Cut Rules and Explicit Substitutions

René Vestergaard
Heriot-Watt University
http://www.cee.hw.ac.uk/~jrvest/

Joe Wells*
Heriot-Watt University
http://www.cee.hw.ac.uk/~jbw/

May 20, 1999

Abstract

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 variations 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 a similar result 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. 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 [21]. 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 [21]. 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

*Supported by EPSRC grant GR/L 36963 and NSF grants CCR–9417382 and CCR–9806747.

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 [21].

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.

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 [21] ("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 [27] in the case of the simply typed λ-calculus [12] and the N-system NJ and by Curry [16] 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 [1]. 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 [44], unification [17], representation of incomplete proofs [29, 35], proof theory [26], and functional compiler back-ends [24].

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 substitu-

tions:

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

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

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 [2, 4], work which discusses the relationship between LJ and NJ [54, 41, 42], and other work [37].

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:

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

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

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 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 [20], 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 another example, Pfenning [40, 39] represents a cut elimination proof in Elf.

A second example of this approach is the work of Wadler [53], 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 (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 third example of this approach is the work of Herbelin [25]. 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.

A final example of this approach is in the very recent work of Cerrito and Kesner [11]. 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. In the application of their system for programming languages this seems to be beneficial.

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 [8], but in a manner which does not correspond to what goes on in cut elimina-

tion proofs. Dychoff and Pinto [19] as well as Schwichtenberg [47] 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 between L-systems and N-system [41] are axiomatised by *permutative conversions*. They are concerned with completeness (SN and CR) of permutative conversions.

L-System Formulations The proof theory and the behavior of normalization of an L-system depends on subtle details of its formulation. Gentzen [21] designed the first L-system, called G1 by many later researchers [49]. The intuitionistic fragment of G1, called LJ by Gentzen, has a $\rightarrow$-left rule somewhat like this one:[1]

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

Unfortunately, G1 also needs a contraction rule:

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

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. In particular, 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 contraction or cut elimination from succeeding. 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 [34] devised his G3 system,[2] which we call GK3, the intuitionistic fragment of which has a $\rightarrow$-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 multiset.

Later, Dragalin [18] invented another system, called G3 by Troelstra and Schwichtenberg [49], the intuitionistic fragment of which has a $\rightarrow$-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 Dragalin's G3, both cut and contraction can now be eliminated, but the intermediate stages of the contraction elimination procedure need an inversion principle or rule. Unfortunately, as shown by one of us (Vestergaard) [50], inversion leads to a computational anomoly, because it implies that the functions it is used on are constant functions. Thus, Dragalin's G3 is 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 [10]. Using indices avoids the need to work with α-equivalence classes. When carrying out substitution and also using names, renaming by α-conversion is needed 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 introduce the equivalent of fresh "names" by shifting all other indices.

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

[2] Unfortunately, many systems are known by this name.

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., $C\lambda\xi\phi$ [9]; the calculi of categorical combinators [14]; $\lambda\sigma, \lambda\sigma_{\Uparrow}, \lambda\sigma_{SP}$ [1, 15, 43] (the $\lambda\sigma$-family); $\lambda\nu$ [3], $\lambda\zeta$ [28] (the $\lambda\sigma$-descendants); $\lambda s, \lambda s_e, \lambda t$ [30, 31, 32]. Explicit-substitution calculi for the λ-calculus with names include calculi such as $\lambda bxgc$ and its variants [6, 7, 46, 45].

1.2 Our Contribution

We present a fragment $GK\exists i_{dB}$ of the intuitionistic portion of Kleene's G3 with ordered assumptions. We define two variations on the cut rule, (Cut_{ns}) and (Cut_{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 $GK\exists i_{dB} + (\text{Weak}) + (\text{Reloc}) + (\text{Cut}_{\text{ns}})$ and $GK\exists i_{dB} + (\text{Weak})_{k,1} + (\text{Cut}_{\text{cs}})$. For both of these logics, we define proof term annotations and rewrite rules, yielding the calculi λ^{ns} and λ^{cs}. The two variations allow showing correspondences with the substitution mechanisms of two existing systems of explicit substitution, λs and λt.

