Fundamenta Informaticae 170 (2019) 139–176
DOI 10.3233/FI-2019-1858
IOS Press

Strong Normalization for Truth Table Natural Deduction

Herman Geuvers[*]
Institute for Computing and Information Science
Radboud University Nijmegen, The Netherlands
herman@cs.ru.nl

Iris van der Giessen
Department of Philosophy and Religious Studies
Utrecht University, The Netherlands
i.vandergiessen@uu.nl

Tonny Hurkens
Haps, The Netherlands
hurkens@science.ru.nl

Abstract. We present a proof of strong normalization of proof-reduction in a general system of natural deduction called *truth table natural deduction*. In previous work, we have defined truth table natural deduction, which is a method for deriving intuitionistic derivation rules for a connective from its truth table. This yields natural deduction rules for each connective separately. Moreover, these rules adhere to a standard format which gives rise to a general notions of detour and permutation conversion for natural deductions. The aim is to remove all convertibilities and obtain a deduction in normal form. In general, conversion of truth table natural deductions is non-deterministic, which makes it more challenging to study. It has already been shown that this conversion is weakly normalizing. To prove strong normalization, we construct a conversion-preserving translation from deductions to terms in an extension of simply typed lambda calculus

[*]Address for correspondence: Institute for Computing and Information Science, Radboud University Nijmegen, The Netherlands

This research has partially been supported by Cost Action CA15123 EUTypes.
The paper is partially funded by the Netherlands Organisation for Scientific Research under grant 639.073.807.

which we call *parallel simply typed lambda calculus* and which we prove to be strongly normalizing. This makes it possible to get a grip on the non-deterministic character of conversion in the intuitionistic truth table natural deduction system.

Keywords: Natural deduction; truth tables; intuitionistic logic; detour conversion; permutation conversion; strong normalization

1. Introduction

Natural deduction originates from the work by the German mathematician Gerhard Gentzen (1909-1945): *Untersuchungen über das logische Schliessen*, 1935 [1], where he studies mathematical foundations and proof theory. His most famous contributions are natural deduction and sequent calculus. He constructed both logical systems in an attempt to prove the consistency of number theory. He first defined natural deduction, for which he tried to prove consistency using the cut elimination theorem, which he called the Hauptsatz. However, he did not manage to prove the Hauptsatz for natural deduction. Therefore he introduced the sequent calculus for which he proved the Hauptsatz. Later, in 1956, the Swedish logician Dag Prawitz (born in 1936) gave a direct proof of the cut elimination theorem for natural deduction in *Natural deduction: a proof-theoretical study* [2].

The main idea of natural deduction is to capture mathematical reasoning in a formal system of reasoning rules in a natural way. This results in a system where propositions are derived by applying deduction rules to a set of assumptions. Gentzen distinguished so-called introduction and elimination rules, where the elimination rules arise from the introduction rules in a natural manner known as the inversion principle. The Hauptsatz states that elimination from introduced formulas can be avoided, which results in 'normal' derivations. These normal forms correspond to cut-free derivations in sequent calculus and normalization corresponds to cut elimination. Since the work of Gentzen and Prawitz, various other natural deduction systems have been introduced.

This paper provides a further proof-theoretic analysis of the *truth table natural deduction system* that has been introduced in [3] by two authors of the present paper. The main idea of this system is that propositional derivation rules for a connective can be derived in a canonical way from its definition via a truth table. This results in derivation rules in a 'standard' format. We use a tree format with sequents in the nodes and leaves, where all elimination and introduction rules have the 'standard' form:

$$\frac{\Gamma \vdash A_1 \ \ldots \ \Gamma \vdash A_n \qquad \Gamma, B_1 \vdash D \ \ldots \ \Gamma, B_m \vdash D}{\Gamma \vdash D}$$

So if the conclusion of a rule is $\Gamma \vdash D$, then the premises of the rule can be one of the forms:

- $\Gamma, B \vdash D$: we still need to prove D from Γ, but we are now also allowed to use B as additional assumption. We call B a *case*.
- $\Gamma \vdash A$: instead of proving D from Γ, we now need to prove A from Γ. We call A a *lemma*.

Both classical propositional logic and intuitionistic propositional logic can be defined in this way.

The analysis in this paper focuses on the proof-theoretic properties for intuitionistic logic, and extends the work in [3] and [4]. The standard form for intuitionistic logic is very suitable for defining *detour conversion* (get rid of an introduction rule immediately followed by an elimination rule) and *permutation conversion* (get rid of an elimination rule immediately followed by an elimination rule). The aim is to remove all detour and permutation convertibilities and obtain a normal derivation, where elimination rules are only applied to assumptions. It has already been shown that conversion is weakly normalizing: there is a strategy that always converts a deduction to one in normal form. Deductions in normal form have the *subformula property*: every formula that occurs in a deduction in normal form of the judgement $\Gamma \vdash A$ is a subformula of A or of a formula in Γ. This implies nice properties such as consistency of the logic and decidability [3, 4].

The main contribution of this paper is the strong normalization of the combination of detour and permutation conversion: no matter in which order one converts a deduction, one always reaches a normal deduction. This is an non-trivial result, because conversion in the intuitionistic truth table system is non-deterministic: in some detour conversion steps there are choices to be made [3, 4]. Moreover, conversion does not lead to unique normal forms and therefore the system does not satisfy the Church-Rosser property.

The proof of strong normalization is based on the work of De Groote [5] who has proved strong normalization for proof-reduction in the Prawitz system of natural deduction. The proof of De Groote is based on a conversion-preserving translation from the Prawitz system to simply typed lambda calculus, which is strongly normalizing. Conversion in the Prawitz system is easier in the sense that it is deterministic. For the translation of the truth table system, we introduce an extended variant of the simply typed lambda calculus, which we call *parallel simply typed lambda calculus* and which we prove to be strongly normalizing. This enables us to get a grip on the non-deterministic character of the conversions.

The paper is structured as follows. Section 2 defines the truth table natural deduction system for intuitionistic propositional logic. It gives some examples and it recaps some well-known definitions. It also introduces the notions of detour conversion and permutation conversion, the study of which is the main topic of this paper. The material presented in Section 2 basically comes from [3] and [4].

Section 3 introduces a term calculus for the truth table natural deduction system. Here, deductions are interpreted as terms and detour conversion and permutation conversion correspond to term-reductions. This has been described in [4] and it extends the well-known Curry-Howard formulas-as-types isomorphism to these new connectives and deduction rules: a deduction of a formula A is interpreted as a *proof-term* of type A. Just as in [4], we analyze conversions on deductions by studying reductions on proof-terms.

Sections 4 and 5 contain the main contribution of the paper: a proof of strong normalization of the combination of detour conversion and permutation conversion. We prove this by analyzing the corresponding reductions on proof-terms, as defined in Section 3. In Section 4 we present the parallel simply typed lambda calculus and in Section 5 we give a comprehensive study of the method of De Groote applied to the intuitionistic truth table natural deduction system in order to prove strong normalization. We end the paper with a section on related and future work.

2. The truth table natural deduction system

In this section, we introduce the truth table natural deduction system for intuitionistic propositional logic developed in [3]. The main idea of the system is that propositional derivation rules for a connective can be derived in a canonical way from its definition via a truth table. In this way, it is possible to derive elimination rules and introduction rules, where the introduction rules come in an intuitionistic and a classical variant. In this paper we focus on the intuitionistic rules and we call the natural deduction system that we obtain the *truth table natural deduction system*.

For the well-known connectives, the system is equivalent (the same formulas are derivable) to the well-known one introduced by Gentzen [1] and studied e.g. by Prawitz [2]. A difference is that the deduction rules that we derive are in a ‘standard format’ as described in Section 1. This makes it easier to study generic properties and define general constructions on derivations, like normalization.

This section is basically a recap of the definitions introduced in [3]. For a Kripke semantics of the intuitionistic system and for information about the classical system we refer to [3] and [4].

We consider an arbitrary set of connectives, each with a fixed *arity*. Each connective c, say of arity n, is defined via its truth table t_c, which is a function of arity n, $t_c : \{0,1\}^n \to \{0,1\}$. The values of this function can be represented in a table where each row represents an application of the function, that is, for $a_i \in \{0,1\}$, we get the row $a_1 \ldots a_n \mid t_c(a_1,\ldots,a_n)$. If c is an n-ary connective, the truth table has 2^n rows and $n+1$ columns. So $\bot$ and $\top$ are considered as 0-ary connectives. We will also look at the well-known connectives $\vee, \wedge, \to$ (of arity 2) and $\neg$ (of arity 1). In [3] and [4] other connectives like the 3-ary connectives $\mathsf{if\text{-}then\text{-}else}$ and most are studied as examples.

The language of the truth table system is built from *propositional letters*, $p_0, p_1, p_2, \ldots$, and *connectives*, each with fixed arity, $c_0, c_1, c_2, \ldots$.

Definition 2.1. Let $\mathcal{C}$ be a set of connectives, each with a fixed arity. The set $\mathrm{PROP}_{\mathcal{C}}$ of propositions over $\mathcal{C}$ is the smallest set such that

1. for p a propositional letter, we have $p \in \mathrm{PROP}_{\mathcal{C}}$,
2. for $c \in \mathcal{C}$ of arity n and $A_1,\ldots,A_n \in \mathrm{PROP}_{\mathcal{C}}$, we have $c(A_1,\ldots,A_n) \in \mathrm{PROP}_{\mathcal{C}}$.

We work in propositional logic, but we prefer to speak of *formulas* instead of propositions. We write formulas with capital letters A, B, C, etc. For formulas with a special role, we use Greek letters Φ, Ψ. For connectives $\bot$ and $\top$, $\bot\,()$ and $\top()$ are formulas. Since the parentheses are superfluous, we write $\bot$ and $\top$ for these formulas.

We now define how to derive intuitionistic natural deduction rules from truth tables defined in [3].

Definition 2.2. Let c be an n-ary connective with a truth table t_c. We write $\phi = c(p_1,\ldots,p_n)$ where $p_1,\ldots,p_n$ are proposition letters and we write $\Phi = c(A_1,\ldots A_n)$ where $A_1,\ldots,A_n$ are formulas. We define a derivation rule for each row in the truth table t_c of c. If the truth value of a row in t_c equals 0, then we obtain an *elimination rule* (el). If the row gives a 1, then it gives rise to an *introduction rule* (in). The rules are defined in the following way, where the row of the truth table is given on the left-hand side.

$$\begin{array}{c|c} p_1 \dots p_n & \phi \\ \hline a_1 \dots a_n & 0 \end{array} \quad \mapsto \quad \frac{\Gamma \vdash \Phi \qquad \dots \quad \Gamma \vdash A_k \quad \dots \qquad \dots \quad \Gamma, A_l \vdash D \quad \dots}{\Gamma \vdash D}\,\text{el}$$

where A_k ranges over all formulas where $a_k = 1$ and A_l ranges over all formulas where $a_l = 0$.

$$\begin{array}{c|c} p_1 \dots p_n & \phi \\ \hline b_1 \dots b_n & 1 \end{array} \quad \mapsto \quad \frac{\dots \quad \Gamma \vdash A_j \quad \dots \qquad \dots \quad \Gamma, A_i \vdash \Phi \quad \dots}{\Gamma \vdash \Phi}\,\text{in}$$

where A_j ranges over all formulas where $b_j = 1$ and A_i ranges over all formulas where $b_i = 0$.

For an n-ary connective c, there are 2^n rules, since there are 2^n rows in the truth table. The rules in Definition 2.2 are given in sequent notation including a set Γ containing auxiliary assumptions. We only specify Γ when necessary.

In an elimination rule, we call $\Gamma \vdash \Phi$ the *major premise*, and the other premises are called *minor premises*. Recall the *standard form* for the derivation rules.

$$\frac{\Gamma \vdash A_1 \ \dots \ \Gamma \vdash A_n \qquad \Gamma, B_1 \vdash D \ \dots \ \Gamma, B_m \vdash D}{\Gamma \vdash D}$$

We call the A's a *lemma* and the B's a *case*. If we look at the rules in Definition 2.2, we see that A_j occurs as a lemma, if $a_j = 1$ in truth table t_c, and A_i occurs as a case, if $a_i = 0$ in t_c. This standard form simplifies the definition and study of properties like normalization, as we will see later. Most of the well-known natural deduction rules adhere to this standard form, with the implication-introduction rules being a notable exception. Our introduction rules for implication are different but equivalent, as shown in [3, 4].

Example 2.3. Consider the truth tables of $\vee$, $\to$ and $\neg$.

A	B	$A \vee B$	$A \to B$
0	0	0	1
0	1	1	1
1	0	1	0
1	1	1	1

A	$\neg A$
0	1
1	0

We derive the following intuitionistic rules, labeled by their corresponding entries in the rows of the truth tables. Disjunction $\vee$ has one elimination rule and three introduction rules.

$$\frac{\vdash A \vee B \qquad A \vdash D \qquad B \vdash D}{\vdash D}\,\vee\text{-el}_{00} \qquad \frac{\vdash A \qquad B \vdash A \vee B}{\vdash A \vee B}\,\vee\text{-in}_{10}$$

$$\frac{A \vdash A \vee B \qquad \vdash B}{\vdash A \vee B}\,\vee\text{-in}_{01} \qquad \frac{\vdash A \qquad \vdash B}{\vdash A \vee B}\,\vee\text{-in}_{11}$$

Implication $\to$ also has one elimination rule and three introduction rules.

$$\frac{A \vdash A \to B \qquad B \vdash A \to B}{\vdash A \to B}\,\to\text{-in}_{00} \qquad \frac{\vdash A \to B \qquad \vdash A \qquad B \vdash D}{\vdash D}\,\to\text{-el}_{10}$$

$$\frac{A \vdash A \to B \qquad \vdash B}{\vdash A \to B}\,\to\text{-in}_{01} \qquad \frac{\vdash A \qquad \vdash B}{\vdash A \to B}\,\to\text{-in}_{11}$$

The introduction and elimination rule for negation are given by:

$$\frac{A \vdash \neg A}{\vdash \neg A}\ \neg\text{-in}_0 \qquad\qquad \frac{\vdash \neg A \qquad \vdash A}{\vdash D}\ \neg\text{-el}_1$$

Note that the introduction rule for $\neg$ only has a case and the elimination rule only has one minor lemma. See Appendix A for the rules of the other usual connectives $\wedge$, $\bot$ and $\top$.

Definition 2.4. Let $\mathcal{C}$ be a set of connectives, each with a fixed arity. We define the *intuitionistic truth table natural deduction system* for $\mathcal{C}$, $\text{IPC}_{\mathcal{C}}$, as follows.

1. We have the *axiom rule*

$$\frac{}{\Gamma \vdash A}\ \text{axiom, if } A \in \Gamma$$

2. $\text{IPC}_{\mathcal{C}}$ contains the elimination rules and intuitionistic introduction rules for all connectives in $\mathcal{C}$.

We write $\Gamma \vdash_{\text{IPC}_{\mathcal{C}}} A$ if $\Gamma \vdash A$ is derivable using the rules of $\text{IPC}_{\mathcal{C}}$. Usually we will just write $\Gamma \vdash A$ if the set of connectives is clear. A *derivation* in $\text{IPC}_{\mathcal{C}}$ is a tree built up using the rules in $\text{IPC}_{\mathcal{C}}$ whose leaves are instances of the axiom rule or instances of connective rules without any premises. We use Greek capital letters Π or Σ to denote derivations. If judgement $\Gamma \vdash A$ is the result of an axiom rule, $\Gamma \vdash A$ is called an *assumption*. If the end-sequent in a derivation Π is of the form $\Gamma \vdash D$ we say that D is *derived from* Γ.

Often it is possible to reduce the number of rules. In the following we will see the reduced formats as described in [3]. We first look at two standard lemmas. The first lemma is the weakening property. The second lemma shows how to put derivations together in one derivation.

Lemma 2.5. (Weakening)
If Π is a derivation of $\Gamma \vdash A$ and $\Gamma \subseteq \Delta$, then Π is also a derivation of $\Delta \vdash A$.

Proof:
Proof by simple induction on the derivation of $\Gamma \vdash A$. □

Lemma 2.6. If $\Gamma \vdash A$ and $\Delta, A \vdash B$, then $\Gamma, \Delta \vdash B$.

Proof:
This is proved by induction on the derivation of $\Delta, A \vdash B$. □

This lemma can be written in a more suggestive way using proof-trees. If Σ is a derivation of $\Gamma \vdash A$ and Π of $\Delta, A \vdash B$, then the derivation of $\Gamma, \Delta \vdash B$ becomes:

$$\begin{array}{c} \begin{array}{ccc} \Sigma & & \Sigma \\ \Gamma \vdash A & \dots & \Gamma \vdash A \end{array} \\ \Pi \\ \Gamma, \Delta \vdash B \end{array}$$

This is possible, because the only place in Π where the hypothesis A can be used is an instance of the axiom rule of the shape $\Delta', A \vdash A$ for some $\Delta' \supseteq \Delta$. In Π we replace each $\Delta', A \vdash A$ by the derivation Σ of $\Delta', \Gamma \vdash A$, which is possible due to weakening.

The following two lemmas state the reductions to the optimized rules. For proofs see [3]. Note that Lemma 2.6 is needed for Lemma 2.8.

