# Chapter udf

# Propositions as Types

This is a *very experimental* draft of a chapter on the Curry–Howard correspondence. It needs more explanation and motivation, and there are probably errors and omissions. The proof of normalization should be reviewed and expanded. There are no examples for the product type. Permutation and simplification conversions are not covered. It will make a lot more sense once there is also material on the (typed) lambda calculus which is basically presupposed here. Use with extreme caution.

## pty.1 Introduction

int:pty:int:sec

The lambda calculus is a simple model of computation. It operates on terms built from variables. If $N$ and $M$ are terms, then $(NM)$ is also a term ("$N$ applied to $M$"). If $N$ is a term, then $\lambda x.\, N$ is a term. If $N$ contains the variable $x$, then the corresponding occurrences of $x$ in $\lambda x.\, N$ are bound by the $\lambda x$ operator, and so are no longer free. A term of the form $(\lambda x.\, NM)$ is said to $\beta$-contract to $N[M/x]$ (the result of replacing every free occurrence of $x$ in $N$ by $M$). A term $N$ $\beta$-converts to $N'$ if $N'$ results from $N$ by $\beta$-contracting a subterm, and we write $N \to N'$. A computation in the $\lambda$-calculus is a sequence $N \to N_1 \to N_2 \to \ldots$. If this sequence ends in a term $N_k$ where no subterm can be $\beta$-contracted, it is called *normal* or a normal form of $N$. Think of computation in the $\lambda$-calculus as such sequences. The computation halts if it results in a normal form, and the normal form is the result of the computation. This simple model of computations turns out to be Turing complete: every partial recursive function can be computated by a term in the $\lambda$-calculus.

Haskell Curry and William Howard independently discovered a close similarity between natural deduction and the $\lambda$-calculus. Intuitively, a term $\lambda x.\, N$ is a function: $x$ is the argument; $N$ tells us how to compute the value given $x$. A term $(\lambda x.\, NM)$ then is applying the function $\lambda x.\, N$ to an argument $M$, and $\beta$-contraction tells us how to evaluate it: replace $x$ in $N$ by $M$ (and keep evaluating the resulting term). Now think of these as functions with a fixed domain

and range. If the domain of $\lambda x.\, N$ is $A$ and and its range $B$, then we should think of the variable $x$ as only taking arguments from $A$ and $N$ as producing values from $B$. Consequently, $(\lambda x.\, NM)$ only should make sense if $\lambda x.\, N$ is a function $A \to B$, and $M \in A$. If we want to add this information to the $\lambda$-calculus we obtain the *typed* $\lambda$-calculus. In the typed $\lambda$-calculus not every expression $(\lambda x.\, NM)$ is legal, but only those where the "types" of $\lambda x.\, N$ and $M$ fit together, i.e., $\lambda x.\, N$ has type $A \to B$ and $M$ has type $A$. The type of $\lambda x.\, N$ is $A \to B$ only if the type of $x$ is $A$ and the type of $N$ is $B$. In general, not every expression $(M_1 M_2)$ is legal. Only those terms $(M_1 M_2)$ are legal where $M_1$ has type $A \to B$ and $M_2$ has type $A$; if so, then $(M_1 M_2)$ has type $B$.

These typing rules now clearly correspond to derivations in natural deduction, where the types are represented by formulas, and the derivations themselves to $\lambda$-terms. For instance, a derivation of $\varphi \to \psi$ corresponds to a term $\lambda x^{\varphi}.\, N$, which has the function type $\varphi \to \psi$. The inference rules of natural deduction correspond to typing rules in the typed lambda calculus, e.g.,

$$\dfrac{\begin{array}{c}[\varphi]^x \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\, x\, {\to}\text{Intro} \qquad \text{corresponds to} \qquad \dfrac{x : \varphi \;\Rightarrow\; N : \psi}{\Rightarrow\; \lambda x^{\varphi}.\, N : \varphi \to \psi}$$

where the rule on the right means that if $x$ is of type $\varphi$ and $N$ is of type $\psi$, then $\lambda x^{\varphi}.\, N$ is of type $\varphi \to \psi$.

The $\to$Elim rule corresponds to the typing rule for composition terms, i.e.,

$$\dfrac{\varphi \to \psi \qquad \varphi}{\psi}\, {\to}\text{Elim} \quad \text{corresponds to} \quad \dfrac{\Rightarrow\; M_1 : \varphi \to \psi \qquad \Rightarrow\; M_2 : \varphi}{\Rightarrow\; (M_1 M_2) : \psi}$$

If a $\to$Intro rule is followed immediately by a $\to$Elim rule, the derivation can be simplified:

$$\dfrac{\dfrac{\begin{array}{c}[\varphi]^x \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\, x\, {\to}\text{Intro} \qquad \begin{array}{c}\vdots \\ \varphi\end{array}}{\psi}\, {\to}\text{Elim} \qquad \to \qquad \begin{array}{c}\vdots \\ \varphi \\ \vdots \\ \psi\end{array}$$

which corresponds to $\beta$-contraction of lambda terms

$$(\lambda x^{\varphi}.\, NM) \quad \to \quad N[M/x].$$

Similar correspondences hold between the rules for $\wedge$ and "product" types, and between the rules for $\vee$ and "sum" types.

This correspondence between terms in the simply typed lambda calculus and natural deduction derivations is called the "Curry–Howard", "proofs-as-programs," or "propositions as types" correspondence. In addition to formulas

(propositions) corresponding to types, and proofs to terms, we can summarize the correspondences as follows:

| logic | program |
|---|---|
| proposition/formula | type |
| proof | term |
| assumption | variable |
| discharged assumption | bound variable |
| undischarged assumption | free variable |
| $\varphi \to \psi$ | function type |
| $\varphi \wedge \psi$ | product type |
| $\varphi \vee \psi$ | sum type |
| $\bot$ | empty type |

The Curry–Howard correspondence is one of the cornerstones of automated proof assistants and type checkers for programs, since checking a proof witnessing a proposition (as we did above) amounts to checking if a program (term) has the declared type.

## pty.2 Proof Terms

int:pty:ter: sec

We label assumptions in **N1i** with variables (e.g., $\varphi^x$) as discharge labels. In **N2i**, every sequent $\Gamma \Rightarrow \varphi$ has a list of variable–formula pairs in the context $\Gamma$ on the left. Thus, in **N2i**, every sequent in a derivation carries not only the information what the derivation so far has derived (i.e., $\varphi$), but also which assumptions are open at each point (i.e., $\Gamma$). What if we also included in each sequent the information *how* $\varphi$ was derived. To this end, we can define a *term-labelled* system **tN2i**, in which sequents have the form $\Gamma \Rightarrow N : \varphi$. $\Gamma$ and $\varphi$ are as before: a set of pairs of the form $x : \psi$ and the formula the sub-derivation ending in the sequent has shown to follow from the assumptions in $\Gamma$. $N$ will be a certain term that encodes the derivation ending in the sequent.