Both variations share several properties. The proof term annotations use de Bruijn indices to manage assumptions. Both variations on the cut rule introduce an explicit substitution. The $\rightarrow$-left rule binds the result of a function application. 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 ($GK\exists i_{dB} + (\text{Weak}) + (\text{Reloc}) + (\text{Cut}_{\text{ns}})$ or $GK\exists i_{dB} + (\text{Weak})_{k,1} + (\text{Cut}_{\text{cs}})$).
- A slight restriction (either λ^{ns}_{-} or λ^{cs}_{-}) of each calculus (either λ^{ns} or λ^{cs}) which removes critical pairs of the rewrite rules satisfies (1) SR, (2) the confluence or Church-Rosser (CR) property, and (3) the strong normalizing (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 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:

- 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 previous work, we treat all 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 [38, 22]. This gives a solid foundation for our strong cut-elimination result, in contrast to some other work in 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 $GK3i_{dB} + (\mathrm{Weak}) + (\mathrm{Reloc}) + (\mathrm{Cut_{ns}})$ and $GK3i_{dB} + (\mathrm{Weak})_{k,1} + (\mathrm{Cut_{cs}})$ 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. 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. [36] 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 [13] 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. This is the only other work we are aware of which combines de Bruijn indices with a presentation of cut elimination. However, the logic is an N-system and hence orthogonal to our goals.

2 Logical Deduction and Computation

We define $GK3i_{dB}$, a proof-term-annotated L-system for the implicative fragment of intuitionistic logic. The inference rules of $GK3i_{dB}$ 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 $GK3i_{dB}$ 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, 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 $GK3i_{dB}$ and we will sometimes take $GK3i_{dB}$ to refer to the logic or 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 [34], which we call GK3. Kleene's original definition effectively took contexts to be multisets of formulae, but we take contexts to be lists of formulae. (We will briefly consider contexts as multisets in Section 3.3).

For the $(\rightarrow \mathrm{L})$ 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

Indices, formulae (τ)

$$\begin{array}{rcl} k, m, l, v, n, i & ::= & 0 \mid 1 \mid \ldots \\ \tau, \sigma, \nu, \rho & ::= & \alpha \mid \tau \to \tau \end{array}$$

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

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

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

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

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

Figure 2: The logical rules, together called *GK3i*$_{dB}$.

4) without needing a contraction rule (cf. [49] 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 ($\mathrm{Cut}_{\mathrm{cs}}$) rule is context-sharing, like ($\to$L). In contrast, the ($\mathrm{Cut}_{\mathrm{ns}}$) rule can express completely non-context-sharing cuts in the case when Γ' is empty. We avoid restricting ($\mathrm{Cut}_{\mathrm{ns}}$) to always be 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, ($\mathrm{Cut}_{\mathrm{ns}}$) has some built-in ability to do 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). We consider both the context-sharing and non-context-sharing versions of cut because they are both commonly used and also because their natural propogation rules turn out to behave similarly to the rules of some existing calculi of explicit substitution (see Section 3).

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.

Theorem 1 (Admissibility) *Cuts + Struct are admissible for GK3i*$_{dB}$.

Proof Sketch: From the proof theory viewpoint, for the uses of rules in *Cuts + Struct* 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. By the Curry-Howard isomorphism, we can specify this via rewrite rules on proof terms, which we give

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

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

Figure 3: The cut rules, together called *Cuts*.