Lemma 2.7. Two derivation rules of the form

$$\frac{\vdash A_1 \ldots \vdash A_n \quad B_1 \vdash D \ldots B_m \vdash D \quad \Phi \vdash D}{\vdash D}\,, \quad \frac{\vdash A_1 \ldots \vdash A_n \quad \vdash \Phi \quad B_1 \vdash D \ldots B_m \vdash D}{\vdash D}$$

are equivalent to the rule

$$\frac{\vdash A_1 \ldots \vdash A_n \qquad B_1 \vdash D \ldots B_m \vdash D}{\vdash D}\,.$$

Lemma 2.8. The following rules are equivalent:

$$\frac{\vdash A_1 \;\ldots\; \vdash A_n \qquad B \vdash D}{\vdash D} \quad \text{and} \quad \frac{\vdash A_1 \;\ldots\; \vdash A_n}{\vdash B}.$$

Optimization does not always result in unique optimized rules, but for the usual connectives $\wedge, \vee, \to$ and $\neg$ it does.

Example 2.9. In Example 2.3, we saw the rules for $\vee, \to$ and $\neg$ derived from the truth tables, following Definition 2.2. The optimized rules for those connectives are given below. The rules for $\neg$ are already in optimized form, therefore we will not repeat them here.

$$\frac{\vdash A \vee B \qquad A \vdash D \qquad B \vdash D}{\vdash D}\,\vee\text{-el} \qquad \frac{\vdash A}{\vdash A \vee B}\,\vee\text{-in}_1 \qquad \frac{\vdash B}{\vdash A \vee B}\,\vee\text{-in}_2$$

$$\frac{\vdash A \to B \qquad \vdash A}{\vdash B}\,\to\text{-el} \qquad \frac{\vdash B}{\vdash A \to B}\,\to\text{-in}_1 \qquad \frac{A \vdash A \to B}{\vdash A \to B}\,\to\text{-in}_2$$

The optimized rules for $\vee$ correspond to the well-known Prawitz rules [2]. Note that the rules for $\to$ are essentially different from those of Prawitz, but they are proved to be equivalent in the sense that they prove the same formulas [3, 4]. See Appendix A for the optimized rules of the connectives $\wedge$, $\bot$ and $\top$.

We now introduce so-called *detour* and *permutation* conversion of intuitionistic derivations which have already been defined in [4]. In [4], normalization properties have been established, such as strong normalization for detour conversion and for permutation conversion separately, and weak normalization for the combination of the two.

In a normal derivation all major premises of elimination rules are assumptions. A *convertibility* is a pattern in a derivation for which a major premise of an elimination rule is not an assumption. These are also sometimes called *cuts*, but this terminology is usually reserved for sequent calculus. There are two different convertibilities, which we call *detour convertibilities* and *permutation convertibilities*. (In [3], they are called *direct cuts* and *indirect cuts* respectively, but we prefer to stick to the terminology of 'convertibility', as we have done in [4].)

A detour convertibility is an introduction rule of a formula Φ immediately followed by an elimination rule of Φ. We will see that there are two possibilities circumventing such patterns. Before giving the definitions, we illustrate it with an example, where we intuitively see why one wants to avoid detour convertibilities. We also see that detour conversion is non-deterministic.

Example 2.10. Consider the non-optimized rules for $\vee$ from Example 2.3. An example of a detour convertibility of $\vee$ is.

$$\dfrac{\dfrac{\begin{array}{c}\Sigma_A\\ \Gamma\vdash A\end{array}\qquad \begin{array}{c}\Sigma_B\\ \Gamma\vdash B\end{array}}{\Gamma\vdash A\vee B}\,\vee\text{-in}_{11} \qquad \begin{array}{c}\Pi_A\\ \Gamma,A\vdash D\end{array}\qquad \begin{array}{c}\Pi_B\\ \Gamma,B\vdash D\end{array}}{\Gamma\vdash D}\,\vee\text{-el}_{00}$$

We see that first introducing $A \vee B$ and then eliminating it is not necessary and undesirable. Note also that the derivation does not satisfy the subformula property. It can be replaced by two different derivations:

$$\begin{array}{c}\Sigma_A\\ \Gamma\vdash A\\ \Pi_A\\ \Gamma\vdash D\end{array}\quad \text{or} \quad \begin{array}{c}\Sigma_B\\ \Gamma\vdash B\\ \Pi_B\\ \Gamma\vdash D\end{array}$$

This means that we can make a choice in the conversion, which means that it is non-deterministic.

Definition 2.11. Let c be an n-ary connective with an elimination rule and an intuitionistic introduction rule derived from truth table t_c. So we have the following rows in the truth table.

p_1	$\dots$	p_n	$c(p_1,\dots,p_n)$
a_1	$\dots$	a_n	0
b_1	$\dots$	b_n	1

A *detour convertibility* is a pattern in which the consequence of an introduction rule is the major premise of an elimination rule. This has the following form with subtrees Σ_j, Σ_i, Π_k and Π_l, where $\Phi = c(A_1,\dots,A_n)$.

$$\dfrac{\dfrac{\begin{array}{c}\Sigma_j\\ \dots\Gamma\vdash A_j\dots\end{array}\qquad \begin{array}{c}\Sigma_i\\ \dots\Gamma,A_i\vdash\Phi\dots\end{array}}{\Gamma\vdash\Phi}\,\text{in} \qquad \begin{array}{c}\Pi_k\\ \dots\Gamma\vdash A_k\dots\end{array}\qquad \begin{array}{c}\Pi_l\\ \dots\Gamma,A_l\vdash D\dots\end{array}}{\Gamma\vdash D}\,\text{el}$$

Here, A_j ranges over all formulas where $b_j = 1$ and A_i ranges over all formulas where $b_i = 0$ in the row of the truth table. Similarly, A_k ranges over all formulas where $a_k = 1$ and A_l ranges over all formulas where $a_l = 0$.

Definition 2.12. (Detour conversion)
A *detour conversion* is the operation of replacing a detour convertibility from Definition 2.11 in a derivation by one of the following derivations. In general there are several (but at least one) ways to do this. The two general patters for detour conversion are as follows: (1) when a case of the elimination rule matches with a lemma of the introduction rule; (2) when a lemma of the elimination matches with a case of the introduction rule.

1. If $j' = l'$ for some j', l' which means $A_{j'} = A_{l'}$:

$$\begin{array}{c}\begin{array}{ccc}\Sigma_{j'} & & \Sigma_{j'}\\ \Gamma\vdash A_{j'} & \dots & \Gamma\vdash A_{j'}\end{array}\\ \Pi_{l'}\\ \Gamma\vdash D\end{array}$$

2. If $i' = k'$ for some i', k' which means $A_{i'} = A_{k'}$:

$$\dfrac{\begin{array}{c}\begin{array}{ccc}\Pi_{k'} & & \Pi_{k'}\\ \Gamma \vdash A_{i'} & \dots & \Gamma \vdash A_{i'}\end{array}\\ \Sigma_{i'}\\ \Gamma \vdash \Phi\end{array} \qquad \begin{array}{c}\Pi_k\\ \dots\Gamma \vdash A_k\dots\end{array} \qquad \begin{array}{c}\Pi_l\\ \dots\Gamma, A_l \vdash D\dots\end{array}}{\Gamma \vdash D}\ \text{el}$$

Both diagrams yield correct derivations because of Lemma 2.6.

In the two rows of the truth table, as depicted in Definition 2.11, each m where $a_m \neq b_m$ yields a 'matching case' of Definition 2.12, i.e. a possibility for a detour conversion. So there can be several 'matching cases' $l' = j'$ or $k' = i'$. This means that detour conversion is non-deterministic. In [3] and [4], the matching cases are just written as $j = l$ and $i = k$, but in this paper, instead, we write $j' = l'$ and $i' = k'$ to indicate that j', l', i', k' are chosen entries.

Detour convertibility and detour conversion can also be defined in the same way for the optimized rules [3, 4].

Example 2.13. Example 2.10 shows matching case (1) of Definition 2.12. Here we see an example of matching case (2) with optimized rules for $\to$.

$$\dfrac{\dfrac{\begin{array}{c}\Sigma\\ \Gamma, A \vdash A \to B\end{array}}{\Gamma \vdash A \to B}\ \to\text{-in}_2 \qquad \begin{array}{c}\Pi\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ \to\text{-el} \quad \text{is replaced by} \quad \dfrac{\begin{array}{c}\Pi\\ \Gamma \vdash A\\ \Sigma\\ \Gamma \vdash A \to B\end{array} \qquad \begin{array}{c}\Pi\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ \to\text{-el}$$

Now we turn to the permutation convertibilities. A permutation convertibility is a pattern in which the consequence of an elimination rule is the major premise of another elimination rule. It is important to look at such patterns, because it may block a detour convertibility: between the introduction and elimination rule can be several other elimination rules. To solve this blockage, we can permute one elimination rule over the other. This justifies the name of permutation convertibility.

Definition 2.14. Let c be an n-ary connective and let c' be an n'-ary connective with elimination rules r and r' respectively. So we have the following rows in truth tables t_c and $t_{c'}$:

p_1	$\dots$	p_n	$c(p_1, \dots, p_n)$
b_1	$\dots$	b_n	0

p_1	$\dots$	$p_{n'}$	$c'(p_1, \dots, p_{n'})$
a_1	$\dots$	$a_{n'}$	0

A *permutation convertibility* is a pattern of the following form, in which $\Phi = c(B_1, \dots B_n)$ and $\Psi = c'(A_1, \dots, A_{n'})$.

$$\dfrac{\dfrac{\begin{array}{c}\Sigma\\ \Gamma \vdash \Psi\end{array} \qquad \begin{array}{c}\Sigma_j\\ \dots\Gamma \vdash A_j\dots\end{array} \qquad \begin{array}{c}\Sigma_i\\ \dots\Gamma, A_i \vdash \Phi\dots\end{array}}{\Gamma \vdash \Phi}\ \text{el}_{r'} \qquad \begin{array}{c}\Pi_k\\ \dots\Gamma \vdash B_k\dots\end{array} \qquad \begin{array}{c}\Pi_l\\ \dots\Gamma, B_l \vdash D\dots\end{array}}{\Gamma \vdash D}\ \text{el}_r$$

Here, A_j ranges over all formulas where $a_j = 1$ and A_i ranges over all formulas where $a_i = 0$. Similarly, B_k ranges over all formulas where $b_k = 1$ and B_l ranges over all formulas where $b_l = 0$.

For optimized rules, permutation convertibilities are defined in a similar way.

Example 2.15. This example with optimized rules is also stated in [4].

$$\dfrac{\dfrac{\Gamma \vdash A \vee B \qquad \dfrac{\Gamma, A, C \vdash C \to D}{\Gamma, A \vdash C \to D}\,{\to}\text{-in}_2 \qquad \Gamma, B \vdash C \to D}{\Gamma \vdash C \to D}\,\vee\text{-el} \qquad \Gamma \vdash C}{\Gamma \vdash D}\,{\to}\text{-el}$$

In this example we have a permutation convertibility where the conclusion of $\vee$-el is the major premise of $\to$-el. In this derivation, this permutation convertibility blocks the detour convertibility of the combination of $\to$-in$_2$ and $\to$-el.

Definition 2.16. (Permutation conversion)
A *permutation conversion* is defined by replacing a permutation convertibility from Definition 2.14 by the following derivation, where the two elimination rules are permuted.

$$\dfrac{\begin{matrix}\Sigma\\ \Gamma \vdash \Psi\end{matrix} \qquad \begin{matrix}\Sigma_j\\ \dots\Gamma \vdash A_j\dots\end{matrix} \qquad \dots \dfrac{\begin{matrix}\Sigma_i\\ \Gamma, A_i \vdash \Phi\end{matrix} \qquad \begin{matrix}\Pi_k\\ \dots\Gamma, A_i \vdash B_k\dots\end{matrix} \qquad \begin{matrix}\Pi_l\\ \dots\Gamma, A_i, B_l \vdash D\dots\end{matrix}}{\Gamma, A_i \vdash D}\,\text{el}_r \dots}{\Gamma \vdash D}\,\text{el}_{r'}$$

This is a correct derivation, because of the weakening property, that is, Π_k is a derivation of $\Gamma, A_i \vdash B_k$ since it was a derivation of $\Gamma \vdash B_k$. Similarly, Π_l is a derivation of $\Gamma, A_i, B_l \vdash D$.

Example 2.17. Now we look at the permutation conversion of the derivation of Example 2.15. We get the following derivation.

$$\dfrac{\Gamma \vdash A \vee B \qquad \dfrac{\dfrac{\Gamma, A, C \vdash C \to D}{\Gamma, A \vdash C \to D}\,{\to}\text{-in}_2 \qquad \Gamma, A \vdash C}{\Gamma, A \vdash D}\,{\to}\text{-el} \qquad \dfrac{\Gamma, B \vdash C \to D \qquad \Gamma, B \vdash C}{\Gamma, B \vdash D}\,{\to}\text{-el}}{\Gamma \vdash D}\,\vee\text{-el}$$

We observe that we have created a detour convertibility of the combination of $\to$-in$_2$ and $\to$-el. Consequently, this can be reduced with a detour conversion as in Example 2.13.

Example 2.18. An interesting case is a connective (for example negation) whose truth table has a row of the form $t_c(1, \dots, 1) = 0$. Such a connective has an elimination rule without any case in its premises. We observe that in a permutation conversion with this elimination rule, parts of the derivation disappear in the conversion. This is for instance the case in the following permutation conversion with $\neg$.

$$\dfrac{\dfrac{\vdash \neg B \qquad \vdash B}{\vdash A \vee A}\,\neg\text{-el} \qquad A \vdash A \qquad A \vdash A}{\vdash A}\,\vee\text{-el} \qquad \text{is replaced by} \qquad \dfrac{\vdash \neg B \qquad \vdash B}{\vdash A}\,\neg\text{-el}$$

Definition 2.19. A derivation in IPC$_{\mathcal{C}}$ is *normal* if every major premise of an elimination rule is an assumption.

Equivalently, a derivation is normal if it does not contain any detour or permutation convertibilities. It can be shown that a normal derivation satisfies the subformula property, see [4] for details.

We have defined the detour and permutation conversions for non-optimized rules. It is also possible to define those conversions for the optimized rules, which is done in [4].

In the definition of detour conversion (Definition 2.12), we saw that detour conversion is potentially non-deterministic: in case there are more 'matching cases', there are several ways to eliminate the detour. This can actually be exploited to show that IPC$_\mathcal{C}$ does not satisfy the Church-Rosser property. The Church-Rosser property says that if derivation Σ converts to Π_1 and to Π_2, then there is a Π_3, such that both Π_1 and Π_2 can be converted to Π_3. Informally speaking, the Church-Rosser property says that the order of the conversions does not make a difference to the eventual result. This is also known as confluence.

The Church-Rosser property fails whenever $t_c(a_1, \ldots, a_n) \neq t_c(b_1, \ldots, b_n)$ for $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ that are 'sufficiently different', so we need to talk about the distance between two sequences of the same length, which we just define by $|(a_1, \ldots, a_n) - (b_1, \ldots, b_n)| := \Sigma_{i=1}^{n} |a_i - b_i|$.

Proposition 2.20. Let $\mathcal{C}$ be a set of connectives. IPC$_\mathcal{C}$ satisfies the Church-Rosser property if and only if for each n-ary connective c in $\mathcal{C}$, and all $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$, we have

$$|(a_1, \ldots, a_n) - (b_1, \ldots, b_n)| \geq 2 \quad \text{implies} \quad t_c(a_1, \ldots, a_n) = t_c(b_1, \ldots, b_n).$$

The proof is postponed until the next section (Proposition 3.9), where we can give the proof using proof-terms.

Corollary 2.21. If $\mathcal{C}$ contains a non-constant connective of arity 3 or higher, IPC$_\mathcal{C}$ does not satisfy the Church-Rosser property.

Proof:
Using Proposition 2.20, it suffices to show that, for any connective c of arity $n \geq 3$,

$$c \text{ is constant} \quad \Longleftrightarrow \quad |\vec{a} - \vec{b}| \geq 2 \text{ implies } t_c(\vec{a}) = t_c(\vec{b}) \text{ for all } \vec{a}, \vec{b},$$

where $\vec{a}$ is short notation for $(a_1, \ldots, a_n)$.

The part $\Rightarrow$ is obvious. For the part $\Leftarrow$, it suffices to consider a connective of arity 3. We can depict the rows of the truth table for c in the diagram below, which can be seen as an ordering of the triples, based on the pointwise ordering $0 \leq 1$, but its main purpose is that all triples that only differ on one position are connected by a line.

$$\begin{tikzcd}
\text{(IV)} & & 1,1,1 \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
\text{(III)} & 1,1,0 \arrow[d, dash] \arrow[dr, dash] & 1,0,1 \arrow[dl, dash] \arrow[dr, dash] & 0,1,1 \arrow[dl, dash] \arrow[d, dash] \\
\text{(II)} & 1,0,0 \arrow[dr, dash] & 0,1,0 \arrow[d, dash] & 0,0,1 \arrow[dl, dash] \\
\text{(I)} & & 0,0,0 &
\end{tikzcd}$$

Now suppose that for all $\vec{a}, \vec{b}$ with $|\vec{a} - \vec{b}| \geq 2$ we have $t_c(\vec{a}) = t_c(\vec{b})$. Then all triples on level (I) and (III) have the same truth value, because $|\vec{a} - \vec{b}| \geq 2$ for each pair of these triples. Similarly all triples on levels (II) and (IV) have the same truth value and the two triples on level (I) and (IV) have the same truth value. As a consequence, all triples have the same truth value and c is constant. $\square$

It should be remarked that detour and permutation conversion for the well-known Prawitz style natural deduction is Church Rosser. So, if one optimizes the rules for $\wedge, \vee, \rightarrow, \neg$, some of the non-determinism gets somehow resolved: the non-deterministic choices are fixed so one ends up with a derivation that has a unique normal form. Also note that there are other connectives, for example most, where detour conversion is also non-confluent for the optimized deduction rules, and there is no way to resolve the non-determinism other then fixing an arbitrary ordering upon the arguments of most. See [4] for more details.

