An Algorithmic Interpretation of a Deep Inference System

Kai Brünnler and Richard McKinley

Institut für angewandte Mathematik und Informatik
Neubrückstr. 10, CH – 3012 Bern, Switzerland

Abstract. We set out to find something that corresponds to deep inference in the same way that the lambda-calculus corresponds to natural deduction. Starting from natural deduction for the conjunction-implication fragment of intuitionistic logic we design a corresponding deep inference system together with reduction rules on proofs that allow a fine-grained simulation of beta-reduction.

1 Introduction

The *Curry-Howard-Isomorphism* states that intuitionistic natural deduction derivations with the operation of detour-elimination behave exactly like lambda terms with beta reduction. For an introduction, see the book [4] by Girard, Lafont and Taylor. Since the lambda calculus expresses algorithms, the lambda calculus is thus an *algorithmic interpretation* of intuitionistic natural deduction. We want to find an algorithmic interpretation for *deep inference*, a formalism which has been introduced by Guglielmi [5]. So far, no deep inference system that we are aware of has an algorithmic interpretation. In fact, while they typically have cut elimination procedures, the cut elimination steps are not given in the form of simple reduction rules on proof terms.

The natural starting point for algorithmic interpretation of deep inference is of course a system for intuitionistic logic. There already exists such a system, by Tiu [11]. However, it focuses on locality of inference rules, and not on algorithmic interpretation. Its cut elimination proof works via translation to the sequent calculus. We design another deep inference system for intuitionistic logic, with the specific aim of staying as close to natural deduction as possible, because there the algorithmic interpretation is well-understood. We then give a definition of proof terms for this system. The general way of building proof terms for deep inference is already present in [1]. We equip these proof terms with reduction rules that allow us to simulate beta-reduction. We give translations from natural deduction to deep inference and back and prove a weak form of weak normalisation.

The principal way of composing our proof terms is not function application, as in the lambda calculus, but is function composition, as in composition of arrows in a category. So it is a system of what should be called *categorical combinators*. In fact, it turns out that our proof terms are very similar to some categorical combinators that Curien designed in the eighties, in order to serve as a target for the compilation of functional programming languages [2]. A very accessible introduction to those combinators and how they led to the development of explicit substitution calculi, like the $\lambda\sigma$-calculus, can be found in Hardin [7].

The difference between our combinators and Curien's is in the presentation of the defining adjunctions of a cartesian closed category. In our presentation proof terms can be thought of graphically: they are built using vertical composition (the usual composition of morphisms) and horizontal composition (the connectives).

2 A Deep Inference System for Intuitionistic Logic

Formulas, denoted by A, B, C, D, are defined as follows

$$A ::= a \mid (A \wedge A) \mid (A \supset A) \quad ,$$

where a is a propositional variable. As usual, conjunction binds stronger than implication and is left-associative, implication is right-associative. A *formula context*, denoted by $C\{\ \}$, is a formula with exactly one occurrence of the special propositional variable $\{\ \}$, called *the hole* or *the empty context*. The formula $C\{A\}$ is obtained by replacing the hole in $C\{\ \}$ by A. As usual, a context is *positive* if the number of implications we pass from the left on the path from the hole to the root is even. A context is *negative* if that number is odd, and is *strictly positive* if that number is zero.

A deep inference rule is a term rewriting rule on formulas. A rule is written

$$\rho \frac{A}{B} \quad ,$$

where ρ is the name of the rule and A and B are formulas containing schematic variables. A is the *premise* and B is the *conclusion* of the rule. In term rewriting A would be the left-hand-side or the redex and B would be the right-hand-side or the contractum. A *system* is a set of rules. An *instance* of a formula containing schematic variables is obtained by replacing the schematic variables by formulas. An *instance* of an inference rule as above is

$$\rho \frac{C\{A'\}}{C\{B'\}} \quad ,$$

where $C\{\ \}$ is a context, the formula A' is an instance of A and B' is an instance of B. A deep inference derivation is a sequence of rule instances composed in the obvious way. In term rewriting terminology a derivation is just a reduction sequence from one formula to another one using the given inference rules as rewrite rules. Of course, this definition only applies when the context is positive, since applying a rule in a negative context is generally unsound. For a negative context $C\{\ \}$, an instance of ρ will have the form

$$\rho \frac{C\{B'\}}{C\{A'\}} \quad .$$

$$\mathsf{c}\,\frac{A}{A \wedge A} \qquad\qquad \mathsf{i}\,\frac{B}{A \supset (B \wedge A)}$$

$$\mathsf{w}_1\,\frac{A \wedge B}{A} \quad \mathsf{w}_2\,\frac{A \wedge B}{B} \qquad\qquad \mathsf{e}\,\frac{(A \supset B) \wedge A}{B}$$

Fig. 1. A deep inference system for intuitionistic logic

Before seeing examples of derivations, let us look at a specific system for the conjunction-implication fragment of intuitionistic logic: the system in Figure 1. Because we like to think of the pair $\mathsf{w}_1, \mathsf{w}_2$ as one rule, there are essentially four rules: $\mathsf{c}, \mathsf{w}, \mathsf{i}, \mathsf{e}$, or, respectively: *contraction*, *weakening*, *implication introduction* and *implication elimination*. We can think of contraction as conjunction introduction and of weakening as conjunction elimination. Implication elimination can also be called *evaluation*. Categorically, each introduction rule is the unit and each elimination rule the counit of an adjunction. The system is designed with one goal in mind: to stay as close as possible to natural deduction, the home ground for algorithmic interpretation of proofs.

Let us now look at two examples of derivations. Notice how the inference rules apply deeply inside a context, as opposed to, say, rules in natural deduction. Notice also how the derivation on the right contains a "detour", a single instance of w_1 would also do the job.

