Ms})$ and thus β-reduces to $\overline{cut_3(cut_4(\overline{M'}, x.\overline{M}), \overline{Ms})}$. By Lemma 8, this equals the RHS. ■

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

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 $< \infty$)*

$$\|cut_3(\lambda x.M, (M' :: Ms))\| > \|cut_3(cut_4(M', x.M), Ms)\|.$$ ■

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

Proposition 20. *The following hold:*

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

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) $Ms = cut_1(Ms'', Ms''')$: similar to the first two parts of case 2(c) below.
   (d) $Ms = cut_2(M'', x.Ms''')$: similar to the case 2(d) below.
2. (a) $M = \lambda x.M''$: routine;
   (b) $M = (x; Ms'')$: routine;
   (c) $M = cut_3(M'', Ms'')$: there are three cases:
      i. the B-reduction is of M'' to M''': by inductive hypothesis, $\overline{M''} \rightsquigarrow_\beta^* \overline{M'''}$, whence, by Lemma 15 (1), $\{\overline{M''}\}\overline{Ms''} \rightsquigarrow_\beta^* \{\overline{M'''}\}\overline{Ms''}$. Now,

$$\overline{cut_3(M'', Ms'')} = \overline{cut_3(\overline{M''}, \overline{Ms''})} = \{\overline{M''}\}\overline{Ms''}$$

      by Lemma 8 and Proposition 9 (3) respectively; and similarly

$$\overline{cut_3(M''', Ms'')} = \overline{cut_3(\overline{M'''}, \overline{Ms''})} = \{\overline{M'''}\}\overline{Ms''}$$

      whence $\overline{M} \rightsquigarrow_\beta^* \overline{M'}$.
      ii. the B-reduction is of Ms'' to Ms''': similar, using Lemma 15 (2).
      iii. the B-reduction is at the root of M: we use the Projection Lemma.
   (d) $M = cut_4(M'', x.M''')$: 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 β-reduction in the λ-calculus.

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

Proof. Suppose that $M \rightsquigarrow^*_{ES+CC+B} M_1$ and $M \rightsquigarrow^*_{ES+CC+B} M_2$. Then, by Lemma 8 and Proposition 20, both $\overline{M} \rightsquigarrow_\beta^* \overline{M_1}$ and $\overline{M} \rightsquigarrow_\beta^* \overline{M_2}$, whence, by confluence of β-reduction in the λ-calculus, for some (pure) term M° we have $\overline{M_1} \rightsquigarrow_\beta^* M^\circ$ and $\overline{M_2} \rightsquigarrow_\beta^* M^\circ$. But then, both $M_1 \rightsquigarrow^*_{ES+CC+B} M^\circ$ and also $M_2 \rightsquigarrow^*_{ES+CC+B} M^\circ$. (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, β-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 Ms.)
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). 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 $M \rightsquigarrow_{ES+CC} M'$ then $\| M \| = \| M' \|$;*
2. *If $Ms \rightsquigarrow_{ES+CC} Ms'$ then $\| Ms \| = \| Ms' \|$;*
3. *If $M \rightsquigarrow_{B} M'$ then $\| M \| \geq \| M' \|$;*
4. *If $Ms \rightsquigarrow_{B} Ms'$ 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 $cut_1(cut_2(M, x.Ms), cut_2(M, x.Ms'))$, the two cut_2 subterms are the only cosily embedded proper subterms; and in a term of the form $cut_2(M, x.Ms)$, 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)* $\| cut_1(Ms, Ms') \| \geq \| Ms \|$;
   (b) $\| cut_1(Ms, Ms') \| \geq \| Ms' \|$;
5. $\| cut_3(M, Ms) \| \geq \| M \|$;
6. $\| cut_3(M, Ms) \| \geq \| Ms \|$.

Proof. 1. Because $\overline{(M :: Ms)} = (\overline{M} :: \overline{Ms})$;
2. Because $\overline{(x; Ms)} = (x; \overline{Ms})$;
3. Because $\overline{\lambda x.M} = \lambda x.\overline{M}$;
4. Because, by Proposition 9 (1), $\overline{cut_1(Ms, Ms')} = \overline{Ms} @ \overline{Ms'}$;
5. Since $\overline{cut_3(M, Ms)} = \overline{cut_3(\overline{M}, \overline{Ms})}$ 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 $\| (x; Ms'' @ Ms) \| = \| Ms'' \| + \| Ms \|$;
   (b) $M = \lambda x.M'$; so $cut_3(M, Ms)$ is pure and has Ms as a sub-term, whence $\| cut_3(M, Ms) \| \geq \| Ms \|$;