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

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

Figure 4: The structural rules, together called *Struct*.

in Figures 5, 6, 7, and 8. The proof is by a (standard) case analysis on the last rule applied in the premise(s). For *Cuts* there are two 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_-)). Then we prove (by standard argument) the innermost strategy for the propagation rules terminates using the sizes of cut-formulae and derivations for innermost cut occurrences. Details are available in [51]. □

Corollary 2 (Hauptsätze) *Every end sequent derived from $GK3i_{dB} + Struct + Cuts$ can be derived from $GK3i_{dB}$.*

The above results are presented from the proof theory viewpoint. Now we present them again from the computation viewpoint.

Definition 3 (Proof Terms) *Let Λ^G be the set of valid proof terms for $G \subseteq GK3i_{dB} + Cuts + Struct$.*

Definition 4 (Calculi of Proof Terms) *Let*

λ^{ns} be the terms
$\Lambda^{\mathrm{ns}} = \Lambda^{GK3i_{dB}+(\mathrm{Weak})+(\mathrm{Reloc})+(\mathrm{Cut}_{\mathrm{ns}})}$ *with rewriting relation*
$\to_{\mathrm{ns}} = \to_w \cup \to_r \cup \to_{c_{ns}}$, *and*

λ^{cs} be the terms
$\Lambda^{\mathrm{cs}} = \Lambda^{GK3i_{dB}+(\mathrm{Weak})_{k,1}+(\mathrm{Cut}_{\mathrm{cs}})}$ *with rewriting relation* $\to_{\mathrm{cs}} = \to_w \cup \to_{c_{cs}}$.

Observe that λ^{cs} does not need all of the possible instances of (Weak).

Theorem 5 *The calculi λ^{ns}, λ^{cs} enjoy SR and WIN. Their normal-form fragment is $\Lambda^{GK3i_{dB}}$ (which is "Cuts + Struct"-free).*

2.1 Strengthening the Results

We will now strengthen the already obtained results in a non-proof-theoretical direction using the following result [38].

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, (Bind$^1_{\mathrm{X}}$) with any of (Realign$_{\mathrm{X}}$), (Abstr$_{\mathrm{X}}$), (Bind$^2_{\mathrm{X}}$), and (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 (Bind$^3_{\mathrm{ns}}$) and (Bind$^3_{\mathrm{cs}}$) presented in Figure 9 which are restrictions of (Bind$^1_{\mathrm{ns}}$) and (Bind$^1_{\mathrm{cs}}$). Let λ^{ns}_- and λ^{cs}_- be the result of replacing (Bind$^1_{\mathrm{ns}}$) and (Bind$^1_{\mathrm{cs}}$) in λ^{ns} and λ^{cs}

$$
\begin{array}{rcl}
\mathcal{W}_k^m(n) & \to_w & \left\{ \begin{array}{ll} n & \text{if } n \leq k \\ n+m & \text{if } n > k \end{array} \right. \\
\mathcal{W}_k^m(\lambda e) & \to_w & \lambda \mathcal{W}_{k+1}^m(e) \\
\mathcal{W}_k^m(\text{bind}_1\ (n\, e) \text{ in } e') & \to_w & \left\{ \begin{array}{ll} \text{bind}_1\ (n\, \mathcal{W}_k^m(e)) \text{ in } \mathcal{W}_{k+1}^m(e') & \text{if } n \leq k \\ \text{bind}_1\ ((n+m)\, \mathcal{W}_k^m(e)) \text{ in } \mathcal{W}_{k+1}^m(e') & \text{if } n > k \end{array} \right.
\end{array}
$$

