experience in a list-processing language such as LISP or PROLOG; you can safely ignore these examples if you want to.

**Pairs.** If $M$ and $N$ are lambda terms, we define the pair $\langle M, N\rangle$ to be the lambda term $\lambda z.zMN$. We also define two terms **left** $= \lambda p.p(\lambda xy.x)$ and **right** $= \lambda p.p(\lambda xy.y)$. We observe the following:

$$\begin{array}{rcl} \textbf{left}\,\langle M, N\rangle & \twoheadrightarrow_\beta & M \\ \textbf{right}\,\langle M, N\rangle & \twoheadrightarrow_\beta & N \end{array}$$

The terms **left** and **right** are called the left and right *projections*.

**Tuples.** The encoding of pairs easily extends to arbitrary $n$-tuples. If $M_1, \ldots, M_n$ are terms, we define the $n$-tuple $\langle M_1, \ldots, M_n\rangle$ as the lambda term $\lambda z.zM_1 \ldots M_n$, and we define the $i$th projection $\pi_i^n = \lambda p.p(\lambda x_1 \ldots x_n.x_i)$. Then

$$\pi_i^n\langle M_1, \ldots, M_n\rangle \quad \twoheadrightarrow_\beta \quad M_i, \text{for all } 1 \leqslant i \leqslant n.$$

**Lists.** A list is different from a tuple, because its length is not necessarily fixed. A list is either empty (“nil”), or else it consists of a first element (the “head”) followed by another list (the “tail”). We write **nil** for the empty list, and $H :: T$ for the list whose head is $H$ and whose tail is $T$. So, for instance, the list of the first three numbers can be written as $1 :: (2 :: (3 :: \textbf{nil}))$. We usually omit the parentheses, where it is understood that ”::” associates to the right. Note that every list ends in **nil** .

In the lambda calculus, we can define **nil** $= \lambda xy.y$ and $H :: T = \lambda xy.xHT$. Here is a lambda term that adds a list of numbers:

$$\textbf{addlist}\ l = l(\lambda h\,t.\,\textbf{add}\ h(\textbf{addlist}\ t))(\overline{0}).$$

Of course, this is a recursive definition, and must be translated into an actual lambda term by the method of Section 3.3. In the definition of **addlist** , $l$ and $t$ are lists of numbers, and $h$ is a number. If you are very diligent, you can calculate the sum of last weekend’s Canadian lottery results by evaluating the term

$$\textbf{addlist}\,(\overline{4} :: \overline{22} :: \overline{24} :: \overline{32} :: \overline{42} :: \overline{43} :: \textbf{nil}\,).$$

Note that lists enable us to give am alternative encoding of the natural numbers: We can encode a natural number as a list of booleans, which we interpret as the binary digits 0 and 1. Of course, with this encoding, we would have to carefully redesign our basic functions, such as successor, addition, and multiplication.

However, if done properly, such an encoding would be a lot more efficient (in terms of number of $\beta$-reductions to be performed) than the encoding by Church numerals.

**Trees.** A binary tree is a data structure that can be one of two things: either a *leaf*, labeled by a natural number, or a *node*, which has a left and a right subtree. We write **leaf** $(N)$ for a leaf labeled $N$, and **node** $(L, R)$ for a node with left subtree $L$ and right subtree $R$. We can encode trees as lambda terms, for instance as follows:

$$\textbf{leaf}\,(n) = \lambda xy.xn, \qquad \textbf{node}\,(L, R) = \lambda xy.yLR$$

As an illustration, here is a program (i.e., a lambda term) that adds all the numbers at the leafs of a given tree.

$$\textbf{addtree}\ t = t(\lambda n.n)(\lambda l\,r.\,\textbf{add}\,(\textbf{addtree}\ l)(\textbf{addtree}\ r)).$$

**Exercise 11.** This is a voluntary programming exercise.

(a) Write a lambda term that calculates the length of a list.

(b) Write a lambda term that calculates the depth (i.e., the nesting level) of a tree. You may need to define a function **max** that calculates the maximum of two numbers.

(c) Write a lambda term that sorts a list of numbers. You may assume given a term **less** that compares two numbers.

