Under consideration for publication in Math. Struct. in Comp. Science

Cut Rules and Explicit Substitutions

RENÉ VESTERGAARD[1 †] and JOE WELLS[2 ‡]

Heriot-Watt University, Edinburgh, Scotland
[1] *http://www.cee.hw.ac.uk/~jrvest/*
[2] *http://www.cee.hw.ac.uk/~jbw/*

Received 18 April 2000

We introduce a method to associate calculi of proof terms and rewrite rules with cut elimination procedures for logical deduction systems (i.e., Gentzen-style sequent calculi) in the case of intuitionistic logic. We illustrate this method using two different versions of the cut rule for a variant of the intuitionistic fragment of Kleene's logical deduction system G3. Our systems are in fact calculi of explicit substitution, where the cut rule introduces an explicit substitution and the left-$\to$ rule introduces a binding of the result of a function application. Cut propagation steps of cut elimination correspond to propagation of explicit substitutions and propagation of weakening (to eliminate it) corresponds to propagation of index-updating operations. We prove various subject reduction, termination, and confluence properties for our calculi.
Our calculi improve on some earlier calculi for logical deduction systems in a number of ways. By using de Bruijn indices, our calculi qualify as first-order term rewriting systems (TRS's), allowing us to correctly use certain results for TRS's about termination. Unlike in some other calculi, each of our calculi has only one cut rule and we do not need unusual features of sequents.
We show that the substitution and index-updating mechanisms of our calculi work the same way as the substitution and index-updating mechanisms of Kamareddine and Ríos' λs and λt, two well known systems of explicit substitution for the standard λ-calculus. By a change in the format of sequents, we obtain similar results for a known λ-calculus with variables and explicit substitutions, Rose's $\lambda bxgc$.

1. Introduction

1.1. *Background and Motivation*

Natural and Logical Deduction. We assume logic systems are presented in sequent style, i.e., they allow derivations of sequents $\Gamma \vdash \tau$, where Γ is the *antecedent* or *assumptions*, generally a collection of formulae, and τ is the *succedent* or *consequence*. In many logics (e.g., in some presentations of classical logic), succedents are also collections of formulae.

† The work was completed while this, the corresponding, author visited LFCS, University of Edinburgh.
‡ Supported by EPSRC grant GR/L 36963 and NSF grants CCR–9417382 and CCR–9806747.

Because this paper deals exclusively with intuitionistic logic (in fact, only the implicative fragment), we require succedents to be a single consequent formula.

Natural deduction systems, which we choose to call N-systems, are symbolic logics generally given via introduction and elimination rules for the logical connectives which operate on the right, i.e., they manipulate the succedent formula. Examples are Gentzen's NJ and NK (Gentzen 1935). Logical deduction systems are given via left-introduction and right-introduction rules for the logical connectives. Although others have called these systems "sequent calculi", we call them L-systems to avoid confusion with other systems given in sequent style. Examples are Gentzen's LK and LJ (Gentzen 1935). In this paper we are primarily interested in L-systems. The advantage of N-systems is that they seem closer to actual reasoning, while L-systems on the other hand seem to have an easier proof theory. L-systems are often extended with a "cut" rule as part of showing that for a given L-system and N-system, the derivations of each system can be encoded in the other. For example, NK proves the same as LK + cut (Gentzen 1935).

Proof Normalization. A system is *consistent* when it is impossible to prove false, i.e., derive absurdity from zero assumptions. A system is *analytic* (has the *analycity* property) when there is an effective method to decompose any conclusion sequent into simpler premise sequents from which the conclusion can be obtained by some rule in the system such that the conclusion is derivable iff the premises are derivable (Mäenpää 1993).

To achieve the goals of consistency and analycity, it has been customary to consider *normal-form* fragments of logics, which ideally only include derivations satisfying the *subformula property*. This property can be simply stated as "every formula appearing in the derivation also appears as a subformula in the final sequent". Having a set of normal derivations with the subformula property has clear benefits: (1) the space of possible derivations of a sequent is easier to search, and (2) it can be shown that the sequent having absurdity as a consequence of nothing is not the result of a normal derivation using the simple reason that the only subformula of absurdity is itself. Normalization is the process of taking any derivation and finding a derivation in the normal-form fragment with the same final sequent. A logic with an effective normalization procedure will thus ideally be consistent and analytic.

For N-systems, a normal derivation is one where an introduction rule is not followed by an elimination rule that eliminates the connective just introduced. For L-systems, the notion of normal is simpler: a derivation is normal if it is cut-free. Normalization for an L-system shows the cut rule is admissible, i.e., unnecessary for provability of propositions. Gentzen's Hauptsatz (Gentzen 1935) ("main statement") is cut elimination for LK, i.e., LK + cut = LK, and, hence, in turn showing the consistency of LK + cut and therefore also of NK.

Curry-Howard Correspondence. The process of normalization can be used to obtain a correspondence between a logic and a calculus. The correspondence was observed by Howard (Howard 1969) in the case of the simply typed λ-calculus (Church 1941) and the N-system NJ and by Curry (Curry and Feys 1958) in the similar case of combinators and a particular Hilbert-style logic, Combinatory Logic. (A Hilbert-style logic generally only has axioms and *modus ponens* for rules and dispenses with assumptions.) The correspondence maps formulae into types and proofs into terms. Derived formulae are associated with proof terms and assumptions are associated with variables. Rewriting (a.k.a. reduction) of terms is proof transformation and rewriting normal forms are normal proofs. The correspondence is sometimes an isomorphism where each proof corresponds with exactly one term, after proofs are identified up to α-conversion and excessive context. An isomorphism is easy to obtain by making the appropriate term constructor for each proof rule. In the case of an isomorphism, typed terms form a succinct notation for proofs.

Explicit Substitution Calculi. A calculus with rewrite rules which involve substitution, e.g., the λ-calculus, can be simulated by a system with additional rules and term constructors which carries out the substitutions in a stepwise and incremental manner (Abadi *et al.* 1991). Among other benefits, explicit substitution makes rewrite steps closer to actual machine operations and allows more laziness in evaluation. Explicit-substitution calculi have been used for, among other things, work on proof-search (Ritter *et al.* 1996), unification (Dowek *et al.* 1995), representation of incomplete proofs (Muñoz Hurtado 1996b; Magnusson 1995), proof theory (Herbelin 1995), and functional compiler backends (Hardin *et al.* 1996).

Curry-Howard Correspondence for L-Systems. While the Curry-Howard Correspondence is straightforward and, hence, well developed for N-systems, it has also been applied to L-systems. Although right-introduction rules are generally treated the same way as for N-systems, there has been considerable variation in the treatment of the cut rule and left-introduction rules. An older approach associates cut and left-introduction rules with the results of meta-level substitutions:

$$\text{(cut)} \qquad \frac{\Gamma \vdash e_1 : \sigma \quad x : \sigma, \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_2[x{:=}e_1] : \tau}$$

$$(\to\text{-left}) \quad \frac{\Gamma \vdash e_1 : \sigma \quad x : \tau, \Gamma \vdash e_2 : \rho}{z : \sigma \to \tau, \Gamma \vdash e_2[x{:=}(z e_1)] : \rho}$$

Here the expression $e_2[x{:=}e_1]$ denotes the *result* of a substitution rather than a new kind of syntax. This approach allows a certain kind of comparison with N-systems, as the set of proof terms can be the same. Examples of this approach include work on L-systems for intuitionistic linear logic with proof terms in N-system style (Abramsky 1993; Benton *et al.* 1993), work which discusses the relationship between LJ and NJ (Prawitz 1965; Zucker 1974; Pottinger 1977) — in particular the latter, and work that addresses uniqueness of normal forms for cut elimination (Mints 1996).

A more recent approach gives explicit syntax for cut and the left-introduction rules which can be viewed as an explicit substitution or alternatively as a variable binding "let" construct. So where the earlier approach used a meta-level substitution, this approach might use rules like this:

$$(\text{cut}) \qquad \frac{\Gamma \vdash e_1 : \sigma \quad x : \sigma, \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_2\{\!\{x := e_1\}\!\} : \tau}$$

$$(\rightarrow\text{-left}) \qquad \frac{\Gamma \vdash e_1 : \sigma \quad x : \tau, \Gamma \vdash e_2 : \rho}{z : \sigma \rightarrow \tau, \Gamma \vdash \text{let } x = (z\ e_1) \text{ in } e_2 : \rho}$$

Here, the expressions $e_2\{\!\{x := e_1\}\!\}$ and (let $x = (z\ e_1)$ in e_2) denote new syntactic constructs rather than the results of meta-level substitutions. Generally in this approach, rewriting corresponds (closely) to the cut propagation of the cut elimination procedure. The results we present in this paper follow this approach.

One example of this approach is by Gallier (Gallier 1993), who gives a term assignment for a system similar to LJ. Some example rewrite rules are given, but there is no detailed study and no comparison with the explicit substitution literature.

In a second example, Pfenning (Pfenning 1995; Pfenning 1991) represents cut elimination proofs syntactically in Elf but with no furher study of the induced rewrite relations. Observe that while Pfenning indeed treats the cut and left rules at the object-level, he effectively settles for a meta-level treatment of the employed structural rules. This is partly accomplished by representing multi-sets of assumptions with sets of variable names (although the two formats are not equivalent) and by subsequently considering derivations equal up-to the appropriate notion of α. Among other things, this means (1) that Pfenning's languages cannot be TRS's — rather, they are presented as higher-order abstract syntax, and (2) that the languages impose additional structure on the derivations, even when considering derivations equal up-to α.

A third example of this approach is the work of Wadler (Wadler 1993), who gives a proof term assignment for an L-system as well as rewrite rules. The largest differences between Wadler's approach and ours are that he, as Pfenning, (1) uses names for variables instead of de Bruijn indices and (2) does not give explicit syntax to the structural rules, which in his calculus are contraction and weakening. In Wadler's calculus, a let-binding which is syntactically evident can be spontaneously duplicated due to the presence of a use of contraction the position of which is *not* syntactically evident. Similarly, a let-binding can be spontaneously annihilated due to the presence of a use of weakening the position of which is not syntactically evident.

A fourth example of this approach is the work of Herbelin (Herbelin 1994). Herbelin presents an L-system LJT, a restriction of LJ, where one of the assumptions in each sequent may be designated as being in the "stoup". The axiom, $\rightarrow$-left, and contraction rules must use the stoup assumption. There are two different cut rules, C_H and C_M, one of which uses the stoup assumption. There is no weakening rule. The calculus $\bar{\lambda}$ which Herbelin defines to be in correspondence with LJT has two sorts, argument lists and terms, with explicit substitutions for both sorts. The proof rules are then divided into two categories, those for applicative contexts (i.e., argument lists) and those for terms. Each of the cut rules must actually be specialized into two versions, one for each sort, so there end up being four cut rules total. Herbelin shows that the normal forms of his system are isomorphic to normal N-system derivations, although the non-normal forms differ quite substantially.

Espírito Santo has just displayed a refinement of LJT + cut whose derivations are in one-to-one correspondence with (all) NJ derivations. Furthermore, the refined LJT admits a big-step cut elimination procedure that matches "β-reduction" exactly. That is, the refined LJT + normalization is isomorphic to NJ + normalization (Espírito Santo 2000). His approach is orthogonal to the present approach in that we study (standard) L-systems that are not isomorphic to NJ but rather appear to be refinements of NJ. In fact, it could reasonably be said that we study the additional structure of L-systems which is not present in (standard) N-systems.

A final (hybrid) example of this approach is in the very recent work of Cerrito and Kesner (Cerrito and Kesner 1999). They develop a typed calculus in which the pattern matching of ML-like languages is given meaning via rewrite rules which are essentially performing cut elimination. When one erases the term annotations from their typing rules, the logic that results is an L-system with implication ($\rightarrow$), conjunction ($\times$), and disjunction ($+$). Because of choices they make in formulating their system, it appears that both cut and contraction are not admissible rules of their system. They specifically note that their rewrite rules do not eliminate all uses of cut and hence amount to an incompletely specified proof theory. However, in the application of their system for programming languages this seems to be both beneficial and justified.

There are other approaches to assigning terms to the cut rule and the left-introduction rules which do not neatly fit into the categories given above. Breazu-Tannen et al. associate left-introduction rules with pattern matching constructs (Breazu Tannen *et al.* 1993), but in a manner which does not correspond to what goes on in cut elimination proofs. Dyckhoff and Pinto (Dyckhoff and Pinto 1999) as well as Schwichtenberg (Schwichtenberg 1999) give syntactic constructs to left rules in L-systems without cut. They show the equivalence classes of L-system proofs induced by an epimorphic mapping, the Zucker-Pottinger epimorphism (Zucker 1974; Pottinger 1977), between L-systems and N-system are axiomatised by *permutative conversions.* They are concerned with completeness of permutative conversions, i.e., with presenting systems for doing permutative conversions that always terminate (strong normalization) and furthermore do so in a manner that makes the result independent of the particular order of the individual conversions (confluence).

L-System Formulations. The proof theory and the behavior of normalization of an L-system depends on subtle details of its formulation. Gentzen (Gentzen 1935) designed the first L-system, called G1 by many later researchers (Troelstra and Schwichtenberg 1996). The intuitionistic fragment of G1, called LJ by Gentzen, has a $\rightarrow$-left rule somewhat like this one:[†]

$$(\rightarrow\text{-left})\quad \frac{\Gamma \vdash \sigma \quad \tau, \Gamma \vdash \rho}{\sigma \rightarrow \tau, \Gamma \vdash \rho}$$

[†] In Gentzen's system, the premises of the $\rightarrow$-left rule do not share assumptions.

Unfortunately, G1 needs a contraction rule in order to eliminate cut:

$$\text{(contraction)} \quad \frac{\Gamma, \tau, \tau \vdash \sigma}{\Gamma, \tau \vdash \sigma}$$

The impossibility of eliminating uses of contraction from a G1 proof means that proof search must consider contraction, introducing non-determinism and making it more difficult to bound the size of the search space. Also, handling contraction as part of a Curry-Howard isomorphism is painful, especially when it cannot be eliminated. Generally, the straightforward term annotation for contraction is an explicit duplication of a variable, e.g., (duplicate x as y, z in e) which can require intricate propagation rules. Also, in the presence of any such explicit duplication, all other rules which allow sharing assumptions between multiple premises may need term annotations that record similar variable duplicating information. Getting this right can be quite tricky and not doing it may prevent cut or contraction elimination from succeeding correctly. Another issue for G1 is that cut elimination by propagation may need, in intermediate stages, a more powerful *multicut* rule.

To eliminate the need for contraction, Kleene (Kleene 1952) devised his G3 system,[‡] which we call GK3, the intuitionistic fragment of which has a →-left rule somewhat like this one:

$$(\rightarrow\text{-left}) \quad \frac{\sigma \rightarrow \tau, \Gamma \vdash \sigma \quad \tau, \sigma \rightarrow \tau, \Gamma \vdash \rho}{\sigma \rightarrow \tau, \Gamma \vdash \rho}$$

The difference from G1 is that this rule requires the formula constructed by this rule in the antecedent of the derived sequent to already be present in the antecedents of both premise sequents. This paper will use the implicative, intuitionistic fragment of GK3 with only one change — we will treat the antecedent as a sequence rather than as a set as (effectively) done by Kleene. Furthermore, we will consider the effect of adding cut rules to the calculus, something which Kleene did not do.

Later, Dragalin (Dragalin 1979) invented another system, adapted as G3 by Troelstra and Schwichtenberg (Troelstra and Schwichtenberg 1996), the intuitionistic fragment of which has a →-left rule somewhat like this one:

$$(\rightarrow\text{-left}) \quad \frac{\sigma \rightarrow \tau, \Gamma \vdash \sigma \quad \tau, \Gamma \vdash \rho}{\sigma \rightarrow \tau, \Gamma \vdash \rho}$$

The difference from GK3 is that this rule requires the constructed formula to already be present in the antecedent of the left premise, but not in the antecedent of the right premise. With Troelstra and Schwictenberg's G3, both cut and contraction can now be eliminated, but the intermediate stages of the elimination procedure need an inversion principle or rule.[§] Unfortunately, as shown by one of us (Vestergaard 1998b), inversion leads to a computational anomaly, because it implies that the functions it is used on are constant functions thus, e.g., collapsing the Church numerals represented within the

‡ Unfortunately, many systems are known by the name G3.

§ Both the context-sharing and the non-context-sharing cut rule can be eliminated. The former needs inversion in its own right while the latter needs contraction which then needs inversion.

system. Troelstra and Schwichtenberg's G3 (as well as Dragalin's original system) is hence a poor candidate for attempting to form a Curry-Howard isomorphism.

De Bruijn Indices. Traditionally, the λ-calculus and similar calculi have represented variables with names. De Bruijn introduced the idea of representing variables with indices instead of names (de Bruijn 79). Using indices avoids the need to work with α-equivalence classes. When carrying out substitution and also using names, renaming by α-conversion is traditionally used to correctly preserve bindings during substitution. When using indices instead of names, this job is performed by index-updating operators. The essential feature of indices is that index-updating operators can obtain an equivalent effect to that of "choosing a fresh name", only in a fully specified manner, by shifting all other indices.