The terms we assign to each sequent are built from variables $x$, $y$, ..., using function symbols which encode the inferences carried out so far. We need a separate function symbol for each inference rule. The function symbol corresponding to introduction rules are called "constructors" and those corresponding to elimination rules "destructors." Each function symbol takes as many arguments as the rule has premises. For instance, the function symbols for $\wedge$Intro and $\wedge\text{Elim}_1$ might be $c^\wedge$ (2-place) and $d_1^\wedge$ (1-place). In **tN2i** these rules then become

$$\frac{\Gamma_1 \Rightarrow N : \varphi \qquad \Gamma_2 \Rightarrow M : \psi}{\Gamma_1, \Gamma_2 \Rightarrow c^\wedge(N, m) : \varphi \wedge \psi}\,\wedge\text{Intro} \qquad \frac{\Gamma \Rightarrow N : \varphi \wedge \psi}{\Gamma \Rightarrow d_1^\wedge(N) : \varphi}\,\wedge\text{Elim}_1$$

If an inference rule discharges an assumption, i.e., the rule has a premise that indicates a labelled formula $x : \varphi$ in a premise that is not in the context of the conclusion, we have to indicate this in the proof term as well. The function symbols associated with such rules *bind* the corresponding variables.

For instance, the $\to$Intro rule goes from the premise $x : \varphi, \Gamma \Rightarrow \psi$ to $\Gamma \Rightarrow \varphi \to \psi$. The constructor $c^{\to}$ takes one argument, but also must indicate that the variable $x$ becomes bound. This is the proof term's way of expressing that the assumption with label $x$ has been discharged. For instance, if the proof term for the premise is $N$, we could write the proof term for the conclusion as $c^{\to}([x]N)$ and the rule as

$$\frac{x : \varphi, \Gamma \Rightarrow N : \psi}{\Gamma \Rightarrow c^{\to}([x]N) : \varphi \to \psi} \to\text{Intro}$$

In order to be able to fully reconstruct a derivation, we need some additional information. If the premise formulas of a rule don't fully determine what the conclusion formula is, we have to add that information to the term. This is the case in $\to$Intro, so we will write $c^{\to}([x : \varphi]N)$. It is also the case in $\vee$Intro and $\bot_I$, so we use function symbols that carry that information, e.g.,

$$\frac{\Gamma \Rightarrow N : \varphi}{\Gamma \Rightarrow c_1^{\vee\psi} : \varphi \vee \psi} \vee\text{Intro}_1 \qquad \frac{\Gamma \Rightarrow N : \bot}{\Gamma \Rightarrow d_{\varphi}^{\bot}(N) : \varphi} \bot_I$$

With these ideas it is possible to construct a sequent-style natural deduction system in which every sequent is of the form $\Gamma \Rightarrow N : \varphi$, where $N$ is a *proof term*.

It turns out that there is a close connection between such proof terms and terms in the so-called *typed lambda calculus*. To respect this connection, we'll use symbols used in the literature on the lambda calculus rather than the generic constructor and destructor symbols above. E.g., instead of writing $c^{\to}([x : \varphi]N)$ we write $\lambda x^{\varphi}.\, N$, instead of $d^{\to}(N, M)$ we write $(NM)$, $\langle N, M\rangle$ instead of $c^{\wedge}(N, M)$, and $\pi_i(N)$ instead of $d_i^{\wedge}(N)$.

**Definition pty.1.** Proof terms are inductively generated by the following rules:

1. Every variable $x$ is a proof term (Axioms).
2. If $x$ is a variable, $\varphi$ is a formula, and $N$ is a proof term, then $\lambda x^{\varphi}.\, N$ is also a proof term ($\to$Intro).
3. If $N$ and $M$ are proof terms, then $(NM)$ is also a proof term ($\to$Elim).
4. If $N$ and $M$ are proof terms, then $\langle N, M\rangle$ is a proof term ($\wedge$Intro).
5. If $N$ is a proof term, then $\pi_i(N)$ is also a proof term, where $i$ is 1 or 2 ($\wedge\text{Elim}_i$).
6. If $N$ is a proof term, and $\varphi$ is a formula, then $\iota_i^{\varphi}(M)$ is a proof term, where $i$ is 1 or 2 ($\vee\text{Intro}_i$).
7. If $M$, $N_1$, $N_2$ are proof terms, and $x_1$, $x_2$ are variables, then $\delta\, M\, x_1.N_1\, x_2.N_2$ is a proof term ($\vee$Elim).
8. If $N$ is a proof term and $\varphi$ is a formula, then $\varepsilon^{\varphi}(N)$ is proof term ($\bot_I$).

**Axioms:**

$$x : \varphi \Rightarrow x : \varphi$$

**Inference Rules:**

$$\frac{x : \varphi, \Gamma \Rightarrow N : \psi}{\Gamma \Rightarrow \lambda x^{\varphi}. N : \varphi \to \psi} \to\text{Intro}$$

$$\frac{\Gamma_1 \Rightarrow N : \varphi \to \psi \qquad \Gamma_2 \Rightarrow M : \varphi}{\Gamma_1, \Gamma_2 \Rightarrow (NM) : \psi} \to\text{Elim}$$

$$\frac{\Gamma_1 \Rightarrow N_1 : \varphi \qquad \Gamma_2 \Rightarrow N_2 : \psi}{\Gamma_1, \Gamma_2 \Rightarrow \langle N_1, N_2 \rangle : \varphi \wedge \psi} \wedge\text{Intro}$$

$$\frac{\Gamma \Rightarrow N : \varphi \wedge \psi}{\Gamma \Rightarrow \pi_1(N) : \varphi} \wedge\text{Elim}_1 \qquad \frac{\Gamma \Rightarrow N : \varphi \wedge \psi}{\Gamma \Rightarrow \pi_2(N) : \psi} \wedge\text{Elim}_2$$

$$\frac{\Gamma \Rightarrow N : \varphi}{\Gamma \Rightarrow \iota_1^{\psi}(N) : \varphi \vee \psi} \vee\text{Intro}_1 \qquad \frac{\Gamma \Rightarrow N : \psi}{\Gamma \Rightarrow \iota_2^{\varphi}(N) : \varphi \vee \psi} \vee\text{Intro}_2$$

$$\frac{\Gamma_1 \Rightarrow M : \varphi \vee \psi \qquad x : \varphi, \Gamma_2 \Rightarrow N_1 : \chi \qquad y : \psi, \Gamma_3 \Rightarrow N_2 : \chi}{\Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \delta\, M\, x.N_1\, y.N_2 : \chi} \vee\text{Elim}$$

$$\frac{\Gamma \Rightarrow N : \bot}{\Gamma \Rightarrow \varepsilon^{\varphi}(N) : \varphi} \bot_I$$

Table pty.1: Rules of the propositional term-labelled intuitionistic natural deduction system **tN2ip**.

int:pty:ter:tab:tN2ip

Not every proof term is the proof term of a derivation. For instance, a term $\langle N, M \rangle$ encodes a proof ending in a $\wedge$Intro inference, so its end-formula is a conjunction $\varphi \wedge \psi$. This can't be the major premise of a $\to$Elim rule, so $(\langle N, M \rangle P)$ can't be the proof term of a correct proof. Only those proof terms that follow the rules of **tN2ip** are *correct* proof terms. These rules are given in Table pty.1.