3. The Curry-Howard isomorphism

Following the Curry-Howard isomorphism [6, 7], formulas correspond to types and derivations correspond to proof-terms. So far, we have considered the truth table system in a proof theoretical context. The short notation of proof-terms for derivation gives the great advantage of studying proof-terms over derivation trees. In addition, detour and permutation conversions correspond to reductions of terms. This makes it easier to prove strong normalization in Section 5.

The type system that we define is based on the λ-calculus. For each set of connectives $\mathcal{C}$ we define a system $\lambda_\mathcal{C}$, where for each derivation rule in IPC$_\mathcal{C}$, we give a typing rule in $\lambda_\mathcal{C}$. The types in $\lambda_\mathcal{C}$ are exactly the formulas. In addition, we define reductions in $\lambda_\mathcal{C}$ that correspond to detour and permutation conversions [4].

Definition 3.1. Let $\mathcal{C}$ be a set of connectives. The *types* in $\lambda_\mathcal{C}$ are the formulas involving connectives from $\mathcal{C}$, so each $A \in \text{PROP}_\mathcal{C}$ is a type of $\lambda_\mathcal{C}$. The abstract syntax for *proof-terms* in $\lambda_\mathcal{C}$ is

$$M ::= x \mid M \cdot_r [\overline{M}; \overline{\lambda x.M}] \mid \{\overline{M}; \overline{\lambda x.M}\}_r$$

where x ranges over typed variables and r ranges over the rules of all connectives in $\mathcal{C}$. We write $\overline{M}$ to mean a finite sequence of terms. For the sake of readability, we prefer to leave out the specific types of the variables x in abstractions. Let a *context* Γ be a set of type declarations of the form $x : A$. The terms are typed using the derivation rules of IPC$_\mathcal{C}$ as follows. So the rules are derived from the truth table, just as in Definition 2.2.

$$\frac{}{\Gamma \vdash x : A}\ \text{axiom, if } x : A \in \Gamma$$

$$\frac{\Gamma \vdash M : \Phi \qquad \ldots \Gamma \vdash N_k : A_k \ldots \qquad \ldots \Gamma, x_l : A_l \vdash O_l : D \ldots}{\Gamma \vdash M \cdot_r [\overline{N}; \overline{\lambda x.O}] : D}\ r, \text{el}$$

$$\frac{\ldots \Gamma \vdash N_j : A_j \ldots \qquad \ldots \Gamma, y_i : A_i \vdash M_i : \Phi \ldots}{\Gamma \vdash \{\overline{N}; \overline{\lambda y.M}\}_r : \Phi}\ r, \text{in}$$

Both in the elimination and introduction rules, we prefer to use capital letter M for terms of type Φ, where $\Phi = c(A_1, \dots, A_n)$ for the concerned connective c. In the elimination rules, $\overline{N}$ is the sequence of terms N_k for the 1-entries in the truth table t_c and $\overline{\lambda x.O}$ is the sequence of the $\lambda x_l.O_l$'s for the 0-entries. To stick to a same terminology, we call the N_k's a *lemma* and the $\lambda x_l.O_l$'s a *case*. Similarly, for the introduction rules.

If it is clear from the context which rule is applied, we omit r in the elimination and introduction term. The method of Definition 3.1 can also be applied to the optimized rules in a straightforward way.

Example 3.2. The typing rules of disjunction are as follows, with their corresponding terms.

$$\frac{\vdash M : A \vee B \quad x : A \vdash O_1 : D \quad y : B \vdash O_2 : D}{\vdash M \cdot [\,; \lambda x.O_1, \lambda y.O_2] : D} \vee\text{-el} \qquad \frac{\vdash N_A : A \quad y : B \vdash M : A \vee B}{\vdash \{N_A\,;\, \lambda y.M\} : A \vee B} \vee\text{-in}_{10}$$

$$\frac{x : A \vdash M : A \vee B \qquad \vdash N_B : B}{\vdash \{N_B\,;\, \lambda x.M\} : A \vee B} \vee\text{-in}_{01} \qquad \frac{\vdash N_A : A \qquad \vdash N_B : B}{\vdash \{N_A, N_B;\,\} : A \vee B} \vee\text{-in}_{11}$$

We see that derivations in IPC$_{\mathcal{C}}$ directly correspond to proof-terms in $\lambda_{\mathcal{C}}$. Now we define term reduction rules that correspond to detour and permutation conversion. In the definition of detour reduction we use the notion of substituting term N for x in M, writing $M[x := N]$. Substitution in $\lambda_{\mathcal{C}}$ is defined in the standard way as it is done in (typed) lambda calculus, so we leave out the formal definition and turn directly to the definition of detour reduction.

Definition 3.3. (Detour reduction)
Consider a term of a detour convertibility as defined in Definition 2.11. Define *detour reduction* in $\lambda_{\mathcal{C}}$ as follows. It should be understood that $N_{j'}$ is in the sequence $\overline{N}$, etc.

1. $j' = l'$ for some j', l', that is, $N_{j'} : A_{j'}$ and $x_{l'} : A_{l'}$ with $A_{j'} = A_{l'}$:

$$\{\overline{N}\,;\, \overline{\lambda y.M}\} \cdot [\overline{P}\,;\, \overline{\lambda x.Q}] \longrightarrow_{\mathrm{D}} Q_{l'}[x_{l'} := N_{j'}]$$

2. $i' = k'$ for some i', k', that is, $y_{i'} : A_{i'}$ and $P_{k'} : A_{k'}$ with $A_{i'} = A_{k'}$:

$$\{\overline{N}\,;\, \overline{\lambda y.M}\} \cdot [\overline{P}\,;\, \overline{\lambda x.Q}] \longrightarrow_{\mathrm{D}} M_{i'}[y_{i'} := P_{k'}] \cdot [\overline{P}\,;\, \overline{\lambda x.Q}]$$

3. and if $P \longrightarrow_{\mathrm{D}} R$, then

   (a) $P \cdot [\overline{N}\,;\, \overline{\lambda x.O}] \longrightarrow_{\mathrm{D}} R \cdot [\overline{N}\,;\, \overline{\lambda x.O}]$
   (b) $M \cdot [\overline{N}, P, \overline{N'}\,;\, \overline{\lambda x.O}] \longrightarrow_{\mathrm{D}} M \cdot [\overline{N}, R, \overline{N'}\,;\, \overline{\lambda x.O}]$
   (c) $M \cdot [\overline{N}\,;\, \overline{\lambda x.O}, \lambda x.P, \overline{\lambda x'.O'}] \longrightarrow_{\mathrm{D}} M \cdot [\overline{N}\,;\, \overline{\lambda x.O}, \lambda x.R, \overline{\lambda x'.O'}]$
   (d) $\{\overline{N}, P, \overline{N'}\,;\, \overline{\lambda y.M}\} \longrightarrow_{\mathrm{D}} \{\overline{N}, R, \overline{N'}\,;\, \overline{\lambda y.M}\}$
   (e) $\{\overline{N}\,;\, \overline{\lambda y.M}, \lambda y.P, \overline{\lambda y'.M'}\} \longrightarrow_{\mathrm{D}} \{\overline{N}\,;\, \overline{\lambda y.M}, \lambda y.R, \overline{\lambda y'.M'}\}$

Cases (1) and (2) are the base cases of the detour reduction. In these cases, the term on the left is called the *redex*. Case (3) represents the extension of the reduction to subterms.

Example 3.4. There are four detour reductions of terms with disjunction from Example 3.2, namely

$$\begin{aligned}
\{N_B\ ;\ \lambda x.M\}_{01} \cdot_\vee [\,;\lambda x.O_1, \lambda y.O_2] &\longrightarrow_{\mathrm{D}} O_2[y := N_B] \\
\{N_A\ ;\ \lambda y.M\}_{10} \cdot_\vee [\,;\lambda x.O_1, \lambda y.O_2] &\longrightarrow_{\mathrm{D}} O_1[x := N_A] \\
\{N_A, N_B;\ \}_{11} \cdot_\vee [\,;\lambda x.O_1, \lambda y.O_2] &\longrightarrow_{\mathrm{D}} O_1[x := N_A] \\
\{N_A, N_B;\ \}_{11} \cdot_\vee [\,;\lambda x.O_1, \lambda y.O_2] &\longrightarrow_{\mathrm{D}} O_2[y := N_B]
\end{aligned}$$

The last two lines represent the detour conversion of Example 2.10, which shows that the detour reduction is non-deterministic.

Now we define the permutation reductions.

Definition 3.5. (Permutation reduction)
Consider a term of a permutation convertibility as defined in Definition 2.14. We define *permutation reduction* in $\lambda_{\mathcal{C}}$ as follows.

1. $(M \cdot [\overline{N}; \overline{\lambda x.O}]) \cdot [\overline{P}; \overline{\lambda y.Q}] \longrightarrow_{\mathrm{P}} M \cdot [\overline{N}; \overline{\lambda x.(O \cdot [\overline{P}; \overline{\lambda y.Q}])}]$
2. We extend the definition to subterms in the same way as in Definition 3.3 (3) with $\longrightarrow_{\mathrm{D}}$ replaced by $\longrightarrow_{\mathrm{P}}$.

Case (1) is the base case, where we call the left term a *redex*.

Definition 3.6. A term is in *normal form* if it contains no redex.

Due to the Curry-Howard isomorphism, normal derivations in IPC$_{\mathcal{C}}$ correspond to normal forms in $\lambda_{\mathcal{C}}$. We have the following lemma about normal forms [4].

Lemma 3.7. Term P is in normal form if one of the following holds.

1. $P = x$, where x is a variable,
2. $P = x \cdot [\overline{N}; \overline{\lambda x.O}]$ with all N_k and O_l in normal form and x a variable,
3. $P = \{\overline{N}; \overline{\lambda y.M}\}$ with all N_j and M_i in normal form.

We also summarize the main results that have been proved about weak and strong normalization in [4].

Theorem 3.8. In $\lambda_{\mathcal{C}}$ we have the following normalization results.

1. The reduction $\longrightarrow_{\mathrm{P}}$ is strongly normalizing.
2. The reduction $\longrightarrow_{\mathrm{D}}$ is strongly normalizing.
3. The union of $\longrightarrow_{\mathrm{P}}$ and $\longrightarrow_{\mathrm{D}}$ is weakly normalizing.

In the rest of this paper we will prove that the union of $\longrightarrow_{\mathrm{P}}$ and $\longrightarrow_{\mathrm{D}}$ is strongly normalizing. From the above results we will only use the simplest one, that reduction $\longrightarrow_{\mathrm{P}}$ is strongly normalizing.

We now prove that, under certain precise conditions, the union of $\longrightarrow_{\mathrm{P}}$ and $\longrightarrow_{\mathrm{D}}$ is Church Rosser, thereby proving Proposition 2.20. In the proof, we use that the union of $\longrightarrow_{\mathrm{P}}$ and $\longrightarrow_{\mathrm{D}}$ is strongly normalizing (which we will actually prove in the rest of this paper).

Proposition 3.9. Let $\mathcal{C}$ be a set of connectives and let $\longrightarrow_{\mathrm{P}}$ and $\longrightarrow_{\mathrm{D}}$ be the term-reductions defined from that following Definitions 3.3 and 3.5. These reductions satisfy the Church-Rosser property if and only if for each n-ary connective c in $\mathcal{C}$, and all $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$, we have

$$|(a_1, \ldots, a_n) - (b_1, \ldots, b_n)| \geq 2 \quad \text{implies} \quad t_c(a_1, \ldots, a_n) = t_c(b_1, \ldots, b_n).$$

Proof:
Suppose we have an n-ary connective c and $\vec{a} = (a_1, \ldots, a_n)$ and $\vec{b} = (b_1, \ldots, b_n)$ with $|\vec{a} - \vec{b}| \geq 2$ and $t_c(a_1, \ldots, a_n) \neq t_c(b_1, \ldots, b_n)$. We show that the Church-Rosser property fails. Let $1 \leq g < h \leq n$ be such that $a_g \neq b_g$ and $a_h \neq b_h$. Consider the proof-term

$$\{\overline{N} \ ; \ \overline{\lambda y.M}\} \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}] : \Phi$$

arising from an introduction of $\Phi = c(A_1, \ldots, A_n)$ and then an elimination of Φ with conclusion Φ. (So we take $D := \Phi$ in the elimination rule.) Make sure that each Q_l is a different term in normal form not containing x_l. The easiest way to do this is by letting them all be variables of the appropriate type, declared in Γ. Let the M_i also all be different variables of the appropriate type, declared in Γ, different from all the Q_l. Take the P_k and the N_k also in normal form. Now, the above term reduces in two of the ways below, depending on what a_g and a_h are exactly.

$$\begin{array}{lll}
\{\overline{N} \ ; \ \overline{\lambda y.M}\} \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}] & \longrightarrow_{\mathrm{D}} & Q_g[x_g := N_g] \\
\{\overline{N} \ ; \ \overline{\lambda y.M}\} \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}] & \longrightarrow_{\mathrm{D}} & Q_h[x_h := N_h] \\
\{\overline{N} \ ; \ \overline{\lambda y.M}\} \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}] & \longrightarrow_{\mathrm{D}} & M_g[y_g := P_g] \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}] \\
\{\overline{N} \ ; \ \overline{\lambda y.M}\} \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}] & \longrightarrow_{\mathrm{D}} & M_h[y_h := P_h] \cdot [\overline{P} \ ; \ \overline{\lambda x.Q}]
\end{array}$$

Each of these is a term in normal form, and they are all different. So Church-Rosser fails.

The other way around, we rely on standard theory from the field of Term Rewriting (see [8]). If for each connective c and all $\vec{a}$ and $\vec{b}$, with $|\vec{a} - \vec{b}| \geq 2$ we have $t_c(a_1, \ldots, a_n) = t_c(b_1, \ldots, b_n)$, then the only critical pairs arise through an overlap of $\longrightarrow_{\mathrm{D}}$ and $\longrightarrow_{\mathrm{P}}$ and an overlap of $\longrightarrow_{\mathrm{P}}$ and $\longrightarrow_{\mathrm{P}}$. Each of these is easily solvable, which yields weak-Church-Rosser, and thereby Churh-Rosser because we have strong normalization. See [8] for the details on critical pairs, weak-Church-Rosser and how it implies Church-Rosser. □

4. Parallel simply typed λ-calculus

In Section 5, we prove strong normalization of reduction (the combination of detour and permutation) in $\lambda_{\mathcal{C}}$ by generalizing the proof of strong normalization by De Groote [5] for Prawitz natural deduction.

The proof of De Groote uses a translation to simply typed lambda calculus, $\lambda^{\to}$. As the truth table natural deduction system is not confluent, we have extended $\lambda^{\to}$ with an extra term-constructor to make it possible to easily keep track of different choices. The parallel calculus makes it possible to combine different proofs of a formula. These proofs will stand parallel to each other, which justifies the name *parallel simply typed λ-calculus*. So in the present section we define and study the parallel simply typed lambda calculus, denoted by p$\lambda^{\to}$. As the simply typed lambda calculus $\lambda^{\to}$ is a very well-known system, we will not repeat its definition and the basic results about $\lambda^{\to}$, but refer to standard literature like [7, 9].

Definition 4.1. (Parallel simply typed λ-calculus)
The types in the *parallel simply typed λ-calculus* are of the form $A ::= a \mid A \to A$, where a ranges over the *atomic types*. The abstract syntax for proof-terms in the *parallel simply typed λ-calculus* is

$$M ::= x \mid (MM) \mid \lambda x.M \mid (M_1 || \ldots || M_n) \text{ for } (n > 1)$$

where x ranges over variables. The terms are typed using the following derivation rules with *context* Γ. The axiom rule, application and abstraction are the same for $\lambda^{\to}$.

$$\frac{}{\Gamma \vdash x_i : A_i} \text{ if } x_i : A_i \in \Gamma, \text{ axiom} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \text{ application}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \text{ abstraction} \qquad \frac{\Gamma \vdash M_1 : A \qquad \ldots \qquad \Gamma \vdash M_n : A}{\Gamma \vdash (M_1 || \ldots || M_n) : A} \; n > 1, \text{ parallel}$$

We use the following terminology and notations. We use capital letters A, B, C to represent types. We reserve capital letters M, N, O, P, Q for terms in p$\lambda^{\to}$. When M is of the form $(M_1 || \ldots || M_n)$, we call M a *parallel term*.

Parallel terms can be used to store all information of a proof. For instance, it may be the case that a formula can be proved in several ways, and one wants to record those different proofs.

Example 4.2. In a proof of $A \to B \to (A \to C) \to (B \to C) \to C$ one can apply the information from A or the information from B to conclude C. A 'parallel proof' that records both options would look like this, written in a notation where formulas in brackets indicate discharged assumptions.

$$\cfrac{\cfrac{\cfrac{[A \to C] \qquad [A]}{C} \qquad \cfrac{[B \to C] \qquad [B]}{C}}{C}}{A \to B \to (A \to C) \to (B \to C) \to C} \text{ 4 abstractions}$$

In IPC, this is not a valid derivation, as it combines the two sub-derivations of C in one, for which there is no rule in IPC. The proof-term that would correspond to this derivation tree in p$\lambda^{\to}$ is $M := \lambda x.\lambda y.\lambda z.\lambda w.(zx || wy)$, where we see that the two options for proving C are both recorded.