De Bruijn Indices and Explicit Substitution. In explicit substitution, the index-updating operators become term-level constructs with their own rewrite rules. Among explicit-substitution calculi simulating the λ-calculus, many use de Bruijn indices, e.g., $C\lambda\xi\phi$ (de Bruijn 1978); the calculi of categorical combinators (Curien 1993); $\lambda\sigma, \lambda\sigma_{\Uparrow}, \lambda\sigma_{SP}$ (Abadi *et al.* 1991; Curien *et al.* 1996; Ríos 1993) (the $\lambda\sigma$-family); $\lambda\nu$ (Benaissa *et al.* 1996), $\lambda\zeta$ (Muñoz Hurtado 1996a) (the $\lambda\sigma$-descendants); $\lambda s, \lambda s_e, \lambda t$ (Kamareddine and Ríos 1995; Kamareddine and Ríos 1997; Kamareddine and Ríos 1998). Explicit-substitution calculi for the λ-calculus with names include calculi such as $\lambda bxgc$ and its variants (Bloo 1997; Bloo and Rose 1995; Rose 1996b; Rose 1996a).

1.2. *Our Contribution*

We present a fragment, $GK3i_{dB}$, of the intuitionistic portion of Kleene's G3 with ordered assumptions. We define two versions of the cut rule, $(\mathrm{Cut}_{\mathrm{ns}})$ and $(\mathrm{Cut}_{\mathrm{cs}})$, which differ in how they handle sharing of assumptions between their premises, as well as a weakening rule, (Weak), and an assumption permuting rule, (Reloc). We then consider the logics $GK3i_{dB} + (\mathrm{Weak}) + (\mathrm{Reloc}) + (\mathrm{Cut}_{\mathrm{ns}})$ and $GK3i_{dB} + (\mathrm{Weak})_{k,1} + (\mathrm{Cut}_{\mathrm{cs}})$. For both of these logics, we define proof term annotations and induce rewrite rules on them from the (standard) cut elimination proofs of the logics, yielding the calculi λ^{ns} and λ^{cs}. These allow us to show correspondences with the substitution mechanisms of two existing systems of explicit substitution, λs and λt.

The two variations share several properties. The proof term annotations use de Bruijn indices to manage assumptions. Both versions of the cut rule introduce an explicit substitution. The $\rightarrow$-left rule binds the result of the application of a named function. In the rewrite rules, cut propagation becomes explicit substitution propagation and weakening propagation becomes index-updating propagation. For both variations, we are able to prove the following:

- Each calculus (either λ^{ns} or λ^{cs}) has the subject reduction (SR) property. Thus, the proof transformations represented by the rewrite rules are correct.
- Each calculus (either λ^{ns} or λ^{cs}) satisfies the weak innermost normalization (WIN) property on well typed terms. Furthermore, normal forms are cut-free. Thus, there is

a cut elimination procedure for the logic ($GK3i_{dB}$ + (Weak) + (Reloc) + ($\mathrm{Cut}_{\mathrm{ns}}$) or $GK3i_{dB}$ + $(\mathrm{Weak})_{k,1}$ + ($\mathrm{Cut}_{\mathrm{cs}}$)).

- A structural analysis of each calculus (either λ^{ns} or λ^{cs}), results in a slightly restricted calculus (either $\lambda^{\mathrm{ns}}_{-}$ or $\lambda^{\mathrm{cs}}_{-}$) which still is complete with regards to cut elimination but which has no critical pairs. The equivalent property at the level of the cut elimination proofs is that of a complete and non-overlapping case-splitting in the induction step. As a result, the restricted calculus (either $\lambda^{\mathrm{ns}}_{-}$ or $\lambda^{\mathrm{cs}}_{-}$) satisfies (1) SR, (2) cut-freeness of the normal forms, (3) the confluence or Church-Rosser (CR) property, and (4) the strong normalization (SN) property. Thus, for the restricted calculus, procedures for cut elimination by cut propagation can use any strategy of choosing which cut to propagate next. This is often called "strong" cut elimination.
  The third property, (CR), contradicts the commonly held view that cut-elimination is inherently non-confluent. The property is obtained at the cost of fewer equations in the induced equational theory of derivations.
- The way each calculus (either λ^{ns} or λ^{cs}) propagates explicit substitions corresponds closely to the way a well known system of explicit substitution for the λ-calculus (either λs or λt) propagates substitutions.

The advantages of our approach over some other L-systems with explicit syntax for cut and the left-introduction rules are the following:

- We study the computational meaning of "off-the-shelf" proof theory and give a meaningful account of it in terms of well-established computational structures.
- In comparison with Herbelin's LJT logic and its associated $\bar{\lambda}$-calculus, we have only one syntactic sort and one rule for cut. We do not need any term constructors corresponding to argument list prefixing or concatenation. Also, as our sequents do not have a stoup, we do not have the restriction that the axiom and $\rightarrow$-left rules must use only the stoup formula, so we avoid the need to use Herbelin's somewhat awkward encoding of LJ into LJT which requires the introduction of applicative contexts.
- Unlike some of the previous work we have outlined, we treat all employed structural rules explicitly. This aids in understanding what the computational meaning of these rules might be. Also, by having term constructors for every rule, our terms can be seen as a succinct syntax for derivations.
- Our calculi are the first for L-systems to represent variables by de Bruijn indices instead of by names.¶ The structural rules of our logics are associated with term-level index-updating operators. Explicit index updating allows our calculi to be first-order term rewriting systems (TRS's). If we relied on implicit α-conversion to do this renaming, then our systems would be higher-order rewrite systems, but would not qualify as first-order. The fact that our calculi are first-order TRS's allows us to safely and correctly use results in the literature for first-order TRS's which give conditions when WIN implies SN (O'Donnell 1977; Gramlich 1995). This gives a solid foundation for our strong cut-elimination result, in contrast to some other work in

¶ Except for (Herbelin 1995) in the case of the highly specialized LJT.

the literature which incorrectly attempts to use theorems about first-order systems for higher-order systems.

— The two calculi λ^{ns} and λ^{cs} help to illustrate our contention that *the structure of inference rules matters*. The difference in the assumption maintenance mechanism of the logics $\mathit{GK3i}_{dB}$ + (Weak) + (Reloc) + ($\mathrm{Cut}_{\mathrm{ns}}$) and $\mathit{GK3i}_{dB}$ + $(\mathrm{Weak})_{k,1}$ + ($\mathrm{Cut}_{\mathrm{cs}}$) is reflected in the way that λ^{ns} and λ^{cs} update indices and introduce the weakening and assumption reordering operators.

— We show a weak correspondence with the existing explicit substitution calculi λs and λt. This weak correspondence shows that the mechanisms for propagating explicit substitutions and updating indices in λ^{ns} and λ^{cs} are nearly the same as the mechanisms of well known and established explicit substitution calculi, λs and λt.[‖] We also discuss a very similar weak correspondence with Rose's $\lambda bxgc$.
These correspondences with the substitution mechanisms of established explicit substitution calculi are the first shown for a calculus for an L-system.

1.3. *Additional Related Work*

Other related works, somewhat orthogonal to the results here, are as follows. Maraist et al. (Maraist *et al.* 1995) present a typed let-calculus. The type system is an N-system using a cut rule to type the let-construct. The proof theory of the type system is not considered (apart from the statement of a SR-result) and no comparative results as to the explicit-substitution nature of the let construct are given. In fact, meta-level substitution is used in reducing the let construct. Di Cosmo and Kesner (Di Cosmo and Kesner 1997) encode calculi with explicit substitution into proof nets in order to show the calculi SN. Pagano develops a logic corresponding to the $\lambda\sigma_{\Uparrow}$ explicit substitution calculus (Pagano 1998). While the calculus uses de Bruijn indices, the logic is an N-system and the work is thus orthogonal to our goals.

2. Logical Deduction and Computation

We define $\mathit{GK3i}_{dB}$, a proof-term-annotated L-system for the implicative fragment of intuitionistic logic. The inference rules of $\mathit{GK3i}_{dB}$ are given in Figure 2 using the operations and conventions for formulae, contexts, and metavariables given in Figure 1. When the rules of a logic are annotated with terms and a fixed order is imposed on the assumptions in a sequent, the antecedent (material to the left of "$\vdash$") is called a *context*. The system $\mathit{GK3i}_{dB}$ is more than a logic because it has proof terms. It is not exactly an inductive definition of a set of (explicitly typed) terms of a calculus either, because it is context-sensitive due to the assumptions prescribing a set of permissible indices. However, both a logic and a set of terms are easily extracted from the definition of $\mathit{GK3i}_{dB}$ and we will

‖ In fact, we consider these parallel correspondences to be curiously instructive as there *prima facie* is no direct relationship between the non-context-sharing versus context-sharing nature of the cut rules and the centralized versus incremental index-updating strategies of the explicit substitution calculi.

Formulae (τ)

$$\tau, \sigma, \nu, \rho \quad ::= \quad \alpha \mid \tau \to \tau$$

Contexts (Γ), context-smash ($\oplus$)

$$\begin{array}{rcl} \Gamma & ::= & \tau_1, \ldots, \tau_k \\ |\tau_1, \ldots, \tau_k| & = & k \\ \Gamma^1 \oplus \Gamma^2 & = & \begin{cases} \Gamma^2 & \text{if } \Gamma^1 \text{ is a prefix of } \Gamma^2 \\ \Gamma^1 & \text{if } \Gamma^2 \text{ is a proper prefix of } \Gamma^1 \\ \text{undefined} & \text{otherwise} \end{cases} \end{array}$$

Fig. 1. Contexts (Γ) as finite but unbounded lists of formulae.

$$\frac{}{\Gamma^1, \tau, \Gamma^2 \vdash n : \tau}\,(\mathrm{Ax})_n \quad \text{if } |\Gamma^1| = n - 1 \qquad \frac{\tau, \Gamma \vdash e : \sigma}{\Gamma \vdash \lambda e : \tau \to \sigma}\,(\to\mathrm{R})$$

$$\frac{\Gamma \vdash e_1 : \tau \qquad \sigma, \Gamma \vdash e_2 : \nu}{\Gamma \vdash \mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2 : \nu}\,(\to\mathrm{L})_k \quad \text{if } \Gamma = \Gamma^1, \tau \to \sigma, \Gamma^2 \text{ and } |\Gamma^1| = k - 1$$

Fig. 2. The logical rules, together called $GK3i_{dB}$.

sometimes take $GK3i_{dB}$ to refer exclusively to the logic or to the set of terms. We define the *end sequent* of a derivation to be its final sequent with the proof term removed.

The system $GK3i_{dB}$ is a refinement of the implicative fragment of Kleene's G3 (Kleene 1952), which we call GK3. Kleene's original definition used lists for antecedents but effectively took them to be sets of formulae in a technical manner. We adapt the original definition as contexts-as-lists and retain it throughout — except for Section 3.3 where we briefly consider contexts as sets/multisets. We also consider cut and its elimination from the system, unlike Kleene.

For the ($\to$L) rule, notice that the (left) introduced formula is required to already be present in the assumptions of both premises. This allows proving the admissibility proofs of the structural and cut rules (in Figures 3 and 4) without needing an explicit contraction rule (cf. (Troelstra and Schwichtenberg 1996) and the discussion on L-System Formulations in Section 1). The subscript 1 on "bind_1" in the term annotation serves merely as a reminder that this form binds index 1 in the body e_2 to the result of the function call $(k\, e_1)$.

We will independently consider the two different cut rules for $GK3i_{dB}$ given in Figure 3. Both are indexed by the level (position in the context) of the assumed formula of the right premise they "cut out". They differ only in the way they manage the two contexts in the premises. The ($\mathrm{Cut_{cs}}$) rule is context-sharing, like ($\to$L). In contrast, the ($\mathrm{Cut_{ns}}$) rule can express completely non-context-sharing cuts in the case when Γ^2 is empty. We avoid restricting ($\mathrm{Cut_{ns}}$) to always be completely non-context-sharing because then we would need a contraction rule (see the discussion on L-System Formulations in Section 1 for why we avoid having a contraction rule). Instead, ($\mathrm{Cut_{ns}}$) has some built-in ability to do

$$\frac{\Gamma^1 \vdash e_1 : \tau \qquad \Gamma, \tau, \Gamma^2 \vdash e_2 : \sigma}{\Gamma, (\Gamma^1 \oplus \Gamma^2) \vdash e_2\{\!\{k := e_1\}\!\} : \sigma} \, (\mathrm{Cut}_{\mathrm{ns}})_k \qquad \text{if } \Gamma^1 \oplus \Gamma^2 \text{ is defined and } |\Gamma| = k-1$$

$$\frac{\Gamma^1, \Gamma^2 \vdash e_1 : \tau \qquad \Gamma^1, \tau, \Gamma^2 \vdash e_2 : \sigma}{\Gamma^1, \Gamma^2 \vdash e_2[\![k := e_1]\!] : \sigma} \, (\mathrm{Cut}_{\mathrm{cs}})_k \qquad \text{if } |\Gamma^1| = k-1$$

Fig. 3. The cut rules, together called *Cuts*.

$$\frac{\Gamma^1, \Gamma^2 \vdash e : \sigma}{\Gamma^1, \Gamma, \Gamma^2 \vdash \mathcal{W}^m_k(e) : \sigma} \, (\mathrm{Weak})_{k,m} \qquad \text{if } |\Gamma^1| = k \text{ and } |\Gamma| = m$$

$$\frac{\Gamma^1, \Gamma^2, \tau, \Gamma^3 \vdash e : \sigma}{\Gamma^1, \tau, \Gamma^2, \Gamma^3 \vdash \mathcal{R}^k_l(e) : \sigma} \, (\mathrm{Reloc})_{k,l} \qquad \text{if } |\Gamma^1, \Gamma^2| = k-1 \text{ and } |\Gamma^1| = l-1$$

Fig. 4. The structural rules, together called *Struct*.

the equivalent of contraction, exactly the amount needed for correct cut propagation (see the (β_{ns}) rule in Figure 7 which introduces new cuts needing the contraction ability). *A priori* there is no guarantee that this is possible and some work needs to be done to spell out all cases. The issue is that (both kinds of) cut propagation, which moves cuts up in derivations, has to ensure that addition of new assumptions is allowed by the propagated cut rule in the places prescribed by the discharging mechanisms of the "stationary" rules.

We consider both a context-sharing and a non-context-sharing version of the cut rule because, conceptually, both are commonly used and also because their natural propagation rules turn out to behave similarly to the rules of some existing calculi of explicit substitution (see Section 3). However, the particular definition of the non-context-sharing cut rule is new, albeit *natural*, and due to the first author (Vestergaard 1998a).

The structural rules, (Weak) and (Reloc) given in Figure 4, are necessary in intermediate steps during the cut elimination procedure. Eliminating the ($\mathrm{Cut}_{\mathrm{ns}}$) rule requires both structural rules while eliminating ($\mathrm{Cut}_{\mathrm{cs}}$) only requires a restriction of (Weak). The structural rules are themselves admissible for $GK3i_{dB}$ (with elimination procedures given in Figures 5 and 6). A rule name without its indices, e.g., (Weak), stands for the collection of the rule at all possible values of the indices.

Honouring our commitment to "off-the-shelf" proof theory, we now present the result, namely cut elimination, that provides the basis for the rest of the article. The result will initially be given in its traditional form showing that occurrences of cut can be propagated towards the leafs and ultimately eliminated (Gentzen 1935). Following on, we present the Hauptsatz as a corollary of cut elimination before we, finally, restate the cut elimination result in the TRS framework. The latter step involves a complete change of view so the primary issue becomes the constructive content of the proof of cut elimination rather than the result of carrying the proof out, so to speak.

$$\mathcal{W}_k^m(n) \quad \to_w \quad \begin{cases} n & \text{if } n \leq k \\ n+m & \text{if } n > k \end{cases}$$

$$\mathcal{W}_k^m(\lambda e) \quad \to_w \quad \lambda \mathcal{W}_{k+1}^m(e)$$

$$\mathcal{W}_k^m(\text{bind}_1\ (n\,e) \text{ in } e') \quad \to_w \quad \begin{cases} \text{bind}_1\ (n\,\mathcal{W}_k^m(e)) \text{ in } \mathcal{W}_{k+1}^m(e') & \text{if } n \leq k \\ \text{bind}_1\ ((n+m)\,\mathcal{W}_k^m(e)) \text{ in } \mathcal{W}_{k+1}^m(e') & \text{if } n > k \end{cases}$$

Fig. 5. Rewrite rules for $\mathcal{W}_k^m(\,\cdot\,)$ (propagation of (Weak)).

$$\mathcal{R}_l^k(n) \quad \to_r \quad \begin{cases} n & \text{if } n < l \lor n > k \\ l & \text{if } n = k \\ n+1 & \text{if } l \leq n < k \end{cases}$$

$$\mathcal{R}_l^k((\lambda e)) \quad \to_r \quad \lambda \mathcal{R}_{l+1}^{k+1}(e)$$

$$\mathcal{R}_l^k((\text{bind}_1\ (n\,e) \text{ in } e')) \quad \to_r \quad \begin{cases} \text{bind}_1\ (n\,(\mathcal{R}_l^k(e))) \text{ in } \mathcal{R}_{l+1}^{k+1}(e') & \text{if } n < l \lor n > k \\ \text{bind}_1\ (l\,(\mathcal{R}_l^k(e))) \text{ in } \mathcal{R}_{l+1}^{k+1}(e') & \text{if } n = k \\ \text{bind}_1\ ((n+1)\,(\mathcal{R}_l^k(e))) \text{ in } \mathcal{R}_{l+1}^{k+1}(e') & \text{if } l \leq n < k \end{cases}$$

Fig. 6. Rewrite rules for $\mathcal{R}_l^k(\,\cdot\,)$ (propagation of (Reloc)).