## pty.3 Converting Derivations to Proof Terms

int:pty:pt:sec

The process of converting natural deduction derivations to proof terms, i.e., translating derivations in **N2i** to derivations in **tN2i**, is extremely straightforward. It simply amounts to writing a proof term to the left of each formula on the right of a sequent in the derivation, resulting in sequents of the form $\Gamma \Rightarrow M : \varphi$. We'll then say that $M$ *witnesses* $\varphi$ in context $\Gamma$. Which proof term to write is entirely determined by the rules of **tN2i**.

**Proposition pty.2.** *Every derivation $\delta$ of $\Gamma \Rightarrow \varphi$ in* **N2i** *corresponds to a derivation $\delta'$ of $\Gamma \Rightarrow N : \varphi$ in* **tN2i**. *The proof term $N$ is uniquely determined by $\delta$.*

*Proof.* We proceed by induction on the height of $\delta$. If $\delta$ is the axiom $x : \varphi \Rightarrow \varphi$, then $\delta'$ is the axiom $x : \varphi \Rightarrow x : \varphi$. If $\delta$ ends in an inference rule, the inductive hypothesis provides proof terms for each premise, and **tN2i**-derivations of sequents including these proof terms corresponding to each premise. We apply the corresponding rule of **tN2i**; the corresponding proof term is determined by the **tN2i** inference rule. □

**Example pty.3.** Consider the very simple derivation in **N1i** of $(\varphi \wedge \psi) \to (\psi \wedge \varphi)$:

$$\dfrac{\dfrac{\dfrac{[\varphi \wedge \psi]^x}{\psi}\wedge\text{Elim}_2 \quad \dfrac{[\varphi \wedge \psi]^x}{\varphi}\wedge\text{Elim}_1}{\psi \wedge \varphi}\wedge\text{Intro}}{(\varphi \wedge \psi) \to (\psi \wedge \varphi)}\,{\scriptstyle x}\,\to\text{Intro}$$

In **N2i** it looks like this:

$$\dfrac{\dfrac{\dfrac{x : \varphi \wedge \psi \;\Rightarrow\; \varphi \wedge \psi}{x : \varphi \wedge \psi \;\Rightarrow\; \psi}\wedge\text{Elim}_2 \quad \dfrac{x : \varphi \wedge \psi \;\Rightarrow\; \varphi \wedge \psi}{x : \varphi \wedge \psi \;\Rightarrow\; \varphi}\wedge\text{Elim}_1}{x : \varphi \wedge \psi \;\Rightarrow\; \psi \wedge \varphi}\wedge\text{Intro}}{\Rightarrow\; (\varphi \wedge \psi) \to (\psi \wedge \varphi)}\to\text{Intro}$$

We assign proof terms from the top down. The proof terms for the axioms are just the same as the variable in the context, $x$. The proof terms associated with $\wedge\text{Elim}_i$ are then determined as $\pi_2(x)$ and $\pi_1(x)$. When we apply $\wedge$Intro, we combine these two proof terms: $\langle \pi_2(x), \pi_1(x) \rangle$. Finally, the $\to$Intro rule applies a $\lambda$-abstraction, binding $x$ and recording that $x$ labelled the assumption $\varphi \wedge \psi$: $\lambda x^{\varphi \wedge \psi}. \langle \pi_2(x), \pi_1(x) \rangle$. The derivation in **tN2i** then is:

$$\dfrac{\dfrac{\dfrac{x : \varphi \wedge \psi \;\Rightarrow\; x : \varphi \wedge \psi}{x : \varphi \wedge \psi \;\Rightarrow\; \pi_2(x) : \psi}\wedge\text{Elim}_2 \quad \dfrac{x : \varphi \wedge \psi \;\Rightarrow\; x : \varphi \wedge \psi}{x : \varphi \wedge \psi \;\Rightarrow\; \pi_1(x) : \varphi}\wedge\text{Elim}_1}{x : \varphi \wedge \psi \;\Rightarrow\; \langle \pi_2(x), \pi_1(x) \rangle : \psi \wedge \varphi}\wedge\text{Intro}}{\Rightarrow\; \lambda x^{\varphi \wedge \psi}. \langle \pi_2(x), \pi_1(x) \rangle : (\varphi \wedge \psi) \to (\psi \wedge \varphi)}\to\text{Intro}$$

## pty.4 Recovering Derivations from Proof Terms

int:pty:tp:sec

Now let us consider the other direction: translating a proof term $N$ back to a derivation in **N2i**. In general, it is only possible to do this relative to a context that contains pairs $x : \varphi$ for every variable $x$ that occurs free in the given proof term. But let's start with an example where there are no free variables, namely the term

$$\lambda x^{\varphi \wedge \psi}. \langle \pi_2(x), \pi_1(x) \rangle$$

It is of the form $\lambda x^{\varphi\wedge\psi}.\, N_1$. Since the only rule of **tN2i** that produces a term of this form is $\to$Intro, we know that the derivation encoded by $N$ has to end with $\to$Intro, i.e., it ends in

$$\dfrac{x : \varphi \wedge \psi \;\Rightarrow\; \langle \pi_2(x), \pi_1(x)\rangle : \chi}{\Rightarrow\; \lambda x^{\varphi\wedge\psi}.\, \langle \pi_2(x), \pi_1(x)\rangle : (\varphi \wedge \psi) \to \chi}\,{\to}\text{Intro}$$

We don't yet know what $\chi$ is, but we do know that the antecedent is $\varphi \wedge \psi$, and that the context of the premise has to include $x : \varphi \wedge \psi$.