Example 1.

$$\mathsf{c}\,\cfrac{A \wedge B}{\mathsf{w}_2\,\cfrac{(A \wedge B) \wedge (A \wedge B)}{\mathsf{w}_1\,\cfrac{B \wedge (A \wedge B)}{B \wedge A}}} \qquad\qquad \mathsf{i}\,\cfrac{A \wedge B}{\mathsf{w}_1\,\cfrac{(B \supset (A \wedge B)) \wedge B}{\mathsf{e}\,\cfrac{(B \supset A) \wedge B}{A}}}$$

We now introduce *proof terms*, or just *terms*, to capture deep inference derivations. Proof terms are denoted by R, T, U, V and are defined as follows:

$$R ::= \mathsf{id} \mid \rho \mid (R \,.\, R) \mid (R \wedge R) \mid (R \supset R)$$

where id is *identity*, ρ is the name of an inference rule from Figure 1, $(R_1 \,.\, R_2)$ is *(sequential) composition* and $(R_1 \wedge R_2)$ and $(R_1 \supset R_2)$ are *conjunction* and *implication*. Both conjunction and implication are also referred to as *parallel composition*. Sequential composition binds stronger than parallel composition and is left-associative. Unnecessary parentheses may be dropped.

Some proof terms can be typed. The typing judgement $A \xrightarrow{R} B$ says that the term R can have the type $A \to B$, so R has premise A and conclusion B. In that case R is called *typeable* and the triple consisting of A, R, B is called a *typed*

$$A \xrightarrow{\mathsf{id}} A \qquad \frac{A \xrightarrow{R} B \quad B \xrightarrow{T} C}{A \xrightarrow{R.T} C}$$

$$\frac{A \xrightarrow{R} C \quad B \xrightarrow{T} D}{A \wedge B \xrightarrow{R \wedge T} C \wedge D} \qquad \frac{C \xrightarrow{R} A \quad B \xrightarrow{T} D}{A \supset B \xrightarrow{R \supset T} C \supset D}$$

Fig. 2. Typing rules for proof terms

term. Typing judgements are derived by the typing rules in Figure 2 relative to a given set of typing axioms. A typing axiom types an inference rule name: we have $A \xrightarrow{\rho} B$ where A and B are instances of the premise and the conclusion of ρ, respectively. The only set of inference rules (or: typing axioms) we consider here is the one in Figure 1.

Example 2. Consider the following two terms R and T, which correspond to the derivations in Example 1:

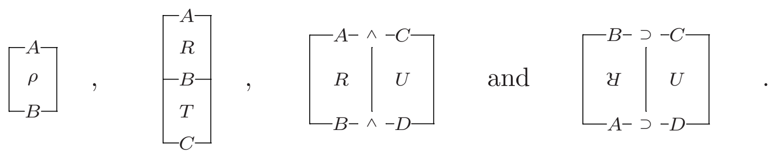
$$\mathsf{c} \,.\, (\mathsf{w}_2 \wedge \mathsf{id}) \,.\, (\mathsf{id} \wedge \mathsf{w}_1) \qquad \text{and} \qquad (\mathsf{i} \wedge \mathsf{id}) \,.\, ((\mathsf{id} \supset \mathsf{w}_1) \wedge \mathsf{id}) \,.\, \mathsf{e}$$

It is easy to see that they can be typed as $A \wedge B \xrightarrow{R} B \wedge A$ and $A \wedge B \xrightarrow{T} A$.

Clearly, there is canonical way of turning deep inference derivations into proof terms, as suggested by the examples above, and also a straightforward way of turning proof terms into deep inference derivations (that requires us to choose some order among parallel rewrites):

Proposition 1. *Given two formulas A, B and a system of inference rules, there is a derivation from A to B in that system iff there is a proof term R such that $A \xrightarrow{R} B$ can be derived from the typing axioms corresponding to the given system.*

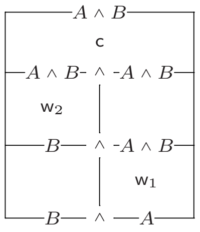
Having introduced these typing derivations, we replace them immediately by a more economical and suggestive notation, where we compose inference rules vertically and horizontally. Let ρ be an inference rule $A \xrightarrow{\rho} B$, and let $A \xrightarrow{R} B$, $B \xrightarrow{T} C$, $C \xrightarrow{U} D$. Then the typing derivations for ρ, $R \,.\, T$, $R \wedge U$ and $R \supset U$ are represented as the tiles



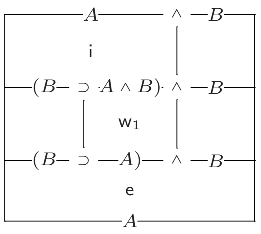
$$
\begin{array}{ll}
 & R\,.\,(T\,.\,U) \to (R\,.\,T)\,.\,U \\
 & R\,.\,\mathsf{id} \to R \leftarrow \mathsf{id}\,.\,R \\
(\mathsf{bur}) & \mathsf{id} \wedge \mathsf{id} \to \mathsf{id} \leftarrow \mathsf{id} \supset \mathsf{id} \\
 & (R \wedge T)\,.\,(U \wedge V) \to R\,.\,U \wedge T\,.\,V \\
 & (R \supset T)\,.\,(U \supset V) \to U\,.\,R \supset T\,.\,V \\
(\mathsf{nw}) & \begin{array}{l}(R \wedge T)\,.\,\mathsf{w}_1 \to \mathsf{w}_1\,.\,R \\ (R \wedge T)\,.\,\mathsf{w}_2 \to \mathsf{w}_2\,.\,T\end{array} \\
(\mathsf{nc}) & R\,.\,\mathsf{c} \to \mathsf{c}\,.\,(R \wedge R) \\
(\beta_\wedge) & \mathsf{c}\,.\,\mathsf{w}_1 \to \mathsf{id} \leftarrow \mathsf{c}\,.\,\mathsf{w}_2 \\
(\mathsf{ni}) & R\,.\,\mathsf{i} \to \mathsf{i}\,.\,(\mathsf{id} \supset (R \wedge \mathsf{id})) \\
(\beta_\supset) & (\mathsf{i}\,.\,(\mathsf{id} \supset R) \wedge T)\,.\,\mathsf{e} \to (\mathsf{id} \wedge T)\,.\,R
\end{array}
$$

Fig. 3. System beta

Example 3. Here are the tile representations of the derivations from the first example:



and



A $\wedge$ B
i
$(B \supset A \wedge B)$ $\wedge$ B
w_1
$(B \supset A)$ $\wedge$ B
e
A

.

2.1 Reduction

Some reduction rules are shown in Figure 3. They were chosen for the single purpose of allowing us to simulate β-reduction of the simply-typed lambda calculus, the best-understood algorithmic interpretation of a logical system. In particular, the rules were not chosen to make sense categorically: some naturality equations are missing, extensionality is missing and the rule for beta reduction is more general than one would expect.