The separation of these three versions of essentially the same result serves to highlight their complementary views: structural proof theory, semantical ditto, and term rewriting, respectively. Having stated all three views, we can then go on to invoke each in turn as we see fit and as circumstances dictate.

Theorem 1 (Admissibility). *Cuts+Struct* are (constructively) admissible for $GK3i_{dB}$.

Proof. From the viewpoint of structural proof theory, that is for the uses of *Cuts* + *Struct* rules in a derivation, we show how to propagate at least one non-leaf use towards the leaves and how to eliminate leaf uses while preserving the derivation's end sequent.

The proof proceeds by a standard case analysis on the last rule applied in the premise(s). By the Curry-Howard isomorphism, we can present the individual cases via rewrite rules on proof terms. The rewrite rules are given in Figures 5, 6, 7, and 8.[††]

For *Cuts* there are two general cases, each with a number of sub-cases: (1) at least one premise has a *non-principal cut-formula* ((Realign$_-$), (Abstr$_-$), (Bind$^1_-$), (Bind$^2_-$) of Figures 7, and 8) and (2) both have principal cut-formulae ((Subst$_-$), (Reindex$_-$), (β_-)) — the cases subsume cut on axioms, something which sometimes is considered to be an independent third case.

Having thus outlined the details of the case-splitting, we prove (by standard argument) that the innermost strategy for the propagation rules terminates using the sizes of cut-formulae and derivations for innermost cut occurrences. In the case of *Struct* it is simply

[††] For readability, we refer the reader to Figure 11. The figure contains a transcription of the cut rewrite rules to the variables-as-names situation.

$$
\begin{array}{ll}
n\{\!\!\{k := e\}\!\!\} & \\
\qquad \to_{c_{ns}} \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases} & (\mathrm{Realign_{ns}}) \\
(\lambda e')\{\!\!\{k := e\}\!\!\} & \\
\qquad \to_{c_{ns}} \lambda(e'\{\!\!\{k+1 := e\}\!\!\}) & (\mathrm{Abstr_{ns}}) \\
e\{\!\!\{k := (\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2)\}\!\!\} & \\
\qquad \to_{c_{ns}} \mathrm{bind}_1\ ((n+k-1)\,(\mathcal{W}_0^{k-1}(e_1)))\ \mathrm{in}\ \mathcal{R}_1^k((\mathcal{W}_k^1(e))\{\!\!\{k := e_2\}\!\!\}) & (\mathrm{Bind^1_{ns}}) \\
(\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2)\{\!\!\{k := e\}\!\!\} & \\
\qquad \to_{c_{ns}} \begin{cases} \mathrm{bind}_1\ (n\, e_1\{\!\!\{k := e\}\!\!\})\ \mathrm{in}\ (e_2\{\!\!\{k+1 := e\}\!\!\}) & \text{if } n < k \\ \mathrm{bind}_1\ ((n-1)\, e_1\{\!\!\{k := e\}\!\!\})\ \mathrm{in}\ (e_2\{\!\!\{k+1 := e\}\!\!\}) & \text{if } n > k \end{cases} & (\mathrm{Bind^2_{ns}}) \\
k\{\!\!\{k := e\}\!\!\} & \\
\qquad \to_{c_{ns}} \mathcal{W}_0^{k-1}(e) & (\mathrm{Subst_{ns}}) \\
(\mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2)\{\!\!\{k := m\}\!\!\} & \\
\qquad \to_{c_{ns}} \mathrm{bind}_1\ ((m+k-1)\, e_1\{\!\!\{k := m\}\!\!\})\ \mathrm{in}\ (e_2\{\!\!\{k+1 := m\}\!\!\}) & (\mathrm{Reindex_{ns}}) \\
(\mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2)\{\!\!\{k := \lambda e\}\!\!\} & \\
\qquad \to_{c_{ns}} e_2\{\!\!\{k+1 := \lambda e\}\!\!\}\{\!\!\{1 := (\mathcal{W}_1^{k-1}(e))\{\!\!\{1 := e_1\{\!\!\{k := \lambda e\}\!\!\}\}\!\!\}\}\!\!\} & (\beta_{\mathrm{ns}})
\end{array}
$$

Fig. 7. Rewrite rules for $\cdot\{\!\!\{k := \cdot\}\!\!\}$ (propagation of ($\mathrm{Cut_{ns}}$)).

$$
\begin{array}{ll}
n[\![k := e]\!] & \\
\qquad \to_{c_{cs}} \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases} & (\mathrm{Realign_{cs}}) \\
(\lambda e')[\![k := e]\!] & \\
\qquad \to_{c_{cs}} \lambda(e'[\![k+1 := \mathcal{W}_0^1(e)]\!]) & (\mathrm{Abstr_{cs}}) \\
e[\![k := (\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2)]\!] & \\
\qquad \to_{c_{cs}} \mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ \mathcal{W}_0^1(e)[\![k+1 := e_2]\!] & (\mathrm{Bind^1_{cs}}) \\
(\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2)[\![k := e]\!] & \\
\qquad \to_{c_{cs}} \begin{cases} \mathrm{bind}_1\ (n\, e_1[\![k := e]\!])\ \mathrm{in}\ (e_2[\![k+1 := \mathcal{W}_0^1(e)]\!]) & \text{if } n < k \\ \mathrm{bind}_1\ ((n-1)\, e_1[\![k := e]\!])\ \mathrm{in}\ (e_2[\![k+1 := \mathcal{W}_0^1(e)]\!]) & \text{if } n > k \end{cases} & (\mathrm{Bind^2_{cs}}) \\
k[\![k := e]\!] & \\
\qquad \to_{c_{cs}} e & (\mathrm{Subst_{cs}}) \\
(\mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2)[\![k := m]\!] & \\
\qquad \to_{c_{cs}} \mathrm{bind}_1\ (m\, e_1[\![k := m]\!])\ \mathrm{in}\ (e_2[\![k+1 := m+1]\!]) & (\mathrm{Reindex_{cs}}) \\
(\mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2)[\![k := \lambda e]\!] & \\
\qquad \to_{c_{cs}} e_2[\![k+1 := \mathcal{W}_0^1(\lambda e)]\!][\![1 := e[\![1 := e_1[\![k := \lambda e]\!]]\!]]\!] & (\beta_{\mathrm{cs}})
\end{array}
$$

Fig. 8. Rewrite rules for $\cdot[\![k := \cdot]\!]$ (propagation of ($\mathrm{Cut_{cs}}$)).

based on the height of the derivation, while in the case of *Cuts* we use a lexicographical ordering of formula-size and derivation size on (top-most) cut occurrences, see, e.g., (Troelstra and Schwichtenberg 1996).

The elimination of an occurrence of a cut rule is interleaved with the elimination of occurrences of structural rules but not *vice versa*. Furthermore, the elimination of an occurrence of a *Struct*-rule does not increase the derivation size (nor any formula size).

Hence, the proof is a constructive admissibility proof for all of *Cuts* + *Struct* as well as for the individual rules of *Cuts* + *Struct*. Details are available in Appendices A, B, C, and D. □

As noted, the preceding theorem is concerned with structural proof theory (i.e., with

$$e\{\!\!\{k := (\mathrm{bind}_1\ (n\,e_1)\ \mathrm{in}\ e_2)\}\!\!\}$$
$$\rightarrow_{c_{ns}} \mathrm{bind}_1\ ((n+k-1)\,(\mathcal{W}_0^{k-1}(e_1)))\ \mathrm{in}\ \mathcal{R}_1^k((\mathcal{W}_k^1(e))\{\!\!\{k := e_2\}\!\!\}) \quad (\mathrm{Bind}^3_{\mathrm{ns}})$$
$$\text{where } e = (\mathrm{bind}_1\ (k\,e_1')\ \mathrm{in}\ e_2')$$

$$e[\![k := (\mathrm{bind}_1\ (n\,e_1)\ \mathrm{in}\ e_2)]\!]$$
$$\rightarrow_{c_{cs}} \mathrm{bind}_1\ (n\,e_1)\ \mathrm{in}\ ((\mathcal{W}_0^1(e))[\![k+1 := e_2]\!]) \quad (\mathrm{Bind}^3_{\mathrm{cs}})$$
$$\text{where } e = (\mathrm{bind}_1\ (k\,e_1')\ \mathrm{in}\ e_2')$$

Fig. 9. Replacements for $(\mathrm{Bind}^1_{\mathrm{ns}})$ and $(\mathrm{Bind}^1_{\mathrm{cs}})$ to remove critical pairs.

derivations). Turning to semantical proof theory (i.e., to the axiomatized consequence relation), we have:

Corollary 2 (Hauptsätze[‡‡]). The sets of end-sequents derivable from $\mathit{GK3i}_{dB}+\mathit{Struct}+\mathit{Cuts}$ and $\mathit{GK3i}_{dB}$ (i.e., the consequence relations they axiomatize) coincide.

Having thus seen two proof theoretical versions, one structural and one semantical, of essentially the same result, we present yet another version, this time from a computational viewpoint. The intention is to give first-class status to all constructive details of the proof of Theorem 1.

Definition 3 (Proof Terms). Let Λ^G be the set of valid proof terms for $G \subseteq \mathit{GK3i}_{dB} + \mathit{Cuts} + \mathit{Struct}$.

Definition 4 (Calculi of Proof Terms). Let

λ^{ns} be the terms $\Lambda^{\mathrm{ns}} = \Lambda^{\mathit{GK3i}_{dB}+(\mathrm{Weak})+(\mathrm{Reloc})+(\mathrm{Cut}_{\mathrm{ns}})}$ with rewriting relation $\rightarrow_{\mathrm{ns}}=\rightarrow_w \cup \rightarrow_r \cup \rightarrow_{c_{ns}}$, and

λ^{cs} be the terms $\Lambda^{\mathrm{cs}} = \Lambda^{\mathit{GK3i}_{dB}+(\mathrm{Weak})_{k,1}+(\mathrm{Cut}_{\mathrm{cs}})}$ with rewriting relation $\rightarrow_{\mathrm{cs}}=\rightarrow_w \cup \rightarrow_{c_{cs}}$.

Observe that λ^{cs} does not need all of the possible instances of (Weak). Now, with these basic definitions in place, Theorem 1 transcribes as follows:

Theorem 5. The (implicitly typed) calculi λ^{ns}, λ^{cs} enjoy SR and WIN. Their normal-form fragment is $\Lambda^{\mathit{GK3i}_{dB}}$ (which is "*Cuts* + *Struct*"-free).

Proof. Exactly the same proof as for Theorem 1. □

2.1. *Strengthening the Results*

We will now strengthen the already obtained results in a direction which pertains mainly to the study of the induced equational theory. In doing so, we will also address some long-standing proof theoretical issues with cut elimination procedures that are not directly related to consistency considerations: termination and confluence properties. We will be using the following result (O'Donnell 1977).

[‡‡] The fact that the Hauptsätze are corollaries to the cut elimination results is a direct consequence of the fact that, at this level, semantical proof theory is (naively) given as simply the "logical meaning" of structural proof theory.

Theorem 6 (O'Donnell). For an orthogonal (left-linear and non-overlapping rules) TRS, WIN $\Leftrightarrow$ SN.

The calculi λ^{ns} and λ^{cs} qualify as first-order TRS's. Unfortunately, they contain critical pairs, specifically, $(\mathrm{Bind}^1_{\mathrm{X}})$ with any of $(\mathrm{Realign}_{\mathrm{X}})$, $(\mathrm{Abstr}_{\mathrm{X}})$, $(\mathrm{Bind}^2_{\mathrm{X}})$, and $(\mathrm{Subst}_{\mathrm{X}})$ of Figures 7 and 8, where X ranges over ns and cs. Correspondingly, it is well known in the proof theory community that "logical deduction proofs do not have unique normal forms". However, this is easy to fix using the rules $(\mathrm{Bind}^3_{\mathrm{ns}})$ and $(\mathrm{Bind}^3_{\mathrm{cs}})$ presented in Figure 9 which are restrictions of $(\mathrm{Bind}^1_{\mathrm{ns}})$ and $(\mathrm{Bind}^1_{\mathrm{cs}})$. Let $\lambda^{\mathrm{ns}}_{-}$ and $\lambda^{\mathrm{cs}}_{-}$ be the result of replacing $(\mathrm{Bind}^1_{\mathrm{ns}})$ and $(\mathrm{Bind}^1_{\mathrm{cs}})$ in λ^{ns} and λ^{cs} respectively by $(\mathrm{Bind}^3_{\mathrm{ns}})$ and $(\mathrm{Bind}^3_{\mathrm{cs}})$. These new calculi retain the nice properties proven for λ^{ns} and λ^{cs} but are also confluent and have unique normal forms. Furthermore, cut elimination terminates regardless of the cut propagation strategy.

Theorem 7. The (implicitly typed) calculi $\lambda^{\mathrm{ns}}_{-}$ and $\lambda^{\mathrm{cs}}_{-}$ are orthogonal TRS's and have the SR, SN, and CR properties. Their normal-form fragment is $\Lambda^{GK3i_{dB}}$.

Proof. Orthogonality is easily checked, cf. Appendix E. SR follows directly from Theorem 5. SN is obtained via Theorem 6 while CR is obtained immediately by orthogonality of the rules. Every innermost occurrence of the syntax constructors corresponding to *Cuts* + *Struct* is a redex hence giving $\Lambda^{GK3i_{dB}}$ as the normal-form fragment of the calculi. □

Corollary 8. The (implicitly typed) calculi $\lambda^{\mathrm{ns}}_{-}$ and $\lambda^{\mathrm{cs}}_{-}$ have uniquely existing normal forms (i.e., every term rewrites to exactly one normal form in a finite number of steps).

Proof. Directly from SN and CR. □

In summation, a slight restriction of one of the cut propagation rules formulated as a rewriting rule has allowed us to strengthen the traditional proof theoretical results while still performing cut elimination. The restriction was based on a simple structural analysis of the calculi of proof terms and corresponds to an adjustment of the case-splitting in the induction step of the admissibility proof of cut. The simplicity is due to our use of proof term calculi and would, for example, appear obfuscated at the level of derivations proper.

To stress the structural dimension of the results, i.e., the proper separation of the logical content of the Hauptsätze and the language with which we express it, we point out that we in fact have the following result.

Theorem 9. The untyped versions of $\lambda^{\mathrm{ns}}_{-}$ and $\lambda^{\mathrm{cs}}_{-}$ have the CR property.

Proof. The untyped rewriting relations are still orthogonal, cf. the proof of Theorem 7. □

2.2. *Assessing the Results*

The main proof theoretical achievements of our pursuits, besides their correctness: SR, WN (WIN), and cut-freeness of normal forms, are as follows.

— The formalization of cut elimination procedures is traditionally considered an inaccessible problem, e.g., requiring the use of *higher-order abstract syntax* (Pfenning 1995). Our general approach to formalizing L-systems uses the TRS-framework and hence renders the cut elimination procedures immediately amenable to formalization.

— Confluence is normally considered out-of-bounds for cut elimination procedures. This has partly been rationalized by the presence of (incomplete) permutative conversions with distinct normal forms and partly by over-specified cut elimination procedures taking advantage of this.
We have obtained confluence results for (the underlying untyped languages of) cut elimination procedures at the expense of equating fewer derivations. Future work aims to incorporate (complete) permutative conversions to rectify this, cf. (Mints 1996; Dyckhoff and Pinto 1999; Schwichtenberg 1999).
We do not know of any other confluence results for cut elimination in a standard L-system apart from one that can be extracted from (Mints 1996) in the typed case.

— General SN results for cut-elimination procedures obtained through generic TRS results have also been studied by Tahhan Bittar (Tahhan Bittar 1996) and Selhab (Selhab 1998). Dragalin (Dragalin 1979) presents a strong cut-elimination procedure in its own right (that is, without using TRS results). A closer comparison with these results, although interesting, falls outside the scope of the present paper.

3. Relations to Known Calculi

This section compares the explicit substitution machinery of λ^{ns} and λ^{cs} with similar machinery previously developed by others, the λs and λt calculi of Kamareddine and Ríos. The key to this comparison is the following observation. Instead of using an explicit binding construct for ($\rightarrow$L), we could have relied on ordinary λ-binding and application syntax. Hence, instead of writing: "$\mathrm{bind}_1\ (n\,e)$ in e'", we could have written: "$(\lambda e')\,(n\,e)$".

3.1. *Translating to λs and λt.*

The λs and λt calculi, defined in Figure 10, are explicit substitution versions of the untyped λ-calculus with de Bruijn indices developed by Kamareddine and Ríos (Kamareddine and Ríos 1995; Kamareddine and Ríos 1998). Both the λs and λt calculi are confluent on closed terms,[§§] preserve strong normalization (PSN) w.r.t. the λ-calculus, and have an SN simply-typed version. In addition, for λs there is a confluent extension to open terms (Kamareddine and Ríos 1997) which, unfortunately, is not PSN w.r.t. the λ-calculus (Guillaume 1997) but is WN in the simply-typed case. The λt-calculus is slightly closer than λs to the style of $\lambda\upsilon$ and $\lambda\sigma$ in that it does index updating in smaller increments.