Just as in $\lambda^{\to}$, we can define substitution in the usual way and we can show that the substitution lemma holds in p$\lambda^{\to}$. The substitution lemma is proved by induction on derivations, like many other lemmas in this section.

Lemma 4.3. (Substitution lemma)
Assume that $\Gamma, x : B, \Delta \vdash M : A$ and $\Gamma \vdash N : B$, then $\Gamma, \Delta \vdash M[x := N] : A$.

For p$\lambda^{\rightarrow}$ we extend β-reduction with a rule for parallel terms.

Definition 4.4. ($\beta_{||}$-reduction)
The $\beta_{||}$-*reduction* in p$\lambda^{\rightarrow}$ is defined by adding to the well-known β-rule of $\lambda^{\rightarrow}$ the following reduction step

$$(M_1||\dots||M_n)N \rightarrow_{||} (M_1N||\dots||M_nN)$$

and extend to subterms. As usual, terms of the form $(\lambda x.M)N$ and $(M_1||\dots||M_n)N$ are $\beta_{||}$-*redexes*. If M reduces in zero, one or more steps to N, we write $M \twoheadrightarrow_{\beta_{||}} N$. This means that $\twoheadrightarrow_{\beta_{||}}$ is the reflexive and transitive closure of $\rightarrow_{\beta_{||}}$. If M reduces in one or more steps we write $M \overset{+}{\rightarrow}_{\beta_{||}} N$, which is the transitive closure of $\rightarrow_{\beta_{||}}$.

Strong normalization of p$\lambda^{\rightarrow}$ can be proved in different ways. It can be proved directly, by constructing a model using the well-known saturated sets method of Tait [10]. This is an adaptation of the well-known proof of strong normalization for $\lambda^{\rightarrow}$ and can be found in detail in [11]. Another proof proceeds by translating p$\lambda^{\rightarrow}$-terms to $\lambda^{\rightarrow}$ in a reduction preserving way, and then we conclude strong normalization for p$\lambda^{\rightarrow}$ from strong normalization for $\lambda^{\rightarrow}$. We follow this approach. To define the translation, we need enough constants, which we can just assume without a problem, as it does not affect the meta-theory.

Convention 4.5. For each atomic type a and $n \geq 1$ we assume a constant $\mathbf{c}_a^n : a \rightarrow \dots \rightarrow a \rightarrow a$, where the type contains exactly $n + 1$ occurrences of a. So $\mathbf{c}_a^n$ expects n terms of type a to produce a term of type a.

Definition 4.6. We define the following translation $F(-)$ from well-typed p$\lambda^{\rightarrow}$-terms to well-typed $\lambda^{\rightarrow}$-terms. $F(x) := x$, $F(MN) := F(M)F(N)$, $F(\lambda x.M) := \lambda x.F(M)$ and

$$F((M_1||\dots||M_n)) := \lambda x_1 \dots x_m.\mathbf{c}_a^n(F(M_1)\vec{x})\dots(F(M_n)\vec{x})$$

if $M_1 : A_1 \rightarrow \dots \rightarrow A_m \rightarrow a$.

In this definition, $\vec{x}$ denotes the vector $x_1 \dots x_m$. The definition makes use of the fact that every $\lambda^{\rightarrow}$-type is of the form $A_1 \rightarrow \dots A_m \rightarrow a$ for some atomic type a. So, the arguments to $\mathbf{c}_a^n$ are all of atomic type a.

Lemma 4.7. The translation $F(-)$ preserves types and reduction steps. That is, if $\Gamma \vdash M : A$ in p$\lambda^{\rightarrow}$, then $\Gamma \vdash F(M) : A$ in $\lambda^{\rightarrow}$ and moreover, if $M \rightarrow_{\beta_{||}} N$ in p$\lambda^{\rightarrow}$, then $F(M) \rightarrow_{\beta} F(N)$ in $\lambda^{\rightarrow}$.

Proof:
The proof of the first part is a straightforward induction on the derivation of $\Gamma \vdash M : A$ in p$\lambda^{\rightarrow}$. The second is by induction on the definition of $M \rightarrow_{\beta_{||}} N$ in p$\lambda^{\rightarrow}$ (Definition 4.4). The only interesting

case is when $M = (M_1||\dots||M_n)P \to_{\beta_{||}} (M_1 P||\dots||M_n P) = N$. Then

$$\begin{aligned} F((M_1||\dots||M_n)P) &= (\lambda y.\lambda x_1 \dots x_m.\mathbf{c}_a^n(F(M_1)y\vec{x})\dots(F(M_n)y\vec{x}))F(P) \\ &\to_\beta \lambda x_1 \dots x_m.\mathbf{c}_a^n(F(M_1)F(P)\vec{x})\dots(F(M_n)F(P)\vec{x}) = F((M_1P||\dots||M_nP)). \end{aligned}$$

□

The following theorem is an immediate consequence of Lemma 4.7 and the fact that $\lambda^{\to}$ is strongly normalizing.

Theorem 4.8. (Strong normalization for $\mathrm{p}\lambda^{\to}$**)**
In parallel simply typed lambda calculus, $\beta_{||}$-reduction is strongly normalizing.

The following proposition is checked by induction on the derivation. See [11] for the details.

Proposition 4.9. (Subject reduction property)
Parallel simply typed lambda calculus satisfies the subject reduction property. That is, if $\Gamma \vdash M : A$ and $M \to_{\beta_{||}} N$, then $\Gamma \vdash N : A$.

In $\mathrm{p}\lambda^{\to}$, one can record a number of terms of the same type as a 'tupled term', but one cannot select a term from such a tuple. Therefore, the parallel simply typed lambda calculus also satisfies the Church-Rosser property. A proof can be given by adapting the well-known method that Takahashi has developed to prove the Church-Rosser property in $\lambda^{\to}$ [12]. The proof of Church-Rosser is detailed in [11].

To translate $\lambda_{\mathcal{C}}$-terms and their reductions to $\mathrm{p}\lambda^{\to}$-terms and reductions on these, we need to consider *parallel subterms* of $\mathrm{p}\lambda^{\to}$-terms.

Definition 4.10. We write $M' \sqsubseteq M$ to denote that M' is *parallel subterm* of M. Relation $\sqsubseteq$ is inductively defined by the following rules.

1. $M \sqsubseteq M$,
2. If $N \sqsubseteq M_i$ for some i, then $N \sqsubseteq (M_1||\dots||M_n)$,
3. If $N_i \sqsubseteq M_i$ for all i, then $(N_1||\dots||N_n) \sqsubseteq (M_1||\dots||M_n)$,
4. If $P \sqsubseteq Q$, then $\lambda x.P \sqsubseteq \lambda x.Q$,
5. If $P \sqsubseteq Q$ and $M \sqsubseteq N$, then $PM \sqsubseteq QN$.

Example 4.11. Recall the proof-term $M = \lambda x.\lambda y.\lambda z.\lambda w.(zx||wy)$ in Example 4.2. It has the following parallel subterms: the term itself, $\lambda x.\lambda y.\lambda z.\lambda w.zx$ and $\lambda x.\lambda y.\lambda z.\lambda w.wy$. So note that being a parallel subterm is really different from being a subterm: zx is a subterm of M but not a parallel subterm. On the other hand, $\lambda x.\lambda y.\lambda z.\lambda w.zx$ is not a subterm of M.

Lemma 4.12. The relation $\sqsubseteq$ is a partial order, that is, $\sqsubseteq$ is reflexive, antisymmetric and transitive, and it commutes with substitution: if $M \sqsubseteq N$ and $P \sqsubseteq Q$, then $M[x := P] \sqsubseteq N[x := Q]$.

Proof:
Reflexivity is immediate and for transitivity – the main property that we will use – if $M \sqsubseteq N$ and $N \sqsubseteq P$, one proves $M \sqsubseteq P$ by induction on the definition of $N \sqsubseteq P$. Full details can be found in [11]. That $\sqsubseteq$ commutes with substitution is proved by induction on $M \sqsubseteq N$. □

The following lemma states that reduction can be extended to 'parallel superterms'.

Lemma 4.13. Let M, N and P be p$\lambda^{\rightarrow}$-terms. If $M \rightarrow_{\beta_{||}} N$ and $M \sqsubseteq P$, then there is a p$\lambda^{\rightarrow}$-term Q such that $P \xrightarrow{+}_{\beta_{||}} Q$ and $N \sqsubseteq Q$.

Proof:
The proof is by induction on the definition of $M \sqsubseteq P$. The interesting case is when $M = M_1 M_2 = (\lambda x.M')M_2$ is a $\beta_{||}$-redex itself. If $P = M$ or P is a parallel term containing M it is clear. For $P = P_1 P_2$ with $M_1 \sqsubseteq P_1$, $M_2 \sqsubseteq P_2$ and $M = (\lambda x.M')M_2 \rightarrow_{\beta_{||}} M'[x := M_2] = N$, there are two interesting cases for $M_1 \sqsubseteq P_1$.

- Case (4) in Definition 4.10. Then $P_1 = \lambda x.P'$ for some P', define $Q = P'[x := P_2]$. Then we have $P = (\lambda x.P')P_2 \rightarrow_{\beta_{||}} Q$ and $N = M'[x := M_2] \sqsubseteq Q$, because $M' \sqsubseteq P'$ and $M_2 \sqsubseteq P_2$ (and we have the substitution property in Lemma 4.12).
- Case (2) in Definition 4.10. Then $P_1 = (P'_1 || \dots || P'_n)$ such that $M_1 \sqsubseteq P'_i$ for some i. Again there are two cases. If P'_i is a parallel term, then repeat the process. Note that this eventually ends, since there are finitely many nested subterms. If $P'_i = \lambda x.P''$ with $M' \sqsubseteq P''$, then define

  $$Q = (P'_1 P_2 || \dots || P''[x := P_2] || \dots || P'_n P_2).$$

  We have $P = (P'_1 || \dots || P'_n)P_2 \rightarrow_{\beta_{||}} (P'_1 P_2 || \dots || (\lambda x.P'')P_2 || \dots || P'_n P_2) \rightarrow_{\beta_{||}} Q$. And we also have $N = M'[x := M_2] \sqsubseteq Q$, because $M' \sqsubseteq P''$ and $M_2 \sqsubseteq P_2$. □

5. Strong normalization

In [4], weak normalization of IPC$_{\mathcal{C}}$ with permutation and detour conversions is proved by studying the term reduction in $\lambda_{\mathcal{C}}$. The proof consists in proving strong normalization separately for detour reduction and for permutation reduction. The procedure for weak normalization consists of first doing a full permutation reduction and then doing a detour reduction step of 'highest rank'. This is a fairly complicated process and proof, and in [4] no proof of strong normalization for the union of detour and permutation reduction is given.

Here we present a detailed proof of strong normalization of the union of detour and permutation reduction. Our proof is based on the work of De Groote [5] who gives a detailed proof for strong normalization of Prawitz natural deduction. His proof is based on a translation from natural deduction to simply typed lambda calculus, $\lambda^{\rightarrow}$. The clue of the proof is that a permutation conversion in the Prawitz system corresponds to equality (α-equivalence) in simply typed lambda calculus and that a detour conversion corresponds to β-reduction in simply typed lambda calculus. Therefore, an infinite conversion would translate to an infinite reduction in $\lambda^{\rightarrow}$, because there cannot be infinitely many consecutive permutation conversions. As $\lambda^{\to}$ is strongly normalizing, there cannot be such an infinite conversion of derivations in Prawitz natural deduction.

In [5], it is not made explicit where the translations of connectives to $\lambda^{\to}$-types arise. So, it is not straightforward to generalize this to arbitrary connectives. A limitation is that [5] excludes the permutation conversion for the $\bot$-rule. This gives normal proofs that do not satisfy the subformula property, as illustrated in the example below. We have created a method to generalize the approach of De Groote to full IPC$_{\mathcal{C}}$ (for any set of connectives $\mathcal{C}$), including the permutation for the $\bot$-rule.

Example 5.1. Below are two examples of derivations (terms) in normal form following De Groote, but they do not have the subformula property.

$$\dfrac{\dfrac{\vdash \bot}{\vdash A \vee A}\,\bot\text{-el} \qquad A \vdash A \qquad A \vdash A}{\vdash A}\,\vee\text{-el} \qquad\qquad \dfrac{\dfrac{\vdash \neg B \qquad \vdash B}{\vdash A \vee A}\,\neg\text{-el} \qquad A \vdash A \qquad A \vdash A}{\vdash A}\,\vee\text{-el}$$

$$x : \bot \vdash\ x \cdot [;] \cdot [;\lambda p.p, \lambda q.q] : A \qquad\qquad x : \neg B, y : B\ \vdash\ x \cdot [y;] \cdot [;\lambda p.p, \lambda q.q] : A$$

This situation occurs with connectives c that have the following row in their truth table (that yields an elimination rule): $t_c(1, \ldots, 1) = 0$, which is not uncommon. These are exactly the connectives having an elimination rule with no case in the premises.

We define a translation from the truth table system to parallel simply typed lambda calculus, p$\lambda^{\to}$. In the translation, we need to preserve all possible reductions in $\lambda_{\mathcal{C}}$. This is tricky when subterms can be thrown away, which can happen in some permutation reduction rules, exactly for 'negative connectives' like $\bot$ and $\neg$, see Example 2.18. The translation of De Groote needs some special treatment when we also permute those 'negative connectives'. We therefore single out two types of permutations, which is a refinement of Definition 3.3.

Definition 5.2. (Positive and negative permutation reductions)

1. *Positive permutation reduction*: assume there is at least one case $\lambda x_l.O_l$.

$$(M \cdot [\overline{N}; \overline{\lambda x.O}]) \cdot_r [\overline{P}; \overline{\lambda y.Q}] \ \longrightarrow_{\text{Ppos}}\ M \cdot [\overline{N}; \overline{\lambda x.(O \cdot_r [\overline{P}; \overline{\lambda y.Q}])}]$$

2. *Negative permutation reduction*: in case $\overline{\lambda x.O}$ is empty

$$(M \cdot [\overline{N};\ ]) \cdot_r [\overline{P}; \overline{\lambda y.Q}] \ \longrightarrow_{\text{Pneg}}\ M \cdot [\overline{N};\ ]$$

In this format, De Groote only considers reductions corresponding to the positive permutation reduction. Our generalization to $\lambda_{\mathcal{C}}$ of the approach of De Groote [5] proves strong normalization only for detour and positive permutation reductions of the truth table system. This is Theorem 5.24.

The translation is not totally sufficient for the negative permutation reduction, which has two reasons. First, we have to slightly change the translation based on De Groote for elimination rules with no cases by the reason illustrated above. Second, this adjustment does not ensure that a negative permutation step in $\lambda_{\mathcal{C}}$ corresponds to a $\twoheadrightarrow_{\beta_{||}}$ step in p$\lambda^{\to}$. However, we have found a manner to prove strong normalization for detour and all permutation reductions using Theorem 5.24. This is our main result in Theorem 5.28.

We define a translation for types (formulas) of $\lambda_{\mathcal{C}}$ and we define a corresponding translation of terms. First we translate types. We use a negative translation of formulas. De Groote bases his negative translation on the translation induced by Plotkin's call-by-name CPS-translation [13], but in [5], it is not made explicit how the translation arises. A closer reading of [5] reveals that the translation can be derived from the elimination rules of the concerned connective. We generalize this strategy to $\lambda_{\mathcal{C}}$ and it turns out to be effective to prove strong normalization.

Definition 5.3. (Type translation)
Let o be a distinguished atomic type in p$\lambda^{\to}$. For every type A in p$\lambda^{\to}$, we denote

$$\sim A := A \to o.$$

The negative translation $\langle\!\langle \Phi \rangle\!\rangle$ of any formula Φ of IPC$_{\mathcal{C}}$ is defined inductively as follows.

- If $\Phi = A$ a proposition letter, $\langle\!\langle A \rangle\!\rangle := \sim\sim A$.
- If $\Phi = c(A_1, \dots, A_n)$ for a connective $c \in \mathcal{C}$, with elimination rules $r_1, \dots, r_t$,

  $$\langle\!\langle \Phi \rangle\!\rangle := \sim(E_1 \to \cdots \to E_t \to o),$$

  where E_s is the *elimination pattern* for rule r_s defined by

  $$E_s = \langle\!\langle A_{k_1} \rangle\!\rangle \to \cdots \to \langle\!\langle A_{k_m} \rangle\!\rangle \to \ \sim\langle\!\langle A_{l_1} \rangle\!\rangle \to \cdots \to \sim\langle\!\langle A_{l_{n-m}} \rangle\!\rangle \to o,$$

  where the A_k's are the formulas where $a_k = 1$ and the A_l's are the formulas where $a_l = 0$ in the row of the truth table t_c that corresponds with elimination rule r_s.

  There are two special cases. If there is no elimination rule for c, then $\langle\!\langle \Phi \rangle\!\rangle := \sim o$. If c is a 0-ary connective with an elimination rule, then $\langle\!\langle \Phi \rangle\!\rangle := \sim\sim o$. Note that $\bot$ is the only connective with this property.

Looking at the definition, we can make the following easy but convenient observation

$$\begin{array}{lcl} \langle\!\langle \Phi \rangle\!\rangle & = & \sim\Phi^\circ \text{ with} \\ & & \Phi^\circ \ = \ \sim\Phi \text{ if } \Phi \text{ is a propostion letter,} \\ & & \Phi^\circ \ = \ E_1 \to \cdots \to E_t \to o \text{ if } \Phi = c(A_1, \dots, A_n) \text{ for some connective } c, \\ & & \qquad\quad \text{with } E_1, \dots, E_t \text{ as in Definition 5.3.} \end{array}$$

It is also possible to translate types using the optimized rules. Optimization is defined in Lemma 2.7 and Lemma 2.8. To do this, we only consider rules optimized by Lemma 2.7, because a rule optimized by Lemma 2.8 is not in the correct form. However, this does not matter since this optimization does not reduce the number of rules and such a rule is easily rewritten back in the correct form using Lemma 2.8 itself.