# 4 The Church-Rosser Theorem

## 4.1 Extensionality, $\eta$-equivalence, and $\eta$-reduction

In the untyped lambda calculus, any term can be applied to another term. Therefore, any term can be regarded as a function. Consider a term $M$, not containing the variable $x$, and consider the term $M' = \lambda x.Mx$. Then for any argument $A$, we have $MA =_\beta M'A$. So in this sense, $M$ and $M'$ define "the same function". Should $M$ and $M'$ be considered equivalent as terms?

The answer depends on whether we want to accept the principle that "if $M$ and $M'$ define the same function, then $M$ and $M'$ are equal". This is called the principle of *extensionality*, and we have already encountered it in Section 1.1. Formally, the extensionality rule is the following:

$$(ext_\forall) \quad \frac{\forall A.MA = M'A}{M = M'}.$$

In the presence of the axioms ($\xi$), (*cong*), and ($\beta$), it can be easily seen that $MA = M'A$ is true for *all* terms $A$ if and only if $Mx = M'x$, where $x$ is a fresh variable. Therefore, we can replace the extensionality rule by the following equivalent, but simpler rule:

$$(ext) \quad \frac{Mx = M'x, \text{ where } x \notin FV(M, M')}{M = M'}.$$

Note that we can apply the extensionality rule in particular to the case where $M' = \lambda x.Mx$, where $x$ is not free in $M$. As we have remarked above, $Mx =_\beta M'x$, and thus extensionality implies that $M = \lambda x.Mx$. This last equation is called the $\eta$-law (eta-law):

$$(\eta) \quad M = \lambda x.Mx, \text{ where } x \notin FV(M).$$

In fact, ($\eta$) and (*ext*) are equivalent in the presence of the other axioms of the lambda calculus. We have already seen that (*ext*) and ($\beta$) imply ($\eta$). Conversely, assume ($\eta$), and assume that $Mx = M'x$, for some terms $M$ and $M'$ not containing $x$ freely. Then by ($\xi$), we have $\lambda x.Mx = \lambda x.M'x$, hence by ($\eta$) and transitivity, $M = M'$. Thus (*ext*) holds.

We note that the $\eta$-law does not follow from the axioms and rules of the lambda calculus that we have considered so far. In particular, the terms $x$ and $\lambda y.xy$ are not $\beta$-equivalent, although they are clearly $\eta$-equivalent. We will prove that $x \neq_\beta \lambda y.xy$ in Corollary 4.5 below.

Single-step $\eta$-reduction is the smallest relation $\to_\eta$ satisfying (*cong*$_1$), (*cong*$_2$), ($\xi$), and the following axiom (which is the same as the $\eta$-law, directed left to right):

$$(\eta) \quad M \to_\eta \lambda x.Mx, \text{ where } x \notin FV(M).$$

Single-step $\beta\eta$-reduction $\to_{\beta\eta}$ is defined as the union of the single-step $\beta$- and $\eta$-reductions, i.e., $M \to_{\beta\eta} M'$ iff $M \to_\beta M'$ or $M \to_\eta M'$. Multi-step $\eta$-reduction $\twoheadrightarrow_\eta$, multi-step $\beta\eta$-reduction $\twoheadrightarrow_{\beta\eta}$, as well as $\eta$-equivalence $=_\eta$ and $\beta\eta$-equivalence $=_{\beta\eta}$ are defined in the obvious way as we did for $\beta$-reduction and equivalence. We also get the evident notions of $\eta$-normal form, $\beta\eta$-normal form, etc.

## 4.2 Statement of the Church-Rosser Theorem, and some consequences

**Theorem** (Church and Rosser, 1936)**.** *Let $\twoheadrightarrow$ denote either $\twoheadrightarrow_\beta$ or $\twoheadrightarrow_{\beta\eta}$. Suppose $M$, $N$, and $P$ are lambda terms such that $M \twoheadrightarrow N$ and $M \twoheadrightarrow P$. Then there exists a lambda term $Z$ such that $N \twoheadrightarrow Z$ and $P \twoheadrightarrow Z$.*

In pictures, the theorem states that the following diagram can always be completed:

$$\begin{tikzcd} & M \arrow[dl, two heads] \arrow[dr, two heads] & \\ N \arrow[dr, two heads, dotted] & & P \arrow[dl, two heads, dotted] \\ & Z & \end{tikzcd}$$

This property is called the *Church-Rosser property*, or *confluence*. Before we prove the Church-Rosser Theorem, let us highlight some of its consequences.