Both λs and λt are untyped calculi while λ^{ns} and λ^{cs}, as presented, are simply-typed. However, we will be treating the type-free versions of λ^{ns} and λ^{cs} in the following. We

[§§] The closed terms are the usual terms.

λs			λt		
$a \in \Lambda s$	$::=$	$n \mid a\,a \mid \lambda a \mid a\sigma^i a \mid \varphi^i_k a$ where $i, n \geq 1, k \geq 0$	$b \in \Lambda t$	$::=$	$n \mid b\,b \mid \lambda b \mid b\varsigma^i b \mid \theta_k b$ where $i, n \geq 1, k \geq 0$
$(\lambda a_1)\,a_2$	$\rightarrow_{\lambda s}$	$a_1\sigma^1 a_2$	$(\lambda b_1)\,b_2$	$\rightarrow_{\lambda t}$	$b_1\varsigma^1 b_2$
$(\lambda a_1)\sigma^i a_2$	$\rightarrow_{\lambda s}$	$\lambda(a_1\sigma^{i+1}a_2)$	$(\lambda b_1)\varsigma^i b_2$	$\rightarrow_{\lambda t}$	$\lambda(b_1\varsigma^{i+1}(\theta_0 b_2))$
$(a_1\,a_2)\sigma^i a_3$	$\rightarrow_{\lambda s}$	$(a_1\sigma^i a_3)\,(a_2\sigma^i a_3)$	$(b_1\,b_2)\varsigma^i b_3$	$\rightarrow_{\lambda t}$	$(b_1\varsigma^i b_3)\,(b_2\varsigma^i b_3)$
$n\sigma^i a$	$\rightarrow_{\lambda s}$	$\begin{cases} n-1 & \text{if } n > i \\ \varphi^i_0 a & \text{if } n = i \\ n & \text{if } n < i \end{cases}$	$n\varsigma^i b$	$\rightarrow_{\lambda t}$	$\begin{cases} n-1 & \text{if } n > i \\ b & \text{if } n = i \\ n & \text{if } n < i \end{cases}$
$\varphi^i_k n$	$\rightarrow_{\lambda s}$	$\begin{cases} n & \text{if } n \leq k \\ n+i-1 & \text{if } n > k \end{cases}$	$\theta_k n$	$\rightarrow_{\lambda t}$	$\begin{cases} n & \text{if } n \leq k \\ n+1 & \text{if } n > k \end{cases}$
$\varphi^i_k(\lambda a)$	$\rightarrow_{\lambda s}$	$\lambda(\varphi^i_{k+1}a)$	$\theta_k(\lambda b)$	$\rightarrow_{\lambda t}$	$\lambda(\theta_{k+1}b)$
$\varphi^i_k(a_1\,a_2)$	$\rightarrow_{\lambda s}$	$(\varphi^i_k a_1)\,(\varphi^i_k a_2)$	$\theta_k(b_1\,b_2)$	$\rightarrow_{\lambda t}$	$(\theta_k b_1)\,(\theta_k b_2)$

Fig. 10. The λs and λt calculi.

define the following translations from the terms of λ^{ns} and λ^{cs} into the terms of λs and λt, respectively.

Definition 10. $\mathcal{S} : \Lambda^{GK3i_{dB}+(\mathrm{Cut}_{\mathrm{ns}})+(\mathrm{Weak})} \longrightarrow \Lambda s$ is given as follows

$$\begin{array}{rcl} \mathcal{S}(n) & = & n \\ \mathcal{S}(\lambda e) & = & \lambda\mathcal{S}(e) \\ \mathcal{S}(\mathrm{bind}_1\ (n\,e) \text{ in } e') & = & (\lambda\mathcal{S}(e'))\,(n\,\mathcal{S}(e)) \\ \mathcal{S}(e\{\!\{k := e'\}\!\}) & = & \mathcal{S}(e)\sigma^k\mathcal{S}(e') \\ \mathcal{S}(\mathcal{W}^m_k(e)) & = & \varphi^{m+1}_k\mathcal{S}(e) \end{array}$$

Definition 11. $\mathcal{T} : \Lambda^{\mathrm{cs}} \longrightarrow \Lambda t$ is given as follows

$$\begin{array}{rcl} \mathcal{T}(n) & = & n \\ \mathcal{T}(\lambda e) & = & \lambda\mathcal{T}(e) \\ \mathcal{T}(\mathrm{bind}_1\ (n\,e) \text{ in } e') & = & (\lambda\mathcal{T}(e'))\,(n\,\mathcal{T}(e)) \\ \mathcal{T}(e[\![k := e']\!]) & = & \mathcal{T}(e)\varsigma^k\mathcal{T}(e') \\ \mathcal{T}(\mathcal{W}^1_k(e)) & = & \theta_k\mathcal{T}(e) \end{array}$$

Observe that $\mathcal{S}$ does not translate λ^{ns} terms of the form $\mathcal{R}^k_l(e)$. The λs calculus does not have a construct with the correct index-updating behavior to be the target of the translation. The calculi of explicit substitution which we consider as targets for the translation of λ^{ns} and λ^{cs} do not support either the simple exchange rule or the slightly more powerful relocation rule (equivalent to multiple uses of exchange involving the same formula occurrence) that appears as (Reloc) in λ^{ns}. Thus, the statements that follow will (seemingly) say less about the relationship between λ^{ns} and λs than they will about the relationship between λ^{cs} and λt.

As can be seen from Figure 7, uses of the (Reloc) rule are only introduced in one case. The need for (Reloc) in that case could be avoided altogether by using a more general rule than (→L) allowing any formula in the right premise to be discharged, rather than just the first. However, this would merely be moving the difficulty, as such a more general rule also does not seem to have a translation working exclusively at the object-level, cf. the discussion on L-System Formulations in Section 1. If, instead, we had considered a translation that used the appropriate notion of meta-level rather than object-level substitution for the bind-construct, the contentious use of the (Reloc) rule would be absorbed into the action of the substitution operation and the mapping could be made total. We have not done so in order to stay in line with the rest of the article which has taken a "term model" approach but it can be done (Vestergaard 1999). Besides, as we shall see, the chosen approach has its own merits.

We now define restrictions and slight modifications of λs and λt to operate on the images of the (relevant) terms of λ^{ns} and λ^{cs} under $\mathcal{S}$ and $\mathcal{T}$. Let therefore $\Lambda s^- = \mathcal{S}(\Lambda^{GK3i_{dB}+(\mathrm{Cut_{ns}})+(\mathrm{Weak})})$ and $\Lambda t^- = \mathcal{T}(\Lambda^{\mathrm{cs}})$. Define $a \to_{\lambda s^-} a'$ whenever for $a \in \Lambda s^-$ there is some a'' such that $a \to_{\lambda s} a''$ and $a'' \to^*_{\lambda s} a'$ is a rewrite sequence of minimal length such that $a' \in \Lambda s^-$. Define $b \to_{\lambda t^-} b'$ similarly. Theorems 14 and 15, apart from establishing the simulation results, also prove that there always is one rewrite sequence with the stated property in both cases.

The restricted calculi λs^- and λt^- are then $\lambda s^- = (\Lambda s^-, \to_{\lambda s^-})$ and $\lambda t^- = (\Lambda t^-, \to_{\lambda t^-})$. Thus, we have restricted all applications to be of the form $(\lambda\cdot)\,(n\,\cdot)$ and we treat such constructs as atomic with respect to the rewrite rules. Let $\to_{\varphi^-}$ be the rewrite relation using only (minimal) rewrite sequences with rules of λs^- pertaining to the φ operator and let $\to_{\theta^-}$ similarly use only (minimal) rewrite sequences with rules of λt^- pertaining to the θ operator.

Proposition 12. For $e, e' \in \Lambda^{GK3i_{dB}+(\mathrm{Cut_{ns}})+(\mathrm{Weak})}$ we have

$$e \to_w e' \quad \Leftrightarrow \quad \mathcal{S}(e) \to_{\varphi^-} \mathcal{S}(e')$$

Proof. See Appendix F. □

That is, the index-updating mechanisms of λs and λ^{ns} coincide, except for the lack of (Reloc).

Proposition 13. For $e, e' \in \Lambda^{\mathrm{cs}}$ we have

$$e \to_w e' \quad \Leftrightarrow \quad \mathcal{T}(e) \to_{\theta^-} \mathcal{T}(e')$$

Proof. In effect a special case of the above. □

Although the above propositions show the index-updating mechanisms correspond, the rules (Bind^{1}_{-}) of Figure 7 and 8 (and any specialisation of them) are neither supported by λs nor by λt (but see Section 3.2 below). Hence we are led to consider the calculi without these rules. Define therefore the restricted calculus $\lambda^{\mathrm{ns}-}$ to be λ^{ns} without $(\mathrm{Bind}^1_{\mathrm{ns}})$ and the restricted calculus $\lambda^{\mathrm{cs}-}$ to be λ^{cs} without $(\mathrm{Bind}^1_{\mathrm{cs}})$.

Theorem 14. *λs simulates $\lambda^{\mathrm{ns}-}$, i.e., for $e, e' \in \Lambda^{GK3i_{dB}+(\mathrm{Cut}_{\mathrm{ns}})+(\mathrm{Weak})}$:*

$$e \rightarrow_{\mathrm{ns}-} e' \quad \Rightarrow \quad \mathcal{S}(e) \rightarrow_{\lambda s^-} \mathcal{S}(e')$$

Proof. See Appendix G and Proposition 12. □

Theorem 15. *λt simulates $\lambda^{\mathrm{cs}-}$, i.e., for $e, e' \in \Lambda^{\mathrm{cs}}$:*

$$e \rightarrow_{\mathrm{cs}-} e' \quad \Rightarrow \quad \mathcal{T}(e) \rightarrow_{\lambda t^-} \mathcal{T}(e')$$

Proof. See Appendix H and Proposition 13. □

3.2. *Relating Full Cut Elimination*

So we have seen that most of the syntax and rules of λ^{ns} map onto syntax and rules of λs and all of the syntax and most of the rules of λ^{cs} map onto syntax and rules of λt. Still, this leaves out the rewrite rules (Bind^1_{-}). Taking advantage of the totality on syntax of the latter mapping, $\mathcal{T}$, we can however translate $(\mathrm{Bind}^1_{\mathrm{cs}})$ into λt as suggested by the mapping:

$$e\varsigma^k((\lambda e_2)\,(n\,e_1)) \quad \rightarrow \quad (\lambda(\theta_0 e)\varsigma^{k+1} e_2)\,(n\,e_1)$$

The essential core of this rule in λ-calculus notation is

$$(\lambda x.M)((\lambda y.N)(z\,P)) \rightarrow (\lambda y.(\lambda x.M)N)(z\,P)$$

Note that this rewrite rule appears (subject to variations of syntax and various restrictions) in a wide variety of places in the literature. (See (Kfoury and Wells 1995) for a discussion of some of these occurrences.) This suggests that we could add such a rule to λt to thus make λt partly L-system-like rather than predominantly N-system-like. It remains to verify the properties of such a modified λt.

Alternatively, and preferably, we could map the bind-construct to λs and λt via a meta-level version of their substitution mechanisms. This would lead to "λs and λt as homomorphic images of cut propagation" thus showing that the Zucker-Pottinger epimorphism (Zucker 1974; Pottinger 1977) factors through calculi with explicit substitution (Vestergaard 1999).

We are presently working on both these issues where especially the latter is found to be interesting from both the perspectives of proof theory and explicit substitution calculi.

3.3. *The Format of Contexts; $\lambda bxgc$*

The two cut rules we have considered are also admissible when we consider contexts as sets of variables, each mapping to a formula. This is not equivalent to having contexts as multisets of formulae.¶¶ In particular, the former kind of formalization performs implicit

¶¶ That is not to say that the two cut rules are not admissible with contexts as multisets; they are!

$$e \in \Lambda^{GK3i_{var}+(\mathrm{Cut})} \quad ::= \quad x \mid \lambda x.e \mid \mathrm{let}\ y = (f\,e)\ \mathrm{in}\ e \mid e\langle\!\langle x := e\rangle\!\rangle$$
$$x, y, f, g \quad \in \quad Var$$

$$\begin{array}{lr}
y\langle\!\langle x := e\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} y & (\mathrm{Realign}_{\mathrm{var}}) \\
(\lambda y.e')\langle\!\langle x := e\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} \lambda y.e'\langle\!\langle x := e\rangle\!\rangle \quad \text{if } y \notin \mathrm{FV}(e) & (\mathrm{Abstr}_{\mathrm{var}}) \\
e\langle\!\langle x := (\mathrm{let}\ y = (f\,e_1)\ \mathrm{in}\ e_2)\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} \mathrm{let}\ y = (f\,e_1)\ \mathrm{in}\ e\langle\!\langle x := e_2\rangle\!\rangle \quad \text{if } y \notin \mathrm{FV}(e) & (\mathrm{Bind}^1_{\mathrm{var}}) \\
(\mathrm{let}\ y = (f\,e_1)\ \mathrm{in}\ e_2)\langle\!\langle x := e\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} \mathrm{let}\ y = (f\,e_1\langle\!\langle x := e\rangle\!\rangle)\ \mathrm{in}\ e_2\langle\!\langle x := e\rangle\!\rangle \quad \text{if } y \notin \mathrm{FV}(e) & (\mathrm{Bind}^2_{\mathrm{var}}) \\
x\langle\!\langle x := e\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} e & (\mathrm{Subst}_{\mathrm{var}}) \\
(\mathrm{let}\ y = (f\,e_1)\ \mathrm{in}\ e_2)\langle\!\langle f := g\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} \mathrm{let}\ y = (g\,e_1\langle\!\langle f := g\rangle\!\rangle)\ \mathrm{in}\ e_2\langle\!\langle f := g\rangle\!\rangle & (\mathrm{Reindex}_{\mathrm{var}}) \\
(\mathrm{let}\ y = (f\,e_1)\ \mathrm{in}\ e_2)\langle\!\langle f := \lambda x.e\rangle\!\rangle & \\
\quad \to_{GK3i_{var}} e_2\langle\!\langle f := \lambda x.e\rangle\!\rangle\langle\!\langle y := e\langle\!\langle x := e_1\langle\!\langle f := \lambda x.e\rangle\!\rangle\rangle\!\rangle\rangle\!\rangle \quad \text{if } y \notin \mathrm{FV}(\lambda x.e) & (\beta_{\mathrm{var}})
\end{array}$$

Fig. 11. Propagation of (Cut) (obtained from ($\mathrm{Cut}_{\mathrm{ns}}$) and ($\mathrm{Cut}_{\mathrm{cs}}$)) on $GK3i_{var}$ written as the SR-enjoying rewriting rule of the associated calculus of proof terms, $\lambda^{GK3i_{var}}$.

$$a \in \Lambda bxgc \quad ::= \quad x \mid \lambda x.a \mid a\,a \mid a\langle\!| x := a |\!\rangle$$
$$x, y \quad \in \quad Var$$

$$\begin{array}{rcl}
(\lambda x.a_1)\,a_2 & \to_{\lambda bxgc} & a_1\langle\!| x := a_2 |\!\rangle \\
x\langle\!| x := a |\!\rangle & \to_{\lambda bxgc} & a \\
(\lambda y.a_1)\langle\!| x := a_2 |\!\rangle & \to_{\lambda bxgc} & \lambda y.a_1\langle\!| x := a_2 |\!\rangle \qquad y \notin \mathrm{FV}(a_2) \\
(a_1\,a_2)\langle\!| x := a |\!\rangle & \to_{\lambda bxgc} & a_1\langle\!| x := a |\!\rangle\, a_2\langle\!| x := a |\!\rangle \\
a_1\langle\!| x := a_2 |\!\rangle & \to_{\lambda bxgc} & a_1 \qquad \text{if } x \notin \mathrm{FV}(a_1)
\end{array}$$

Fig. 12. The $\lambda bxgc$-calculus.

contraction. In fact, Dragalin's G3 calculus, which collapses the Church numerals as it stands (cf. Section 1.1 – L-System Formulations), becomes well-behaved in the sense of Theorems 14 and 15 when using contexts as sets of variables (Vestergaard 1998b).

Figure 11 shows the collapse of the cut propagations on $GK3i_{dB}$ to cut propagation on a version of Kleene's G3 with contexts as sets of variables, $GK3i_{var}$. We observe that the two cut rules have exactly the same cut propagation procedure. In fact, the two cut rules become inseparable under (Weak)-equivalence which we henceforth will employ. Two derivations are thus considered equivalent if they rewrite to the same derivation by propagating occurrences of (Weak) *towards* their roots (that is, opposite the standard direction). Observe that this notion of (Weak) normal forms takes into account that not all occurrences of (Weak) can make it all the way to the root. It can happen that an occurrence meets an internal rule application that discharge the assumption in question and hence is blocked from further propagation.