The system is called System beta. It has two subsystems that we wish to identify: System bur, the first block of reduction rules, which is labeled with (bur), and System subst, which is obtained from System beta by removing the ($\beta_\supset$)-rule. System bur equationally specifies a category with two bifunctors. From a deep inference point of view, it has nothing to do with the inference rules involved, it just equates derivations which differ due to inessential, bureaucratic detail. System subst is named in accordance with Curien. Consider a β-reduction step in the lambda calculus. There are two things to do: first, remove the application operator and the lambda, and second, carry out the substitution. While the

$$\begin{array}{ll}
(\mathsf{bur}') & \begin{array}{l}(W\,.\,(R \wedge T))\,.\,(U \wedge V) \to W\,.\,(R\,.\,U \wedge T\,.\,V)\\ (W\,.\,(R \supset T))\,.\,(U \supset V) \to W\,.\,(U\,.\,R \supset T\,.\,V)\end{array}\\[1ex]
(\mathsf{nw}') & \begin{array}{l}(W\,.\,(R \wedge T))\,.\,\mathsf{w}_1 \to (W\,.\,\mathsf{w}_1)\,.\,R)\\ (W\,.\,(R \wedge T))\,.\,\mathsf{w}_2 \to (W\,.\,\mathsf{w}_2)\,.\,T)\end{array}\\[1ex]
(\beta_{\supset}{}') & \begin{array}{l}(\mathsf{i} \wedge R)\,.\,\mathsf{e} \to \mathsf{id} \wedge R\\ (W\,.\,(\mathsf{i} \wedge R))\,.\,\mathsf{e} \to W\,.\,(\mathsf{id} \wedge R)\\ (W\,.\,((\mathsf{i}\,.\,(\mathsf{id} \supset R)) \wedge T))\,.\,\mathsf{e} \to (W\,.\,(\mathsf{id} \wedge T))\,.\,R\end{array}
\end{array}$$

Fig. 4. The completion of system beta into system Beta

($\beta_{\supset}$)-rule allows us to do the first step, System subst allows us do the second step.

System beta is not locally confluent, its completion Beta is obtained by adding the rules in Figure 4. Morally, the right thing to do could be to work modulo bur, which would allow us to abandon these extra reduction rules. In this work we formally stay within the free theory. Nevertheless, we think of the terms as deep inference derivations, which are equal modulo associativity and, morally, should be equal modulo bur. System Bur is a completion of System bur and System Subst a completion of System subst, both are obtained by adding the corresponding rules from Figure 4.

For a given subsystem of System Beta we write $R \to T$ if R can be rewritten into T in one step by any rule in the given subsystem, so $\to$ is closed under context and irreflexive. We write $\to^n$ for the composition of $\to$ with itself n-times, and $\twoheadrightarrow$ for the reflexive-transitive closure of $\to$. If no subsystem is specified we mean System Beta itself.

Example 4. Our example terms R and T rewrite as follows:

$$\begin{array}{ll}
 & \mathsf{c}\,.\,(\mathsf{w}_2 \wedge \mathsf{id})\,.\,(\mathsf{id} \wedge \mathsf{w}_1)\\
\to & \mathsf{c}\,.\,(\mathsf{w}_2\,.\,\mathsf{id}) \wedge (\mathsf{id}\,.\,\mathsf{w}_1)\\
\to^2 & \mathsf{c}\,.\,(\mathsf{w}_2 \wedge \mathsf{w}_1)
\end{array}
\quad\text{and}\quad
\begin{array}{ll}
 & (\mathsf{i} \wedge \mathsf{id})\,.\,((\mathsf{id} \supset \mathsf{w}_1) \wedge \mathsf{id})\,.\,\mathsf{e}\\
\to & (\mathsf{i}\,.\,(\mathsf{id} \supset \mathsf{w}_1) \wedge (\mathsf{id}\,.\,\mathsf{id}))\,.\,\mathsf{e}\\
\to & (\mathsf{id} \wedge \mathsf{id}\,.\,\mathsf{id})\,.\,\mathsf{w}_1\\
\to^3 & \mathsf{w}_1
\end{array}$$

Figure 5 shows for most of the reductions in system beta that they preserve typing. For the remaining rules this is easy to check, so we have the following proposition.

Proposition 2 (reduction preserves typing). *Let R and T be proof terms with $R \twoheadrightarrow T$. If $A \xrightarrow{R} B$ then $A \xrightarrow{T} B$.*

By checking critical pairs we get local confluence, strong normalisation for Bur can be obtained by a simple polynomial interpretation, so we have the following proposition.