(c) $M = cut_3(M', Ms'')$; so, by rule 5(c), $\overline{cut_3(M, Ms)} = \overline{cut_3(M', Ms''@Ms)}$ and, by inductive hypothesis, $\| cut_3(M', Ms''@Ms) \| \geq \| Ms''@Ms \| \geq \| Ms \|$. ■

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

Proof. By the lemma, using induction on the number of constructors between the subterm and the term. ■

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

$$\| M \| \not\leq \| cut_2(M, x.[]) \| = 0$$

and

$$\| M \| \not\leq \| cut_4(M, x.\lambda y.(y; [])) \| = 0.$$

Corollary 28. *We have the following:*

1. *For each of the $(ES + CC)$-reduction rules $L \rightsquigarrow R$, and for each non-variable subterm S of R, and instantiation θ of the variables in the rule, we have $\| L^\theta \| \geq \| S^\theta \|$;*
2. *For the B-reduction rule $L \rightsquigarrow R$, and for each non-variable subterm S of R, and instantiation θ 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 $L \rightsquigarrow R$, each non-variable proper subterm of R is cosily embedded.
2. By Lemma 18, assuming $\| L^\theta \| < \infty$, we have $\| L^\theta \| > \| R^\theta \|$; as before, the only non-variable proper sub-term S of R is cosily embedded. ■

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 $M \rightsquigarrow_R M'$. 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 $M^* \rightsquigarrow M''$. 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 $\| M'' \| < \infty$. ■

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 $cut^n > cut^m$ for $n > m$, use the suffices $(4 = 2 > 3 = 1)$ as before for ordering cut operators with the same superfix, 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 "$>_{LPO}$" 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 $M' \rightsquigarrow_{EC+CC+B} M''$, then $M' >_{LPO} M''$. (Similarly for Ms.)*

Proof. It suffices to consider only reductions at root position, since $>_{LPO}$ 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

$$M' = cut_3^m(\lambda x.M, (M''' :: Ms)) \rightsquigarrow_B cut_3^r(cut_4^s(M''', x.M), Ms) = M''$$

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 $M' >_{LPO} cut_4^s(M''', x.M)$ follows because $m > s$ and M''', M are variables properly occurring in M'; that $M' >_{LPO} Ms$ is trivial, the latter being a variable properly occurring in M'. It follows that $M' >_{LPO} M''$. ■

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

Proof. By the well-foundedness of $>_{LPO}$ 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 β-reduction.* ■

11 Comments

It would be interesting to find a direct normalisation proof (in sequent calculus notation) for the simply-typed λ-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 $\overline{\lambda}22$ and $\overline{\lambda}44$ 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 λ_H of terms (similar to our calculus of pure terms);
2. shows that λ_H can be extended to a calculus λ_H^+ of terms which, in our notation, have no instances of cut_1 and 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 cut_1 and 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, cut_3 and 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 commmunication, March 2001) conjectured that our SN result for $(ES + CC + B)$ also follows from the SN result [9] for the $\overline{\lambda}\mu\overline{\mu}$-calculus, mentioning however that a similar conjecture for the strong normalisability of Parigot's $\lambda\mu$-calculus 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 $\lambda\mathbf{x}$-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

1. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. R. Bloo and H. Geuvers. Explicit Substitution: On the Edge of Strong Normalisation. *Theoretical Computer Science*, 211(1–2):375–395, 1999.
3. R. Bloo and K. H. Rose. Preservation of Strong Normalisation in Named Lambda Calculi with Explicit Substitution and Garbage Collection. In Proceedings of CSN'95, 1995, pp 62–72.
4. I. Cervesato and F. Pfenning. A Linear Spine Calculus. School of Computer Science, Carnegie Mellon University, Pittsburgh, Pa., CMU-CS-97-125, 1997.
5. R. Dyckhoff and L. Pinto. Proof Search in Constructive Logic. In S. B. Cooper and J. K. Truss, editors, *Proceedings of the Logic Colloquium 1997*, volume 258 of *London Mathematical Society Lecture Note Series*, pages 53–65. Cambridge University Press, 1997.
6. R. Dyckhoff and L. Pinto. Cut-Elimination and a Permutation-Free Sequent Calculus for Intuitionistic Logic. *Studia Logica*, 60(1):107–118, 1998.
7. J. C. Espírito Santo. Revisiting the Correspondence between Cut Elimination and Normalisation. In *Proceedings of ICALP 2000*, volume 1853 of *LNCS*, pages 600–611. Springer Verlag, 2000.
8. H. Herbelin. A λ-calculus Structure Isomorphic to Sequent Calculus Structure. In *Proceedings of the 1994 conference on Computer Science Logic*, volume 933 of *LNCS*, pages 67–75. Springer Verlag, 1995.
9. H. Herbelin. Explicit Substitutions and Reducibility. Journal of Logic and Computation, Vol 11 (3), pages 429–449, 2001.
10. F. Joachimski and R. Matthes. Short Proofs of Normalisation. Archive for Mathematical Logic, to appear. (Preprint, 1999.)
11. K. H. Rose. Explicit Substitution: Tutorial & Survey. Technical report, BRICS, Department of Computer Science, University of Aarhus, 1996.
12. F. van Raamsdonk and P. Severi. On Normalisation. Technical report, TR CS-R9545, Centrum voor Wiskunde en Informatica, Amsterdam, 1996. (incorporated into [13].)
13. F. van Raamsdonk, P. Severi, M. H. Sörensen and H. Xi. Perpetual Reductions in Lambda Calculus. Information and Computation **149**, pages 173–225, 1999.
14. R. Vestergaard and J. Wells. Cut Rules and Explicit Substitutions. Mathematical Structures in Computer Science, to appear.

A Proof of Proposition 5

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

$$\begin{aligned} h([]) &= 1 \\ h((M :: Ms)) &= h(M) + h(Ms) + 1 \\ h(cut_1(Ms, Ms')) &= h(Ms') + 2 * h(Ms) + 1 \\ h(cut_2(M, x.Ms)) &= h(Ms) * (3 * h(M) + 1) \end{aligned}$$

$$\begin{aligned} h((x; Ms)) &= h(Ms) + 1 \\ h((\lambda x.M)) &= h(M) + 1 \\ h(cut_3(M, Ms)) &= h(Ms) + 2 * h(M) + 1 \\ h(cut_4(M, x.M')) &= h(M') * (3 * h(M) + 1) \end{aligned}$$

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,

   $$cut_1^A(cut_1^B([], Ms), Ms')$$

   reduces by 1(a) to

   $$cut_1^A(Ms, Ms');$$

   we can also reduce it by 5(a) to

   $$cut_1^B([], cut_1^A(Ms, Ms'))$$

   which reduces by 1(a) to

   $$cut_1^A(Ms, Ms').$$

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

   $$cut_2^A(M, x.cut_1^B([], Ms))$$

   reduces by 1(a) to

   $$cut_2^A(M, x.Ms);$$

   we can also reduce it by 5(b) to

   $$cut_1^B(cut_2^A(M, x.[]), cut_2^A(M, x.Ms))$$

   which reduces by 2(a) to

   $$cut_1^B([], cut_2^A(M, x.Ms))$$

   which reduces by 1(a) to

   $$cut_2^A(M, x.Ms).$$

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

$$cut_1^A(cut_1^B((M :: Ms), Ms'), Ms'')$$

reduces by 1(b) to

$$cut_1^A((M :: cut_1^B(Ms, Ms')), Ms'')$$

which reduces by 1(b) to

$$(M :: cut_1^A(cut_1^B(Ms, Ms'), Ms''))$$

which reduces by 5(a) to

$$(M :: cut_1^B(Ms, cut_1^A(Ms', Ms'')));$$

we can also reduce it by 5(a) to

$$cut_1^B((M :: Ms), cut_1^A(Ms', Ms''))$$

which reduces by 1(b) to

$$(M :: cut_1^B(Ms, cut_1^A(Ms', Ms''))).$$

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

$$cut_2^A(M, x.cut_1^B((M' :: Ms), Ms'))$$

reduces by 1(b) to

$$cut_2^A(M, x.(M' :: cut_1^B(Ms, Ms')))$$

which reduces by 2(b) to

$$(cut_4^A(M, x.M') :: cut_2^A(M, x.cut_1^B(Ms, Ms'))$$

which reduces by 5(b) to

$$(cut_4^A(M, x.M') :: cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms')));$$

we can also reduce it by 5(b) to

$$cut_1^B(cut_2^A(M, x.(M' :: Ms)), cut_2^A(M, x.Ms'))$$

which reduces by 2(b) to

$$cut_1^B((cut_4^A(M, x.M') :: cut_2^A(M, x.Ms)), cut_2^A(M, x.Ms'))$$

which reduces by 1(b) to

$$(cut_4^A(M, x.M') :: cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms'))).$$