Example 5.4. Conjunction $\wedge$ has the following two optimized elimination rules, as denoted in Appendix A.

$$\frac{\vdash A \wedge B}{\vdash A}\wedge\text{-el}_1 \quad \text{and} \quad \frac{\vdash A \wedge B}{\vdash B}\wedge\text{-el}_2$$

First we rewrite those rules with Lemma 2.8 to the following.

$$\frac{\vdash A \wedge B \qquad A \vdash D}{\vdash D}\wedge\text{-el}_1 \quad \text{and} \quad \frac{\vdash A \wedge B \qquad B \vdash D}{\vdash D}\wedge\text{-el}_2$$

We have $E_1 = {\sim}\langle\!\langle A\rangle\!\rangle \to o = {\sim\sim}\langle\!\langle A\rangle\!\rangle$ and $E_2 = {\sim}\langle\!\langle B\rangle\!\rangle \to o = {\sim\sim}\langle\!\langle B\rangle\!\rangle$. So

$$\langle\!\langle A \wedge B\rangle\!\rangle = {\sim}({\sim\sim}\langle\!\langle A\rangle\!\rangle \to {\sim\sim}\langle\!\langle B\rangle\!\rangle \to o).$$

This is similar to the definition of De Groote [5].

Example 5.5. Here we state the negative translation of types for connectives $\vee$, $\to$, $\neg$, $\perp$ and $\top$ using the optimized rules. See Appendix A for the optimized elimination rules. Note that some rules have to be rewritten with Lemma 2.8 before translating the type.

- Disjunction: $\langle\!\langle A \vee B\rangle\!\rangle = {\sim}(({\sim}\langle\!\langle A\rangle\!\rangle \to {\sim}\langle\!\langle B\rangle\!\rangle \to o) \to o)$
- Implication: $\langle\!\langle A \to B\rangle\!\rangle = {\sim}((\langle\!\langle A\rangle\!\rangle \to {\sim}\langle\!\langle B\rangle\!\rangle \to o) \to o)$
- Negation: $\langle\!\langle \neg A\rangle\!\rangle = {\sim}({\sim}\langle\!\langle A\rangle\!\rangle \to o) = {\sim\sim\sim}\langle\!\langle A\rangle\!\rangle$
- Bottom: $\langle\!\langle \perp\rangle\!\rangle = (o \to o) \to o = {\sim\sim}o$
- Top: $\langle\!\langle \top\rangle\!\rangle = o \to o$

In [5], the term translation is established in two steps. First a translation $\langle\!\langle M\rangle\!\rangle$ and then refining this definition to $[\![M]\!]$. We proceed in the same way of translating a term M in the truth table system to terms $\langle\!\langle M\rangle\!\rangle$ and $[\![M]\!]$ in p$\lambda^{\to}$.

Convention 5.6. We sometimes write indices i, j, k, l in sequences of terms to make clear that it is an indexed sequence, that is, we write $\overline{N_j}$ instead of $\overline{N}$ to indicate that it is indexed by j.

Definition 5.7. (Term translation $\langle\!\langle M\rangle\!\rangle$)
We define $\langle\!\langle M\rangle\!\rangle$ inductively as follows.

1. (Axiom)
$$\langle\!\langle x\rangle\!\rangle := \lambda h.xh$$

2. (Elimination) For connective c, let $r_1, \ldots, r_t$ be its elimination rules. We distinguish between elimination terms with case and without any case.

$$\langle\!\langle M \cdot_{r_s} [\overline{N_k}; \overline{\lambda x_l.O_l}]\rangle\!\rangle := \lambda h.\langle\!\langle M\rangle\!\rangle(\lambda g_1 \ldots \lambda g_t.g_s\overline{\langle\!\langle N_k\rangle\!\rangle}\,\overline{(\lambda x_l.\langle\!\langle O_l\rangle\!\rangle h)}) \text{ if there is a case } \lambda x_l.O_l$$
$$\langle\!\langle M \cdot_{r_s} [\overline{N_k};\ ]\rangle\!\rangle := \lambda h.\langle\!\langle M\rangle\!\rangle\big((\lambda p.\lambda g_1 \ldots \lambda g_t.g_s\overline{\langle\!\langle N_k\rangle\!\rangle})h\big)$$

3. (Introduction) For connective c, let $r_1, \dots, r_t$ be its elimination rules. For introduction rule r we define

$$\langle\!\langle \{\overline{N_j}; \overline{\lambda y_i.M_i}\}_r \rangle\!\rangle := \lambda h.\big((\overline{\lambda q_j}.\overline{\lambda q_i}.h)\overline{\langle\!\langle N_j\rangle\!\rangle}\,\overline{(\lambda y_i.\langle\!\langle M_i\rangle\!\rangle h)}\big)e^h_1 \dots e^h_t,$$

where $\overline{\lambda q_j}$ should be understood as a sequence of lambda abstractions corresponding to N_j's and $\overline{\lambda q_i}$ corresponding to $(\lambda y_i.\langle\!\langle M_i\rangle\!\rangle h)$'s. Term e^h_s is the possibly parallel term $(\dots)$ defined as follows: if $E_s = \langle\!\langle A_{k_1}\rangle\!\rangle \to \dots \to \langle\!\langle A_{k_m}\rangle\!\rangle \to\ \sim\!\langle\!\langle A_{l_1}\rangle\!\rangle \to \dots \to\ \sim\!\langle\!\langle A_{l_{n-m}}\rangle\!\rangle \to o$ is the elimination pattern for r_s, then e^h_s contains

- $\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}\langle\!\langle N_{j'}\rangle\!\rangle$ for all j' in rule r and l' in r_s such that $j' = l'$,
- and $\overline{\lambda h_k}.\overline{\lambda h_l}.(\lambda y_{i'}.\langle\!\langle M_{i'}\rangle\!\rangle h)h_{k'}$ for all i' in rule r and k' in r_s such that $i' = k'$.

Here, $\overline{\lambda h_k}$ quantifies over all $\langle\!\langle A_k\rangle\!\rangle$ in E_s and $\overline{\lambda h_l}$ over all $\sim\!\langle\!\langle A_l\rangle\!\rangle$. In the proof of Proposition 5.12, we see that the terms e^h_s are well-defined of type E_s, by checking the types of its components.

In the definition there are a lot of *dummy redexes*: redexes where the abstracted variable does not occur in the body and the argument gets thrown away by a β-reduction step. In the elimination term $\langle\!\langle M \cdot_{r_s} [\overline{N_k};\ ]\rangle\!\rangle$ redex $(\lambda p.\dots)h$ is a dummy redex. In the introduction term we have for each N_j and M_i a dummy redex $(\lambda q_j \dots)\langle\!\langle N_j\rangle\!\rangle$ and $(\lambda q_i \dots)(\lambda y_i.\langle\!\langle M_i\rangle\!\rangle h)$. These are necessary in order to preserve all subterms in the translation. This is very important, because reduction can also take place in those subterms. In the elimination term we need to preserve h. In the introduction rule we make sure that each subterm N_j and M_i is preserved in the translation, as they need not be in the e^h_s terms.

In the introduction terms we use parallel terms. This is a new idea which is not present in the approach of De Groote, but it is very useful, because detour reduction in the truth table system is non-deterministic. Using parallel terms, we store each choice of a detour step in the subterm e^h_s. So each possible detour reduction in term M is captured by a $\beta_{||}$-reduction in the translated p$\lambda^{\to}$-term $\langle\!\langle M\rangle\!\rangle$.

Example 5.8. We give a term translation for the non-optimized rules of $\vee$. See Example 3.2 for the typing rules. There is one elimination term and there are three introduction terms of $\vee$. Here we write down the translations of $\vee$-el$_{00}$ and $\vee$-in$_{11}$.

$$\langle\!\langle M \cdot_{\vee} [\,;\lambda x.O_1, \lambda y.O_2]\rangle\!\rangle = \lambda h.\langle\!\langle M\rangle\!\rangle(\lambda g_1.g_1(\lambda x.\langle\!\langle O_1\rangle\!\rangle h)(\lambda y.\langle\!\langle O_2\rangle\!\rangle h))$$
$$\langle\!\langle \{N_A, N_B;\ \}_{11}\rangle\!\rangle = \lambda h.\Big((\lambda p.\lambda q.h)\langle\!\langle N_A\rangle\!\rangle\langle\!\langle N_B\rangle\!\rangle\Big)e_1,$$

with $e_1 = (\lambda h_1.\lambda h_2.(h_1\langle\!\langle N_A\rangle\!\rangle) \mid\mid \lambda h_1.\lambda h_2.(h_2\langle\!\langle N_B\rangle\!\rangle))$.

The last line illustrates the use of the parallel terms. From Example 3.4, we know that there is a choice in the detour reduction with rule $\vee$-in$_{11}$.

The term translation can also be applied to the optimized rules from Lemma 2.7 and Lemma 2.8. When an elimination term is optimized by Lemma 2.8, the term translation has to be slightly modified because such a term has a different form. This proceeds in the following way. A rule can be optimized by Lemma 2.8, only when it has exactly one case, say $x : A \vdash O : D$. Then the term $(\lambda x.\langle\!\langle O\rangle\!\rangle h)$ in the elimination translation is replaced by $(\lambda f.fh)$ where $f : \langle\!\langle A\rangle\!\rangle$.

Example 5.9. We give the term translations of the optimized rules for conjunction, $\wedge$, where we use the type translation in Example 5.4. We have the following optimized rules from Appendix A with the corresponding terms.

$$\frac{\vdash M : A \wedge B}{\vdash M \cdot_{\text{el1}} [\,;\,] : A}\ \wedge\text{-el}_1\ ,\quad \frac{\vdash M : A \wedge B}{\vdash M \cdot_{\text{el2}} [\,;\,] : B}\ \wedge\text{-el}_2 \quad \text{and} \quad \frac{\vdash N_A : A \qquad \vdash N_B : B}{\vdash \{N_A, N_B;\ \}_{\text{in}} : A \wedge B}\ \wedge\text{-in}$$

The p$\lambda^{\to}$-terms after term translation are as follows. We omit the dummy redexes in the introduction term, because each subterm is already translated by a matching case $j' = l'$.

$$\begin{aligned}
&⟪M \cdot_{\text{el1}} [\,;\,]⟫ = \lambda h.⟪M⟫(\lambda g_1.\lambda g_2.g_1(\lambda f.fh)),\\
&⟪M \cdot_{\text{el2}} [\,;\,]⟫ = \lambda h.⟪M⟫(\lambda g_1.\lambda g_2.g_2(\lambda f.fh)),\\
&⟪\{N_A, N_B;\ \}_{\text{in}}⟫ = \lambda h.h(\lambda h_1.h_1⟪N_A⟫)(\lambda h_2.h_2⟪N_B⟫).
\end{aligned}$$

The detour reductions for the optimized rules for $\wedge$ are deterministic, so there is no parallel term in the introduction rule.

Example 5.10. We continue Example 5.5. We give the term translations of the connectives $\vee, \to, \neg$, $\bot$ and $\top$ with the optimized rules. See Appendix A for the rules. In each introduction rule we can omit the dummy redexes, because all information is preserved in the e_s^h terms.

- *Disjunction:* Let $M : A \vee B$, $N_A : A$ and $N_B : B$. Then

$$\begin{aligned}
&⟪M \cdot_{\text{el}} [\,; \lambda x_A.O_1, \lambda x_B.O_2]⟫ = \lambda h.⟪M⟫(\lambda g_1.g_1(\lambda x_A.⟪O_1⟫h)(\lambda x_B.⟪O_2⟫h)),\\
&⟪\{N_A;\ \}_{\text{in1}}⟫ = \lambda h.h(\lambda h_1 \lambda h_2.h_1⟪N_A⟫),\\
&⟪\{N_B;\ \}_{\text{in2}}⟫ = \lambda h.h(\lambda h_1 \lambda h_2.h_2⟪N_B⟫).
\end{aligned}$$

- *Implication:* Let $M : A \to B$, $N_A : A$ and $N_B : B$. Then

$$\begin{aligned}
&⟪M \cdot_{\text{el}} [N_A;\ ]⟫ = \lambda h.⟪M⟫(\lambda g_1.g_1⟪N_A⟫(\lambda f.fh)),\\
&⟪\{N_B;\ \}_{\text{in1}}⟫ = \lambda h.h(\lambda h_1 \lambda h_2.h_2⟪N_B⟫),\\
&⟪\{\,; \lambda y_A.M\}_{\text{in2}}⟫ = \lambda h.h(\lambda h_1 \lambda h_2.(\lambda y_A.⟪M⟫h)h_1).
\end{aligned}$$

- *Negation:* Let $M : \neg A$ and $N_A : A$. Then

$$\begin{aligned}
&⟪M \cdot_{\text{el}} [N_A;\ ]⟫ = \lambda h.⟪M⟫\big(\lambda p.(\lambda g_1.g_1⟪N_A⟫)h\big),\\
&⟪\{\,; \lambda y_A.M\}_{\text{in}}⟫ = \lambda h.h(\lambda h_1.(\lambda y_A.⟪M⟫h)h_1).
\end{aligned}$$

- *Bottom:* Let $M :\bot$. Then $⟪M \cdot_{\text{el}} [\,;\,]⟫ = \lambda h.⟪M⟫\big(\lambda p.(\lambda g_1.g_1)h\big)$.
- *Top:* $⟪\{\,;\,\}_{\text{in}}⟫ = \lambda h.h$.

For the optimized rules for those connectives, the detour reduction is deterministic. This means that there is no parallel term in any of these introduction terms.

It can be shown that translations of types and translations of terms commute with the typing relation, that is, if $M : A$ in $\lambda_{\mathcal{C}}$, then $⟪M⟫ : ⟪A⟫$ in p$\lambda^{\to}$. This is true for non-optimized rules as well as for optimized rule. See Proposition 5.12 for the proof for non-optimized rules.

Example 5.11. We continue Example 5.9. Here we show the correctness of the translation for the optimized rules $\wedge$-el$_1$ and $\wedge$-in. We see that the translation commutes with the typing relation in the case of those rules. First, recall the rules:

$$\frac{\vdash M : A \wedge B}{\vdash M \cdot_{\text{el1}} [\,;\,] : A}\,\wedge\text{-el}_1 \quad \text{and} \quad \frac{\vdash N_A : A \qquad \vdash N_B : B}{\vdash \{N_A, N_B;\,\}_{\text{in}} : A \wedge B}\,\wedge\text{-in}$$

The following show the translated derivations where expressions between brackets are used in abstractions:

$$\dfrac{\langle\!\langle M\rangle\!\rangle : \langle\!\langle A \wedge B\rangle\!\rangle \qquad \dfrac{\dfrac{\dfrac{[g_1 : {\sim}{\sim}\langle\!\langle A\rangle\!\rangle] \qquad \dfrac{\dfrac{[f : \langle\!\langle A\rangle\!\rangle] \qquad [h : A^\circ]}{fh : o}}{\lambda f.fh : {\sim}\langle\!\langle A\rangle\!\rangle}}{g_1(\lambda f.fh) : o}}{\lambda g_2.g_1(\lambda f.fh) : {\sim}{\sim}\langle\!\langle B\rangle\!\rangle \to o}}{\lambda g_1.\lambda g_2.g_1(\lambda f.fh) : {\sim}{\sim}\langle\!\langle A\rangle\!\rangle \to {\sim}{\sim}\langle\!\langle B\rangle\!\rangle \to o}}{\dfrac{\langle\!\langle M\rangle\!\rangle(\lambda g_1.\lambda g_2.g_1(\lambda f.fh)) : o}{\lambda h.\langle\!\langle M\rangle\!\rangle(\lambda g_1.\lambda g_2.g_1(\lambda f.fh)) : \langle\!\langle A\rangle\!\rangle}}$$

and

$$\dfrac{\dfrac{\dfrac{[h : {\sim}{\sim}\langle\!\langle A\rangle\!\rangle \to {\sim}{\sim}\langle\!\langle B\rangle\!\rangle \to o] \qquad \dfrac{\dfrac{[h_1 : {\sim}\langle\!\langle A\rangle\!\rangle] \qquad \langle\!\langle N_A\rangle\!\rangle : \langle\!\langle A\rangle\!\rangle}{h_1\langle\!\langle N_A\rangle\!\rangle : o}}{\lambda h_1.h_1\langle\!\langle N_A\rangle\!\rangle : {\sim}{\sim}\langle\!\langle A\rangle\!\rangle}}{h(\lambda h_1.h_1\langle\!\langle N_A\rangle\!\rangle) : {\sim}{\sim}\langle\!\langle B\rangle\!\rangle \to o} \qquad \dfrac{\dfrac{[h_2 : {\sim}\langle\!\langle B\rangle\!\rangle] \qquad \langle\!\langle N_B\rangle\!\rangle : \langle\!\langle B\rangle\!\rangle}{h_2\langle\!\langle N_B\rangle\!\rangle : o}}{\lambda h_2.h_2\langle\!\langle N_B\rangle\!\rangle : {\sim}{\sim}\langle\!\langle B\rangle\!\rangle}}{h(\lambda h_1.h_1\langle\!\langle N_A\rangle\!\rangle)(\lambda h_2.h_2\langle\!\langle N_B\rangle\!\rangle) : o}}{\lambda h.h(\lambda h_1.h_1\langle\!\langle N_A\rangle\!\rangle)(\lambda h_2.h_2\langle\!\langle N_B\rangle\!\rangle) : \langle\!\langle A \wedge B\rangle\!\rangle}$$

In the following, we write $\langle\!\langle\Gamma\rangle\!\rangle = x_1 : \langle\!\langle B_1\rangle\!\rangle, \ldots, x_n : \langle\!\langle B_n\rangle\!\rangle$ when $\Gamma = x_1 {:} B_1, \ldots, x_n {:} B_n$.