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

$$
\begin{array}{rcl}
\mathcal{R}_l^k(n) & \to_r & \left\{ \begin{array}{ll} n & \text{if } n < l \vee n > k \\ l & \text{if } n = k \\ n+1 & \text{if } l \leq n < k \end{array} \right. \\
\mathcal{R}_l^k((\lambda e)) & \to_r & \lambda \mathcal{R}_{l+1}^{k+1}(e) \\
\mathcal{R}_l^k((\text{bind}_1\ (n\, e) \text{ in } e')) & \to_r & \left\{ \begin{array}{ll} \text{bind}_1\ (n\, (\mathcal{R}_l^k(e))) \text{ in } \mathcal{R}_{l+1}^{k+1}(e') & \text{if } n < l \vee n > k \\ \text{bind}_1\ (l\, (\mathcal{R}_l^k(e))) \text{ in } \mathcal{R}_{l+1}^{k+1}(e') & \text{if } n = k \\ \text{bind}_1\ ((n+1)\, (\mathcal{R}_l^k(e))) \text{ in } \mathcal{R}_{l+1}^{k+1}(e') & \text{if } l \leq n < k \end{array} \right.
\end{array}
$$

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

respectively by ($\text{Bind}_{\text{ns}}^3$) and ($\text{Bind}_{\text{cs}}^3$). 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 *λ_{-}^{ns} and λ_{-}^{cs} are orthogonal TRS's and have the SR, SN, and CR properties. Their normal-form fragment is $\Lambda^{GK3i_{dB}}$.*

Proof Sketch: Orthogonality is easily checked, although it requires a syntactic translation into the TRS framework which uses a countably infinite number of symbols and which translates each rule of λ_{-}^{ns} and λ_{-}^{cs} into a countably infinite number of TRS rules. CR is obtained by orthogonality of the rules while the specific details for obtaining SN via Theorem 6 are given in [51]. □

General SN results for cut-elimination procedures obtained through general TRS results have also been studied by Tahhan Bittar [5] and Selhab [48]. Dragalin [18] 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 similarly 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 ($\to$L), we could have relied on ordinary λ-binding and application syntax. Hence, instead of writing: "$\text{bind}_1\ (n\, e)$ 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 [30, 32]. Both the λs and λt calculi are confluent on closed terms,[3] preserve strong normalisation (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 [31] which, unfortunately, is not PSN w.r.t. the λ-calculus [23] but is WN the simply-typed case. The λt-calculus is slightly closer than λs to the style of λv and $\lambda\sigma$ in that it does index updating in smaller increments.

[3] The closed terms are the usual terms.

$$
\begin{array}{rcll}
n\{\!\{k := e\}\!\} & \to_{c_{ns}} & \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases} & (\text{Realign}_{\text{ns}}) \\
(\lambda e')\{\!\{k := e\}\!\} & \to_{c_{ns}} & \lambda(e'\{\!\{k+1 := e\}\!\}) & (\text{Abstr}_{\text{ns}}) \\
e\{\!\{k := (\text{bind}_1\ (n\, e_1)\ \text{in}\ e_2)\}\!\} & \to_{c_{ns}} & \text{bind}_1\ ((n+k-1)\,(\mathcal{W}_0^{k-1}(e_1)))\ \text{in}\ \mathcal{R}_1^k((\mathcal{W}_k^1(e))\{\!\{k := e_2\}\!\}) & (\text{Bind}^1_{\text{ns}}) \\
(\text{bind}_1\ (n\, e_1)\ \text{in}\ e_2)\{\!\{k := e\}\!\} & \to_{c_{ns}} & \begin{cases} \text{bind}_1\ (n\, e_1\{\!\{k := e\}\!\})\ \text{in}\ (e_2\{\!\{k+1 := e\}\!\}) & \text{if } n < k \\ \text{bind}_1\ ((n-1)\, e_1\{\!\{k := e\}\!\})\ \text{in}\ (e_2\{\!\{k+1 := e\}\!\}) & \text{if } n > k \end{cases} & (\text{Bind}^2_{\text{ns}}) \\
k\{\!\{k := e\}\!\} & \to_{c_{ns}} & \mathcal{W}_0^{k-1}(e) & (\text{Subst}_{\text{ns}}) \\
(\text{bind}_1\ (k\, e_1)\ \text{in}\ e_2)\{\!\{k := m\}\!\} & \to_{c_{ns}} & \text{bind}_1\ ((m+k-1)\, e_1\{\!\{k := m\}\!\})\ \text{in}\ (e_2\{\!\{k+1 := m\}\!\}) & (\text{Reindex}_{\text{ns}}) \\
(\text{bind}_1\ (k\, e_1)\ \text{in}\ e_2)\{\!\{k := \lambda e\}\!\} & \to_{c_{ns}} & e_2\{\!\{k+1 := \lambda e\}\!\}\{\!\{1 := (\mathcal{W}_1^{k-1}(e))\{\!\{1 := e_1\{\!\{k := \lambda e\}\!\}\}\!\}\}\!\} & (\beta_{\text{ns}})
\end{array}
$$