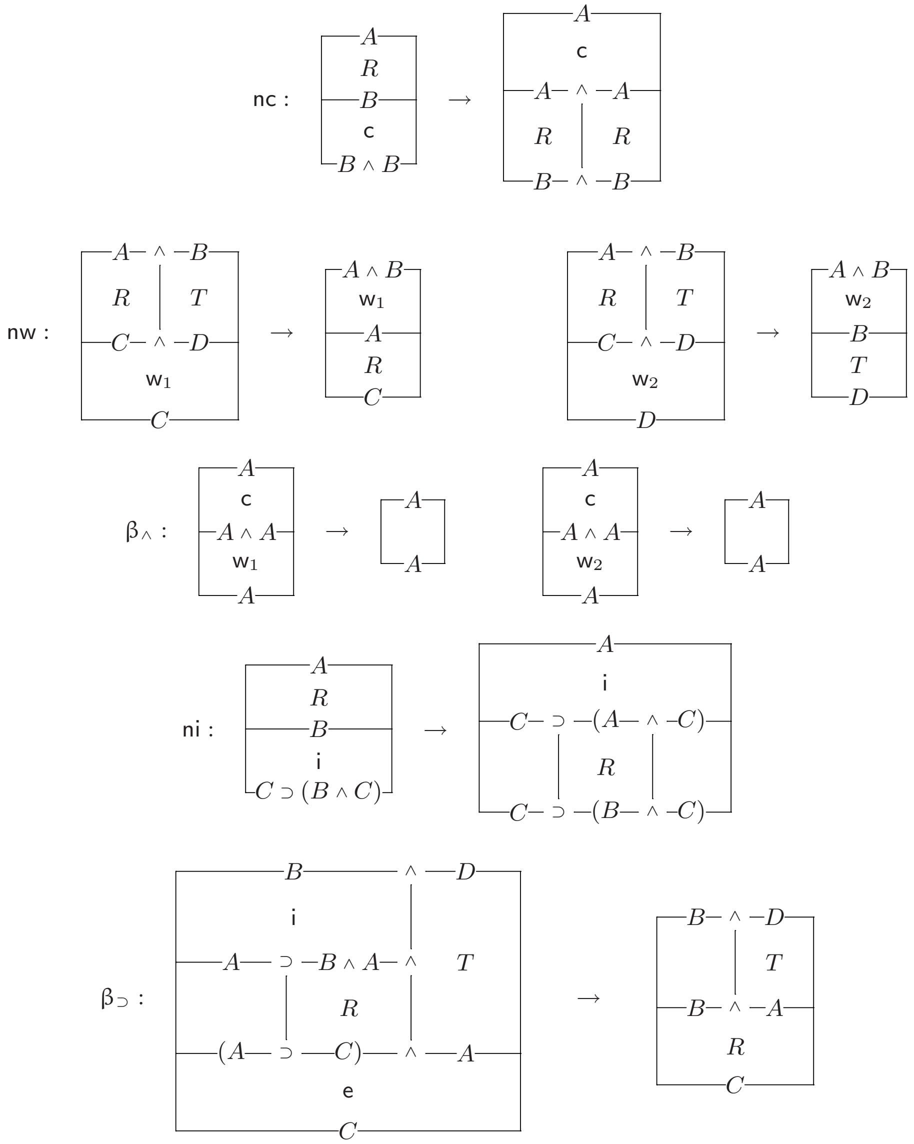


Fig. 5. Reduction rules with typing

Proposition 3.
(i) Systems Bur*,* Subst *and* Beta *are locally confluent.*
(ii) System Bur *is confluent and strongly normalising.*

Remark 1. System Subst, and thus also System Beta, is not strongly normalising. We have the following cycle:

$$\begin{array}{rcl} \mathsf{c}\,.\,\mathsf{w}_1\,.\,\mathsf{c}\,.\,\mathsf{w}_1 & \to & \mathsf{c}\,.\,(\mathsf{c}\,.\,\mathsf{w}_1 \wedge \mathsf{c}\,.\,\mathsf{w}_1)\,.\,\mathsf{w}_1 \\ & \to & \mathsf{c}\,.\,\mathsf{w}_1\,.\,(\mathsf{c}\,.\,\mathsf{w}_1) \\ & \to & \mathsf{c}\,.\,\mathsf{w}_1\,.\,\mathsf{c}\,.\,\mathsf{w}_1 \qquad . \end{array}$$

The situation is different from Curien's system, where the subsystem for carrying out substitutions is strongly normalising. The confluence proofs for Curien's systems, that we know of, use strong normalisation of the subsystem which carries out substitutions, so they do not seem to directly apply in our setting. For the moment we do not know whether our system is confluent. In any case we do not see the failure of strong normalisation as a major defect. The problem is now to find a natural and liberal strategy which ensures termination.

3 The Relation with Natural Deduction

There is an obvious inductive translation of a natural deduction derivation into a deep inference derivation. It yields a deep inference derivation with the same conclusion as the natural deduction derivation and which has as its premise the conjunction of all premises of the natural deduction derivation. Since our inference rules are all sound and since a suitable replacement theorem holds for intuitionistic logic, we also know that we can also embed deep inference into natural deduction. So translations in both directions exist. However, they only work on derivations, not on the underlying untyped terms. What we would like to have in both directions is a translation of untyped terms which has the property of preserving typing. However, the obvious inductive translation of derivations is not even well-defined on their underlying untyped terms. Consider a standard sequent-style natural deduction system with additive context treatment and without structural rules. The two axiom instances $A \vdash A$ and $B, A \vdash A$ are different derivations, that should be translated into $A \xrightarrow{\mathsf{id}} A$ and $B \wedge A \xrightarrow{\mathsf{w}_2} A$, respectively. However, both axiom instances have the same underlying pure term, namely just a variable. Clearly, taking the underlying pure lambda term loses too much information of the original derivation. To keep that information we very slightly extend the syntax of lambda terms. We mark a variable if it corresponds to an axiom of the first kind and we will not mark it if it corresponds to an axiom of the second kind. The marked variables behave as usual except that they are not allowed to be bound.

We consider λ-terms with de Bruijn indices, introduced in [3]. They are defined as follows, where $n \geq 1$:

$$M ::= n \mid n^{\cdot} \mid (\lambda M) \mid (M\,M) \mid (\pi_1 M) \mid (\pi_2 M) \mid \langle M, M \rangle \qquad ,$$

$$\Gamma, A, \Delta \vdash i^{(\cdot)} : A \quad \text{where } i = |A, \Delta| \text{ and } i \text{ is marked iff } |\Gamma| = 0$$

$$\wedge_I \frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \wedge B} \qquad \supset_I \frac{\Gamma, A \vdash M : B}{\Gamma \vdash \lambda M : A \supset B}$$

$$\wedge_E \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \pi_1 M : A} \quad \wedge_E \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \pi_2 M : B} \qquad \supset_E \frac{\Gamma \vdash M : A \supset B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

Fig. 6. Typing rules for the name-free λ-calculus

and where in a given term an occurrence of $n^{\cdot}$, a *marked* index, is in the scope of at most $n-1$ λ's. The reduction rules for β-reduction together with substitution $M[n \leftarrow N]$ and lifting t^n_i are defined as follows:

$$\begin{array}{ll}
 & m[n \leftarrow N] \qquad = \begin{cases} m-1 & m > n \\ t_0^{n-1}(N) & m = n \\ m & m < n \end{cases} \\
\pi_1\langle M, N\rangle \to M & \\
\pi_2\langle M, N\rangle \to N & m^{\cdot}[n \leftarrow N] \qquad = (m-1)^{\cdot} \\
(\lambda M)N \to M[1 \leftarrow N] & (M_1\, M_2)[n \leftarrow N] = (M_1[n \leftarrow N]\, M_2[n \leftarrow N]) \\
 & (\lambda M)[n \leftarrow N] \qquad = (\lambda M[n+1 \leftarrow N])
\end{array}$$

$$\begin{array}{ll}
t^n_i(m) \ = \begin{cases} m+n & m > i \\ m & m \leq i \end{cases} & t^n_i(M\, N) = (t^n_i(M)\, t^n_i(N)) \\
 & t^n_i(\lambda M) \ \ = (\lambda\, t^n_{i+1}(M)) \\
t^n_i(m^{\cdot}) = (m+n)^{\cdot} &
\end{array}$$