Now the proof term associated with the premise is already determined: $\langle \pi_2(x), \pi_1(x)\rangle$. The derivation *it* encodes must end with $\wedge$Intro. We still don't know what $\chi$ is, but because it is the conclusion of $\wedge$Intro, we know it must be a conjunction, i.e., $\chi \equiv (\chi_1 \wedge \chi_2)$. We continue the reconstruction:

$$\dfrac{\dfrac{x : \varphi \wedge \psi \;\Rightarrow\; \pi_2(x) : \chi_1 \qquad x : \varphi \wedge \psi \;\Rightarrow\; \pi_1(x) : \chi_2}{x : \varphi \wedge \psi \;\Rightarrow\; \langle \pi_2(x), \pi_1(x)\rangle : \chi_1 \wedge \chi_2}\,{\wedge}\text{Intro}}{\Rightarrow\; \lambda x^{\varphi\wedge\psi}.\, \langle \pi_2(x), \pi_1(x)\rangle : (\varphi \wedge \psi) \to (\chi_1 \wedge \chi_2)}\,{\to}\text{Intro}$$

The proof term $\pi_2(x)$ means that top sequent on the left was obtained by $\wedge\text{Elim}_2$, i.e., the premise is of the form $\theta \wedge \chi_1$. On the right, we can determine that the inference leading to $\chi_2$ must be $\wedge\text{Elim}_1$, and that the premise is of the form $\chi_2 \wedge \alpha$.

$$\dfrac{\dfrac{\dfrac{x : \varphi \wedge \psi \;\Rightarrow\; x : \theta \wedge \chi_1}{x : \varphi \wedge \psi \;\Rightarrow\; \pi_2(x) : \chi_1}\,{\wedge}\text{Elim}_2 \qquad \dfrac{x : \varphi \wedge \psi \;\Rightarrow\; x : \chi_2 \wedge \alpha}{x : \varphi \wedge \psi \;\Rightarrow\; \pi_1(x) : \chi_2}\,{\wedge}\text{Elim}_1}{x : \varphi \wedge \psi \;\Rightarrow\; \langle \pi_2(x), \pi_1(x)\rangle : \chi_1 \wedge \chi_2}\,{\wedge}\text{Intro}}{\Rightarrow\; \lambda x^{\varphi\wedge\psi}.\, \langle \pi_2(x), \pi_1(x)\rangle : (\varphi \wedge \psi) \to (\chi_1 \wedge \chi_2)}\,{\to}\text{Intro}$$

Since the proof terms of the topmost sequents are now variables, we know that they must be axioms. This determines $\chi_1$, $\chi_2$, $\theta$, and $\alpha$: $\theta \equiv \varphi$ and $\chi_1 \equiv \psi$ or else the left sequent is not an axiom, and $\chi_2 \equiv \varphi$ and $\alpha \equiv \psi$, or else the right sequent isn't. We have recovered the proof fully:

$$\dfrac{\dfrac{\dfrac{x : \varphi \wedge \psi \;\Rightarrow\; x : \varphi \wedge \psi}{x : \varphi \wedge \psi \;\Rightarrow\; \pi_2(x) : \psi}\,{\wedge}\text{Elim}_2 \qquad \dfrac{x : \varphi \wedge \psi \;\Rightarrow\; x : \varphi \wedge \psi}{x : \varphi \wedge \psi \;\Rightarrow\; \pi_1(x) : \varphi}\,{\wedge}\text{Elim}_1}{x : \varphi \wedge \psi \;\Rightarrow\; \langle \pi_2(x), \pi_1(x)\rangle : \psi \wedge \varphi}\,{\wedge}\text{Intro}}{\Rightarrow\; \lambda x^{\varphi\wedge\psi}.\, \langle \pi_2(x), \pi_1(x)\rangle : (\varphi \wedge \psi) \to (\psi \wedge \varphi)}\,{\to}\text{Intro}$$

## pty.5 Types

int:pty:typ:sec

A proof term such as $(MN)$ by itself does not uniquely determine the proof it is a proof term of. That's because the free variables, i.e., the proof terms of axioms, do not specify the formulas in the corresponding axioms. However, if we specify these in a context $\Gamma$, the proof *is* uniquely determined.

**Definition pty.4.** A *context* for a proof term $N$ is a set $\Gamma$ such that for every free variable of $N$, $\Gamma$ contains exactly one pair $x : \varphi$.

Note that the definition does not require that $\Gamma$ contains $x : \varphi$ *only* for free variables of $N$. This, e.g., $\Gamma = \{x : \varphi, y : \psi, z : \chi\}$ is a context for $\langle x, y\rangle$.

**Definition pty.5.** Suppose $\Gamma$ is a context for $N$. The *type* of $N$ (relative to $\Gamma$) is inductively defined as follows: int:pty:typ: defn:type

1. The type of $x$ is $\varphi$ iff $x : \varphi \in \Gamma$.
2. If $N$ has type $\psi$ relative to $x : \varphi, \Gamma$, then the type of $\lambda x^{\varphi}. N$ is $\varphi \to \psi$.
3. If $N$ has type $\varphi \to \psi$ and $M$ has type $\varphi$ then $(NM)$ has type $\psi$
4. If $N$ has type $\varphi$ and $M$ has type $\psi$, then $\langle N, M\rangle$ has type $\varphi \wedge \psi$.
5. If $N$ has type $\varphi_1 \wedge \varphi_2$, then $\pi_i(N)$ has type $\varphi_i$.
6. If $N$ has type $\varphi$, then $\iota_i^{\psi}(N)$ has type $\varphi \vee \psi$ if $i = 1$ and type $\psi \vee \varphi$ if $i = 2$.
7. If $M$ has type $\varphi \vee \psi$, $N_1$ has type $\chi$ relative to $x_1 : \varphi, \Gamma$, and $N_2$ has type $\chi$ relative to $x_2 : \psi, \Gamma$, then $\delta\, M\, x_1.N_1\, x_2.N_2$ has type $\chi$.
8. If $N$ has type $\bot$, then $\varepsilon^{\varphi}(N)$ has type $\varphi$.

**Proposition pty.6.** *Every proof term $N$ has exactly one type relative to a context $\Gamma$ for $N$, if it has a type at all (i.e., is a correct proof term).*

*Proof.* By induction on $N$. □

We write $\Gamma \Rightarrow N : \varphi$ if the type of $N$ relative to $\Gamma$ is $\varphi$. The clauses in Definition pty.5 should seem familiar: they are precisely the rules of **tN2i** with one slight difference: the context $\Gamma$ is the same throughout. We can codify the definition as a calculus **tN3i** with the rules given in Table pty.2.

Proof terms are terms in a version of the $\lambda$-calculus, the *typed $\lambda$-calculus with products, sums, and empty type*. The classical $\lambda$-calculus only has term built from variables, *$\lambda$-abstracts* $\lambda x.\, N$ and application $(MN)$, and it lacks the annotation by $\varphi$ of the typed $\lambda$-abstract $\lambda x^{\varphi}.\, N$. The $\lambda$-calculus is a model of computation which is Turing complete (i.e., every partial computable function can be represented in it by a term). The version of the $\lambda$-calculus given by proof terms and typing rules is a restricted model of computation, but the basic ideas are the same. It is restricted by the type assignment: only *well-typed* terms, i.e., proof terms with types (relative to a context $\Gamma$) are legal terms. For instance, $\lambda x.\, (xx)$ is a legal term in the classical untyped $\lambda$-calculus, but $\lambda x^{\varphi}.\, (xx)$ is not well-typed for any $\varphi$. That's because the typing rules require that in order for $(xx)$ to have a type, the first $x$ would have to have type $\varphi \to \psi$ and the second $x$ type $\varphi$, and that's impossible since $\varphi$ is a proper sub-formula of $\varphi \to \psi$.