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

$$
\begin{array}{rcll}
n[\![k := e]\!] & \to_{c_{cs}} & \begin{cases} n & \text{if } n < k \\ n-1 & \text{if } n > k \end{cases} & (\text{Realign}_{\text{cs}}) \\
(\lambda e')[\![k := e]\!] & \to_{c_{cs}} & \lambda(e'[\![k+1 := \mathcal{W}_0^1(e)]\!]) & (\text{Abstr}_{\text{cs}}) \\
e[\![k := (\text{bind}_1\ (n\, e_1)\ \text{in}\ e_2)]\!] & \to_{c_{cs}} & \text{bind}_1\ (n\, e_1)\ \text{in}\ \mathcal{W}_0^1(e)[\![k+1 := e_2]\!] & (\text{Bind}^1_{\text{cs}}) \\
(\text{bind}_1\ (n\, e_1)\ \text{in}\ e_2)[\![k := e]\!] & \to_{c_{cs}} & \begin{cases} \text{bind}_1\ (n\, e_1[\![k := e]\!])\ \text{in}\ (e_2[\![k+1 := \mathcal{W}_0^1(e)]\!]) & \text{if } n < k \\ \text{bind}_1\ ((n-1)\, e_1[\![k := e]\!])\ \text{in}\ (e_2[\![k+1 := \mathcal{W}_0^1(e)]\!]) & \text{if } n > k \end{cases} & (\text{Bind}^2_{\text{cs}}) \\
k[\![k := e]\!] & \to_{c_{cs}} & e & (\text{Subst}_{\text{cs}}) \\
(\text{bind}_1\ (k\, e_1)\ \text{in}\ e_2)[\![k := m]\!] & \to_{c_{cs}} & \text{bind}_1\ (m\, e_1[\![k := m]\!])\ \text{in}\ (e_2[\![k+1 := m+1]\!]) & (\text{Reindex}_{\text{cs}}) \\
(\text{bind}_1\ (k\, e_1)\ \text{in}\ e_2)[\![k := \lambda e]\!] & \to_{c_{cs}} & e_2[\![k+1 := \mathcal{W}_0^1(\lambda e)]\!][\![1 := e[\![1 := e_1[\![k := \lambda e]\!]]\!]\!] & (\beta_{\text{cs}})
\end{array}
$$

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

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

Definition 8 $\mathcal{S} : \Lambda^{GK3i_{dB}+(\text{Cut}_{\text{ns}})+(\text{Weak})} \longrightarrow \Lambda s$ *is given as follows*

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

Definition 9 $\mathcal{T} : \Lambda^{\text{cs}} \longrightarrow \Lambda t$ *is given as follows*

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

Observe that $\mathcal{S}$ does not translate λ^{ns} terms of the form $\mathcal{R}_l^k(e)$. 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 considered 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) that appears as (Reloc) in λ^{ns}. Thus, the statements that follow will 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 ($\to$L) which allowed 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.