A *typing context*, denoted by Γ or Δ, is a finite sequence of formulas. For typing context Γ its length is denoted by $|\Gamma|$ and the conjunction of all its formulas, in the given order and associated to the left, is denoted by $\wedge\Gamma$. Our typing system for lambda terms is given in Figure 6. Notice that it is impossible to type any term in an empty context, because that would require us to abstract over a marked index, which is not allowed. Let $\top$ denote $a \supset a$, for some atom a. Notice that whenever $\Gamma \vdash M : A$ and M' is obtained from M by removing all markings, then $\top, \Gamma \vdash M' : A$

Natural deduction to deep inference. We define a function $\underline{\quad}_{\mathsf{D}}$ from λ-terms to deep inference proof terms. We write R^n for $n > 0$ to denote R sequentially composed with itself n times. An expression $R^0 \mathbin{.} T$ or $T \mathbin{.} R^0$ denotes just T.

$$\begin{array}{ll}
\underline{m^{\cdot}}_{\mathsf{D}} = \begin{cases} \mathsf{id} & m = 1 \\ \mathsf{w}_1^{m-1} & m > 1 \end{cases} & \begin{array}{ll} \underline{\lambda M}_{\mathsf{D}} & = \mathsf{i} \mathbin{.} (\mathsf{id} \supset \underline{M}_{\mathsf{D}}) \\ \underline{MN}_{\mathsf{D}} & = \mathsf{c} \mathbin{.} (\underline{M}_{\mathsf{D}} \wedge \underline{N}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \\ \underline{\pi_n M}_{\mathsf{D}} & = \underline{M}_{\mathsf{D}} \mathbin{.} \mathsf{w}_n \end{array} \\
\underline{m}_{\mathsf{D}} \ = \mathsf{w}_1^{m-1} \mathbin{.} \mathsf{w}_2 & \ \underline{\langle M, N\rangle}_{\mathsf{D}} = \mathsf{c} \mathbin{.} (\underline{M}_{\mathsf{D}} \wedge \underline{N}_{\mathsf{D}})
\end{array}$$

It is straightforward to check that the embedding preserves typing, so we omit the proof, even though it is very instructive:

Theorem 1 (the embedding preserves typing). *If* $\Gamma \vdash M : A$ *then* $\wedge\Gamma \xrightarrow{\underline{M}_{\mathsf{D}}} A$.

We now come to the main theorem: System Beta can simulate β-reduction. The proof is of course similar to the proof of a similar result for Curien's combinators in [2]. We write $\mathsf{id}_n(R)$ for $(\ldots(R \wedge \underbrace{\mathsf{id}) \ldots \wedge \mathsf{id}}_{n \text{ times}})$.

Theorem 2 (the embedding preserves reduction).
(i) If $M \twoheadrightarrow_\beta N$ *then* $\underline{M}_{\mathsf{D}} \twoheadrightarrow \underline{N}_{\mathsf{D}}$.
(ii) $\mathsf{id}_{n-1}(\mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}})) \mathbin{.} \underline{M}_{\mathsf{D}} \twoheadrightarrow \underline{M[n \leftarrow N]}_{\mathsf{D}}$
(iii) $\mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} \underline{M}_{\mathsf{D}} \twoheadrightarrow \underline{t_i^n(M)}_{\mathsf{D}}$

Proof. The first claim follows from the following diagram, which relies on (ii). A similar diagram works for the projection–pairing reduction.

$$\begin{tikzcd}
(\lambda M)N \arrow[r, "\underline{\ \ }_{\mathsf{D}}"] \arrow[dd, "\beta"'] & \mathsf{c} \mathbin{.} (\mathsf{i} \mathbin{.} (\mathsf{id} \supset \underline{M}_{\mathsf{D}}) \wedge \underline{N}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \arrow[d, "\beta_{\supset}{}'"] \\
 & \mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}}) \mathbin{.} \underline{M}_{\mathsf{D}} \arrow[d, "(ii)"] \\
M[1 \leftarrow N] \arrow[r, "\underline{\ \ }_{\mathsf{D}}"] & \underline{M[1 \leftarrow N]}_{\mathsf{D}}
\end{tikzcd}$$

We now prove (ii), by induction on M. We see the cases for an index, an application, and an abstraction. The cases for a marked index, for pairing and for projection are straightforward.

$$\mathsf{id}_{n-1}(\mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}})) \mathbin{.} \underline{m}_{\mathsf{D}} \quad = \quad \mathsf{id}_{n-1}(\mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}})) \mathbin{.} \mathsf{w}_1^{m-1} \mathbin{.} \mathsf{w}_2$$
$$\twoheadrightarrow \begin{cases}
\begin{array}{l} \mathsf{w}_1^{n-1} \mathbin{.} \mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}}) \mathbin{.} \mathsf{w}_1^{m-n} \mathbin{.} \mathsf{w}_2 \\ \twoheadrightarrow \mathsf{w}_1^{m-2} \mathbin{.} \mathsf{w}_2 = \underline{m-1}_{\mathsf{D}} = \underline{m[n \leftarrow N]}_{\mathsf{D}} \end{array} & m > n \\[2ex]
\begin{array}{l} \mathsf{w}_1^{n-1} \mathbin{.} \mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}}) \mathbin{.} \mathsf{w}_2 \\ \twoheadrightarrow \mathsf{w}_1^{n-1} \mathbin{.} \underline{N}_{\mathsf{D}} \overset{(iii)}{\twoheadrightarrow} \underline{t_0^{n-1}(N)}_{\mathsf{D}} = \underline{m[n \leftarrow N]}_{\mathsf{D}} \end{array} & m = n \\[2ex]
\begin{array}{l} \mathsf{w}_1^{m-1} \mathbin{.} \mathsf{id}_{n-m}(\mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}})) \mathbin{.} \mathsf{w}_2 \\ \twoheadrightarrow \mathsf{w}_1^{m-1} \mathbin{.} \mathsf{w}_2 = \underline{m}_{\mathsf{D}} = \underline{m[n \leftarrow N]}_{\mathsf{D}} \end{array} & m < n
\end{cases}$$