We can think of types intuitively as the kind of thing named by a term of that type. So a term of type $\varphi \wedge \psi$ picks out a pair with first component of type $\varphi$ and second component of type $\psi$, i.e., an element of the product $\varphi \times \psi$.

**Axioms:**

$$\Gamma \Rightarrow x : \varphi \text{ if } x : \varphi \in \Gamma$$

**Inference Rules:**

$$\frac{x : \varphi, \Gamma \Rightarrow N : \psi}{\Gamma \Rightarrow \lambda x^{\varphi}.\, N : \varphi \to \psi} \to\text{Intro}$$

$$\frac{\Gamma \Rightarrow N : \varphi \to \psi \qquad \Gamma \Rightarrow M : \varphi}{\Gamma \Rightarrow (NM) : \psi} \to\text{Elim}$$

$$\frac{\Gamma \Rightarrow N_1 : \varphi \qquad \Gamma \Rightarrow N_2 : \psi}{\Gamma \Rightarrow \langle N_1, N_2 \rangle : \varphi \wedge \psi} \wedge\text{Intro}$$

$$\frac{\Gamma \Rightarrow N : \varphi \wedge \psi}{\Gamma \Rightarrow \pi_1(N) : \varphi} \wedge\text{Elim}_1 \qquad \frac{\Gamma \Rightarrow N : \varphi \wedge \psi}{\Gamma \Rightarrow \pi_2(N) : \psi} \wedge\text{Elim}_2$$

$$\frac{\Gamma \Rightarrow N : \varphi}{\Gamma \Rightarrow \iota_1^{\psi}(N) : \varphi \vee \psi} \vee\text{Intro}_1 \qquad \frac{\Gamma \Rightarrow N : \varphi}{\Gamma \Rightarrow \iota_2^{\psi}(N) : \psi \vee \varphi} \vee\text{Intro}_2$$

$$\frac{\Gamma \Rightarrow M : \varphi \vee \psi \qquad x : \varphi, \Gamma \Rightarrow N_1 : \chi \qquad y : \psi, \Gamma \Rightarrow N_2 : \chi}{\Gamma \Rightarrow \delta\, M\, x.N_1\, y.N_2 : \chi} \vee\text{Elim}$$

$$\frac{\Gamma \Rightarrow N : \bot}{\Gamma \Rightarrow \varepsilon^{\varphi}(N) : \varphi} \bot_I$$

Table pty.2: Rules of the propositional term-labelled intuitionistic natural deduction system **tN3ip** serving as a type assignment system.

int:pty:typ:tab:tN3ip

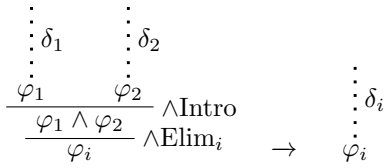
A term of type $\varphi \to \psi$ is a function from things of type $\varphi$ to things of type $\psi$. A term of type $\varphi \vee \psi$ is either something of type $\varphi$ or something of type $\psi$, plus the information which one it is. This is like the disjoint union of the two sets $\varphi$ and $\psi$ defined as $\{\langle i, x\rangle : i = 1 \text{ or } 2, x \in \varphi \text{ if } i = 1, x \in \psi \text{ if } i = 2\}$. The type $\bot$ is the empty set, i.e., if a term has that type it can't name anything. Since natural deduction (including **tN3i**) is consistent, there is no derivation of $\bot$ unless there are open assumptions, and so there is no closed proof term (term without free variables) that has type $\bot$.

The operators of the typed $\lambda$-calculus produce terms that intuitively name the right sort of thing, provided the terms it is applied to name certain things. E.g., "if $N_1 : \varphi$ and $N_2 : \psi$, then $\langle N_1, N_2\rangle : \varphi \wedge \psi$" says that if $N_1$ names somethign of type $\varphi$ and $N_2$ names something of type $\psi$, then $\langle N_1, N_2\rangle$ names something of type $\varphi \wedge \psi$.

## pty.6 Reduction

In natural deduction derivations, an introduction rule that is followed by an elimination rule is a detour that can be "reduced". For instance, an ∧Intro followed by ∧Elim "cancel out," since the conclusion of ∧Elim is always one of the conjuncts of the conjunction that ∧Intro produces, and the premises of ∧Intro are both conjuncts. Thus if we need a derivation of either conjunct, we can simply use that derivation without introducing the conjunction followed by eliminating it. We have

int:pty:red:sec



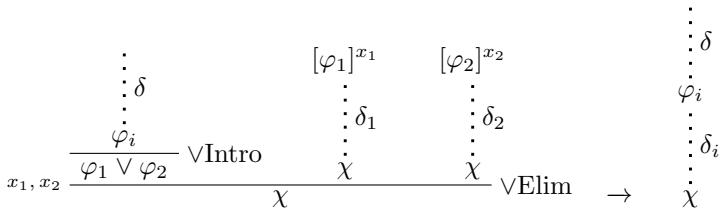

We can consider a similar relation of reduction on the associated proof terms suggested by this simplification. If $N_1$ is the proof term for $\delta_1$ and $N_2$ the one for $\delta_2$, then we can say the proof term for the proof on the left reduces (in one step) to the proof term for the proof on the right:

$$\pi_i(\langle N_1, N_2, \rangle) \to N_i$$

In the typed lambda calculus, this is the *beta reduction* rule for the product type.

In natural deduction, a pair of inferences such as a ∧Intro folloed by ∧Elim, i.e., a pair that is subject to reduction, is called a *cut*. In the typed lambda calculus the term on the left of → is called a *redex*, and the term to the right is called the *reductum*. Unlike untyped lambda calculus, where only $(\lambda x.\, N)Q$ is considered to be redex, in the typed lambda calculus the syntax is extended to terms involving $\langle N, M\rangle$, $\pi_i(N)$, $\iota_i(\varphi)N$, $\delta\, N\, x_1.M_1\, x_2.M_2$, and $\varepsilon^{\varphi}(N)$, with corresponding redexes.

Similarly we have reduction for disjunction:

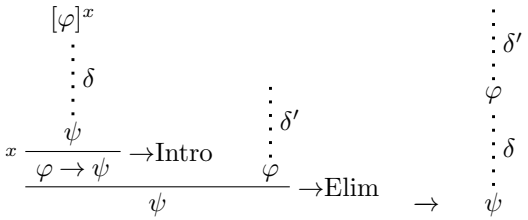



This corresponds to a reduction on proof terms:

$$\delta\, \iota_i^{\varphi_i}(M)\, x_1.N_1\, x_2.N_2 \to N_i[M/x_i]$$

where $M$ is the proof term of $\delta$, and $N_i$ is the proof term of $\delta_i$. This is the beta reduction rule of for sum types. Here, $M[N_i/x]$ means replacing all free occurrences of the variable $x$ in $M$ by $N_i$.