5. Rules 2(a), 2(b) have no overlaps.
6. Rule 3(a) overlaps with 5(c): thus,

$$cut_3^A(cut_3^B((x; Ms), Ms'), Ms'')$$

reduces by 3(a) to

$$cut_3^A((x; cut_1^B(Ms, Ms')), Ms'')$$

which reduces by 3(a) to

$$(x; cut_1^A(cut_1^B(Ms, Ms'), Ms''))$$

which reduces by 5(a) to

$$(x; cut_1^B(Ms, cut_1^A(Ms', Ms'')));$$

we can also reduce it by 5(c) to

$$cut_3^B((x; Ms), cut_1^A(Ms', Ms''))$$

which reduces by 3(a) to

$$(x; cut_1^B(Ms, cut_1^A(Ms', Ms''))).$$

7. Rule 3(a) overlaps with 5(d). Consider two cases:
   (a)
   $$cut_4^A(M, x.cut_3^B((x; Ms), Ms'))$$
   reduces by 3(a) to
   $$cut_4^A(M, x.(x; cut_1^B(Ms, Ms')))$$
   which reduces by 4(b) to
   $$cut_3^A(M, cut_2^A(M, x.cut_1^B(Ms, Ms')))$$
   which reduces by 5(b) to
   $$cut_3^A(M, cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms')));$$
   we can also reduce it by 5(d) to
   $$cut_3^B(cut_4^A(M, x.(x; Ms)), cut_2^A(M, x.Ms'))$$
   which reduces by 4(b) to
   $$cut_3^B(cut_3^A(M, cut_2^A(M, x.Ms)), cut_2^A(M, x.Ms'))$$
   which reduces by 5(c) to
   $$cut_3^A(M, cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms'))).$$
   (b) Let $y \neq x$.
   $$cut_4^A(M, x.cut_3^B((y; Ms), Ms'))$$
   reduces by 3(a) to
   $$cut_4^A(M, x.(y; cut_1^B(Ms, Ms')))$$
   which reduces by 4(a) to
   $$(y; cut_2^A(M, x.cut_1^B(Ms, Ms')))$$
   which reduces by 5(b) to
   $$(y; cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms')));$$
   we can also reduce it by 5(d) to
   $$cut_3^B(cut_4^A(M, x.(y; Ms)), cut_2^A(M, x.Ms'))$$
   which reduces by 4(a) to
   $$cut_3^B((y; cut_2^A(M, x.Ms)), cut_2^A(M, x.Ms'))$$
   which reduces by 3(a) to
   $$(y; cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms'))).$$
8. Rule 3(b) overlaps with 5(c): thus,
   $$cut_3^A(cut_3^B(\lambda x.M, []), Ms)$$
   reduces by 3(b) to
   $$cut_3^A(\lambda x.M, Ms);$$
   we can also reduce it by 5(c) to
   $$cut_3^B(\lambda x.M, cut_1^A([], Ms))$$
   which reduces by 1(a) to
   $$cut_3^B(\lambda x.M, Ms).$$
   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,