The garbage collection rule of $\lambda bxgc$ (in the last line of Figure 12) corresponds, besides subsuming the case: $y\langle\!| x := a |\!\rangle$ with $x \neq y$, to the straighforward propagation of a cut on a newly weakened assumption:

$$\dfrac{\Gamma \vdash e' : \sigma \qquad \dfrac{\Gamma \vdash e : \tau}{x : \sigma, \Gamma \vdash e : \tau}\,(\mathrm{Weak_{var}})_x \quad \text{if } x \notin \mathrm{FV}(e)}{\Gamma \vdash e\langle\!\langle x := e' \rangle\!\rangle : \tau}\,(\mathrm{Cut})_x$$

The above derivation will be taken to $\rightarrow_{GK3i_{var}}$-reduce to the derivation ending in $\Gamma \vdash e : \tau$.

Let $\mathcal{X} : \Lambda^{GK3i_{var}+(\mathrm{Cut})} \longrightarrow \Lambda bxgc$ be defined parallel to $\mathcal{S}$ and $\mathcal{T}$ (Definitions 10 and 11):

$$\begin{array}{rcl}
\mathcal{X}(x) & = & x \\
\mathcal{X}(\lambda x.e) & = & \lambda x.\mathcal{X}(e) \\
\mathcal{X}(\text{let } x = (f\, e_1) \text{ in } e_2) & = & (\lambda x.\mathcal{X}(e_2))\,(f\,\mathcal{X}(e_1)) \\
\mathcal{X}(e_1\langle\!\langle x := e_2 \rangle\!\rangle) & = & \mathcal{X}(e_1)\langle\!| x := \mathcal{X}(e_2) |\!\rangle
\end{array}$$

And let $\lambda^{GK3i_{var}-}$ be $\lambda^{GK3i_{var}}$ minus Line $(\mathrm{Bind}^1_{\mathrm{var}})$ of Figure 11 but including the above garbage collection rule. Finally, let $\rightarrow_{\lambda bxgc^-}$ be the restriction of $\rightarrow_{\lambda bxgc}$ to the image of $\mathcal{X}$ (parallel to the cases for λs^- and λt^-).

Theorem 16. $\lambda bxgc$ simulates $\lambda^{GK3i_{var}-}$, i.e., for $e, e' \in \Lambda^{GK3i_{var}+(\mathrm{Cut})}$:

$$e \rightarrow_{GK3i_{var}-} e' \quad\Rightarrow\quad \mathcal{X}(e) \rightarrow_{\lambda bxgc^-} \mathcal{X}(e')$$

Proof. Parallel to the proofs of Theorems 14 and 15. □

We conclude by pointing out that garbage collection rules also can be introduced naturally into λs and λt along the (proof-theoretical) lines suggested above, cf. (Vestergaard 1998a).

3.4. *Assessing the Results*

Having thus studied the direct relationships between cut elimination calculi and explicit substitution calculi, a few comments are in order.

— We have presented the very first simulation results between cut elimination procedures and explicit substitution calculi. The results formally justify the slogans: weakening is index-updating, cut is explicit substitution. In fact, the cut elimination procedures we consider are "off-the-shelf" while the explicit substitution calculi are well-established in their own right.
Furthermore, the major developments of the paper are done with cut elimination procedures and explicit substitution calculi that are completely specified at the object-level. This means among other things that they explicitly resolve binding during reduction and thus qualify as first-order TRS's. It also means that more details need

to be in accordance to establish the above slogans hence strenghtening the weight the slogans carry.

— The cut elimination calculus $\lambda^{GK3i_{var}}$ using variables-as-names is obtained from both the cut elimination calculi λ^{ns}, λ^{cs} using variables-as-indices by using (variable) notation that *a priori* guarantees correct binding resolution.
  The latter two calculi differ in the way they manage assumptions during cut elimination. In fact, they differ so radically that while the contentious $(\text{Bind}^1_{\text{cs}})$ rule of λ^{cs} (as well as $(\text{Bind}^1_{\text{var}})$ of $\lambda^{GK3i_{var}}$) can be mapped on the object-level to a rule of generalized reduction for the corresponding explicit substitution calculus, λt ($\lambda bxgc$), this is not possible for $(\text{Bind}^1_{\text{ns}})$ of λ^{ns} without adding syntax to λs. One lesson to learn is that the abstract variables-as-names paradigm subsumes (in the formal sense just described) a number of somewhat different concrete representations of variables.

— Quite peculiarly, we also established that the context-sharing versus non-context-sharing nature of the cut rules is directly related to the incremental versus centralized index updating mechanisms of the explicit substitution calculi (λt and λs).

— The apparent weakness of the simulation results, their failure to address one particular rule configuration when doing cut elimination, can be overcome by using a simple meta-level approach that eradicates the above mentioned difficulties (Vestergaard 1999). This issue is left as future work.

4. Conclusion and Future Work

We have studied Curry-Howard Isomorphisms between L-systems and λ-like calculi with explicit substitutions. We constructed such isomorphisms by inducing rewrite rules on calculi of proof terms from admissibility proofs of cut rules based on cut propagation. Many aspects of these calculi are interesting in their own right as we saw in Section 2. In a wider perspective, Section 3 focused on relating the calculi to other known computational structures as cuts-as-explicit-substitutions.

As a first, we have shown an interesting correspondence between the dynamics of the cut and structural rules of L-systems on one hand and known explicit substitution calculi *à la* de Bruijn on the other. We have also used known results on first-order TRS's (O'Donnell 1977) in a correct way to obtain strong cut elimination results as well as confluence.

In carrying this work out, some issues have arisen which we have left to future work. One issue is the study of complete (and oriented) axiomatizations of the equational theory of cut elimination by incorporating permutative conversions. Another issue is the study of cut elimination in L-systems with higher-order type constructors. Typed calculi with explicit substitutions for such constructs have proven difficult in N-system settings. Yet another issue is to further investigate the correspondence between the proof theory of L-systems and N-systems based on calculi with explicit substitutions. And a final issue is the further study of the use of the proof theory of L-systems to interrelate known calculi with explicit substitutions.

Appendix A. Proof details of Theorem 1 for (Weak)-Propagation

Preceded by (Ax) :

$$\dfrac{\dfrac{}{\Gamma^1,\Gamma^2 \vdash n : \tau}\,(\mathrm{Ax})_n}{\Gamma^1,\Gamma,\Gamma^2 \vdash \mathcal{W}_k^m(n) : \tau}\,(\mathrm{Weak})_{k,m} \qquad \text{where } |\Gamma^1| = k \text{ and } |\Gamma| = m$$

rewrites to:

$$\dfrac{}{\Gamma^1,\Gamma,\Gamma^2 \vdash n' : \tau}\,(\mathrm{Ax})_{n'} \qquad \text{where } n' = \begin{cases} n & \text{if } n \le k \\ n+m & \text{if } n > k \end{cases}$$

Preceded by ($\to$**R**) :

$$\dfrac{\dfrac{\tau,\Gamma^1,\Gamma^2 \vdash e : \sigma}{\Gamma^1,\Gamma^2 \vdash \lambda e : \tau \to \sigma}\,(\to\mathrm{R})}{\Gamma^1,\Gamma,\Gamma^2 \vdash \mathcal{W}_k^m(\lambda e) : \tau \to \sigma}\,(\mathrm{Weak})_{k,m} \qquad \text{where } |\Gamma^1| = k \text{ and } |\Gamma| = m$$

rewrites to:

$$\dfrac{\dfrac{\tau,\Gamma^1,\Gamma^2 \vdash e : \sigma}{\tau,\Gamma^1,\Gamma,\Gamma^2 \vdash \mathcal{W}_{k+1}^m(e) : \sigma}\,(\mathrm{Weak})_{k+1,m}}{\Gamma^1,\Gamma,\Gamma^2 \vdash \lambda\mathcal{W}_{k+1}^m(e) : \tau \to \sigma}\,(\to\mathrm{R})$$

Preceded by ($\to$**L**) :

$$\dfrac{\dfrac{\Gamma^1,\Gamma^2 \vdash e_1 : \tau \qquad \sigma,\Gamma^1,\Gamma^2 \vdash e_2 : \nu}{\Gamma^1,\Gamma^2 \vdash \mathrm{bind}_1\ (n\, e_1) \text{ in } e_2 : \nu}\,(\to\mathrm{L})_n}{\Gamma^1,\Gamma,\Gamma^2 \vdash \mathcal{W}_k^m(\mathrm{bind}_1\ (n\, e_1) \text{ in } e_2) : \nu}\,(\mathrm{Weak})_{k,m} \qquad \begin{array}{l} \text{where } \Gamma^1,\Gamma^2 = \tau_1,\ldots,\tau_{n-1},\tau \to \sigma,\tau_{n+1},\ldots \\ \text{where } |\Gamma^1| = k \text{ and } |\Gamma| = m \end{array}$$

rewrites to:

$$\dfrac{\dfrac{\Gamma^1,\Gamma^2 \vdash e_1 : \tau}{\Gamma^1,\Gamma,\Gamma^2 \vdash \mathcal{W}_k^m(e_1) : \tau}\,(\mathrm{Weak})_{k,m} \qquad \dfrac{\sigma,\Gamma^1,\Gamma^2 \vdash e_2 : \nu}{\sigma,\Gamma^1,\Gamma,\Gamma^2 \vdash \mathcal{W}_{k+1}^m(e_2) : \nu}\,(\mathrm{Weak})_{k+1,m}}{\Gamma^1,\Gamma,\Gamma^2 \vdash \mathrm{bind}_1\ (n'\, \mathcal{W}_k^m(e_1)) \text{ in } \mathcal{W}_{k+1}^m(e_2) : \nu}\,(\to\mathrm{L})_{n'}$$

$$\text{where } n' = \begin{cases} n & \text{if } n \le k \\ n+m & \text{if } n > k \end{cases}$$

Appendix B. Proof details of Theorem 1 for (Reloc)-Propagation

Preceded by (Ax) :

$$\dfrac{\dfrac{}{\Gamma^1,\Gamma^2,\sigma,\Gamma^3 \vdash n : \tau}\,(\mathrm{Ax})_n}{\Gamma^1,\sigma,\Gamma^2,\Gamma^3 \vdash \mathcal{R}_l^k(n) : \tau}\,(\mathrm{Reloc})_{k,l} \qquad \text{where } |\Gamma^1,\Gamma^2| = k-1 \text{ and } |\Gamma^1| = l-1$$

rewrites to:

$$\frac{}{\Gamma^1, \sigma, \Gamma^2, \Gamma^3 \vdash n' : \tau}\,(\mathrm{Ax})_{n'} \qquad \text{where } n' = \begin{cases} n & \text{if } n < l \ \vee \ n > k \\ l & \text{if } n = k \\ n+1 & \text{if } l \leq n < k \end{cases}$$

Preceded by $(\to\mathbf{R})$:

$$\frac{\dfrac{\tau_1, \Gamma^1, \Gamma^2, \sigma, \Gamma^3 \vdash e : \tau_2}{\Gamma^1, \Gamma^2, \sigma, \Gamma^3 \vdash \lambda e : \tau_1 \to \tau_2}\,(\to\mathrm{R})}{\Gamma^1, \sigma, \Gamma^2, \Gamma^3 \vdash \mathcal{R}^k_l(\lambda e) : \tau_1 \to \tau_2}\,(\mathrm{Reloc})_{k,l} \qquad \text{where } |\Gamma^1, \Gamma^2| = k-1 \text{ and } |\Gamma^1| = l-1$$

rewrites to:

$$\frac{\dfrac{\tau_1, \Gamma^1, \Gamma^2, \sigma, \Gamma^3 \vdash e : \tau_2}{\tau_1, \Gamma^1, \sigma, \Gamma^2, \Gamma^3 \vdash \mathcal{R}^{k+1}_{l+1}(e) : \tau_2}\,(\mathrm{Reloc})_{k+1,l+1}}{\Gamma^1, \sigma, \Gamma^2, \Gamma^3 \vdash \lambda\mathcal{R}^{k+1}_{l+1}(e) : \tau_1 \to \tau_2}\,(\to\mathrm{R})$$

Preceded by $(\to\mathbf{L})$:

With $\Gamma^1, \Gamma^2, \tau, \Gamma^3$ given as $\sigma_1, \ldots, \sigma_{n-1}, \tau_1 \to \tau_2, \sigma_{n+1}, \ldots$ we have:

$$\frac{\dfrac{\Gamma^1, \Gamma^2, \tau, \Gamma^3 \vdash e_1 : \tau_1 \qquad \tau_2, \Gamma^1, \Gamma^2, \tau, \Gamma^3 \vdash e_2 : \tau_3}{\Gamma^1, \Gamma^2, \tau, \Gamma^3 \vdash \mathrm{bind}_1\ (n\, e_1) \text{ in } e_2 : \tau_3}\,(\to\mathrm{L})_n}{\Gamma^1, \tau, \Gamma^2, \Gamma^3 \vdash \mathcal{R}^k_l(\mathrm{bind}_1\ (n\, e_1) \text{ in } e_2) : \tau_3}\,(\mathrm{Reloc})_{k,l} \qquad \text{where } |\Gamma^1, \Gamma^2| = k-1 \text{ and } |\Gamma^1| = l-1$$

rewrites to:

$$\frac{\dfrac{\Gamma^1, \Gamma^2, \tau, \Gamma^3 \vdash e_1 : \tau_1}{\Gamma^1, \tau, \Gamma^2, \Gamma^3 \vdash \mathcal{R}^k_l(e_1) : \tau_1}\,(\mathrm{Reloc})_{k,l} \qquad \dfrac{\tau_2, \Gamma^1, \Gamma^2, \tau, \Gamma^3 \vdash e_2 : \tau_3}{\tau_2, \Gamma^1, \tau, \Gamma^2, \Gamma^3 \vdash \mathcal{R}^{k+1}_{l+1}(e_2) : \tau_3}\,(\mathrm{Reloc})_{k+1,l+1}}{\Gamma^1, \tau, \Gamma^2, \Gamma^3 \vdash \mathrm{bind}_1\ (n'\, \mathcal{R}^k_l(e_1)) \text{ in } \mathcal{R}^{k+1}_{l+1}(e_2) : \tau_3}\,(\to\mathrm{L})_{n'}$$

$$\text{where } n' = \begin{cases} n & \text{if } n < l \ \vee \ n > k \\ l & \text{if } n = k \\ n+1 & \text{if } l \leq n < k \end{cases}$$

Appendix C. Proof details of Theorem 1 for $(\mathrm{Cut_{ns}})$-Propagation

We will call a formula occurrence *principal* if it is newly introduced by a $(\to\mathrm{R})$- or a $(\to\mathrm{L})$-rule occurrence or if it is the indexed assumption or the consequent formula in an (Ax)-rule occurrence. For ease of reference we will henceforth say "principal formula" and omit the word "occurrence". We will also say, e.g., principal $(\to\mathrm{R})$ with the obvious meaning.

In the following case analysis we have superscripted the new cut occurrences by **f** and **d** if they are ordered to the originating cut occurrence by formula-size or by derivation size, respectively. In the latter case the formula-size is unchanged.

We first do case analysis on occurrences of the cut rule with at least one non-principal cut-formula then those with two principal cut formulae.

(Ax) **on the left with non-principal cut-formula** :

Not possible.

(Ax) **on the right with non-principal cut-formula** :

$$\dfrac{\Gamma^1 \vdash e : \tau \qquad \dfrac{}{\Gamma, \tau, \Gamma^2 \vdash n : \sigma}\,(\mathrm{Ax})_n}{\Gamma, (\Gamma^1 \oplus \Gamma^2) \vdash n\{\!\{k := e\}\!\} : \sigma}\,(\mathrm{Cut}_{\mathrm{ns}})_k \qquad \text{where } |\Gamma| + 1 = k \neq n$$

rewrites to:

$$\dfrac{}{\Gamma, (\Gamma^1 \oplus \Gamma^2) \vdash n' : \sigma}\,(\mathrm{Ax})_{n'} \qquad \text{where } n' = \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases}$$

($\to$**R**) **on the left with non-principal cut-formula** :

Not possible.