$$\begin{aligned}
\mathsf{id}_{n-1}(\mathsf{c} \mathbin{.} (\mathsf{id} \wedge \underline{N}_{\mathsf{D}})) \mathbin{.} \underline{M_1 M_2}_{\mathsf{D}} &= \mathsf{id}_{n-1}(\ldots) \mathbin{.} \mathsf{c} \mathbin{.} (\underline{M_1}_{\mathsf{D}} \wedge \underline{M_2}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \\
&\twoheadrightarrow \mathsf{c} \mathbin{.} (\mathsf{id}_{n-1}(\ldots) \mathbin{.} \underline{M_1}_{\mathsf{D}} \wedge \mathsf{id}_{n-1}(\ldots) \mathbin{.} \underline{M_2}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \\
&\twoheadrightarrow \mathsf{c} \mathbin{.} (\underline{M_1[n \leftarrow N]}_{\mathsf{D}} \wedge \underline{M_2[n \leftarrow N]}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \\
&= \underline{M_1[n \leftarrow N] M_2[n \leftarrow N]}_{\mathsf{D}} = \underline{(M_1 M_2)[n \leftarrow N]}_{\mathsf{D}}
\end{aligned}$$

$$\begin{aligned}
\mathsf{id}_{n-1}(\ldots) \mathbin{.} \underline{\lambda M}_{\mathsf{D}} &= \mathsf{id}_{n-1}(\ldots) \mathbin{.} \mathsf{i} \mathbin{.} (\mathsf{id} \supset \underline{M}_{\mathsf{D}}) \\
&\twoheadrightarrow \mathsf{i} \mathbin{.} (\mathsf{id} \supset \mathsf{id}_n(\ldots)) \mathbin{.} (\mathsf{id} \supset \underline{M}_{\mathsf{D}}) \\
&\twoheadrightarrow \mathsf{i} \mathbin{.} (\mathsf{id} \supset \mathsf{id}_n(\ldots) \mathbin{.} \underline{M}_{\mathsf{D}}) \\
&\twoheadrightarrow \mathsf{i} \mathbin{.} (\mathsf{id} \supset \underline{M[n+1 \leftarrow N}_{\mathsf{D}}) \\
&= \underline{\lambda(M[n+1 \leftarrow N])}_{\mathsf{D}} = \underline{(\lambda M)[n \leftarrow N]}_{\mathsf{D}}
\end{aligned}$$

We now prove (iii), again by induction on M. We again see the cases for an index, an application and an abstraction, the cases for a marked index, a pairing and a projection are straightforward.

$$\mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} \underline{m}_{\mathsf{D}} = (\ldots(\mathsf{w}_1^n \wedge \underbrace{\mathsf{id}) \ldots \wedge \mathsf{id}}_{i \text{ times}}) \mathbin{.} \mathsf{w}_1^{m-1} \mathbin{.} \mathsf{w}_2 \twoheadrightarrow$$
$$\begin{cases} \mathsf{w}_1^{m-1} \mathbin{.} \mathsf{w}_2 = \underline{m}_{\mathsf{D}} = \underline{t_i^n(m)}_{\mathsf{D}} & m \leq i \\ \mathsf{w}_1^{m-1+n} \mathbin{.} \mathsf{w}_2 = \underline{m+n}_{\mathsf{D}} = \underline{t_i^n(m)}_{\mathsf{D}} & m > i \end{cases} .$$

$$\begin{aligned}
\mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} \underline{M_1 M_2}_{\mathsf{D}} &= \mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} (\mathsf{c} \mathbin{.} (\underline{M_1}_{\mathsf{D}} \wedge \underline{M_2}_{\mathsf{D}}) \mathbin{.} \mathsf{e}) \\
&\twoheadrightarrow \mathsf{c} \mathbin{.} (\mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} \underline{M_1}_{\mathsf{D}} \wedge \mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} \underline{M_2}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \\
&\twoheadrightarrow \mathsf{c} \mathbin{.} (\underline{t_i^n M_1}_{\mathsf{D}} \wedge \underline{t_i^n M_2}_{\mathsf{D}}) \mathbin{.} \mathsf{e} \\
&= \underline{t_i^n M_1}_{\mathsf{D}} \underline{t_i^n M_2}_{\mathsf{D}} = \underline{t_i^n(M_1 M_2)}_{\mathsf{D}} \quad .
\end{aligned}$$

$$\begin{aligned}
\mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} \underline{\lambda N}_{\mathsf{D}} &= \mathsf{id}_i(\mathsf{w}_1^n) \mathbin{.} (\mathsf{i} \mathbin{.} (\mathsf{id} \supset \underline{N}_{\mathsf{D}})) \\
&\twoheadrightarrow \mathsf{i} \mathbin{.} (\mathsf{id} \supset \mathsf{id}_{i+1}(\mathsf{w}_1^n) \mathbin{.} \underline{N}_{\mathsf{D}}) \\
&\twoheadrightarrow \mathsf{i} \mathbin{.} (\mathsf{id} \supset \underline{t_{i+1}^n N}_{\mathsf{D}}) = \underline{\lambda t_{i+1}^n N}_{\mathsf{D}} = \underline{t_i^n(\lambda N)}_{\mathsf{D}} \quad .
\end{aligned}$$

□

Definition 1. *Let a proof term T be* essentially in normal form *if each reduction sequence in system* Beta *starting from T only contains instances of the rules* $R \mathbin{.} \mathsf{id} \to R$*,* $\mathsf{id} \mathbin{.} R \to R$*,* $\mathsf{id} \wedge \mathsf{id} \to \mathsf{id}$ *and* $\mathsf{id} \supset \mathsf{id} \to \mathsf{id}$.