We now define restrictions and slight modifications of λs and λt to operate on the images of the terms of λ^{ns} and λ^{cs} under $\mathcal{S}$ and $\mathcal{T}$. Let $\Lambda s^- = \mathcal{S}(\Lambda^{\text{ns}})$ and $\Lambda t^- = \mathcal{T}(\Lambda^{\text{cs}})$. 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 the shortest rewrite sequence such that $a' \in \Lambda s^-$. Define $b \to_{\lambda t^-} b'$ similarly. The restricted cal-

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

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

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

λs

$$a \in \Lambda s \quad ::= \quad n \mid a\, a \mid \lambda a \mid a\sigma^i a \mid \varphi_k^i a \qquad \text{where } i, n \geq 1, k \geq 0$$

$$\begin{array}{rcl}
(\lambda a_1)\, a_2 & \to_{\lambda s} & a_1\sigma^1 a_2 \\
(\lambda a_1)\sigma^i a_2 & \to_{\lambda s} & \lambda(a_1\sigma^{i+1} a_2) \\
(a_1\, a_2)\sigma^i a_3 & \to_{\lambda s} & (a_1\sigma^i a_3)\,(a_2\sigma^i a_3) \\
n\sigma^i a & \to_{\lambda s} & \begin{cases} n-1 & \text{if } n > i \\ \varphi_0^i a & \text{if } n = i \\ n & \text{if } n < i \end{cases} \\
\varphi_k^i n & \to_{\lambda s} & \begin{cases} n & \text{if } n \leq k \\ n+i-1 & \text{if } n > k \end{cases} \\
\varphi_k^i(\lambda a) & \to_{\lambda s} & \lambda(\varphi_{k+1}^i a) \\
\varphi_k^i(a_1\, a_2) & \to_{\lambda s} & (\varphi_k^i a_1)\,(\varphi_k^i a_2)
\end{array}$$

λt

$$b \in \Lambda t \quad ::= \quad n \mid b\, b \mid \lambda b \mid b\varsigma^i b \mid \theta_k b \qquad \text{where } i, n \geq 1, k \geq 0$$

$$\begin{array}{rcl}
(\lambda b_1)\, b_2 & \to_{\lambda t} & b_1\varsigma^1 b_2 \\
(\lambda b_1)\varsigma^i b_2 & \to_{\lambda t} & \lambda(b_1\varsigma^{i+1}(\theta_0 b_2)) \\
(b_1\, b_2)\varsigma^i b_3 & \to_{\lambda t} & (b_1\varsigma^i b_3)\,(b_2\varsigma^i b_3) \\
n\varsigma^i b & \to_{\lambda t} & \begin{cases} n-1 & \text{if } n > i \\ b & \text{if } n = i \\ n & \text{if } n < i \end{cases} \\
\theta_k n & \to_{\lambda t} & \begin{cases} n & \text{if } n \leq k \\ n+1 & \text{if } n > k \end{cases} \\
\theta_k(\lambda b) & \to_{\lambda t} & \lambda(\theta_{k+1} b) \\
\theta_k(b_1\, b_2) & \to_{\lambda t} & (\theta_k b_1)\,(\theta_k b_2)
\end{array}$$

Figure 10: The λs and λt calculi.

culi 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 rules of λs^- pertaining to the φ operator and let $\to_{\theta^-}$ similarly use only rules of λt^- pertaining to the θ operator.

Proposition 10 *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: Straightforward inspection. □

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

Proposition 11 *For* $e, e' \in \Lambda^{\mathrm{cs}}$ *we have*

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

Although the above propositions show the index-updating mechanisms correspond, the rules $(\mathrm{Bind}^1_{_})$ 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 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 12 *λs simulates $\lambda^{\mathrm{ns}-}$, i.e., for* $e, e' \in \Lambda^{GK3i_{dB}+(\mathrm{Cut_{ns}})+(\mathrm{Weak})}$*:*

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

Theorem 13 *λt simulates $\lambda^{\mathrm{cs}-}$, i.e., for* $e, e' \in \Lambda^{\mathrm{cs}}$*:*

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

Proofs are available in [51].