Proposition 5.12. If $\Gamma \vdash M : A$ in λ_C, then $\langle\!\langle\Gamma\rangle\!\rangle \vdash \langle\!\langle M\rangle\!\rangle : \langle\!\langle A\rangle\!\rangle$ in parallel simply typed lambda calculus.

Proof:
The proof is by induction on the derivation of $\Gamma \vdash M : A$.

1. The (Axiom) rule is straightforward
2. (Elimination) We consider $\vdash M \cdot_{r_s} [\overline{N_k}; \overline{\lambda x_l.O_l}] : D$, which is derived with elimination rule r_s with premises $\vdash M : \Phi$, $\vdash N_k : A_k$ and $x_l : A_l \vdash O_l : D$, where $\Phi = c(A_1, \ldots A_n)$ and c has t elimination rules. We want to prove that $\vdash \langle\!\langle M \cdot_{r_s} [\overline{N_k}; \overline{\lambda x_l.O_l}]\rangle\!\rangle : \langle\!\langle D\rangle\!\rangle$. Recall that $\langle\!\langle\Phi\rangle\!\rangle = {\sim}(E_1 \to \cdots \to E_t \to o)$ with elimination patterns

$$E_u = \langle\!\langle A_{k_1}\rangle\!\rangle \to \cdots \to \langle\!\langle A_{k_m}\rangle\!\rangle \to {\sim}\langle\!\langle A_{l_1}\rangle\!\rangle \to \cdots \to {\sim}\langle\!\langle A_{l_{n-m}}\rangle\!\rangle \to o,$$

where the A_k's are the formulas where $a_k = 1$ and the A_l's are the formulas where $a_l = 0$ in the truth table t_c of the corresponding elimination rule r_u.

The added dummy redex in an elimination term where $\overline{\lambda x_l.O_l}$ is empty does not influence the type, so we can consider the situation where we have at least one case $\lambda x_l.O_l$. Induction hypotheses are $\vdash \langle\!\langle M\rangle\!\rangle : \langle\!\langle \Phi\rangle\!\rangle$, $\vdash \langle\!\langle N_k\rangle\!\rangle : \langle\!\langle A_k\rangle\!\rangle$ and $x_l : \langle\!\langle A_l\rangle\!\rangle \vdash \langle\!\langle O_l\rangle\!\rangle : \langle\!\langle D\rangle\!\rangle$ for every k and l. We have the following derivation (where we have omitted some terms). Recall that $\langle\!\langle D\rangle\!\rangle = {\sim}D^\circ$.

$$\dfrac{\langle\!\langle M\rangle\!\rangle : \langle\!\langle \Phi\rangle\!\rangle \qquad \begin{matrix}\dfrac{\dfrac{[g_s : E_s] \quad \dots\langle\!\langle N_k\rangle\!\rangle : \langle\!\langle A_k\rangle\!\rangle\dots \qquad \dots\ \dfrac{\dfrac{x_l : \langle\!\langle A_l\rangle\!\rangle \vdash \langle\!\langle O_l\rangle\!\rangle : \langle\!\langle D\rangle\!\rangle \qquad [h : D^\circ]}{o}}{\lambda x_l.\langle\!\langle O_l\rangle\!\rangle h \ : \ {\sim}\langle\!\langle A_l\rangle\!\rangle}\ \dots}{g_s\overline{\langle\!\langle N_k\rangle\!\rangle}(\overline{\lambda x_l.\langle\!\langle O_l\rangle\!\rangle h}) : o}}{E_t \to o} \\ \vdots \\ E_1 \to \dots \to E_t \to o\end{matrix}}{\dfrac{o}{\lambda h.\langle\!\langle M\rangle\!\rangle(\lambda g_1 \dots \lambda g_t.g_s\overline{\langle\!\langle N_k\rangle\!\rangle}(\overline{\lambda x_l.\langle\!\langle O_l\rangle\!\rangle h})) : \langle\!\langle D\rangle\!\rangle}}$$

We conclude that indeed $\lambda h.\langle\!\langle M\rangle\!\rangle(\lambda g_1 \dots \lambda g_t.g_s\overline{\langle\!\langle N_k\rangle\!\rangle}(\overline{\lambda x_l.\langle\!\langle O_l\rangle\!\rangle h}))$ has type $\langle\!\langle D\rangle\!\rangle$.

3. (Introduction) We consider $\vdash \{\overline{N_j}; \overline{\lambda x_i.M_i}\}_r : \Phi$ which is derived from introduction rule r with premises $\vdash N_j : A_j$ and $x_i : A_i \vdash M_i : \Phi$, where $\Phi = c(A_1, \dots A_n)$ and c has t elimination rules. Dummy redexes do not influence the type, so we consider the situation without these redexes. So we want to prove that $\lambda h.he^h_1 \dots e^h_t : \langle\!\langle \Phi\rangle\!\rangle$. Induction gives $\vdash \langle\!\langle N_j\rangle\!\rangle : \langle\!\langle A_j\rangle\!\rangle$ and $x_i : \langle\!\langle A_i\rangle\!\rangle \vdash \langle\!\langle M_i\rangle\!\rangle : \langle\!\langle \Phi\rangle\!\rangle$ for all j, i. First we prove for every $s \le t$ that

$$E_s = \langle\!\langle A_{k_1}\rangle\!\rangle \to \dots \to \langle\!\langle A_{k_m}\rangle\!\rangle \to \ {\sim}\langle\!\langle A_{l_1}\rangle\!\rangle \to \dots \to {\sim}\langle\!\langle A_{l_{n-m}}\rangle\!\rangle \to o,$$

is inhabited by the term e^h_s with $h : \Phi^\circ = E_1 \to \dots \to E_t \to o$. (Recall that $\langle\!\langle \Phi\rangle\!\rangle = {\sim}\Phi^\circ$.) Term e^h_s can be a single or a parallel term $(\dots)$ with elements that belong to one of the following cases. Note that e^h_s always exists, since there is at least one 'matching case'.

- $j' = l'$ for some j' in introduction rule r and some l' in elimination rule r_s:

$$\dfrac{\dfrac{[h_{l'} : \ {\sim}\langle\!\langle A_{l'}\rangle\!\rangle] \qquad \langle\!\langle N_{j'}\rangle\!\rangle : \langle\!\langle A_{j'}\rangle\!\rangle}{h_{l'}\langle\!\langle N_{j'}\rangle\!\rangle : o}}{\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}\langle\!\langle N_{j'}\rangle\!\rangle : \langle\!\langle A_{k_1}\rangle\!\rangle \to \dots \to \langle\!\langle A_{k_m}\rangle\!\rangle \to \ {\sim}\langle\!\langle A_{l_1}\rangle\!\rangle \to \dots \to {\sim}\langle\!\langle A_{l_{n-m}}\rangle\!\rangle \to o}$$

  So we conclude that such an element in e^h_s has type E_s.
- $i' = k'$ for some i' in introduction rule r and some k' in elimination rule r_s:

$$\dfrac{\dfrac{\dfrac{\dfrac{y_{i'} : \langle\!\langle A_{i'}\rangle\!\rangle \vdash \langle\!\langle M_{i'}\rangle\!\rangle : \langle\!\langle \Phi\rangle\!\rangle \qquad h : \Phi^\circ}{\langle\!\langle M_{i'}\rangle\!\rangle h : o}}{\lambda y_{i'}.\langle\!\langle M_{i'}\rangle\!\rangle h : \ {\sim}\langle\!\langle A_{i'}\rangle\!\rangle} \qquad [h_{k'} : \langle\!\langle A_{k'}\rangle\!\rangle]}{o}}{\overline{\lambda h_k}.\overline{\lambda h_l}.(\lambda y_{i'}.\langle\!\langle M_{i'}\rangle\!\rangle h)h_{k'} : \langle\!\langle A_{k_1}\rangle\!\rangle \to \dots \to \langle\!\langle A_{k_m}\rangle\!\rangle \to \ {\sim}\langle\!\langle A_{l_1}\rangle\!\rangle \to \dots \to {\sim}\langle\!\langle A_{l_{n-m}}\rangle\!\rangle \to o}$$

  We conclude that in this matching case the element in e^h_s has also type E_s.

Each element in the possibly parallel term e^h_s has type E_s, so e^h_s is well-defined with type E_s. We conclude that $\lambda h.he^h_1, \dots, e^h_t$ has type $\langle\!\langle \Phi\rangle\!\rangle$, since h has type Φ°. □

Just as in [5], the translation $\langle\!\langle M \rangle\!\rangle$ of Definition 5.7 has to be modified to another translation which we denote by $\llbracket M \rrbracket$. The problem is that permutation reductions in M do not correspond to $\beta_{||}$-reductions in $\langle\!\langle M \rangle\!\rangle$. Instead, we have the following situation, for some p$\lambda^{\to}$-term R.

$$\begin{tikzcd}
M \arrow[r] \arrow[d, "P"] & \langle\!\langle M \rangle\!\rangle \arrow[dr, "+", "\beta_{||}"'] & \\
N \arrow[r] & \langle\!\langle N \rangle\!\rangle \arrow[r, "+", "\beta_{||}"'] & R
\end{tikzcd}$$

This is not sufficient for translating an infinite reduction in $\lambda_{\mathcal{C}}$ to an infinite $\beta_{||}$-reduction in p$\lambda^{\to}$. Therefore, following De Groote [5], we define a modified translation $\llbracket M \rrbracket$ that improves on the situation described above. The idea of the modified translation is to perform specific additional $\beta_{||}$-reductions in the translation $\llbracket M \rrbracket$. This only works for positive permutation reductions. With the new translation $\llbracket \cdot \rrbracket$, we have the following diagrams.

$$\begin{tikzcd}
M \arrow[r] \arrow[d, "\mathrm{D}"] & \llbracket M \rrbracket \sqsubseteq & K \arrow[d, "+", "\beta_{||}"'] \\
N \arrow[r] & \llbracket N \rrbracket \sqsubseteq & \exists K'
\end{tikzcd}
\qquad
\begin{tikzcd}
M \arrow[r] \arrow[d, "\mathrm{Ppos}"] & \llbracket M \rrbracket \arrow[d, phantom, "||"] \\
N \arrow[r] & \llbracket N \rrbracket
\end{tikzcd}
\qquad
\begin{tikzcd}
M \arrow[r] \arrow[d, "\mathrm{Pneg}"] & \llbracket M \rrbracket \arrow[dr, "+", "\beta_{||}"'] & \\
N \arrow[r] & \llbracket N \rrbracket \arrow[r, "+", "\beta_{||}"'] & R
\end{tikzcd}$$

The left diagram is proved in Proposition 5.22 and the diagram in the middle is Proposition 5.17. This yields strong normalization for $\longrightarrow_{\mathrm{D}} \cup \longrightarrow_{\mathrm{Ppos}}$. After that, we consider the addition of negative permutation reductions separately.

Definition 5.13. (Modified term translation $\llbracket M \rrbracket$)
For every term M in $\lambda_{\mathcal{C}}$, we define term $\llbracket M \rrbracket$ in parallel simply typed lambda calculus by

$$\llbracket M \rrbracket = \lambda h.(M : h),$$

where h is a fresh variable and where the operator : is defined as follows (not to be confused with the typing relation).

1. (Axiom) $x : H := xH$.
2. (Elimination) For connective c, let $r_1, \ldots, r_t$ be its elimination rules. We distinguish between elimination terms with case and without case.

$$M \cdot_{r_s} [\overline{N_k}; \overline{\lambda x_l.O_l}] : H := M : \big(\lambda g_1 \ldots \lambda g_t.g_s \overline{\llbracket N_k \rrbracket}\, \overline{(\lambda x_l.(O_l : H))}\big) \text{ if there is a case } \lambda x_l.O_l$$
$$M \cdot_{r_s} [\overline{N_k};\ ] : H := M : \big((\lambda p.\lambda g_1 \ldots \lambda g_t.g_s \overline{\llbracket N_k \rrbracket})H\big)$$

3. (Introduction) For connective c, let $r_1, \ldots, r_t$ be its elimination rules. For introduction rule r we define

$$\{\overline{N_j}; \overline{\lambda y_i.M_i}\}_r : H := \big((\overline{\lambda q_j}.\overline{\lambda q_i}.H)\overline{\llbracket N_j \rrbracket}\,\overline{(\lambda y_i.(M_i : H))}\big)\llbracket e_1^H \rrbracket \ldots \llbracket e_t^H \rrbracket,$$

where $\overline{\lambda q_j}$ should be understood as a sequence of dummy lambda abstractions corresponding to N_j's and $\overline{\lambda q_i}$ correspond to $\lambda y_i.(M_i : H)$'s. Term $[\![e_s^H]\!]$ is the possibly parallel term $(\dots)$ defined as follows: if $E_s = \langle\!\langle A_{k_1}\rangle\!\rangle \to \dots \to \langle\!\langle A_{k_m}\rangle\!\rangle \to \ {\sim}\langle\!\langle A_{l_1}\rangle\!\rangle \to \dots \to {\sim}\langle\!\langle A_{l_{n-m}}\rangle\!\rangle \to o$ is the elimination pattern for r_s, then $[\![e_s^H]\!]$ contains

- $\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}[\![N_{j'}]\!]$ for all j' in r and l' in r_s such that $j' = l'$,
- and $\overline{\lambda h_k}.\overline{\lambda h_l}.(\lambda y_{i'}.(M_{i'} : H))h_{k'}$ for all i' in r and k' in r_s such that $i' = k'$.

Here, $\overline{\lambda h_k}$ quantifies over all $\langle\!\langle A_k\rangle\!\rangle$ in E_s and $\overline{\lambda h_l}$ over all ${\sim}\langle\!\langle A_l\rangle\!\rangle$.

Example 5.14. In Example 5.8, we gave the non-modified translations of terms corresponding to the two rules for disjunction $\vee$-el$_{00}$ and $\vee$-in$_{11}$. Here we consider a detour redex of those rules and show the difference between the non-modified translation of Definition 5.7 and the modified translation of Definition 5.13.

$$\begin{aligned}&\langle\!\langle \{N_A, N_B, ;\ \}_{11} \cdot_\vee [\ ; \lambda x.O_1, \lambda y.O_2]\rangle\!\rangle \\ &\qquad = \lambda h.\Big(\lambda h'.\big((\lambda p.\lambda q.h')\langle\!\langle N_A\rangle\!\rangle\langle\!\langle N_B\rangle\!\rangle\big)e_1\Big)(\lambda g_1.g_1(\lambda x.\langle\!\langle O_1\rangle\!\rangle h)(\lambda y.\langle\!\langle O_2\rangle\!\rangle h))\end{aligned}$$

with $e_1 = (\lambda h_1.\lambda h_2.(h_1\langle\!\langle N_A\rangle\!\rangle) \mid\mid \lambda h_1.\lambda h_2.(h_2\langle\!\langle N_B\rangle\!\rangle))$ and

$$\begin{aligned}&[\![\{N_A, N_B, ;\ \}_{11} \cdot_\vee [\ ; \lambda x.O_1, \lambda y.O_2]]\!] \\ &\qquad = \lambda h.\Big(\big(\lambda p.\lambda q.(\lambda g_1.g_1(\lambda x.(O_1 : h))(\lambda y.(O_2 : h)))[\![N_A]\!][\![N_B]\!]\Big)[\![e_1]\!]\end{aligned}$$

with $[\![e_1]\!] = (\lambda h_1.\lambda h_2.(h_1[\![N_A]\!]) \mid\mid \lambda h_1.\lambda h_2.(h_2[\![N_B]\!]))$. We see that the second one is somehow related to the first one by a β-reduction on $\lambda h'$.

We have defined the type translation in Definition 5.3 and the modified term translation in Definition 5.13. We have shown that the first term translation commutes with the typing relation. Now we have to show that this also holds for the modified term translation. This relies on the following lemma.

Lemma 5.15. Let M be a term in $\lambda_{\mathcal{C}}$ and let H be a p$\lambda^{\to}$-term, then:

1. $\langle\!\langle M\rangle\!\rangle \twoheadrightarrow_{\beta_{||}} [\![M]\!]$,
2. $\langle\!\langle M\rangle\!\rangle H \twoheadrightarrow_{\beta_{||}} M : H$.