Proposition 4 (the embedding essentially preserves normal form). *If M is in normal form then $\underline{M}_{\mathsf{D}}$ is essentially in normal form.*

Proof. By checking the reduction rules we first observe that, when given two terms R, T which are essentially in normal form, then also the terms $R \supset T$, $R \wedge T$, $\mathsf{i} \,.\, (R \supset T)$ and $\mathsf{c} \,.\, (R \wedge T)$ are essentially in normal form. We prove our proposition by induction on M. Translations of indices are clearly in normal form, and our observation takes care of abstractions and pairings, so we are left with applications and projections. Let M be an application $M_1 M_2$. Then M_1 can not be an abstraction, so it has to be either an index, a projection, a pairing or an application. Say it is an application $N_1 N_2$. Then $\underline{M}_{\mathsf{D}} = \mathsf{c} \,.\, (\underline{M_1}_{\mathsf{D}} \wedge \underline{M_2}_{\mathsf{D}}) \,.\, \mathsf{e}$ with $\underline{M_1}_{\mathsf{D}} = \mathsf{c} \,.\, (\underline{N_1}_{\mathsf{D}} \wedge \underline{N_2}_{\mathsf{D}}) \,.\, \mathsf{e}$. By induction hypothesis $\underline{M_1}_{\mathsf{D}}$ is essentially in normal form, so can only reduce to terms of the form $\mathsf{c} \,.\, U \,.\, \mathsf{e}$ or $\mathsf{c} \,.\, \mathsf{e}$. But then all reductions possible in a reduction sequence starting from $\underline{M}_{\mathsf{D}}$ are those that are either in a reduction sequence starting from $\underline{M_1}_{\mathsf{D}}$ or $\underline{M_2}_{\mathsf{D}}$ and thus $\underline{M}_{\mathsf{D}}$ is essentially in normal form. The other cases are similar.

Deep inference to natural deduction. We define a function $\underline{\quad}_{\mathsf{N}}$ from deep inference proof terms to natural deduction proof terms, i.e. lambda terms. We give a definition using named lambda terms. For a given deep inference proof term the function yields a lambda term with exactly one free variable, named x. The translation from that into a name-free lambda term is as usual, except that exactly those indices that come from occurrences of x are marked.

$$\begin{array}{llll}
\underline{\mathsf{id}}_{\mathsf{N}} = x & & & \\
\underline{\mathsf{w}_n}_{\mathsf{N}} = \pi_n x & \underline{R \,.\, T}_{\mathsf{N}} & = \underline{T}_{\mathsf{N}}[x \leftarrow \underline{R}_{\mathsf{N}}] & \\
\underline{\mathsf{c}}_{\mathsf{N}} = \langle x, x\rangle & \underline{R \wedge T}_{\mathsf{N}} & = \langle \underline{R}_{\mathsf{N}}[x \leftarrow \pi_1 x], \underline{T}_{\mathsf{N}}[x \leftarrow \pi_2 x]\rangle & \\
\underline{\mathsf{i}}_{\mathsf{N}} = \lambda y.\langle x, y\rangle & \underline{R \supset T}_{\mathsf{N}} & = \lambda y.\underline{T}_{\mathsf{N}}[x \leftarrow (x\underline{R}_{\mathsf{N}}[x \leftarrow y])] & \text{(fresh } y\text{)} \\
\underline{\mathsf{e}}_{\mathsf{N}} = \pi_1 x \pi_2 x & & &
\end{array}$$

Also the embedding in this direction preserves typing. Again it is straightforward to check and we have to omit the proof for space reasons.

Theorem 3 (the embedding preserves typing). *If $A \xrightarrow{R} B$ then $A \vdash \underline{R}_{\mathsf{N}} : B$.*

Remark 2. The embedding does not preserve normal form. Consider the normal form $\mathsf{i} \,.\, (\mathsf{id} \supset \mathsf{w}_1)$ which is mapped to $\lambda z.\pi_1(\lambda y\langle x, y\rangle z)$ which is not in normal form. The embedding does not preserve reduction. Consider the term $\mathsf{w}_1 \,.\, \mathsf{i}$ which reduces to $\mathsf{i} \,.\, (\mathsf{id} \supset (\mathsf{w}_1 \wedge \mathsf{id}))$ but $\underline{\mathsf{w}_1 \,.\, \mathsf{i}}_{\mathsf{N}} = \lambda y.\langle \pi_1 x, y\rangle$ is normal. The embedding does not preserve β-convertibility. Consider $\underline{\mathsf{id} \wedge \mathsf{id}}_{\mathsf{N}} = \langle \pi_1 x, \pi_2 x\rangle$ and $\underline{\mathsf{id}}_{\mathsf{N}} = x$. However, if $R \twoheadrightarrow T$ then $\underline{R}_{\mathsf{N}}$ and $\underline{T}_{\mathsf{N}}$ are convertible in lambda calculus with extensionality and surjective pairing.

Now we can use the two embeddings and their preservation of types to show the following theorem:

Theorem 4. *For each typed term there is a term in normal form with the same type.*

Proof. If a term R is typeable $A \xrightarrow{R} B$ then by Theorem 3 $A \vdash \underline{R}_{\mathsf{N}} : B$ and by weak normalisation and subject reduction of the typed lambda calculus $\underline{R}_{\mathsf{N}}$ has a normal form M with $A \vdash M : B$. Now $\underline{M}_{\mathsf{D}}$ is essentially normal by Proposition 4 and typeable $A \xrightarrow{\underline{M}_{\mathsf{D}}} B$ by Theorem 1. Reducing $\underline{M}_{\mathsf{D}}$ in the canonical system formed by the four rules which collapse and remove identity we obtain a term T in normal form with $A \xrightarrow{T} B$.

Of course, while this is weak normalisation for *some* system, it is not weak normalisation for System Beta, since System Beta cannot simulate the effect of translating into the lambda calculus and back. So the problem now is to prove weak normalisation either directly or maybe by using a different embedding into lambda terms.

4 Discussion

Curien's combinators. We first explain the difference between our combinators and the categorical combinators of Curien. Both systems are orientations of a subset of a defining set of equations of a cartesian closed category, see Lambek and Scott [8]. A cartesian closed category (without terminal object) is a category with binary products and exponentials, which correspond to conjunction and implication, respectively. Both of these structures may be defined using an *adjunction*. As explained in MacLane [9], an adjunction may be specified in many different ways, leading to different presentations of a cartesian closed category. Curien's system corresponds to the specification based on one functor, a mapping of arrows, and the counit. Our system corresponds to the more symmetric specification of an adjunction based on two functors and unit and counit. Curien's definition of a cartesian closed category is the one typically found in textbooks, such as [8].