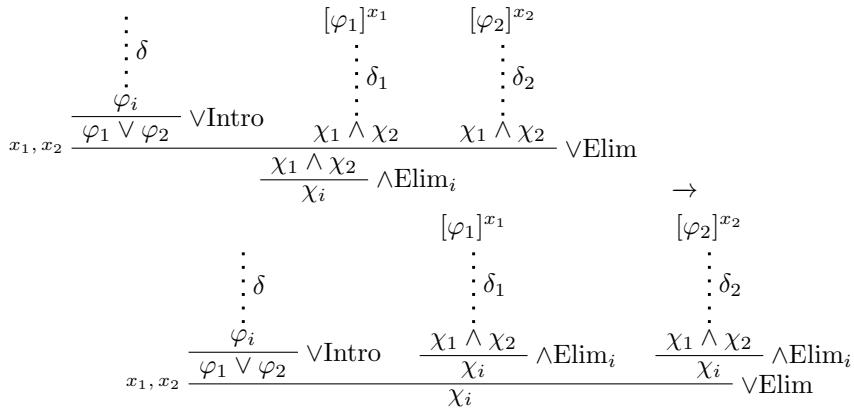
The reduction of the function type corresponds to removal of a detour of →Intro followed by a →Elim.



For proof terms, this amounts to ordinary beta reduction:

$$(\lambda x^{\varphi}.\, NM) \to N[M/x]$$

In addition to reduction conversions on derivations, we can consider permutation conversions. These apply to (sub)derivations that end in a ∨Elim followed by another elimination rule, e.g.:



If the proof terms of $\delta$, $\delta_1$, and $\delta_2$ are $M$, $N_1$, and $N_2$, respectively, the corresponding reduction rule for proof terms, i.e., $\lambda$ terms, is:

$$\pi_i(\delta\, M\, x_1.N_1\, x_2.N_2) \to \delta\, M\, x_1.\pi_i(N_1)\, x_2.\pi_i(N_2)$$

Of course we can eliminate detours not just at the ends of derivations, but also within derivations, by applying the reduction conversion to a sub-derivation ending in a detour. Similarly, $\beta$-reduction applies to subterms of $\lambda$-terms. If $N$ is a subterm of $M$, $N \to N'$, we can replace the subterm $N$ in $M$ by $N'$ to obtain $M'$. In that case we also write $M \to M'$. If there is a sequence $M = M_1 \to M_2 \to M_2 \to \ldots \to M_k = M'$ (including where $k = 1$, i.e., $M = M'$) we write $M \twoheadrightarrow M'$.

A derivation where no conversion can be applied to any sub-derivation is called *normal*. Similarly, a $\lambda$-term where no conversion can be applied (to any sub-term) is called *normal*. A normal $\lambda$-term is also called a *normal form*.

## pty.7 Type Preservation

int:pty:tpr:sec

The reduction and permutation conversions considered can also be defined directly for the systems **tN2i** and **tN3i**. The system **tN3i** defines the type of a $\lambda$-term $N$ relative to the context $\Gamma$ as the formula $\varphi$ such that $\Gamma \Rightarrow N : \varphi$ has a derivation in **tN3i**. The fact that the $\beta$-reduction rules for $\lambda$-terms exactly mirror the reduction and permutation conversions of derivations has an important consequence: the type of a $\lambda$-term relative to a context $\Gamma$ remains the same after $\beta$-conversion. This property is called *type preservation*.

**Proposition pty.7.** *If $N$ has type $\varphi$ relative to $\Gamma$ and $N \to N'$, then $N'$ has type $\varphi$ relative to $\Gamma$.*

*Proof.* By induction on $N$ and the height of derivations $\delta$ of $\Gamma \Rightarrow N : \varphi$ we can easily establish the following: If $M$ is a sub-term of $N$, then $\delta$ contains a sub-derivation $\delta_1$ ending in $\Gamma' \Rightarrow M : \psi$. If $M$ is a redex, then a conversion applies to $\delta_1$. By applying the reduction to the $\delta_1$ we obtain a derivation $\delta_1'$ of $\Gamma' \Rightarrow M' : \psi$. By inspection of the conversions of **tN3i**, $M'$ is the reductum of $M$. Replacing $\delta_1$ in $\delta$ by $\delta_1'$ results in a derivation $\delta'$ of $\Gamma \Rightarrow N' : \varphi$, where $N'$ results from $N$ by replacing the subterm $M$ by $M'$.

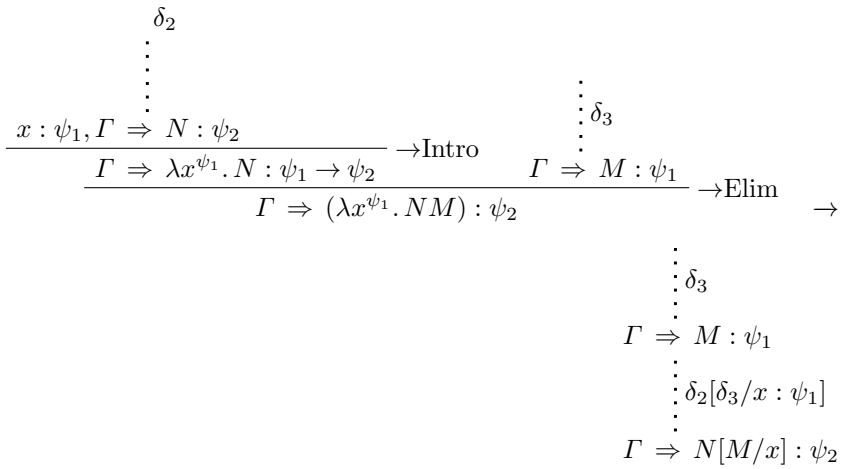
For instance, consider the reduction conversion for $\wedge$Intro followed by $\wedge$Elim:

$$\dfrac{\dfrac{\vdots\ \Gamma \Rightarrow N_1 : \psi_1 \qquad \vdots\ \Gamma \Rightarrow N_2 : \psi_2}{\Gamma \Rightarrow \langle N_1, N_2\rangle : \psi_1 \wedge \psi_2}\,\wedge\text{Intro}}{\Gamma \Rightarrow \pi_i(\langle N_1, N_2\rangle) : \psi_i}\,\wedge\text{Elim}_i \quad \to \quad \vdots\ \Gamma \Rightarrow N_i : \psi_i$$

We see that the proof term of the derivation on the right, i.e., $\delta_1'$ is $N_i$. It is exactly the result of applying $\beta$-conversion to the proof term $\pi_i(\langle N_1, N_2\rangle)$ of the derivation on the left ($\delta_1$).