($\to$**R**) **on the right (necessarily) with non-principal cut-formula** :

$$\dfrac{\Gamma^1 \vdash e : \rho \qquad \dfrac{\tau, \Gamma, \rho, \Gamma^2 \vdash e' : \sigma}{\Gamma, \rho, \Gamma^2 \vdash \lambda e' : \tau \to \sigma}\,(\to\mathrm{R})}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash (\lambda e')\{\!\{k := e\}\!\} : \tau \to \sigma}\,(\mathrm{Cut}_{\mathrm{ns}})_k \qquad \text{where } \Gamma^1 \oplus \Gamma^2 \text{ is defined}$$

rewrites to:

$$\dfrac{\dfrac{\Gamma^1 \vdash e : \rho \qquad \tau, \Gamma, \rho, \Gamma^2 \vdash e' : \sigma}{\tau, \Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e'\{\!\{k+1 := e\}\!\} : \sigma}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{d}}_{k+1}}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \lambda e'\{\!\{k+1 := e\}\!\} : \tau \to \sigma}\,(\to\mathrm{R})$$

($\to$**L**) **on the left (necessarily) with non-principal cut-formula** :

$$\dfrac{\dfrac{\Gamma^1 \vdash e_1 : \tau \qquad \sigma, \Gamma^1 \vdash e_2 : \nu}{\Gamma^1 \vdash \mathrm{bind}_1\ (n\, e_1) \text{ in } e_2 : \nu}\,(\to\mathrm{L})_n \qquad \Gamma, \nu, \Gamma^2 \vdash e : \rho}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e\{\!\{k := (\mathrm{bind}_1\ (n\, e_1) \text{ in } e_2)\}\!\} : \rho}\,(\mathrm{Cut}_{\mathrm{ns}})_k$$

rewrites to:

$$\dfrac{\dfrac{\Gamma^1 \vdash e_1 : \tau}{\Gamma, \Gamma^1 \vdash \mathcal{W}_0^{k-1}(e_1) : \tau}\,(\mathrm{Weak})_{0,k-1} \qquad \dfrac{\dfrac{\sigma, \Gamma^1 \vdash e_2 : \nu \qquad \dfrac{\Gamma, \nu, \Gamma^2 \vdash e : \rho}{\Gamma, \nu, \sigma, \Gamma^2 \vdash \mathcal{W}_k^1(e) : \rho}\,(\mathrm{Weak})_{k,1}}{\Gamma, \sigma, \Gamma^1 \oplus \Gamma^2 \vdash \mathcal{W}_k^1(e)\{\!\{k := e_2\}\!\} : \rho}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{d}}_k}{\sigma, \Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \mathcal{R}_1^k((\mathcal{W}_k^1(e)\{\!\{k := e_2\}\!\})) : \rho}\,(\mathrm{Reloc})_{k,1}}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \mathrm{bind}_1\ ((n+k-1)\, \mathcal{W}_0^{k-1}(e_1)) \text{ in } \mathcal{R}_1^k((\mathcal{W}_k^1(e)\{\!\{k := e_2\}\!\})) : \rho}\,(\to\mathrm{L})_{n+k-1}$$

We observe that for the last rule application to be fully correct we need a weakening lemma which can be different in nature from mere application of a weakening rule, although the latter in this case will suffice (when we at the same time prove that the associated index-updating is the identity). The Weakening Lemma says that *if there is derivation ending in* $\Gamma \vdash e : \tau$ *then there is a derivation ending in* $\Gamma, \Gamma' \vdash e : \tau$. The lemma is straightforwardly correct in the present set-up and we will not prove it here.

(→L) on the right with non-principal cut-formula :

With Γ, ρ, Γ^2 given as $\tau_1, \ldots, \tau_{n-1}, \tau \to \sigma, \tau_{n+1}, \ldots$ we have:

$$\dfrac{\Gamma^1 \vdash e : \rho \qquad \dfrac{\Gamma, \rho, \Gamma^2 \vdash e_1 : \tau \qquad \sigma, \Gamma, \rho, \Gamma^2 \vdash e_2 : \nu}{\Gamma, \rho, \Gamma^2 \vdash \mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2 : \nu}(\to\mathrm{L})_n}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash (\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2)\{\!\{k := e\}\!\} : \nu}(\mathrm{Cut_{ns}})_k \qquad \text{with } |\Gamma| = k-1 \ \wedge\ n \neq k$$

rewrites to:

$$\dfrac{\dfrac{\Gamma^1 \vdash e : \rho \qquad \Gamma, \rho, \Gamma^2 \vdash e_1 : \tau}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_1\{\!\{k := e\}\!\} : \tau}(\mathrm{Cut_{ns}})^{\mathbf{d}}_k \qquad \dfrac{\Gamma^1 \vdash e : \rho \qquad \sigma, \Gamma, \rho, \Gamma^2 \vdash e_2 : \nu}{\sigma, \Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_2\{\!\{k+1 := e\}\!\} : \nu}(\mathrm{Cut_{ns}})^{\mathbf{d}}_{k+1}}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \mathrm{bind}_1\ (n'\,(e_1\{\!\{k := e\}\!\}))\ \mathrm{in}\ e_2\{\!\{k+1 := e\}\!\} : \nu}(\to\mathrm{L})_{n'}$$

$$\text{where } n' = \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases}$$

Principal (Ax) **on the right, any rule on the left :**

$$\dfrac{\Gamma^1 \vdash e : \tau \qquad \dfrac{}{\Gamma, \tau, \Gamma^2 \vdash k : \tau}(\mathrm{Ax})_k}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash k\{\!\{k := e\}\!\} : \tau}(\mathrm{Cut_{ns}})_k$$

rewrites to:

$$\dfrac{\Gamma^1 \vdash e : \tau}{\Gamma, \Gamma^1 \vdash \mathcal{W}_0^{k-1}(e) : \tau}(\mathrm{Weak})_{0,k-1}$$

Again, we need the Weakening Lemma.

Principal (Ax) **on the left, principal (→R) on the right :**

Not possible.

Principal (Ax) **on the left, principal (→L) on the right :**

$$\dfrac{\dfrac{}{\Gamma^1 \vdash m : \tau \to \sigma}(\mathrm{Ax})_m \qquad \dfrac{\Gamma, \tau \to \sigma, \Gamma^2 \vdash e_1 : \tau \quad \sigma, \Gamma, \tau \to \sigma, \Gamma^2 \vdash e_2 : \nu}{\Gamma, \tau \to \sigma, \Gamma^2 \vdash \mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2 : \nu}(\to\mathrm{L})_k \text{ where } |\Gamma| = k-1}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash (\mathrm{bind}_1\ (k\, e_1)\ \mathrm{in}\ e_2)\{\!\{k := m\}\!\} : \nu}(\mathrm{Cut_{ns}})_k$$

rewrites to:

$$\dfrac{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_1\{\!\{k := m\}\!\} : \tau \qquad \dfrac{\dfrac{}{\Gamma^1 \vdash m : \tau \to \sigma}\,(\mathrm{Ax})_m \qquad \sigma, \Gamma, \tau \to \sigma, \Gamma^2 \vdash e_2 : \nu}{\sigma, \Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_2\{\!\{k+1 := m\}\!\} : \nu}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{d}}_{k+1}}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \mathrm{bind}_1\ ((m+k-1)\,(e_1\{\!\{k := m\}\!\}))\ \mathrm{in}\ e_2\{\!\{k+1 := m\}\!\} : \nu}\,(\to\mathrm{L})_{m+k-1}$$

With left premise given as:

$$\dfrac{\dfrac{}{\Gamma^1 \vdash m : \tau \to \sigma}\,(\mathrm{Ax})_m \qquad \Gamma, \tau \to \sigma, \Gamma^2 \vdash e_1 : \tau}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_1\{\!\{k := m\}\!\} : \tau}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{d}}_{k}$$

Principal ($\to$R) on the left, principal ($\to$R) on the right :

Not possible.

Principal ($\to$R) on the left, principal ($\to$L) on the right :

With $|\Gamma| = k-1$ we have:

$$\dfrac{\dfrac{\tau, \Gamma^1 \vdash e : \sigma}{\Gamma^1 \vdash \lambda e : \tau \to \sigma}\,(\to\mathrm{R}) \qquad \dfrac{\Gamma, \tau \to \sigma, \Gamma^2 \vdash e_1 : \tau \qquad \sigma, \Gamma, \tau \to \sigma, \Gamma^2 \vdash e_2 : \nu}{\Gamma, \tau \to \sigma, \Gamma^2 \vdash \mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2 : \nu}\,(\to\mathrm{L})_k}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash (\mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2)\{\!\{k := \lambda e\}\!\} : \nu}\,(\mathrm{Cut}_{\mathrm{ns}})_k$$

rewrites to:

$$\dfrac{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \mathcal{W}_1^{k-1}(e)\{\!\{1 := e_1\{\!\{k := \lambda e\}\!\}\}\!\} : \sigma \qquad \sigma, \Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_2\{\!\{k+1 := \lambda e\}\!\} : \nu}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_2\{\!\{k+1 := \lambda e\}\!\}\{\!\{1 := \mathcal{W}_1^{k-1}(e)\{\!\{1 := e_1\{\!\{k := \lambda e\}\!\}\}\!\}\}\!\} : \nu}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{f}}_1$$

with left premise:

$$\dfrac{\dfrac{\dfrac{\tau, \Gamma^1 \vdash e : \sigma}{\Gamma^1 \vdash \lambda e : \tau \to \sigma}\,(\to\mathrm{R}) \qquad \Gamma, \tau \to \sigma, \Gamma^2 \vdash e_1 : \tau}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_1\{\!\{k := \lambda e\}\!\} : \tau}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{d}}_{k} \qquad \dfrac{\tau, \Gamma^1 \vdash e : \sigma}{\tau, \Gamma, \Gamma^1 \vdash \mathcal{W}_1^{k-1}(e) : \sigma}\,(\mathrm{Weak})_{1,k-1}}{\Gamma, \Gamma^1 \oplus \Gamma^2 \vdash \mathcal{W}_1^{k-1}(e)\{\!\{1 := e_1\{\!\{k := \lambda e\}\!\}\}\!\} : \sigma}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{f}}_1$$

and right premise:

$$\dfrac{\dfrac{\tau, \Gamma^1 \vdash e : \sigma}{\Gamma^1 \vdash \lambda e : \tau \to \sigma}\,(\to\mathrm{R}) \qquad \sigma, \Gamma, \tau \to \sigma, \Gamma^2 \vdash e_2 : \nu}{\sigma, \Gamma, \Gamma^1 \oplus \Gamma^2 \vdash e_2\{\!\{k+1 := \lambda e\}\!\} : \nu}\,(\mathrm{Cut}_{\mathrm{ns}})^{\mathbf{d}}_{k+1}$$

Principal ($\to$L) on the left, any rule on the right :

Not possible.

Appendix D. Proof details of Theorem 1 for $(\mathrm{Cut_{cs}})$-Propagation

We refer to Appendix C for basic definitions.

(Ax) **on the left with non-principal cut-formula** :

Not possible.

(Ax) **on the right with non-principal cut-formula** :

$$\dfrac{\Gamma^1,\Gamma^2 \vdash e:\tau \qquad \dfrac{}{\Gamma^1,\tau,\Gamma^2 \vdash n:\sigma}\,(\mathrm{Ax})_n}{\Gamma^1,\Gamma^2 \vdash n[\![k:=e]\!]:\sigma}\,(\mathrm{Cut_{cs}})_k \qquad \text{where } |\Gamma^1|+1=k\neq n$$

rewrites to:

$$\dfrac{}{\Gamma^1,\Gamma^2 \vdash n':\sigma}\,(\mathrm{Ax})_{n'} \qquad \text{where } n'=\begin{cases} n & \text{if } n<k \\ n-1 & \text{if } n>k \end{cases}$$

$(\to\mathbf{R})$ **on the left with non-principal cut-formula** :

Not possible.

$(\to\mathbf{R})$ **on the right (necessarily) with non-principal cut-formula** :

$$\dfrac{\Gamma^1,\Gamma^2 \vdash e:\nu \qquad \dfrac{\tau,\Gamma^1,\nu,\Gamma^2 \vdash e':\sigma}{\Gamma^1,\nu,\Gamma^2 \vdash \lambda e':\tau\to\sigma}\,(\to\mathrm{R})}{\Gamma^1,\Gamma^2 \vdash (\lambda e')[\![k:=e]\!]:\tau\to\sigma}\,(\mathrm{Cut_{cs}})_k$$

rewrites to:

$$\dfrac{\dfrac{\dfrac{\Gamma^1,\Gamma^2 \vdash e:\nu}{\tau,\Gamma^1,\Gamma^2 \vdash \mathcal{W}_0^1(e):\nu}\,(\mathrm{Weak})_{0,1} \qquad \tau,\Gamma^1,\nu,\Gamma^2 \vdash e':\sigma}{\tau,\Gamma^1,\Gamma^2 \vdash e'[\![k+1:=\mathcal{W}_0^1(e)]\!]:\sigma}\,(\mathrm{Cut_{cs}})^{\mathbf{d}}_{k+1}}{\Gamma^1,\Gamma^2 \vdash \lambda e'[\![k+1:=\mathcal{W}_0^1(e)]\!]:\tau\to\sigma}\,(\to\mathrm{R})$$

$(\to\mathbf{L})$ **on the left (necessarily) with non-principal cut-formula** :

$$\dfrac{\dfrac{\Gamma^1,\Gamma^2 \vdash e_1:\tau \qquad \sigma,\Gamma^1,\Gamma^2 \vdash e_2:\nu}{\Gamma^1,\Gamma^2 \vdash \mathrm{bind}_1\ (n\,e_1)\ \mathrm{in}\ e_2:\nu}\,(\to\mathrm{L})_n \qquad \Gamma^1,\nu,\Gamma^2 \vdash e:\rho}{\Gamma^1,\Gamma^2 \vdash e[\![k:=\mathrm{bind}_1\ (n\,e_1)\ \mathrm{in}\ e_2]\!]:\rho}\,(\mathrm{Cut_{cs}})_k$$

rewrites to:

$$\dfrac{\Gamma^1,\Gamma^2 \vdash e_1 : \tau \qquad \dfrac{\sigma,\Gamma^1,\Gamma^2 \vdash e_2 : \nu \qquad \dfrac{\Gamma^1,\nu,\Gamma^2 \vdash e : \rho}{\sigma,\Gamma^1,\nu,\Gamma^2 \vdash \mathcal{W}_0^1(e) : \rho}\,(\text{Weak})_{0,1}}{\sigma,\Gamma^1,\Gamma^2 \vdash \mathcal{W}_0^1(e)[\![k+1 := e_2]\!] : \rho}\,(\text{Cut}_{\text{cs}})^{\mathbf{d}}_{k+1}}{\Gamma^1,\Gamma^2 \vdash \text{bind}_1\ (n\,e_1)\ \text{in}\ \mathcal{W}_0^1(e)[\![k+1 := e_2]\!] : \rho}\,(\to\text{L})_n$$

($\to$L) on the right with non-principal cut-formula :

With Γ^1,ρ,Γ^2 given as $\tau_1,\ldots,\tau_{n-1},\tau\to\sigma,\tau_{n+1},\ldots$ we have:

$$\dfrac{\Gamma^1,\Gamma^2 \vdash e : \rho \qquad \dfrac{\Gamma^1,\rho,\Gamma^2 \vdash e_1 : \tau \qquad \sigma,\Gamma^1,\rho,\Gamma^2 \vdash e_2 : \nu}{\Gamma^1,\rho,\Gamma^2 \vdash \text{bind}_1\ (n\,e_1)\ \text{in}\ e_2 : \nu}\,(\to\text{L})_n}{\Gamma^1,\Gamma^2 \vdash (\text{bind}_1\ (n\,e_1)\ \text{in}\ e_2)[\![k := e]\!] : \nu}\,(\text{Cut}_{\text{cs}})_k \qquad \text{where } |\Gamma^1|+1 = k \neq n$$

rewrites to:

$$\dfrac{\dfrac{\Gamma^1,\Gamma^2 \vdash e : \rho \qquad \Gamma^1,\rho,\Gamma^2 \vdash e_1 : \tau}{\Gamma^1,\Gamma^2 \vdash e_1[\![k := e]\!] : \tau}\,(\text{Cut}_{\text{cs}})^{\mathbf{d}}_k \qquad \sigma,\Gamma^1,\Gamma^2 \vdash e_2[\![k := \mathcal{W}_0^1(e)]\!] : \nu}{\Gamma^1,\Gamma^2 \vdash \text{bind}_1\ (n'\,(e_1[\![k := e]\!]))\ \text{in}\ e_2[\![k := \mathcal{W}_0^1(e)]\!] : \nu}\,(\to\text{L})_{n'}$$

$$\text{where } n' = \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases}$$

The right premise is given as:

$$\dfrac{\dfrac{\Gamma^1,\Gamma^2 \vdash e : \rho}{\sigma,\Gamma^1,\Gamma^2 \vdash \mathcal{W}_0^1(e) : \rho}\,(\text{Weak})_{0,1} \qquad \sigma,\Gamma^1,\rho,\Gamma^2 \vdash e_2 : \nu}{\sigma,\Gamma^1,\Gamma^2 \vdash e_2[\![k := \mathcal{W}_0^1(e)]\!] : \nu}\,(\text{Cut}_{\text{cs}})^{\mathbf{d}}_k$$

Principal (Ax) **on the right, any rule on the left** :

$$\dfrac{\Gamma^1,\Gamma^2 \vdash e : \tau \qquad \dfrac{}{\Gamma^1,\tau,\Gamma^2 \vdash k : \tau}\,(\text{Ax})_k}{\Gamma^1,\Gamma^2 \vdash k[\![k := e]\!] : \tau}\,(\text{Cut}_{\text{cs}})_k$$

rewrites to:

$$\Gamma^1,\Gamma^2 \vdash e : \tau$$

Principal (Ax) **on the left, principal ($\to$R) on the right** :

Not possible.