Proof:
These statements are proved simultaneously by induction on the structure of M. We only look at the introduction case $\{\overline{N_j}\,;\overline{\lambda y_i.M_i}\}$, as the other cases are similar. Let c be a connective with introduction rule r and elimination rules $r_1, \dots, r_t$. In this proof we use the following induction hypotheses for all j and i:

$$\langle\!\langle N_j\rangle\!\rangle \twoheadrightarrow_{\beta_{||}} [\![N_j]\!] \text{ and } \langle\!\langle M_i\rangle\!\rangle h \twoheadrightarrow_{\beta_{||}} M_i : h.$$

First we look at the p$\lambda^{\to}$-terms $e^h_s = (\dots)$ and $[\![e^h_s]\!] = (\dots)$ in the definitions of the normal term translation (Definition 5.7) and the modified translation (Definition 5.13) for an introduction term. If $\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}\langle\!\langle N_{j'}\rangle\!\rangle$ is in the parallel term e^h_s, then

$$\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}\langle\!\langle N_{j'}\rangle\!\rangle \twoheadrightarrow_{\beta_{||}} \overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}[\![N_{j'}]\!]$$

is in the parallel term $[\![e^h_s]\!]$. And if $\overline{\lambda h_k}.\overline{\lambda h_l}.(\lambda y_{i'}.\langle\!\langle M_{i'}\rangle\!\rangle h)h_{k'}$ is included in e^h_s, then

$$\overline{\lambda h_k}.\overline{\lambda h_l}.(\lambda y_{i'}.\langle\!\langle M_{i'}\rangle\!\rangle h)h_{k'} \twoheadrightarrow_{\beta_{||}} \overline{\lambda h_k}.\overline{\lambda h_l}.(\lambda y_{i'}.(M_{i'} : h))h_{k'}$$

is in the parallel term $[\![e^h_s]\!]$. This means that $e^h_s \twoheadrightarrow_{\beta_{||}} [\![e^h_s]\!]$ for all s. Now we can conclude that

$$\begin{aligned}\langle\!\langle\{\overline{N_j}; \overline{\lambda y_i.M_i}\}\rangle\!\rangle &= \lambda h.\big((\overline{\lambda q_j}.\overline{\lambda q_i}.h)\overline{\langle\!\langle N_j\rangle\!\rangle}\,\overline{(\lambda y_i.\langle\!\langle M_i\rangle\!\rangle h)}\big)e^h_1\dots e^h_t\\ &\twoheadrightarrow_{\beta_{||}} \lambda h.\big((\overline{\lambda q_j}.\overline{\lambda q_i}.h)\overline{[\![N_j]\!]}\,\overline{(\lambda y_i.(M_i : h))}\big)[\![e^h_1]\!]\dots[\![e^h_t]\!] = [\![\{\overline{N_j}; \overline{\lambda y_i.M_i}\}]\!]\end{aligned}$$

and

$$\langle\!\langle\{\overline{N_j}; \overline{\lambda y_i.M_i}\}\rangle\!\rangle H \twoheadrightarrow_{\beta_{||}} \{\overline{N_j}; \overline{\lambda y_i.M_i}\} : H.$$

□

Proposition 5.16. If $\Gamma \vdash M : A$ in $\lambda_{\mathcal{C}}$, then $\langle\!\langle\Gamma\rangle\!\rangle \vdash [\![M]\!] : \langle\!\langle A\rangle\!\rangle$ in parallel simply typed lambda calculus p$\lambda^{\to}$.

Proof:
This follows from Proposition 5.12, Lemma 5.15, and the subject reduction property of the parallel simply typed lambda calculus (Proposition 4.9). □

Now we see that positive permutation reductions correspond to syntactic equality.

Proposition 5.17. Let M and N be terms in $\lambda_{\mathcal{C}}$ such that $M \longrightarrow_{\text{Ppos}} N$. Then

1. $M : H = N : H$, for all p$\lambda^{\to}$-terms H,
2. $[\![M]\!] = [\![N]\!]$.

Proof:
Statement (2) is a direct consequence of (1). For (1), we proceed by induction on the generation of $M \longrightarrow_{\text{Ppos}} N$. We only treat the base case, since the induction steps are easily verified.

We consider the permutation convertibility where there is at least one case of the form $\lambda x.O$:
$(M \cdot [\overline{N}; \overline{\lambda x.O}]) \cdot_{r_s} [\overline{P}; \overline{\lambda y.Q}] \longrightarrow_{\text{Ppos}} M \cdot [\overline{N}; \overline{\lambda x.(O \cdot_{r_s} [\overline{P}; \overline{\lambda y.Q}])}]$.
Write $L = \big(\lambda g_1 \dots \lambda g_t.g_s\overline{[\![P]\!]}\,\overline{(\lambda y.(Q : H))}\big)$, then

$$\begin{aligned}&(M\cdot[\overline{N}; \overline{\lambda x.O}]) \cdot_{r_s} [\overline{P}; \overline{\lambda y.Q}] : H\\ &\quad= (M \cdot [\overline{N}; \overline{\lambda x.O}]) : L\\ &\quad= M : \big(\lambda g_1 \dots \lambda g_{t'}.g_{s'}\overline{[\![N]\!]}\,\overline{(\lambda x.(O : L))}\big)\\ &\quad= M : \big(\lambda g_1 \dots \lambda g_{t'}.g_{s'}\overline{[\![N]\!]}\,\overline{(\lambda x.((O \cdot_{r_s} [\overline{P}; \overline{\lambda y.Q}]) : H))}\big)\\ &\quad= (M \cdot [\overline{N}; \overline{\lambda x.(O \cdot_{r_s} [\overline{P}; \overline{\lambda y.Q}])}]) : H\end{aligned}$$

□

The following lemmas are useful to prove that detour reduction in the truth table system corresponds to $\beta_{||}$-reduction in p$\lambda^{\rightarrow}$ which we establish in Proposition 5.22.

Lemma 5.18. Let M and P be terms in $\lambda_{\mathcal{C}}$ and H be a p$\lambda^{\rightarrow}$-term in which there is no free occurrence of z, then:

1. $(M : H)[z := [\![P]\!]] \twoheadrightarrow_{\beta_{||}} (M[z := P]) : H$,
2. $[\![M]\!][z := [\![P]\!]] \twoheadrightarrow_{\beta_{||}} [\![M[z := P]]\!]$.

Proof:
Property (2) is a direct consequence of (1). The proof of (1) is by induction on the structure of M. See [11] for the details. □

Next lemma is based on Lemma 14 of De Groote [5].

Lemma 5.19. Let H and L be p$\lambda^{\rightarrow}$-terms such that $H \xrightarrow{+}_{\beta_{||}} L$. Then, for any term M in $\lambda_{\mathcal{C}}$, we have $M : H \xrightarrow{+}_{\beta_{||}} M : L$.

Proof:
By induction on M. See [11] for the details. □

Recall the definition of parallel subterm of Definition 4.10, $K \sqsubseteq L$. Since we are working in the parallel simply typed lambda calculus, we need the following lemma.

Lemma 5.20. Let H and L be p$\lambda^{\rightarrow}$-terms, such that $H \sqsubseteq L$. Then, for any term M in $\lambda_{\mathcal{C}}$, we have $(M : H) \sqsubseteq (M : L)$.

Proof:
This is proved by induction on the structure of M. See [11] for the details. □

Proposition 5.21. Let M and N be terms in $\lambda_{\mathcal{C}}$ such that $M \longrightarrow_{\mathrm{D}} N$.

1. For any p$\lambda^{\rightarrow}$-term H there exists a p$\lambda^{\rightarrow}$-term K such that $(M : H) \xrightarrow{+}_{\beta_{||}} K$ and $(N : H) \sqsubseteq K$.
2. There exists a p$\lambda^{\rightarrow}$-term K such that $[\![M]\!] \xrightarrow{+}_{\beta_{||}} K$ and $[\![N]\!] \sqsubseteq K$. This can be illustrated in the following diagram.

$$\begin{tikzcd}
M \arrow[rr] \arrow[d, "\mathrm{D}"] & & {[\![M]\!]} \arrow[d, "\beta_{||}"', "+"] \\
N \arrow[r] & {[\![N]\!]} \arrow[r, phantom, "\sqsubseteq"] & \exists K
\end{tikzcd}$$

Proof:
The proof of (1) is by induction on the definition of $M \longrightarrow_{\mathrm{D}} N$ (Definition 3.3). The induction steps are proved in detail in [11]. Here we treat one of the base cases of the detour reduction.

1. $j' = l'$:
   We consider $\{\overline{N_j}; \overline{\lambda y.M}\} \cdot_{r_s} [\overline{P}; \overline{\lambda x_l.Q_l}] \longrightarrow_{\mathrm{D}} Q_{l'}[x_{l'} := N_{j'}]$.
   In this case we should have a case in the elimination term of the form $\lambda x_l.Q_l$.
   Write $L = \big(\lambda g_1 \dots g_t.g_s \overline{\llbracket P \rrbracket}\, \overline{(\lambda x_l.(Q_l : H))}\big)$, then

$$\begin{aligned}
\{\overline{N_j}; \overline{\lambda y.M}\} \cdot_{r_s} [\overline{P}; \overline{\lambda x_l.Q_l}] : H & \\
&= \{\overline{N_j}; \overline{\lambda y.M}\} : L \\
&= \big((\overline{\lambda q_j}.\overline{\lambda q_i} L)\overline{N_j}\, \overline{\lambda y_i.(M_i : L)}\big)\llbracket e_1^L \rrbracket \dots \llbracket e_t^L \rrbracket \\
&\overset{+}{\rightarrow}_{\beta_{||}} L\llbracket e_1^L \rrbracket, \dots, \llbracket e_t^L \rrbracket \qquad \text{(Delete dummy redexes)} \\
&\overset{+}{\rightarrow}_{\beta_{||}} \llbracket e_s^L \rrbracket \overline{\llbracket P \rrbracket}\, \overline{(\lambda x.(Q : H))} \\
&= (\dots ||(\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}\llbracket N_{j'} \rrbracket)|| \dots)\overline{\llbracket P \rrbracket}\, \overline{(\lambda x.(Q : H))} \qquad \text{(Definition of } \llbracket e_s^L \rrbracket\text{)} \\
&\overset{+}{\rightarrow}_{\beta_{||}} (\dots ||(\overline{\lambda h_k}.\overline{\lambda h_l}.h_{l'}\llbracket N_{j'} \rrbracket)\overline{\llbracket P \rrbracket}\, \overline{(\lambda x.(Q : H))}|| \dots) \\
&\overset{+}{\rightarrow}_{\beta_{||}} (\dots ||(\lambda x_{l'}.(Q_{l'} : H))\llbracket N_{j'} \rrbracket|| \dots) \\
&\overset{+}{\rightarrow}_{\beta_{||}} (\dots ||(Q_{l'} : H)[x_{l'} := \llbracket N_{j'} \rrbracket]|| \dots) \\
&\twoheadrightarrow_{\beta_{||}} (\dots ||(Q_{l'}[x_{l'} := N_{j'}]) : H|| \dots) \qquad \text{(Lemma 5.18)}
\end{aligned}$$

   Define $K = (\dots ||(Q_{l'}[x_{l'} := N_{j'}]) : H|| \dots)$, then we conclude $(Q_{l'}[x_{l'} := N_{j'}] : H) \sqsubseteq K$.

For (2): $\llbracket M \rrbracket = \lambda h.(M : h)$ and by (1) we know that there exists a K', such that $(M : h) \overset{+}{\rightarrow}_{\beta_{||}} K'$ and $(N : h) \sqsubseteq K'$. Let $K = \lambda h.K'$, then $\llbracket M \rrbracket \overset{+}{\rightarrow}_{\beta_{||}} K$ and $\llbracket N \rrbracket = \lambda h.(N : h) \sqsubseteq K$. □

Proposition 5.22. Let M and N be terms in $\lambda_{\mathcal{C}}$ such that $M \longrightarrow_{\mathrm{D}} N$ and $\llbracket M \rrbracket \sqsubseteq K$ for a p$\lambda^{\rightarrow}$-term K. Then there exists a K' such that $K \overset{+}{\rightarrow}_{\beta_{||}} K'$ and $\llbracket N \rrbracket \sqsubseteq K'$. This statement is shown in the following diagram.

$$\begin{tikzcd}
M \arrow[r] \arrow[d, "\mathrm{D}"] & \llbracket M \rrbracket \sqsubseteq & K \arrow[d, "\beta_{||}"', "+"] \\
N \arrow[r] & \llbracket N \rrbracket \sqsubseteq & \exists K'
\end{tikzcd}$$

Proof:
From Proposition 5.21, we know that there exists a term L such that $\llbracket M \rrbracket \overset{+}{\rightarrow}_{\beta_{||}} L$ and $\llbracket N \rrbracket \sqsubseteq L$. By Lemma 4.13 we know that we can reduce K to some term K' such that $L \sqsubseteq K'$. Since relation $\sqsubseteq$ is transitive we conclude $\llbracket N \rrbracket \sqsubseteq K'$. □

Now we look at an example of how the parallel terms are used in a detour reduction.

Example 5.23. Consider the following detour redex $\{N_A, N_B;\ \}_{11} \cdot_{\vee} [\ ; \lambda x.O_1, \lambda y.O_2]$ with non-optimized rules of disjunction (also stated in Example 3.4). There are two possible reductions.

$$\{N_A, N_B;\ \}_{11} \cdot_{\vee} [\ ; \lambda x.O_1, \lambda y.O_2] \longrightarrow_{\mathrm{D}} O_1[x := N_A]$$
$$\{N_A, N_B;\ \}_{11} \cdot_{\vee} [\ ; \lambda x.O_1, \lambda y.O_2] \longrightarrow_{\mathrm{D}} O_2[y := N_B]$$

Recall the modified term translation:

$$\begin{aligned}&\llbracket\{N_A, N_B, ;\ \}_{11} \cdot_\vee [; \lambda x.O_1, \lambda y.O_2]\rrbracket\\ &\quad = \lambda h.\Big(\big(\lambda p.\lambda q.(\lambda g_1.g_1(\lambda x.(O_1 : h))(\lambda y.(O_2 : h))\big)\langle\!\langle N_A\rangle\!\rangle\langle\!\langle N_B\rangle\!\rangle\Big)\llbracket e_1\rrbracket\end{aligned}$$

with $\llbracket e_1\rrbracket = (\lambda h_1.\lambda h_2.(h_1\llbracket N_A\rrbracket) \mid\mid \lambda h_1.\lambda h_2.(h_2\llbracket N_B\rrbracket))$.

The translated term $\beta_{||}$-reduces to

$$K = \lambda h.\big((O_1[x := N_A]) : h \mid\mid (O_2[y := N_B]) : h\big).$$

If we choose to do the first detour reduction, then indeed

$$\llbracket O_1[x := N_A]\rrbracket = \lambda h.((O_1[x := N_A]) : h) \sqsubseteq K.$$

When we would have picked the second possibility, then

$$\llbracket O_2[y := N_B]\rrbracket = \lambda h.((O_2[y := N_B]) : h) \sqsubseteq K.$$

Combining Proposition 5.22 (every $\longrightarrow_{\mathrm{D}}$ step translates to a non-empty $\beta_{||}$-reduction) and Proposition 5.17 (every $\longrightarrow_{\mathrm{Ppos}}$ -step translates to an equality) with Theorem 3.8(1) (there are no infinite $\longrightarrow_{\mathrm{P}}$ -reductions), we can conclude that $\longrightarrow_{\mathrm{D}} \cup \longrightarrow_{\mathrm{Ppos}}$ is strongly normalizing. This is the generalization of Theorem 18 of De Groote [5].

Theorem 5.24. For any set of connectives $\mathcal{C}$, $\lambda_{\mathcal{C}}$ is strongly normalizing with respect to detour and positive permutation reductions, $\longrightarrow_{\mathrm{D}} \cup \longrightarrow_{\mathrm{Ppos}}$.

Proof:
Suppose there is an infinite sequence of detour and positive permutation reductions starting from term M_1 in $\lambda_{\mathcal{C}}$. We draw the following picture.

$$\begin{tikzcd}
M_1 \arrow[d, "\mathrm{D}"] & & & \text{translates to} & K_1 \sqsupseteq \llbracket M_1\rrbracket \arrow[d, "\beta_{||}"', "+"] & & & & \\
M_2 \arrow[r, "\mathrm{Ppos}"] & M_3 \arrow[r, "\mathrm{Ppos}"] & M_4 \arrow[d, "\mathrm{D}"] & & K_2 \sqsupseteq \llbracket M_2\rrbracket & = & \llbracket M_3\rrbracket & = & \llbracket M_4\rrbracket \sqsubseteq K_2 \arrow[d, "\beta_{||}"', "+"] \\
 & & M_5 & & & & & & \llbracket M_5\rrbracket \sqsubseteq K_3 \\
 & & \vdots & & & & & & \vdots
\end{tikzcd}$$

Theorem 3.8(1) says there can be no infinite sequence of $\longrightarrow_{\mathrm{Ppos}}$ -steps, so the sequence contains infinitely many detour steps. Proposition 5.22 and Proposition 5.17 imply that the infinite reduction $M_1 \longrightarrow_{\mathrm{D}} M_2 \longrightarrow_{\mathrm{Ppos}} M_3 \longrightarrow_{\mathrm{Ppos}} M_3 \longrightarrow_{\mathrm{D}} M_4 \ldots$ indeed translates to an infinite sequence $K_1 \overset{+}{\to}_{\beta_{||}} K_2 \overset{+}{\to}_{\beta_{||}} K_3 \ldots$ of $\beta_{||}$-reduction steps. This contradicts the strong normalization property of p$\lambda^{\to}$ (Theorem 4.8). So, there can be no infinite sequence of detour and positive permutation reductions. □

We now have strong normalization of $\lambda_{\mathcal{C}}$ for $\longrightarrow_{\mathrm{D}} \cup \longrightarrow_{\mathrm{Ppos}}$, by adapting and extending the proof strategy of De Groote [5]. De Groote stops here and does not include negative permutations. As mentioned before, that is unsatisfactory, because it does not give the proper notion of normal form. So to fully complete our work we use Theorem 5.24 to prove strong normalization for all reductions $\longrightarrow_{\mathrm{D}} \cup \longrightarrow_{\mathrm{P}}$ in Theorem 5.28. The idea behind the proof is that negative permutation steps do not influence possible other reductions. We see that an infinite reduction path in $\lambda_{\mathcal{C}}$ (including negative permutation) corresponds to an infinite reduction path with $\longrightarrow_{\mathrm{D}} \cup \longrightarrow_{\mathrm{Ppos}}$ steps, by postponing the negative permutation steps.