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. However, this leaves out the rewrite rule $(\mathrm{Bind}^1_{_})$.

Translating this rule into λt as suggested by $\mathcal{T}$ we obtain:

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

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 [33] for a discussion of some of these occurrences.) This suggests that we should consider adding the rule to λt to obtain a better correspondence. It remains to verify the properties of the modified λt.

Alternatively, we could map the bind-construct to λs and λt via a meta-level substitution. This would lead to "λs and λt as homomorphic images of cut propagation" thus showing that the Zucker-Pottinger epimorphism [54, 41] factors through calculi with explicit substitution [52].

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 (Reloc); $\lambda bxgc$

The two cut rules we have considered are also admissible when we consider contexts as multisets of formulae instead of as lists of formulae. We can, unfortunately, not present the formal results relating to these systems here due to space constraints. We would like to say the following, though: (i) by taking contexts as multisets of assumptions we naturally arrive at calculi of proof terms with variable names — the contexts are represented as sets of "assumption classes" identified by variable names, (ii) both cut-elimination calculi *à la* de Bruijn collapse to the same such calculus with variable names, which (iii) is formally related to Rose's $\lambda bxgc$ exactly as stated in Theorems 12 & 13. Furthermore, the garbage collection rule of $\lambda bxgc$ corresponds to the propagation of a cut on a newly weakened assumption, cf. [51] in the case of garbage collection rules for λs and λt.

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, but we have focused on 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. Also, we have also used known results on first-order TRS's [38] in a correct way to obtain strong cut elimination results.

In carrying this work out, some issues have arisen which we have left to future work. One 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. 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 yet another issue is further study of the use of the proof theory of L-systems to interrelate known calculi with explicit substitutions.

4.1 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, and the Logik Oberseminar run by Helmut Schwichtenberg at the LMU, Munich.

References

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

[2] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[3] Zine-El-Abidine Benaissa, Daniel Briaud, Pierre Lescanne, and Jocelyne Rouyer-Degli. $\lambda\nu$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, September 1996.

[4] Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, pages 75–90, Utrecht, The Netherlands, March 1993.

[5] Elias Tahhan Bittar. 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, October 1996.

[6] Roel Bloo. *Preservation of Termination for Explicit Substitution*. PhD thesis, Technical University of Eindhoven, 1997.

[7] Roel Bloo and Kristoffer Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*, November 1995.

[8] Val Breazu-Tannen, Delia Kesner, and Laurence Puel. A typed pattern calculus. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 262–274, 1993.

[9] N.G. de Bruijn. 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, 1978.

[10] N.G. de Bruijn. Lambda calculus notation with namefree formulas involving symbols that represent reference transforming mappings. *Indag. Math.*, 40(3):348–356, 1979.

[11] Serenella Cerrito and Delia Kesner. Pattern matching as cut elimination. In *Proc. 14th Ann. IEEE Symp. Logic in Computer Sci.*, July 1999.

[12] Alonzo Church. *The Calculi of Lambda Conversion*. Princeton University Press, 1941.

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

[14] Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Progress in Theoretical Computer Science. Birkhaüser, 1993.

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

[16] H. B. Curry and R. Feys. *Combinatory Logic*. North-Holland, Amsterdam, 1958.

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

[18] A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, 1988.

[19] Roy Dyckhoff and Luis Pinto. Permutability of proofs in intuitionistic sequent calculi. Draft, November 1996.

[20] Jean Gallier. Constructive logics, Part I: A tutorial on proof systems and typed λ-calculi. *Theoretical Computer Science*, 110:249–339, 1993. Elsevier Science Publishers.

[21] Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969.

[22] Bernhard Gramlich. 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.

[23] Bruno Guillome. The λs_e does not have PSN. Manuscript, November 1997.

[24] Thérèse Hardin, Luc Maranget, and Bruno Pagano. 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.

[25] H. Herbelin. 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, 1995.