Principal (Ax) **on the left, principal ($\to$L) on the right** :

With $|\Gamma^1| = k-1$ we have:

$$\dfrac{\dfrac{}{\Gamma^1,\Gamma^2 \vdash m : \tau\to\sigma}\,(\mathrm{Ax})_m \qquad \dfrac{\Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_1:\tau \qquad \sigma,\Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_2:\nu}{\Gamma^1,\tau\to\sigma,\Gamma^2\vdash \mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2:\nu}\,(\to\mathrm{L})_k}{\Gamma^1,\Gamma^2\vdash(\mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2)[\![k:=m]\!]:\nu}\,(\mathrm{Cut}_{\mathrm{cs}})_k$$

rewrites to:

$$\dfrac{\Gamma^1,\Gamma^2\vdash e_1[\![k:=m]\!]:\tau \qquad \sigma,\Gamma^1,\Gamma^2\vdash e_2[\![k+1:=m+1]\!]:\nu}{\Gamma^1,\Gamma^2\vdash \mathrm{bind}_1\ (m\,e_1[\![k:=m]\!])\ \mathrm{in}\ e_2[\![k+1:=m+1]\!]:\nu}\,(\to\mathrm{L})_m$$

With left premise:

$$\dfrac{\dfrac{}{\Gamma^1,\Gamma^2\vdash m:\tau\to\sigma}\,(\mathrm{Ax})_m \qquad \Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_1:\tau}{\Gamma^1,\Gamma^2\vdash e_1[\![k:=m]\!]:\tau}\,(\mathrm{Cut}_{\mathrm{cs}})^{\mathbf{d}}_k$$

and right premise:

$$\dfrac{\dfrac{}{\sigma,\Gamma^1,\Gamma^2\vdash m+1:\tau\to\sigma}\,(\mathrm{Ax})_{m+1} \qquad \sigma,\Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_2:\nu}{\sigma,\Gamma^1,\Gamma^2\vdash e_2[\![k+1:=m+1]\!]:\nu}\,(\mathrm{Cut}_{\mathrm{cs}})^{\mathbf{d}}_{k+1}$$

Principal ($\to$R) on the left, principal ($\to$R) on the right :

Not possible.

Principal ($\to$R) on the left, principal ($\to$L) on the right :

With $|\Gamma^1| = k-1$ we have:

$$\dfrac{\dfrac{\tau,\Gamma^1,\Gamma^2\vdash e:\sigma}{\Gamma^1,\Gamma^2\vdash \lambda e:\tau\to\sigma}\,(\to\mathrm{R}) \qquad \dfrac{\Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_1:\tau \qquad \sigma,\Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_2:\nu}{\Gamma^1,\tau\to\sigma,\Gamma^2\vdash \mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2:\nu}\,(\to\mathrm{L})_k}{\Gamma^1,\Gamma^2\vdash(\mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2)[\![k:=\lambda e]\!]:\nu}\,(\mathrm{Cut}_{\mathrm{cs}})_k$$

rewrites to:

$$\dfrac{\Gamma^1,\Gamma^2\vdash e[\![1:=e_1[\![k:=\lambda e]\!]]\!]:\sigma \qquad \sigma,\Gamma^1,\Gamma^2\vdash e_2[\![k+1:=\mathcal{W}^1_0(\lambda e)]\!]:\nu}{\Gamma^1,\Gamma^2\vdash e_2[\![k+1:=\mathcal{W}^1_0(\lambda e)]\!][\![1:=e[\![1:=e_1[\![k:=\lambda e]\!]]\!]]\!]:\nu}\,(\mathrm{Cut}_{\mathrm{cs}})^{\mathbf{f}}_1$$

With left premise:

$$\dfrac{\dfrac{\dfrac{\tau,\Gamma^1,\Gamma^2\vdash e:\sigma}{\Gamma^1\vdash\lambda e:\tau\to\sigma}\,(\to\mathrm{R}) \qquad \Gamma^1,\tau\to\sigma,\Gamma^2\vdash e_1:\tau}{\Gamma^1,\Gamma^2\vdash e_1[\![k:=\lambda e]\!]:\tau}\,(\mathrm{Cut}_{\mathrm{cs}})^{\mathbf{d}}_k \qquad \tau,\Gamma^1,\Gamma^2\vdash e:\sigma}{\Gamma^1,\Gamma^2\vdash e[\![1:=e_1[\![k:=\lambda e]\!]]\!]:\sigma}\,(\mathrm{Cut}_{\mathrm{cs}})^{\mathbf{f}}_1$$

and right premise:

$$\dfrac{\dfrac{\dfrac{\tau, \Gamma^1, \Gamma^2 \vdash e : \sigma}{\Gamma^1, \Gamma^2 \vdash \lambda e : \tau \to \sigma}\,(\to\mathrm{R})}{\sigma, \Gamma^1, \Gamma^2 \vdash \mathcal{W}_0^1(\lambda e) : \tau \to \sigma}\,(\mathrm{Weak})_{0,1} \qquad \sigma, \Gamma^1, \tau \to \sigma, \Gamma^2 \vdash e_2 : \nu}{\sigma, \Gamma^1, \Gamma^2 \vdash e_2[\![k+1 := \mathcal{W}_0^1(\lambda e)]\!] : \nu}\,(\mathrm{Cut}_{\mathrm{cs}})^{\mathbf{d}}_{k+1}$$

Principal ($\to$L) on the left, any rule on the right :

Not possible.

Appendix E. Proof details of Theorem 7

View the rewriting rules of the calculi as TRS rewriting rules defined over the (countable) alphabet of function symbols containing all the syntactic constructors, with corresponding arity, plus the integers of arity zero. The variable symbols are e, e', e_1, e_2. There are countably many rewriting rules, each obtained as an instance of the calculus rewriting rules by instantiating the indexes (m, k, n, l) to the integers fulfilling the side-conditions on the rules. The right-hand sides have the additions and subtractions performed. For example, rule $(\mathrm{Bind}^2_{\mathrm{ns}})$ of Figure 7, expands (countably) to:

$$\begin{array}{rcl}
(\mathrm{bind}_1\ (1\,e_1)\ \mathrm{in}\ e_2)\{\!\{2 := e\}\!\} & \to & \mathrm{bind}_1\ (1\,e_1\{\!\{2 := e\}\!\})\ \mathrm{in}\ (e_2\{\!\{3 := e\}\!\}) \\
(\mathrm{bind}_1\ (1\,e_1)\ \mathrm{in}\ e_2)\{\!\{3 := e\}\!\} & \to & \mathrm{bind}_1\ (1\,e_1\{\!\{3 := e\}\!\})\ \mathrm{in}\ (e_2\{\!\{4 := e\}\!\}) \\
(\mathrm{bind}_1\ (2\,e_1)\ \mathrm{in}\ e_2)\{\!\{3 := e\}\!\} & \to & \mathrm{bind}_1\ (2\,e_1\{\!\{3 := e\}\!\})\ \mathrm{in}\ (e_2\{\!\{4 := e\}\!\}) \\
 & \vdots & \\
(\mathrm{bind}_1\ (2\,e_1)\ \mathrm{in}\ e_2)\{\!\{1 := e\}\!\} & \to & \mathrm{bind}_1\ (1\,e_1\{\!\{1 := e\}\!\})\ \mathrm{in}\ (e_2\{\!\{2 := e\}\!\}) \\
(\mathrm{bind}_1\ (3\,e_1)\ \mathrm{in}\ e_2)\{\!\{1 := e\}\!\} & \to & \mathrm{bind}_1\ (2\,e_1\{\!\{1 := e\}\!\})\ \mathrm{in}\ (e_2\{\!\{2 := e\}\!\}) \\
(\mathrm{bind}_1\ (3\,e_1)\ \mathrm{in}\ e_2)\{\!\{2 := e\}\!\} & \to & \mathrm{bind}_1\ (2\,e_1\{\!\{2 := e\}\!\})\ \mathrm{in}\ (e_2\{\!\{3 := e\}\!\}) \\
 & \vdots &
\end{array}$$

The rules are easily seen to be left-linear: no left-hand side contains more than one occurrence of any given variable symbol. Observe that a few rules contain multiple occurrences of a function symbol but that is unrelated to the issue of left-linearity.

A TRS is non-overlapping if no two rewriting rules constitute a critical pair: one left-hand side is a non-variable subterm, up-to replacement of variable symbols, of the other left-hand side. In our case that holds true for pairs of (the countably infinite many) rewriting rules as the mutually exclusive side-conditions on the calculus rules ensure that all TRS rules differ on least one integer as a function symbol.

Appendix F. Proof of Proposition 12

We proceed by structural induction handling both implications simultaneously.

Case $e = \mathcal{W}_k^m(n)$

$$\begin{tikzcd}
\mathcal{W}_k^m(n) \arrow[r, "w"'] \arrow[d, mapsto, "\mathcal{S}"'] & \begin{cases} n & \text{if } n \le k \\ n+m & \text{if } n > k \end{cases} \arrow[d, mapsto, "\mathcal{S}"] \\
\varphi_k^{m+1} n \arrow[r, "\varphi^-"'] & \begin{cases} n & \text{if } n \le k \\ n+(m+1)-1 & \text{if } n > k \end{cases}
\end{tikzcd}$$

The lower leg is well-defined as it is a one-step λs-reduction and, hence, has a (minimal) zero-length completion to reach an element of Λs^-.

Case $e = \mathcal{W}_k^m(\lambda e')$

$$\begin{tikzcd}
\mathcal{W}_k^m(\lambda e') \arrow[r, "w"'] \arrow[d, mapsto, "\mathcal{S}"'] & \lambda \mathcal{W}_{k+1}^m(e') \arrow[d, mapsto, "\mathcal{S}"] \\
\varphi_k^{m+1} \lambda \mathcal{S}(e') \arrow[r, "\varphi^-"'] & \lambda \varphi_{k+1}^{m+1} \mathcal{S}(e')
\end{tikzcd}$$

The lower leg is again well-defined as it is a one-step λs-reduction and, hence, has a (minimal) zero-length completion to reach a term of Λs^-.

Case $e = \mathcal{W}_k^m(\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2)$

$$\begin{tikzcd}
\mathcal{W}_k^m(\mathrm{bind}_1\ (n\, e_1)\ \mathrm{in}\ e_2) \arrow[r, "w"'] \arrow[d, mapsto, "\mathcal{S}"'] & \begin{cases} \mathrm{bind}_1\ (n\, \mathcal{W}_k^m(e_1))\ \mathrm{in}\ \mathcal{W}_{k+1}^m(e_2) & \text{if } n \le k \\ \mathrm{bind}_1\ ((n+m)\, \mathcal{W}_k^m(e_1))\ \mathrm{in}\ \mathcal{W}_{k+1}^m(e_2) & \text{if } n > k \end{cases} \arrow[d, mapsto, "\mathcal{S}"] \\
\varphi_k^{m+1}(\lambda \mathcal{S}(e_2))\,(n\, \mathcal{S}(e_1)) \arrow[r, "\varphi^-"'] & \begin{cases} (\lambda \varphi_{k+1}^{m+1} \mathcal{S}(e_2))\,(n\,(\varphi_k^{m+1} \mathcal{S}(e_1))) & \text{if } n \le k \\ (\lambda \varphi_{k+1}^{m+1} \mathcal{S}(e_2))\,((n+(m+1)-1)(\varphi_k^{m+1} \mathcal{S}(e_1))) & \text{if } n > k \end{cases}
\end{tikzcd}$$

The lower leg comes from the following λs-rewriting sequence:

$$\begin{array}{lll}
\varphi_k^{m+1}(\lambda \mathcal{S}(e_2))\,(n\, \mathcal{S}(e_1)) & \to_{\lambda s} & (\varphi_k^{m+1} \lambda \mathcal{S}(e_2))\,(\varphi_k^{m+1}(n\, \mathcal{S}(e_1))) \\
& \to_{\lambda s} & (\lambda \varphi_{k+1}^{m+1} \mathcal{S}(e_2))\,(\varphi_k^{m+1}(n\, \mathcal{S}(e_1))) \\
& \to_{\lambda s} & (\lambda \varphi_{k+1}^{m+1} \mathcal{S}(e_2))\,((\varphi_k^{m+1} n)\,(\varphi_k^{m+1} \mathcal{S}(e_1))) \\
& \to_{\lambda s} & \begin{cases} (\lambda \varphi_{k+1}^{m+1} \mathcal{S}(e_2))\,(n\,(\varphi_k^{m+1} \mathcal{S}(e_1))) & \text{if } n \le k \\ (\lambda \varphi_{k+1}^{m+1} \mathcal{S}(e_2))\,((n+(m+1)-1)(\varphi_k^{m+1} \mathcal{S}(e_1))) & \text{if } n > k \end{cases}
\end{array}$$

We see that once the first λs-step has been made, we can only reach a Λs^--term if at least the prescribed reductions (or their residuals) are performed. The reason is that the term in consideration is an application that has to be brought into the restricted form allowed in Λs^-. Performing a reduction in, say, $\mathcal{S}(e_1)$ will only lengthen the λs-reduction sequence and, hence, render it a non λs^--reduction.

Appendix G. Proof of Theorem 14

The proof proceeds by a case analysis according to Figure 7.

The argumentation for why the prescribed λs-reduction sequences amount to λs^--reductions steps is similar to the argumentation given in Appendix F.

Case (Realign$_{\text{ns}}$)

$$\begin{array}{rcl} \mathcal{S}(n\{\!\{k := e\}\!\}) & = & n\sigma^k\mathcal{S}(e) \\ & \to_{\lambda s} & \begin{cases} n-1 & \text{if } n > k \\ n & \text{if } n < k \end{cases} \end{array}$$

Case (Abstr$_{\text{ns}}$)

$$\begin{array}{rcl} \mathcal{S}((\lambda e')\{\!\{k := e\}\!\}) & = & (\lambda\mathcal{S}(e'))\sigma^k\mathcal{S}(e) \\ & \to_{\lambda s} & \lambda\mathcal{S}(e')\sigma^{k+1}\mathcal{S}(e) \\ & = & \mathcal{S}(\lambda e'\{\!\{k+1 := e\}\!\}) \end{array}$$

Case (Bind$^2_{\text{ns}}$)

$$\begin{array}{rl} \multicolumn{2}{l}{\mathcal{S}((\text{bind}_1\ (n\, e_1)\ \text{in}\ e_2)\{\!\{k := e\}\!\})} \\ = & ((\lambda\mathcal{S}(e_2))\,(n\,\mathcal{S}(e_1)))\sigma^k\mathcal{S}(e) \\ \to_{\lambda s} & ((\lambda\mathcal{S}(e_2))\sigma^k\mathcal{S}(e))\,((n\,\mathcal{S}(e_1))\sigma^k\mathcal{S}(e)) \\ \to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1}\mathcal{S}(e))\,((n\,\mathcal{S}(e_1))\sigma^k\mathcal{S}(e)) \\ \to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1}\mathcal{S}(e))\,((n\sigma^k\mathcal{S}(e))\,(\mathcal{S}(e_1)\sigma^k\mathcal{S}(e))) \\ \to_{\lambda s} & \begin{cases} (\lambda\mathcal{S}(e_2)\sigma^{k+1}\mathcal{S}(e))\,(n\,(\mathcal{S}(e_1)\sigma^k\mathcal{S}(e))) & \text{if } n < k \\ (\lambda\mathcal{S}(e_2)\sigma^{k+1}\mathcal{S}(e))\,((n-1)\,(\mathcal{S}(e_1)\sigma^k\mathcal{S}(e))) & \text{if } n > k \end{cases} \\ = & \begin{cases} \mathcal{S}(\text{bind}_1\ (n\, e_1\{\!\{k := e\}\!\})\ \text{in}\ e_2\{\!\{k+1 := e\}\!\}) & \text{if } n < k \\ \mathcal{S}(\text{bind}_1\ ((n-1)\, e_1\{\!\{k := e\}\!\})\ \text{in}\ e_2\{\!\{k+1 := e\}\!\}) & \text{if } n > k \end{cases} \end{array}$$

Case (Subst$_{\text{ns}}$)

$$\begin{array}{rcl} \mathcal{S}(k\{\!\{k := e\}\!\}) & = & k\sigma^k\mathcal{S}(e) \\ & \to_{\lambda s} & \varphi_0^k\mathcal{S}(e) \\ & = & \mathcal{S}(\mathcal{W}_0^{k-1}(e)) \end{array}$$

Case (Reindex$_{\text{ns}}$)

$$\mathcal{S}((\text{bind}_1\ (k\, e_1)\ \text{in}\ e_2)\{\!\{k := m\}\!\})$$

$$
\begin{array}{ll}
= & ((\lambda\mathcal{S}(e_2))\,(n\,\mathcal{S}(e_1)))\sigma^k m \\
\to_{\lambda s} & ((\lambda\mathcal{S}(e_2))\sigma^k m)\,((k\,\mathcal{S}(e_1))\sigma^k m) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1} m)\,((k\,\mathcal{S}(e_1))\sigma^k m) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1} m)\,((k\sigma^k m)\,(\mathcal{S}(e_1)\sigma^k m)) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1} m)\,((\varphi_0^k m)\,(\mathcal{S}(e_1)\sigma^k m)) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1} m)\,((m+k-1)\,(\mathcal{S}(e_1)\sigma^k m)) \\
= & \mathcal{S}(\mathrm{bind}_1\ ((m+k-1)\,(e_1\{\!\{k := m\}\!\}))\ \mathrm{in}\ e_2\{\!\{k+1 := m\}\!\})
\end{array}
$$

Case (β_{ns})

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathcal{S}((\mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2)\{\!\{k := \lambda e\}\!\})} \\
= & ((\lambda\mathcal{S}(e_2))\,(k\,\mathcal{S}(e_1)))\sigma^k(\lambda\mathcal{S}(e)) \\
\to_{\lambda s} & ((\lambda\mathcal{S}(e_2))\sigma^k(\lambda\mathcal{S}(e)))\,((k\,\mathcal{S}(e_1))\sigma^k(\lambda\mathcal{S}(e))) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1}(\lambda\mathcal{S}(e)))\,((k\,\mathcal{S}(e_1))\sigma^k(\lambda\mathcal{S}(e))) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1}(\lambda\mathcal{S}(e))) \\
 & ((k\sigma^k(\lambda\mathcal{S}(e)))\,(\mathcal{S}(e_1)\sigma^k(\lambda\mathcal{S}(e)))) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1}(\lambda\mathcal{S}(e))) \\
 & ((\varphi_0^k(\lambda\mathcal{S}(e)))\,(\mathcal{S}(e_1)\sigma^k(\lambda\mathcal{S}(e)))) \\
\to_{\lambda s} & (\lambda\mathcal{S}(e_2)\sigma^{k+1}(\lambda\mathcal{S}(e))) \\
 & ((\lambda\varphi_1^k\mathcal{S}(e))\,(\mathcal{S}(e_1)\sigma^k(\lambda\mathcal{S}(e)))) \\
\to_{\lambda s} & (\mathcal{S}(e_2)\sigma^{k+1}(\lambda\mathcal{S}(e)))\sigma^1 \\
 & ((\lambda\varphi_1^k\mathcal{S}(e))\,(\mathcal{S}(e_1)\sigma^k(\lambda\mathcal{S}(e)))) \\
\to_{\lambda s} & (\mathcal{S}(e_2)\sigma^{k+1}(\lambda\mathcal{S}(e)))\sigma^1 \\
 & ((\varphi_1^k\mathcal{S}(e))\sigma^1(\mathcal{S}(e_1)\sigma^k(\lambda\mathcal{S}(e)))) \\
= & \mathcal{S}(e_2\{\!\{k+1 := \lambda e\}\!\}\{\!\{1 := (\mathcal{W}_1^{k-1}(e))\{\!\{1 := e_1\{\!\{k := \lambda e\}\!\}\}\!\}\}\!\})
\end{array}
$$

Appendix H. Proof of Theorem 15

The proof proceeds by a case analysis according to Figure 8.