The primitives for both systems are summarized in Figure 7. Each of the two rows represents an adjunction, and each column a collection of primitives. Our system takes the functors $\wedge$ and $\supset$ as primitive, while Curien takes the mappings $\langle -, - \rangle$ and Λ. For each adjunction we take both unit and counit, while Curien treats only the counit as a primitive. Of course, both systems include Δ implicitly. The terms of Curien's system are thus built from $\mathsf{id}, \mathsf{w}_1, \mathsf{w}_2, \mathsf{e}$ using arrow composition, and two constructors $\langle -, - \rangle$ and Λ. By the equivalence of the different presentations of an adjunction, we could define Curien's constructors as

$$\begin{aligned} \Lambda(R) \quad &= \quad A \xrightarrow{\mathsf{i}} B \supset (A \wedge B) \xrightarrow{\mathsf{id} \supset R} B \supset C \qquad \text{and} \\ \langle R, T \rangle \quad &= \quad A \xrightarrow{\mathsf{c}} A \wedge A \xrightarrow{R \wedge T} B \wedge C \qquad . \end{aligned}$$

However, since we only have a subset of the defining equations of the adjunctions this will not lead to an embedding of Curien's system (not even the one called CCL_β since it contains a bit of surjective pairing). In particular Beta lacks naturality for the counit e, a part of naturality for the unit i as in $\mathsf{i} \,.\, (\mathsf{id} \supset (R \wedge T)) = \mathsf{i} \,.\, (T \supset (R \wedge \mathsf{id}))$, and the equations $\mathsf{c}.(\mathsf{w}_1 \wedge \mathsf{w}_2) = \mathsf{id}$ and

left adjoint functor right adjoint functor	Hom-bijection	unit counit
$\varDelta : f \mapsto (f,f)$ $\wedge : (f,g) \mapsto f \wedge g$	$\dfrac{(A,A) \xrightarrow{(f,g)} (B,C)}{A \xrightarrow{\langle f,g\rangle} B \wedge C}$	$\mathsf{c} : A \to A \wedge A$ $(\mathsf{w}_1, \mathsf{w}_2) : (A \wedge B, A \wedge B) \to (A, B)$
$- \wedge A : f \mapsto f \wedge \mathsf{id}$ $A \supset - : f \mapsto \mathsf{id} \supset f$	$\dfrac{B \wedge A \xrightarrow{f} C}{B \xrightarrow{\Lambda(f)} A \supset C}$	$\mathsf{i} : B \to A \supset (B \wedge A)$ $\mathsf{e} : (A \supset B) \wedge A \to B$

Fig. 7. Primitives of both systems

$\mathsf{i}.(\mathsf{id} \supset \mathsf{e}) = \mathsf{id}$. Orienting and adding these equations would allow simulation of a lambda calculus with surjective pairing and extensionality and give equational equivalence with Curien's system $\mathsf{CCL}_{\beta\eta\mathsf{SP}}$.

Future work. Adding extensionality is an obvious route for further research. Adding full naturality for i and e is another interesting route: note that our embedding of the lambda calculus stays in the *strictly positive fragment* of proof terms, the fragment where the left-hand side of an implication is always the term id. System Beta never leaves the strictly positive fragment. Full naturality for i and e would allow us to leave that fragment. This gives us a lot of freedom. In explicit substitution calculi when a beta-redex is reduced a substitution arises from it, and then this substitution can be carried out indepently from *other* beta-redexes. In a system with full naturality a substitution could be carried out indepently even from *the very beta-redex that it arises from.* It would also be interesting to use the functor $\wedge$ to more economically embed lambda terms than what is possible with Curien's combinators: by distributing to each subterm not the entire environment, but only those variables of the environment that actually occur. This would correspond to embedding a natural deduction system with multiplicative context treatment, and it would require some kind of exchange combinator, which shuffles around the channels corresponding to the variables. It would also be interesting to study *flow graphs* in the sense of [6] for our proof terms. It is easy to define them, and their acyclicity seems to be the key to a proof of normalisation. Our proof terms also give rise to interaction-style combinators, similar in spirit to those used for optimal reduction, but different because based on function composition instead of function application. An extension with more connectives would be interesting. Notice that the rules to add for disjunction are in perfect duality with those for conjunction:

$$\vee_I \frac{A}{A \vee B} \qquad \vee_I \frac{B}{A \vee B} \qquad\qquad \vee_E \frac{A \vee A}{A} \qquad .$$

We enjoy this improvement over the situation in natural deduction, where we have essentially the same introduction rules, but the following elimination rule:

$$\dfrac{A \vee B \qquad \begin{matrix}[A]\\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ C\end{matrix}}{C} \quad .$$

And finally, classical logic would be interesting. A sensible place would be to start with a system which can simulate reduction in the $\lambda\mu$-calculus [10].

References

1. Kai Brünnler and Stéphane Lengrand. On two forms of bureaucracy in derivations. In Paola Bruscoli, François Lamarche, and Charles Stewart, editors, *Structures and Deduction*, pages 69–80. Technische Universität Dresden, 2005.
2. Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Research Notes in Theoretical Computer Science. Birkhäuser, 2nd edition, 1993.
3. N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, 1972.
4. Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
5. Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1–64, 2007.
6. Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008. `http://arxiv.org/pdf/0709.1205`.
7. Therese Hardin. From categorical combinators to $\lambda\sigma$-calculi, a quest for confluence. Technical report, INRIA Rocquencourt, 1992. Available from `http://hal.inria.fr/inria-00077017/`.
8. J. Lambek and P. J. Scott. *Introduction to higher order categorical logic*. Cambridge University Press, New York, NY, USA, 1986.
9. S. Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer-Verlag, 1971.
10. M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *LPAR 1992*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 1992.
11. Alwen Tiu. A local system for intuitionistic logic. In M. Hermann and A. Voronkov, editors, *LPAR 2006*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 242–256. Springer-Verlag, 2006. `http://users.rsise.anu.edu.au/~tiu/localint.pdf`.