[26] Hugo Herbelin. *Séquents qu'on calcule*. PhD thesis, Université Paris VII, January 1995.

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

[28] César Muñoz Hurtado. 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, 1996.

[29] César Muñoz Hurtado. Proof representation in type theory: State of the art. In *XXII Latin-American Conference of Informatics CLEI Panel*, 1996.

[30] Fairouz Kamareddine and Alejandro Ríos. A λ-calculus à la de Bruijn with explicit substitutions. In *Programming Languages, Implementations, Logics, and Programs*, volume 982, pages 45–62, 1995.

[31] Fairouz Kamareddine and Alejandro Ríos. 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, July 1997.

[32] Fairouz Kamareddine and Alejandro Ríos. Bridging de Bruijn indices and variable names in explicit substitutions calculi. Submitted, February 1998.

[33] Assaf J. Kfoury and J. B. Wells. 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., 1995.

[34] Stephen C. Kleene. *Introduction to Metamathematics*. North-Holland Publishing Co., 1952.

[35] Lena Magnusson. *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, January 1995.

[36] John Maraist, Martin Odersky, David N. Turner, and Philip Wadler. Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. Technical Report 5-95, Fakultät für Informatik, University of Karlsruhe, 1995.

[37] Gregory Mints. Normal forms for sequent derivations. In Piergiorgio Odifreddi, editor, *Kreiseliana; about and around Georg Kreisel*, pages 469–492. Wellesley, Massachusetts, A.K. Peters, 1996.

[38] Michael O'Donnell. *Computing in Systems Described by Equations*. Number 58 in Lecture Notes in Computer Science. Springer-Verlag, 1977.

[39] Frank Pfenning. Logic programming in the LF logical framework. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

[40] Frank Pfenning. A proof of the church-rosser theorem and its representation in a logical framework. Technical Report CMU-CS-92-186, Carnegie Mellon University, Pittsburgh, Pennsylvania, September 1992. To appear in Journal of Automated Reasoning.

[41] Garrel Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977. North-Holland Publishing Company.

[42] Dag Prawitz. *Natural Deduction*. Almquist and Wiksell, Uppsala, 1965.

[43] Alejandro Ríos. *Contribution à l'étude des λ-calculs avec substitutions explicites*. PhD thesis, Université de Paris 7, 1993.

[44] E. Ritter, D. Pym, and L. A. Wallen. On the intuitionistic force of classical search. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, 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, Berlin, Heidelberg, New York, 1996. Springer Verlag.

[45] Kristoffer Rose. Explicit substitutions – tutorial & survey. Technical Report LS-96-13, BRICS, Aarhus University, Denmark, September 1996.

[46] Kristoffer Rose. *Operational Reduction Models for Functional Programming Languages*. PhD thesis, University of Copenhagen, 1996.

[47] Helmut Schwichtenberg. Termination of permutative conversions in intuitionistic Gentzen calculi. Available on the authors homepage: http://www.mathematik.uni-muenchen.de/~schwicht, April 1997.

[48] Sohame Selhab. *Logiques et Réécriture*. PhD thesis, Université Henri Poincaré — Nancy 1 (Spécialité Informatique), Nancy, France, May 1998.

[49] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996.

[50] René Vestergaard. A computational anomaly in the Troelstra-Schwichtenberg G3i(m) system. Available from the author's homepage: http://www.cee.hw.ac.uk/~jrvest/, October 1998.

[51] René Vestergaard. The cut rule and explicit substitutions (author's cut). Available from the author's homepage: http://www.cee.hw.ac.uk/~jrvest/. Also available as TR-1998-9, Dept. of CS, Uni. of Glasgow, April 1998.

[52] René Vestergaard. The correspondence between cut-propagation, β-reduction, and explicit substitutions. Forthcoming, March 1999.

[53] Philip Wadler. A Curry-Howard isomorphism for sequent calculus. Manuscript, December 1993.

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