The argumentation for why the prescribed λt-reduction sequences amount to λt^--reductions steps is similar to the argumentation given in Appendix F.

Case (**Realign**$_{\mathrm{cs}}$)

$$
\begin{array}{lll}
\mathcal{T}(n[\![k := e]\!]) & = & n\varsigma^k\mathcal{T}(e) \\
 & \to_{\lambda t} & \begin{cases} n-1 & \text{if } n > k \\ n & \text{if } n < k \end{cases}
\end{array}
$$

Case (**Abstr**$_{\mathrm{cs}}$)

$$
\mathcal{T}((\lambda e')[\![k := e]\!]) \quad = \quad (\lambda\mathcal{T}(e'))\varsigma^k\mathcal{T}(e)
$$

$$\begin{array}{ll} \to_{\lambda t} & \lambda\mathcal{T}(e')\varsigma^{k+1}(\theta_0\mathcal{T}(e)) \\ = & \mathcal{T}(\lambda e'[\![k+1 := \mathcal{W}_0^1(e)]\!]) \end{array}$$

Case $(\mathbf{Bind}^2_{\mathrm{cs}})$

$$\begin{array}{ll} \multicolumn{2}{l}{\mathcal{T}((\mathrm{bind}_1\ (n\,e_1)\ \mathrm{in}\ e_2)[\![k := e]\!])} \\ = & ((\lambda\mathcal{T}(e_2))\,(n\,\mathcal{T}(e_1)))\varsigma^k\mathcal{T}(e) \\ \to_{\lambda t} & ((\lambda\mathcal{T}(e_2))\varsigma^k\mathcal{T}(e))\,((n\,\mathcal{T}(e_1))\varsigma^k\mathcal{T}(e)) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0\mathcal{T}(e)))\,((n\,\mathcal{T}(e_1))\varsigma^k\mathcal{T}(e)) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0\mathcal{T}(e)))\,((n\varsigma^k\mathcal{T}(e))\,(\mathcal{T}(e_1)\varsigma^k\mathcal{T}(e))) \\ \to_{\lambda t} & \begin{cases} (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0\mathcal{T}(e)))\,(n\,(\mathcal{T}(e_1)\varsigma^k\mathcal{T}(e))) & \text{if } n < k \\ (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0\mathcal{T}(e)))\,((n-1)\,(\mathcal{T}(e_1)\varsigma^k\mathcal{T}(e))) & \text{if } n > k \end{cases} \\ = & \begin{cases} \mathcal{T}(\mathrm{bind}_1\ (n\,e_1[\![k := e]\!])\ \mathrm{in}\ e_2[\![k+1 := \mathcal{W}_0^1(e)]\!]) & \text{if } n < k \\ \mathcal{T}(\mathrm{bind}_1\ ((n-1)\,e_1[\![k := e]\!])\ \mathrm{in}\ e_2[\![k+1 := \mathcal{W}_0^1(e)]\!]) & \text{if } n > k \end{cases} \end{array}$$

Case $(\mathbf{Subst}_{\mathrm{cs}})$

$$\begin{array}{rcl} \mathcal{T}(k[\![k := e]\!]) & = & k\varsigma^k\mathcal{T}(e) \\ & \to_{\lambda t} & \mathcal{T}(e) \end{array}$$

Case $(\mathbf{Reindex}_{\mathrm{cs}})$

$$\begin{array}{ll} \multicolumn{2}{l}{\mathcal{T}((\mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2)[\![k := m]\!])} \\ = & ((\lambda\mathcal{T}(e_2))\,(n\,\mathcal{T}(e_1)))\varsigma^k m \\ \to_{\lambda t} & ((\lambda\mathcal{T}(e_2))\varsigma^k m)\,((k\,\mathcal{T}(e_1))\varsigma^k m) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0 m))\,((k\,\mathcal{T}(e_1))\varsigma^k m) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(m+1))\,((k\,\mathcal{T}(e_1))\varsigma^k m) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(m+1))\,((k\varsigma^k m)\,(\mathcal{T}(e_1)\varsigma^k m)) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(m+1))\,(m\,(\mathcal{T}(e_1)\varsigma^k m)) \\ = & \mathcal{T}(\mathrm{bind}_1\ (m\,(e_1[\![k := m]\!]))\ \mathrm{in}\ e_2[\![k+1 := m+1]\!]) \end{array}$$

Case (β_{cs})

$$\begin{array}{ll} \multicolumn{2}{l}{\mathcal{T}((\mathrm{bind}_1\ (k\,e_1)\ \mathrm{in}\ e_2)[\![k := \lambda e]\!])} \\ = & ((\lambda\mathcal{T}(e_2))\,(k\,\mathcal{T}(e_1)))\varsigma^k(\lambda\mathcal{T}(e)) \\ \to_{\lambda t} & ((\lambda\mathcal{T}(e_2))\varsigma^k(\lambda\mathcal{T}(e)))\,((k\,\mathcal{T}(e_1))\varsigma^k(\lambda\mathcal{T}(e))) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0(\lambda\mathcal{T}(e))))\,((k\,\mathcal{T}(e_1))\varsigma^k(\lambda\mathcal{T}(e))) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0(\lambda\mathcal{T}(e))))\,((k\,\mathcal{T}(e_1))\varsigma^k(\lambda\mathcal{T}(e))) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0(\lambda\mathcal{T}(e)))) \\ & ((k\varsigma^k(\lambda\mathcal{T}(e)))\,(\mathcal{T}(e_1)\varsigma^k(\lambda\mathcal{T}(e)))) \\ \to_{\lambda t} & (\lambda\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0(\lambda\mathcal{T}(e)))) \\ & ((\lambda\mathcal{T}(e))\,(\mathcal{T}(e_1)\varsigma^k(\lambda\mathcal{T}(e)))) \end{array}$$

$$
\begin{array}{ll}
\to_{\lambda t} & (\mathcal{T}(e_2)\varsigma^{k+1}(\theta_0(\lambda\mathcal{T}(e))))\varsigma^1 \\
& (\mathcal{T}(e)\varsigma^1(\mathcal{T}(e_1)\varsigma^k(\lambda\mathcal{T}(e)))) \\
= & \mathcal{T}(e_2[\![k+1 := \mathcal{W}_0^1(\lambda e)]\!][\![1 := e[\![1 := e_1[\![k := \lambda e]\!]]\!]]\!])
\end{array}
$$

Acknowledgements The present work owes a lot to the kind attention of Fairouz Kamareddine. Thanks are also due to Roy Dyckhoff and various anonymous referees for comments on earlier versions of this paper. For invitations to talk about the material presented here, thanks are due to Roy Dyckhoff, the ML-club at the University of Edinburgh, the Logik Oberseminar run by Helmut Schwichtenberg at the LMU, Munich, and the organisers of the Federated Logic Conference Workshop: "The Second International Worskhop on Explicit Substitutions: Theory and Applications to Programs and Proofs" (FLoC/WESTAPP'99).

Finally, all derivations have been type-set using Paul Taylor's proof tree macros.

References

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

Abramsky, S. (1993). Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57.

Benaissa, Z., Briaud, D., Lescanne, P., and Rouyer-Degli, J. (1996). $\lambda\nu$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722.

Benton, N., Bierman, G., de Paiva, V., and Hyland, M. (1993). A term calculus for intuitionistic linear logic. In Bezem, M. and Groote, J.F. editors, *Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, pages 75–90, Utrecht, The Netherlands.

Tahhan Bittar, E. (1996). Strong normalisation proofs for cut elimination in Gentzen's sequent calculi. Prépublication du Laboratoire de Logique Algorithmique et Informatique de Clermont, Numéro 56, Université D'Auvergne.

Bloo, R. (1997). *Preservation of Termination for Explicit Substitution.* PhD thesis, Technical University of Eindhoven.

Bloo, R. and Rose, K. (1995). Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*.

Breazu-Tannen, V., Kesner, D., and Puel, L. (1993). A typed pattern calculus. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 262–274.

de Bruijn, N.G. (1978). A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report TH-Report 78-WSK-03, Department of Mathematics, Technical University of Eindhoven.

de Bruijn, N.G. (1978). Lambda calculus notation with namefree formulas involving symbols that represent reference transforming mappings. *Indag. Math.*, 40(3):348–356.

Cerrito, S. and Kesner, D. (1999). Pattern matching as cut elimination. In *Proc. 14th Ann. IEEE Symp. Logic in Computer Sci.*.

Church, A. (1941). *The Calculi of Lambda Conversion.* Princeton University Press.

Di Cosmo, R. and Kesner, D. (1997). Strong normalization of explicit substitutions via cut elimination in proof nets. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*.

Curien, P.-L. (1993). *Categorical Combinators, Sequential Algorithms and Functional Programming*. Progress in Theoretical Computer Science. Birkhaüser.

Curien, P.-L., Hardin, T., and Lévy, J.-J. (1996). Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2):362–397.

Curry, H.B. and Feys, R. (1958). *Combinatory Logic*. North-Holland, Amsterdam.

Dowek, G., Hardin, T., and Kirchner, C. (1995). Higher-order unification via explicit substitutions. In *Symposium on Logic in Computer Science (LICS '95)*, pages 366–507, Los Alamitos, Ca., USA. IEEE Computer Society Press.

Dragalin, A.G. (1979). Translated as *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, 1988.

Dyckhoff, R. and Pinto, L (1999). Permutability of proofs in intuitionistic sequent calculi. *Theoretical Computer Science*, 212(1–2):141–155

Espírito Santo, J. (2000) Revisiting the correspondence between cut elimination and normalisation. To be presented at ICALP'00.

Gallier, J. (1993). Constructive logics, Part I: A tutorial on proof systems and typed λ-calculi. *Theoretical Computer Science*, 110:249–339, 1993.

Gentzen, G. (1935). Investigations into logical deduction. In Szabo, M. E., editor, *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969.

Gramlich, B. (1995). Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, September 1995. Preliminary version appeared as Relating Innermost, Weak, Uniform and Modular Termination of Term Rewriting Systems, LPAR'92.

Guillaume, B. (1997). The λs_e does not have PSN. Manuscript.

Hardin, T., Maranget, L., and Pagano, B. (1996). Functional back-ends within the lambda-sigma calculus. In *Proceedings of the 1996 ACM SIGPLAN International Conference on Functional Programming*, pages 25–33, 24–26 May 1996.

Herbelin, H. (1994). A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Annual Conference of the European Association for Computer Science Logic, CSL'94, Kazimierz (Poland), Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag.

Herbelin, H. (1995). *Séquents qu'on calcule*. PhD thesis, Université Paris VII.

Howard, W.A., (1969). The formulae-as-types notion of construction. In Seldin, J. P., and Hindley, J. R., editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 470–490. Academic Press, 1980.

Kamareddine, F. and Ríos, A. (1995). A λ-calculus à la de Bruijn with explicit substitutions. In *Programming Languages, Implementations, Logics, and Programs*, volume 982, pages 45–62.

Kamareddine, F. and Ríos, A. (1997). Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420.

Kamareddine, F. and Ríos, A. (1998). Bridging de Bruijn indices and variable names in explicit substitutions calculi. *The Logic Journal of the Interest Group of Pure and Applied Logic*, 6(6):843–874.

Kfoury, A.J. and Wells, J.B. (1995). Addendum to "New notions of reduction and non-semantic proofs of β-strong normalization in typed λ-calculi". Tech. Rep. 95-007, Comp. Sci. Dept., Boston Univ.

Kleene, S.C. (1952). *Introduction to Metamathematics*. North-Holland Publishing Co..

Mäenpää, P. (1993). *The Art of Analysis. Logic and History of Problem Solving.* PhD thesis, University of Helsinki.

Magnusson, L. (1995). *The Implementation of ALF—A Proof Editor Based on Martin-Löf's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Chalmers University of Technology and Göteborg University.

Maraist, J., Odersky, M., Turner, D.N., and Wadler, P. (1995). Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. Technical Report 5-95, Fakultät für Informatik, University of Karlsruhe.

Mints, G. (1996). Normal forms for sequent derivations. In Odifreddi, P., editor, *Kreiseliana; about and around Georg Kreisel*, pages 469–492. Wellesley, Massachusetts, A.K. Peters.

Muñoz Hurtado, C. (1996a). Confluence and preservation of strong normalisation in an explicit substitutions calculus. In *Proceedings, 11*$^{\text{th}}$ *Annual IEEE Symposium on Logic in Computer Science*, pages 440–447.

Muñoz Hurtado, C. (1996b). Proof representation in type theory: State of the art. In *XXII Latin-American Conference of Informatics CLEI Panel.*

O'Donnell, M. (1977). *Computing in Systems Described by Equations.* Number 58 in Lecture Notes in Computer Science. Springer-Verlag.

Pagano, B. (1998). An explicit natural deduction. In proceedings of *The First International Workshop on Explicit Substitution: Theory and Applications to Programs and Proofs*, Tsukuba, Japan.

Pfenning, F. (1991). Logic programming in the LF logical framework. In Huet, G., and Plotkin, G., editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

Pfenning, F. (1995). Structural Cut Elimination. In *10th Annual IEEE Symposium on Logic in Computer Science (LICS'95).*

Pottinger, G. (1977). Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357. North-Holland Publishing Company.

Prawitz, D. (1965). *Natural Deduction.* Almquist and Wiksell, Uppsala.

Ríos, A. (1993). *Contribution à l'étude des λ-calculs avec substitutions explicites.* PhD thesis, Université de Paris 7.

Ritter, E., Pym, D., and Wallen, L.A. (1996). On the intuitionistic force of classical search. In Miglioli, P., Moscato, U., Mundici, D., and Ornaghi, M., editors, *Proc of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 295–311, Springer Verlag.

Rose, K. (1996a). Explicit substitutions – tutorial & survey. Technical Report LS-96-13, BRICS, Aarhus University, Denmark.

Rose, K. (1996b). *Operational Reduction Models for Functional Programming Languages.* PhD thesis, University of Copenhagen.

Schwichtenberg, H. (1999) Termination of permutative conversions in intuitionistic Gentzen calculi. *Theoretical Computer Science*, 212(1–2):247–260.

Selhab, S. (1998). *Logiques et Réécriture.* PhD thesis, Université Henri Poincaré — Nancy 1 (Spécialité Informatique), Nancy, France.

Troelstra, A.S. and Schwichtenberg, H. (1996). *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press.

Vestergaard, R. (1998a). The cut rule and explicit substitutions (author's cut). TR-1998-9, Dept. of CS, Uni. of Glasgow; also available from the author's homepage: http://www.cee.hw.ac.uk/˜jrvest/.

Vestergaard, R. (1998b). A computational anomaly in the Troelstra-Schwichtenberg G3i(m) system. Manuscript; available from the author's homepage: http://www.cee.hw.ac.uk/~jrvest/. See also forthcoming PhD thesis.

Vestergaard, R. (1999). The correspondence between cut-propagation, β-reduction, and explicit substitutions. Manuscript. See also forthcoming PhD thesis.

Wadler, P. (1993). A Curry-Howard isomorphism for sequent calculus. Manuscript.

Zucker, J. (1974). The correspondence between cut-elimination and normalization, Part I & II. *Annals of Mathematical Logic*, 7:1–112,113–155. North-Holland Publishing Company.