$$cut_4^A(\lambda y.M, x.cut_3^B(\lambda z.M', []))$$

reduces by 3(b) to

$$cut_4^A(\lambda y.M, x.\lambda z.M')$$

which reduces by 4(c) to

$$\lambda z.cut_4^A(\lambda y.M, x.M');$$

we can also reduce it by 5(d) to

$$cut_3^B(cut_4^A(\lambda y.M, x.\lambda z.M'), cut_2^A(\lambda y.M, x.[]))$$

which reduces by 2(a) to

$$cut_3^B(cut_4^A(\lambda y.M, x.\lambda z.M'), [])$$

which reduces by 4(c) to

$$cut_3^B(\lambda z.cut_4^A(\lambda y.M, x.M'), [])$$

which reduces by 3(b) to

$$\lambda z.cut_4^A(\lambda y.M, x.M').$$

10. Rules 4(a), 4(b), 4(c) and 4(d) have no overlaps.
11. Rule 5(a) overlaps with itself. Consider

$$cut_1^A(cut_1^B(cut_1^C(Ms, Ms'), Ms''), Ms''')$$

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

$$cut_1^A(cut_1^C(Ms, cut_1^B(Ms', Ms'')), Ms''')$$

which reduces by 5(a) again to

$$cut_1^C(Ms, cut_1^A(cut_1^B(Ms', Ms''), Ms'''))$$

which reduces by 5(a) again to

$$cut_1^C(Ms, cut_1^B(Ms', cut_1^A(Ms'', Ms'''))).$$

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

$$cut_1^B(cut_1^C(Ms, Ms'), cut_1^A(Ms'', Ms'''))$$

which reduces by 5(a) to

$$cut_1^C(Ms, cut_1^B(Ms', cut_1^A(Ms'', Ms'''))).$$

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

$$cut_2^A(M, x.cut_1^B(cut_1^C(Ms, Ms'), Ms''))$$

which reduces by 5(a) to

$$cut_2^A(M, x.cut_1^C(Ms, cut_1^B(Ms', Ms'')))$$

which reduces by 5(b) to

$$cut_1^C(cut_2^A(M, x.Ms), cut_2^A(M, x.cut_1^B(Ms', Ms'')))$$

which reduces by 5(b) to

$$cut_1^C(cut_2^A(M, x.Ms), cut_1^B(cut_2^A(M, x.Ms'), cut_2^A(M, x.Ms'')));$$

we can also reduce it by 5(b) to

$$cut_1^B(cut_2^A(M, x.cut_1^C(Ms, Ms')), cut_2^A(M, x.Ms''))$$

which reduces by 5(b) to

$$cut_1^B(cut_1^C(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms')), cut_2^A(M, x.Ms''))$$

which reduces by 5(a) to

$$cut_1^C(cut_2^A(M, x.Ms), cut_1^B(cut_2^A(M, x.Ms'), cut_2^A(M, x.Ms''))).$$

13. Rule 5(b) has no (further) overlaps.
14. Rule 5(c) overlaps with itself. Consider

$$cut_3^A(cut_3^B(cut_3^C(M, Ms), Ms'), Ms'')$$

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

$$cut_3^A(cut_3^C(M, cut_1^B(Ms, Ms')), Ms'')$$

which reduces by 5(c) to

$$cut_3^C(M, cut_1^A(cut_1^B(Ms, Ms'), Ms''))$$

which reduces by 5(a) to

$$cut_3^C(M, cut_1^B(Ms, cut_1^A(Ms', Ms''))).$$

But also, it reduces by 5(c) at the root position to

$$cut_3^B(cut_3^C(M, Ms), cut_1^A(Ms', Ms''))$$

and again by 5(c) to

$$cut_3^C(M, cut_1^B(Ms, cut_1^A(Ms', Ms''))).$$

15. Rule 5(c) overlaps with 5(d):

$$cut_4^A(M, x.cut_3^B(cut_3^C(M', Ms), Ms'))$$

reduces by 5(c) to

$$cut_4^A(M, x.cut_3^C(M', cut_1^B(Ms, Ms')))$$

which reduces by 5(d) to

$$cut_3^C(cut_4^A(M, x.M'), cut_2^A(M, x.cut_1^B(Ms, Ms')))$$

which reduces by 5(b) to

$$cut_3^C(cut_4^A(M, x.M'), cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms')));$$

we can also reduce it by 5(d) to

$$cut_3^B(cut_4^A(M, x.cut_3^C(M', Ms)), cut_2^A(M, x.Ms'))$$

which reduces by 5(d) to

$$cut_3^B(cut_3^C(cut_4^A(M, x.M'), cut_2^A(M, x.Ms)), cut_2^A(M, x.Ms'))$$

which reduces by 5(c) to

$$cut_3^C(cut_4^A(M, x.M'), cut_1^B(cut_2^A(M, x.Ms), cut_2^A(M, x.Ms'))).$$

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',*

$$cut_1(Ms, Ms') \leadsto^*_{ES+CC} Ms@Ms'.$$

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,*

$$cut_3(M, Ms) \leadsto^*_{ES+CC} \{M\}Ms.$$

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
$$cut_3((y; Ms'), Ms)$$
which reduces by 3(a) to $(y; cut_1(Ms', Ms))$, 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 $cut_3(\lambda x.M', (M'' :: Ms'))$ which is (by definition) the same term as $\{\lambda x.M'\}Ms$ without any reduction.
3. When M is of the form $cut_3(\lambda y.M', M'' :: Ms')$, the LHS of (3) is
$$cut_3(cut_3(\lambda y.M', M'' :: Ms'), Ms)$$
which reduces by 5(c) to
$$cut_3(\lambda y.M', cut_1(M'' :: Ms', Ms))$$
which reduces by 1(b) to
$$cut_3(\lambda y.M', (M'' :: cut_1(Ms', Ms))).$$
which by Lemma 34 reduces to
$$cut_3(\lambda y.M', M'' :: (Ms'@Ms)).$$
which is just $\{M\}Ms$. ■

Lemma 36. *For pure terms M, Ms, M',*
1. $cut_2(M, x.Ms) \leadsto^*_{ES+CC} [M/x]Ms$;
2. $cut_4(M, x.M') \leadsto^*_{ES+CC} [M/x]M'$.