Or consider a $\to$Intro followed by an $\to$Elim inference, and the reduction

conversion applied to it:




Again the proof term of the derivation on the right is exactly the result of $\beta$-reducing the proof term of the derivation on the left. Of course one would have to carefuly verify (by induction on the height of $\delta_2$) that if we replace all axioms of the form $\Gamma' \Rightarrow x : \psi_1$ in $\delta_2$ by the derivation $\delta_3$ of $\Gamma \Rightarrow M : \psi_1$ actually results in a derivation $\delta_2[\delta_3/x : \psi_1]$ of $\Gamma \Rightarrow N[M/x] : \psi_2$. □

The tight correspondence between derivations and $\lambda$-terms has another consequence. If $N$ is a $\lambda$-term of type $\varphi$ relative to a context $\Gamma$ and contains a redex $M$ (i.e., it is not a normal form), then the corresponding **tN3i** proof $\delta$ of $\Gamma \Rightarrow N : \varphi$ contains a sub-derivation (ending in $\Gamma' \Rightarrow M : \psi$) to which a conversion can be applied. If $N \to N'$ by replacing $M$ by its reductum, the corresponding result of applying the reduction to $\delta$ is a derivation of $\Gamma \Rightarrow N' : \varphi$. Hence every $\lambda$-term not in normal form $\beta$-reduces to another. This property is called *progress*.

Progress together with type preservation ensure that $\beta$-reduction of $\lambda$-terms never "gets stuck": if a term is well-typed and isn't in normal form, it reduces to another well-typed term (and that term is of the same type).

## pty.8 Normalization

int:pty:nor:sec

In this section we prove that, via some reduction order, any deduction can be reduced to a normal deduction, which is called the *normalization property*. We will make use of the propositions-as-types correspondence: we show that every proof term can be reduced to a normal form; normalization for natural deduction derivations then follows.

Firstly we define some functions that measure the complexity of terms. The *length* $\mathrm{len}(\varphi)$ of a formulas is defined by

$$\begin{aligned}\mathrm{len}(p) &= 0\\ \mathrm{len}(\varphi \wedge \psi) &= \mathrm{len}(\varphi) + \mathrm{len}(\psi) + 1\\ \mathrm{len}(\varphi \vee \psi) &= \mathrm{len}(\varphi) + \mathrm{len}(\psi) + 1\\ \mathrm{len}(\varphi \to \psi) &= \mathrm{len}(\varphi) + \mathrm{len}(\psi) + 1.\end{aligned}$$

The complexity of a redex $M$ is measured by its *cut rank* $\mathrm{cr}(M)$:

$$\begin{aligned}\mathrm{cr}((\lambda x^{\varphi}.\, N^{\psi})Q) &= \mathrm{len}(\varphi) + \mathrm{len}(\psi) + 1\\ \mathrm{cr}(\mathrm{p}_i(\langle M^{\varphi}, N^{\psi}\rangle)) &= \mathrm{len}(\varphi) + \mathrm{len}(\psi) + 1\\ \mathrm{cr}(\mathrm{case}(\mathrm{in}_i^{\varphi_i}(M^{\varphi_i}), x_1^{\varphi_1}.N_1^{\chi}, x_2^{\varphi_2}.N_2^{\chi})) &= \mathrm{len}(\varphi) + \mathrm{len}(\psi) + 1\end{aligned}$$

The complexity of a proof term is measured by the most complex redex in it, and 0 if it is normal:

$$\mathrm{mr}(M) = \max\{\mathrm{cr}(N) | N \text{ is a sub term of } M \text{ and is redex}\}$$

**Lemma pty.8.** *If $M[N^{\varphi}/x^{\varphi}]$ is a redex and $M \not\equiv x$, then one of the following cases holds:* int:pty:nor: lem:subst

1. *$M$ is itself a redex, or*
2. *$M$ is of the form $\mathrm{p}_i(x)$, and $N$ is of the form $\langle P_1, P_2\rangle$*
3. *$M$ is of the form $\mathrm{case}(i, x_1.P_1, x_2.P_2)$, and $N$ is of the form $\mathrm{in}_i(Q)$*
4. *$M$ is of the form $xQ$, and $N$ is of the form $\lambda x.\, P$*

*In the first case, $\mathrm{cr}(M[N/x]) = \mathrm{cr}(M)$; in the other cases, $\mathrm{cr}(M[N/x]) = \mathrm{len}(\varphi)$).*

*Proof.* Proof by induction on $M$.

1. If $M$ is a single variable $y$ and $y \not\equiv x$, then $y[N/x]$ is $y$, hence not a redex.
2. If $M$ is of the form $\langle N_1, N_2\rangle$, or $\lambda x.\, N$, or $\mathrm{in}_i^{\varphi}(N)$, then $M[N^{\varphi}/x^{\varphi}]$ is also of that form, and so is not a redex.
3. If $M$ is of the form $\mathrm{p}_i(P)$, we consider two cases.
   a) If $P$ is of the form $\langle P_1, P_2\rangle$, then $M \equiv \mathrm{p}_i(\langle P_1, P_2\rangle)$ is a redex, and clearly
   $$M[N/x] \equiv \mathrm{p}_i(\langle P_1[N/x], P_2[N/x]\rangle)$$
   is also a redex. The cut ranks are equal.

b) If $P$ is a single variable, it must be $x$ to make the substitution a redex, and $N$ must be of the form $\langle P_1, P_2\rangle$. Now consider

$$M[N/x] \equiv \mathrm{p}_i(x)[\langle P_1, P_2\rangle/x],$$

which is $\mathrm{p}_i(\langle P_1, P_2\rangle)$. Its cut rank is equal to $\mathrm{cr}(x)$, which is $\mathrm{len}(\varphi)$.

The cases of $\mathrm{case}(N, x_1.N_1, x_2.N_2)$ and $PQ$ are similar. □

**Lemma pty.9.** *If $M$ contracts to $M'$, and $\mathrm{cr}(M) > \mathrm{cr}(N)$ for all proper redex sub-terms $N$ of $M$, then $\mathrm{cr}(M) > \mathrm{mr}(M')$.*

*Proof.* Proof by cases.

1. If $M$ is of the form $\mathrm{p}_i(\langle M_1, M_2\rangle)$, then $M'$ is $M_i$; since any sub-term of $M_i$ is also proper sub-term of $M$, the claim holds.

2. If $M$ is of the form $(\lambda x^\varphi.\, N)Q^\varphi$, then $M'$ is $N[Q^\varphi/x^\varphi]$. Consider a redex in $M'$. Either there is corresponding redex in $N$ with equal cut rank, which is less than $\mathrm{cr}(M)$ by assumption, or the cut rank equals $\mathrm{len}(\varphi)$, which by definition is less than $\mathrm{cr}((\lambda x^\varphi.\, N)Q)$.