Lemma 5.25. Let M_1, M_2, M_3 be terms in $\lambda_{\mathcal{C}}$ such that $M_1 \longrightarrow_{\mathrm{Pneg}} M_2 \longrightarrow_{\mathrm{D}} M_3$. Then there is a term F in $\lambda_{\mathcal{C}}$ such that $M_1 \longrightarrow_{\mathrm{D}} F \twoheadrightarrow_{\mathrm{Pneg}} M_3$. This statement is illustrated by the following diagram.

$$\begin{tikzcd}
M_1 \arrow[r, "\mathrm{D}"'] \arrow[d, "\mathrm{Pneg}"] & \exists F \arrow[d, twoheadrightarrow, "\mathrm{Pneg}"] \\
M_2 \arrow[r, "\mathrm{D}"'] & M_3
\end{tikzcd}$$

Proof:
By induction on the generation of $M_1 \longrightarrow_{\mathrm{Pneg}} M_2$. See [11] for a detailed proof. □

For positive permutations we want to prove a similar statement. We first introduce a special notion to count positive permutation steps.

Definition 5.26. We define the relation $\Longrightarrow^{n}_{\mathrm{Ppos}}$ which counts positive permutation steps as follows. We say $M \Longrightarrow^{n}_{\mathrm{Ppos}} N$ if

$$\begin{aligned}
M &= T \cdot [\overline{R}; \overline{\lambda y.S}] \cdot [\overline{U_1}; \overline{\lambda w_1.V_1}] \cdot \cdots \cdot [\overline{U_n}; \overline{\lambda w_n.V_n}], \\
N &= T \cdot [\overline{R}; \overline{\lambda y.(S \cdot [\overline{U_1}; \overline{\lambda w_1.V_1}] \cdot \cdots \cdot [\overline{U_n}; \overline{\lambda w_n.V_n}])}]
\end{aligned}$$

where in M there is at least one case of the form $\lambda y.S$.

Note that $\Longrightarrow^{1}_{\mathrm{Ppos}}$ is the same as $\longrightarrow_{\mathrm{Ppos}}$.

Lemma 5.27. Let M_1, M_2, M_3 be terms in $\lambda_{\mathcal{C}}$ such that $M_1 \longrightarrow_{\mathrm{Pneg}} M_2 \Longrightarrow^{n}_{\mathrm{Ppos}} M_3$. Then there is a term F in $\lambda_{\mathcal{C}}$ such that $M_1 \Longrightarrow^{m}_{\mathrm{Ppos}} F \twoheadrightarrow_{\mathrm{Pneg}} M_3$ with $m = n$ or $m = n + 1$. This statement is illustrated by the following diagram.

$$\begin{tikzcd}
M_1 \arrow[r, Rightarrow, "m", "\mathrm{Ppos}"'] \arrow[d, "\mathrm{Pneg}"] & \exists F \arrow[d, twoheadrightarrow, "\mathrm{Pneg}"] \\
M_2 \arrow[r, Rightarrow, "n", "\mathrm{Ppos}"'] & M_3
\end{tikzcd}$$

Proof:
By induction on the generation of $M_1 \longrightarrow_{\mathrm{Pneg}} M_2$. See [11] for a detailed proof. □

Finally, we conclude with our main theorem.

Theorem 5.28. (Strong normalization)
For any set of connectives $\mathcal{C}$, $\lambda_{\mathcal{C}}$ is strongly normalizing with respect to detour and permutation reductions, $\longrightarrow_{\text{D}} \cup \longrightarrow_{\text{P}}$.

Proof:
Suppose there is an infinite sequence of detour and permutation reductions starting from term M_1 in $\lambda_{\mathcal{C}}$. We distinguish between negative and positive permutation reductions. There cannot be an infinite sequence of consecutive negative permutation reductions (Theorem 3.8(1)). So we have the diagram below where the vertical direction indicates negative permutation reductions and the horizontal direction the other reductions.

$$\begin{tikzcd}
M_1 \arrow[r, dashed, "\text{D}"] \arrow[d, "\text{Pneg}"] & F_1 \arrow[r, Rightarrow, dashed, "k_1+1"] \arrow[dd, dashed, two heads, "k_1"] & F_2 \arrow[r, dashed, "\text{D}"] \arrow[dd, dashed, two heads, "k_2"] & F_3 \arrow[ddd, dashed, two heads, "k_3"] & \cdots \\
M_2 \arrow[d, "\text{Pneg}"] & & & & \\
M_3 \arrow[r, "\text{D}"] & M_4 \arrow[r, "\text{Ppos}"] & M_5 \arrow[d, "\text{Pneg}"] & & \\
& & M_6 \arrow[r, "\text{D}"] & M_7 & \cdots
\end{tikzcd}$$

We show that the dashed arrows exist such that we get an infinite sequence of detour and positive permutation reductions starting from M_1 on top of the figure. This contradicts Theorem 5.24.

This is proved as follows. From Lemma 5.25 we conclude that if $P_1 \longrightarrow^{k}_{\text{Pneg}} P_2$ in k steps and $P_2 \longrightarrow_{\text{D}} P_3$, then there exists an F such that $P_1 \longrightarrow_{\text{D}} F \twoheadrightarrow_{\text{Pneg}} P_3$. This yields F_1 in the diagram and the first commuting rectangle. From Lemma 5.27 we conclude that if $P_1 \longrightarrow^{k}_{\text{Pneg}} P_2$ in k steps and $P_2 \longrightarrow_{\text{Ppos}} P_3$, then there exists an F such that $P_1 \Longrightarrow^{m}_{\text{Ppos}} F \twoheadrightarrow_{\text{Pneg}} P_3$, with $1 \leq m \leq k+1$. This yields F_2 and the second commuting rectangle. So we can construct an infinite sequence of detour and positive permutation reductions, which is impossible. Therefore $\lambda_{\mathcal{C}}$ is strongly normalizing with respect to detour and permutation reductions $\longrightarrow_{\text{D}} \cup \longrightarrow_{\text{P}}$. □

6. Related and future work

We have studied the *truth table natural deduction system*, a natural deduction system for which the derivation rules are derived from the truth table, as defined in [3, 4]. We have proved that this very general natural deduction system for IPC satisfies strong normalization: any order in which detour conversions and permutation conversions are applied leads to a deduction in normal form. The truth table system is a manner to define natural deduction rules for arbitrary connectives using a standard format. There are various other ways to generalize standard natural deduction, for example as described in [14, 15, 16]. In [4] it has been shown that these can be translated to the truth table system in a conversion-preserving way. This makes the present strong normalization result very general: it applies to all systems of natural deduction. Proofs of weak and strong normalization of IPC appear at various places in the literature, starting with the proof of weak normalization by Prawitz [2], who later also gave a proof of strong normalization in [17]. The proof of strong normalization for IPC of which our proof is a generalization is given by De Groote [5], but we note here (see Example 5.1) that his 'normal proofs' do not have the required subformula property, which our normal proofs do have. Some other direct proofs of strong normalization (that do not use a reduction-preserving translation) are given by Von Plato [18], Troelstra [19] and Simpson [20], who prove strong normalization of intuitionistic predicate logic.

Closely related to the truth table system is the work of Milne [21], but his strategy is slightly different. He starts from the introduction rules which define a certain truth table. From these truth tables, the elimination rules are derived. Milne defines his method for classical logic. The idea that introduction rules are the basis for the definition of the elimination rules is rooted in the inversion principle of Gentzen and Prawitz: 'the introductions represent, as it were, the 'definitions' of the symbols concerned, and the eliminations are no more, in the final analysis, than the consequences of these definitions.' [1] This idea can be generalized to so-called 'general elimination rules'. This is studied by various researchers, such as Von Plato, Read, Francez and Dyckhoff [14, 22, 23]. The idea is that elimination rules are naturally determined by the introduction rules. The method with general elimination rules makes it possible to define deduction rules for arbitrary connectives, where the meaning of the connective lies in the introduction rules. This is different from the truth table system where connectives arise from truth tables. The elimination rules that arise from the 'general elimination' method have a similar shape as the elimination rules we derive from truth tables, in the sense that the conclusion of an elimination rule is an arbitrary formula D instead of a subformula of the major premise. In this way, the standard $\vee$-E rule in the Prawitz system is a general elimination rule. However, the general elimination rules differ from our elimination rules for some connectives, such as for $\wedge$ [4].

An interesting open problem is how to define detour conversion for the classical rules as they have also been defined generically for all connectives in [3]. In a classical deduction one should not only reduce an introduction followed by an elimination but also an elimination followed by an introduction: due to the format of the classical introduction rules, the 'introduced' formula need not be the conclusion of that rule but could be an assumption in the major premise which is eliminated.

Acknowledgments

We thank the anonymous referees for their valuable comments and suggestions for improvements of the paper.

7. Appendix A: Rules from truth tables

We represent the intuitionistic truth table rules for the well-known connectives $\wedge, \neg, \to, \vee, \bot$ and $\top$. Both in plain form from Definition 2.2 (left column) and in optimized form by Lemmas 2.7 and 2.8

(right column). The rules derived from the definition are labeled by the corresponding entries in the truth table. Before we present the rules, we give the well-known truth tables of the connectives.

A	B	$A \vee B$	$A \wedge B$	$A \to B$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1

A	$\neg A$
0	1
1	0

$\bot$	$\top$
0	1

Disjunction $\vee$

$\vee$	From definition	Optimized rules
Elim	$\dfrac{\vdash A \vee B \qquad A \vdash D \qquad B \vdash D}{\vdash D}\ \vee\text{-el}$	$\dfrac{\vdash A \vee B \qquad A \vdash D \qquad B \vdash D}{\vdash D}\ \vee\text{-el}$
Intro	$\dfrac{\vdash A \qquad B \vdash A \vee B}{\vdash A \vee B}\ \vee\text{-in}_{10}$ $\dfrac{\vdash A \qquad \vdash B}{\vdash A \vee B}\ \vee\text{-in}_{11}$ $\dfrac{A \vdash A \vee B \qquad \vdash B}{\vdash A \vee B}\ \vee\text{-in}_{01}$	$\dfrac{\vdash A}{\vdash A \vee B}\ \vee\text{-in}_1$ $\dfrac{\vdash B}{\vdash A \vee B}\ \vee\text{-in}_2$

Conjunction $\wedge$

$\wedge$	From definition	Optimized rules
Elim	$\dfrac{\vdash A \wedge B \qquad A \vdash D \qquad B \vdash D}{\vdash D}\ \wedge\text{-el}_{00}$ $\dfrac{\vdash A \wedge B \qquad A \vdash D \qquad \vdash B}{\vdash D}\ \wedge\text{-el}_{01}$ $\dfrac{\vdash A \wedge B \qquad \vdash A \qquad B \vdash D}{\vdash D}\ \wedge\text{-el}_{10}$	$\dfrac{\vdash A \wedge B}{\vdash A}\ \wedge\text{-el}_1$ $\dfrac{\vdash A \wedge B}{\vdash B}\ \wedge\text{-el}_2$
Intro	$\dfrac{\vdash A \qquad \vdash B}{\vdash A \wedge B}\ \wedge\text{-in}$	$\dfrac{\vdash A \qquad \vdash B}{\vdash A \wedge B}\ \wedge\text{-in}$

Implication $\to$

$\to$	From definition	Optimized rules
Elim	$\dfrac{\vdash A \to B \qquad \vdash A \qquad B \vdash D}{\vdash D}$ $\to$-el	$\dfrac{\vdash A \to B \qquad \vdash A}{\vdash B}$ $\to$-el
Intro	$\dfrac{A \vdash A \to B \qquad B \vdash A \to B}{\vdash A \to B}$ $\to$-in$_{00}$ $\dfrac{\vdash A \qquad \vdash B}{\vdash A \to B}$ $\to$-in$_{11}$ $\dfrac{A \vdash A \to B \qquad \vdash B}{\vdash A \to B}$ $\to$-in$_{01}$	$\dfrac{\vdash B}{\vdash A \to B}$ $\to$-in$_1$ $\dfrac{A \vdash A \to B}{\vdash A \to B}$ $\to$-in$_2$

The rules for negation, top and bottom cannot be further optimized with Lemmas 2.7 and 2.8. That is, the definition gives already the rules in optimized form.

Negation $\neg$

$\neg$	From definition
Elim	$\dfrac{\vdash \neg A \qquad \vdash A}{\vdash D}$ $\neg$-el
Intro	$\dfrac{A \vdash \neg A}{\vdash \neg A}$ $\neg$-in

Bottom $\perp$

$\perp$	From definition
Elim	$\dfrac{\vdash \perp}{\vdash D}$ $\perp$-el
Intro	no introduction rule

Top $\top$

$\top$	From definition
Elim	no elimination rule
Intro	$\dfrac{}{\vdash \top}$ $\top$-in

References

[1] Gentzen G. Untersuchungen über das logische Schliessen. In: The Collected Papers of Gerhard Gentzen, Studies in logic and the foundations of mathematics. North-Holland Pub. Co., 1969 doi:10.1006/inco.2002.3147.

[2] Prawitz D. Natural deduction: a proof-theoretical study. Almqvist & Wiksell, 1965. doi:10.2307/2271676.

[3] Geuvers H, Hurkens T. Deriving Natural Deduction Rules from Truth Tables. In: Logic and Its Applications - 7th Indian Conference, ICLA 2017, Kanpur, India, January 5-7, 2017, Proceedings, volume 10119 of *Lecture Notes in Computer Science*. 2017 pp. 123–138. doi:10.1007/978-3-662-54069-5_10.

[4] Geuvers H, Hurkens T. Proof Terms for Generalized Natural Deduction. In: Abel A, Forsberg FN, Kaposi A (eds.), 23rd International Conference on Types for Proofs and Programs, TYPES 2017, May 29-June 1, 2017, Budapest, Hungary, volume 104 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018 pp. 3:1–3:39. doi:10.4230/LIPIcs.TYPES.2017.3.

[5] de Groote P. On the Strong Normalisation of Intuitionistic Natural Deduction with Permutation-Conversions. *Inf. Comput.*, 2002. **178**(2):441–464. doi:10.1006/inco.2002.3147.

[6] Howard W. The formulae-as-types notion of construction. In: Hindley JR, Seldin JP (eds.), Essays on Combinatory Logic, Lambda Calculus and Formalism, pp. 479–490. Academic Press, London, 1980. Dedicated to Haskell B. Curry on the occasion of his 80th birthday.

[7] Sørensen M, Urzyczyn P. Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics). Elsevier Science Inc., New York, NY, USA, 2006. ISBN 0444520775.

[8] Terese. Term Rewriting Systems, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[9] Nederpelt R, Geuvers H. Type Theory and Formal Proof: An Introduction. Cambridge University Press, 2014. ISBN 9781107036505. URL `https://books.google.nl/books?id=wzTJBAAAQBAJ`.

[10] Tait W. Intensional Interpretations of Functionals of Finite Type I. *J. Symbolic Logic*, 1967. **32**(2):198–212. URL `https://projecteuclid.org:443/euclid.jsl/1183735831`.

[11] van der Giessen I. Natural Deduction Derived from Truth Tables, Master Thesis Mathematics, Radboud University Nijmegen, July 2018. URL `https://www.ru.nl/math/@1060423/master-theses-per-year/`.

[12] Takahashi M. Parallel Reductions in λ-Calculus. *Information and Computation*, 1995. **118**(1):120 – 127. doi:https://doi.org/10.1006/inco.1995.1057.

[13] Plotkin G. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1975. **1**(2):125 – 159. doi:https://doi.org/10.1016/0304-3975(75)90017-1.

[14] von Plato J. Natural deduction with general elimination rules. *Arch. Math. Log.*, 2001. **40**(7):541–567.

[15] Schroeder-Heister P. A Natural Extension of Natural Deduction. *J. Symb. Log.*, 1984. **49**(4):1284–1300.

[16] Negri S, von Plato J. Structural Proof Theory. Cambridge University Press, 2001.

[17] Prawitz D. Ideas and Results in Proof Theory. In: Fenstad J (ed.), 2nd Scandinavian Logic Symposium, pp. 237–309. North-Holland, 1971.

[18] von Plato J. Natural deduction: some recent developments, summary of four lectures at the summer school on proof theory, computation, and complexity. *Technische Universität Dresden*, June 23 - July 4, 2003. URL `http://www.helsinki.fi/~negri/dresum.pdf`.

[19] Troelstra AS. Metamathematical Investigations of intuitionistic arithmetic and analysis, volume 344 of *Lecture Notes in Mathematics*. Springer Verlag, 1973.

[20] Simpson AK. The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D. thesis, University of Edinburgh, 1994.

[21] Milne P. Inversion Principles and Introduction Rules. In: Wansing H (ed.), Dag Prawitz on Proofs and Meaning, pp. 189–224. Springer International Publishing, Cham. ISBN 978-3-319-11041-7, 2015. doi: 10.1007/978-3-319-11041-7_8.

[22] Read S. Harmony and Autonomy in Classical Logic. *Journal of Philosophical Logic*, 2000. **29**(2):123–154. doi:10.1023/A:1004787622057.

[23] Francez N, Dyckhoff R. A Note on Harmony. *Journal of Philosophical Logic*, 2012. **41**(3):613–628. doi:10.1007/s10992-011-9208-0.