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 $cut_4(M, x.M') :: cut_2(M, x.Ms')$. 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 $(y; Ms')$ (for $y \neq x$), the LHS reduces by 4(a) to

$$(y; cut_2(M, x.Ms'))$$

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

$$cut_3(M, cut_2(M, x.Ms'))$$

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

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

which reduces by 5(d) to

$$cut_3(cut_4(M, x.\lambda y.M''), cut_2(M, x.(M''' :: Ms')))$$

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

$$cut_3(\lambda y.cut_4(M, x.M''), (cut_4(M, x.M''') :: cut_2(M, x.Ms')))$$

which by inductive hypothesis (three times) reduces to

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

which is just

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

D Simulation of Lambda-x

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

$$cut_1^A(Ms, []) \leadsto Ms$$
$$cut_3^A(M, []) \leadsto M$$

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

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

$$N ::= x \mid (\lambda x.N) \mid NN \mid N\langle x := N\rangle.$$

In $\lambda\mathtt{x}$ the beta-reduction

$$\beta \qquad (\lambda x.N)N' \leadsto_{\beta} [N'/x]N \tag{1}$$

is replaced by the reduction

$$\mathtt{b} \qquad (\lambda x.N)N' \leadsto_{\mathtt{b}} N\langle x := N'\rangle \tag{2}$$

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

$$\begin{array}{ll}
\mathtt{x}1 & x\langle x := N\rangle \leadsto_{\mathtt{x}1} N \\
\mathtt{x}2 & y\langle x := N\rangle \leadsto_{\mathtt{x}2} y \\
\mathtt{x}3 & (\lambda y.N')\langle x := N\rangle \leadsto_{\mathtt{x}3} \lambda y.\, N'\langle x := N\rangle \\
\mathtt{x}4 & (N'N'')\langle x := N\rangle \leadsto_{\mathtt{x}4} N'\langle x := N\rangle\, N''\langle x := N\rangle
\end{array}$$

We translate $\lambda\mathtt{x}$-terms into Herbelin's calculus as follows:

$$\begin{array}{rcl}
(x)^* & =_{def} & (x; []) \\
(\lambda x.N)^* & =_{def} & \lambda x.N^* \\
(NN')^* & =_{def} & cut_3(N^*, N'^* :: []) \\
(N\langle x := N'\rangle)^* & =_{def} & cut_4(N'^*, x.N^*)
\end{array}$$

The simulation is then as follows:

1. Rule $\mathtt{b}$ is mapped onto the reduction sequence

$$cut_3(\lambda x.N^*, N'^* :: []) \leadsto cut_3(cut_4(N'^*, x.N^*), []) \leadsto cut_4(N'^*, x.N^*)$$

2. Rule $\mathtt{x}1$ is mapped onto the reduction sequence

$$cut_4(N^*, x.(x; [])) \leadsto cut_3(N^*, cut_2(N^*, x.[])) \leadsto cut_3(N^*, []) \leadsto N^*$$

3. Rule $\mathtt{x}2$ is mapped onto the reduction sequence

$$cut_4(N^*, x.(y; [])) \leadsto (y; cut_2(N^*, x.[])) \leadsto (y; [])$$

4. Rule $\mathtt{x}3$ is mapped onto the reduction sequence

$$cut_4(N^*, x.\lambda y.N'^*) \leadsto \lambda y.cut_4(N^*, x.N'^*)$$

5. Rule $\mathtt{x}4$ is mapped onto the reduction sequence

$$\begin{array}{rcl}
cut_4(N^*, x.cut_3(N'^*, N''^* :: [])) & \leadsto & cut_3(cut_4(N^*, x.N'^*), cut_2(N^*, x.N''^* :: [])) \\
& \leadsto & cut_3(cut_4(N^*, x.N'^*), cut_4(N^*, x.N''^*) :: cut_2(N^*, x.[])) \\
& \leadsto & cut_3(cut_4(N^*, x.N'^*), cut_4(N^*, x.N''^*) :: [])
\end{array}$$