3. If $M$ is of the form

$$\mathrm{case}(\mathrm{in}_i(N^{\varphi_i}), x_1^{\varphi_1}.N_1^\chi, x_2^{\varphi_2}.N_2^\chi),$$

then $M' \equiv N_i[N/x_i^{\varphi_i}]$. Consider a redex in $M'$. Either there is corresponding redex in $N_i$ with equal cut rank, which is less than $\mathrm{cr}(M)$ by assumption; or the cut rank equals $\mathrm{len}(\varphi_i)$, which by definition is less than $\mathrm{cr}(\mathrm{case}(\mathrm{in}_i(N^{\varphi_i}), x_1^{\varphi_1}.N_1^\chi, x_2^{\varphi_2}.N_2^\chi))$. □

**Theorem pty.10.** *All proof terms reduce to normal form; all derivations reduce to normal derivations.*

*Proof.* The second follows from the first. We prove the first by complete induction on $m = \mathrm{mr}(M)$, where $M$ is a proof term.

1. If $m = 0$, $M$ is already normal.

2. Otherwise, we proceed by induction on $n$, the number of redexes in $M$ with cut rank equal to $m$.

   a) If $n = 1$, select any redex $N$ such that $m = \mathrm{cr}(N) > \mathrm{cr}(P)$ for any proper sub-term $P$ which is also a redex of course. Such a redex must exist, since any term only has finitely many subterms.

   Let $N'$ denote the reductum of $N$. Now by the lemma $\mathrm{mr}(N') < \mathrm{mr}(N)$, thus we can see that $n$, the number of redexes with $\mathrm{cr}(=)m$ is decreased. So $m$ is decreased (by 1 or more), and we can apply the inductive hypothesis for $m$.

b) For the induction step, assume $n > 1$. the process is similar, except that $n$ is only decreased to a positive number and thus $m$ does not change. We simply apply the induction hypothesis for $n$. □

The fact that typed $\lambda$-terms normalize means that every such term *has a normal form*. If we think of $\beta$-reducing $\lambda$-terms to their normal form as executing a computation and the normal form as the value of the computation, this means that every computation in the typed $\lambda$-calculus eventually halts an produces a value. Moreover, type preservation ensures that the value has the right type. E.g., if $N$ has type $\varphi \to \psi$ (i.e., it is a function that is supposed to work for arguments of type $\varphi$ and produces values of type $\psi$) and we apply it to a term $M$ of type $\varphi$ (the input), $(NM)$ reduces to a term $N'$ (the output) and that output is guaranteed to be of type $\psi$.

In fact, terms in the typed $\lambda$-calculus don't just reduce to normal forms for one way of applying $\beta$-reductions to subterms, but *any* way of applying $\beta$-reductions to subtems eventually terminates in a normal form. This property is called *strong normalization*. It means that not only is there a way to guarantee that a computation eventually halts and produces a value, but *any* way to execute the computation eventually produces a value.

How do we know that different ways of executing a computation produce the *same* value? So far we just know (from type preservation) that all values it produces are of the same type. The typed $\lambda$-calculus has another important property: it is *confluent* or *Church-Rosser*. If $N \twoheadrightarrow N_1$ and $N \twoheadrightarrow N_2$ then there is some term $N'$ such that $N_1 \twoheadrightarrow N'$ and $N_2 \twoheadrightarrow N'$. Recall that $N \twoheadrightarrow N'$ means: $N'$ results from zero or more $\to$-reduction steps. If $N_1$ is in normal form then the only term $N'$ such that $N_1 \twoheadrightarrow N'$ is $N_1$ itself (via zero $\to$-reduction steps). It follows that if $N_1$ and $N_2$ are both in normal form, then $N_1 = N' = N_2$. In other words, since $\twoheadrightarrow$ is strongly normalizing, every way of reducing a term eventually results in a normal form. Since $\twoheadrightarrow$ is Church-Rosser, any two normal forms are equal. Hence, computation in the typed $\lambda$-calculus always terminates and the value (normal form) is the same regardless of how the computation proceeds.

Proving that a relation is confluent is typically difficult. However, we can easily establish the following weaker condition:

**Proposition pty.11 (Weak Church-Rosser Property).** *If $N \to N_1$ and $N \to N_2$ then there is a term $N'$ such that $N_1 \twoheadrightarrow N'$ and $N_2 \twoheadrightarrow N'$.*

*Proof.* The two terms $N_1$ and $N_2$ result from $N$ by replacing the redex $M_1$ or $M_2$ by its reductum, respectively. The redexes $M_1$ and $M_2$ are subterms of $N$, and so cannot overlap. This means that either one is a subterm of the other, or they occur in different non-overlapping places in $N$. Consider the latter case: $N$ is of the form $N(M_1, M_2)$. Replacing $M_1$ by its reductum $M_1'$ results in $N_1$ which is $N(M_1', M_2)$, and replacing $M_2$ by its reductum $M_2'$ results in $N_2$ which is $N(M_1, M_2')$. Both reduce to $N(M_1', M_2')$. For the other case, suppose $M_2$ is a sub-term of $M_1$ (the other case is symmetric) and distinguish cases according

to what kind of redex $M_1$ is. E.g., if $M_1 \equiv \pi_1(\langle O_1, O_2\rangle)$ it reduces to $O_1$. If $M_2$ is a subterm of $O_1$, then converting $M_2$ results in $O_1'$. So converting $M_1$ first and then $M_2$ results in $O_1'$. But converting $M_2$ first results in $\pi_1(\langle O_1', O_2\rangle)$ and this reduces also to $O_2$. □

**Lemma pty.12 (Newman's Lemma).** *If $\twoheadrightarrow$ is strongly normalizing and has the weak Church-Rosser property, it is confluent.*

*Proof.* Since $\twoheadrightarrow$ is strongly normalizing, any reduction sequence results in a normal form. Normal forms are unique iff $\twoheadrightarrow$ is confluent. So suppose some term $N$ has two distinct normal forms $N'$ and $N''$, the result of reduction sequences

$$N = N_1' \to N_2' \to \ldots \to N_k' = N' \text{ and}$$
$$N = N_1'' \to N_2'' \to \ldots \to N_l'' = N''$$

(The reduction sequence must have length $> 0$ since otherwise $N$ would already be in normal form.)

If $N_1' = N_1''$ then it has two distinct normal forms ($N'$ and $N''$). If $N_1' \neq N_1''$ then by weak Church-Rosser there is some $N'''$ such that $N_1' \twoheadrightarrow N'''$ and $N_1'' \twoheadrightarrow N'''$. Since $N' \neq N''$, either $N''' \neq N'$ or $N''' \neq N''$. Consequently, either $N_1'$ or $N_1''$ has two distinct normal forms. We've shown that if $N$ has two distinct normal forms, there is an $N_1$ such that $N \to N_1$ and $N_1$ has two distinct normal forms. Obviously we can keep going and produce $N \to N_1 \to N_2 \to \ldots$, but this is impossible because $\to$ is strongly normalizing. □

# Photo Credits

# Bibliography