**Corollary 4.1.** *If $M =_\beta N$ then there exists some $Z$ with $M, N \twoheadrightarrow_\beta Z$. Similarly for $\beta\eta$.*

*Proof.* Please refer to Figure 1 for an illustration of this proof. Recall that $=_\beta$ is the reflexive symmetric transitive closure of $\to_\beta$. Suppose that $M =_\beta N$. Then there exist $n \geqslant 0$ and terms $M_0, \ldots, M_n$ such that $M = M_0$, $N = M_n$, and

$$\begin{tikzcd}
 & & & M_3 \arrow[dl] \arrow[dr] & & M_5 \arrow[dl] \arrow[dr] & & \\
M_0 \arrow[dr] & & M_2 \arrow[dl] & & M_4 \arrow[ddll, ->>, dotted] & & M_6 \arrow[dr] \arrow[dddlll, ->>, dotted] & \\
 & M_1 \arrow[dr, ->>, dotted] & & & & & & M_7 \arrow[dddlll, ->>, dotted] \\
 & & Z'' \arrow[dr, ->>, dotted] & & & & & \\
 & & & Z' \arrow[dr, ->>, dotted] & & & & \\
 & & & & Z & & &
\end{tikzcd}$$

Figure 1: The proof of Corollary 4.1

for all $i = 1 \ldots n$, either $M_{i-1} \to_\beta M_i$ or $M_i \to_\beta M_{i-1}$. We prove the claim by induction on $n$. For $n = 0$, we have $M = N$ and there is nothing to show. Suppose the claim has been proven for $n-1$. Then by induction hypothesis, there exists a term $Z'$ such that $M \twoheadrightarrow_\beta Z'$ and $M_{n-1} \twoheadrightarrow_\beta Z'$. Further, we know that either $N \to_\beta M_{n-1}$ or $M_{n-1} \to_\beta N$. In case $N \to_\beta M_{n-1}$, then $N \twoheadrightarrow_\beta Z'$, and we are done. In case $M_{n-1} \to_\beta N$, we apply the Church-Rosser Theorem to $M_{n-1}$, $Z'$, and $N$ to obtain a term $Z$ such that $Z' \twoheadrightarrow_\beta Z$ and $N \twoheadrightarrow_\beta Z$. Since $M \twoheadrightarrow_\beta Z' \twoheadrightarrow_\beta Z$, we are done. The proof in the case of $\beta\eta$-reduction is identical. $\square$

**Corollary 4.2.** *If $N$ is a $\beta$-normal form and $N =_\beta M$, then $M \twoheadrightarrow_\beta N$, and similarly for $\beta\eta$.*

*Proof.* By Corollary 4.1, there exists some $Z$ with $M, N \twoheadrightarrow_\beta Z$. But $N$ is a normal form, thus $N =_\alpha Z$. $\square$

**Corollary 4.3.** *If $M$ and $N$ are $\beta$-normal forms such that $M =_\beta N$, then $M =_\alpha N$, and similarly for $\beta\eta$.*

*Proof.* By Corollary 4.2, we have $M \twoheadrightarrow_\beta N$, but since $M$ is a normal form, we have $M =_\alpha N$. $\square$

**Corollary 4.4.** *If $M =_\beta N$, then neither or both have a $\beta$-normal form. Similarly for $\beta\eta$.*

*Proof.* Suppose that $M =_\beta N$, and that one of them has a $\beta$-normal form. Say, for instance, that $M$ has a normal form $Z$. Then $N =_\beta Z$, hence $N \twoheadrightarrow_\beta Z$ by Corollary 4.2. $\square$

**Corollary 4.5.** *The terms $x$ and $\lambda y.xy$ are not $\beta$-equivalent. In particular, the $\eta$-rule does not follow from the $\beta$-rule.*

*Proof.* The terms $x$ and $\lambda y.xy$ are both $\beta$-normal forms, and they are not $\alpha$-equivalent. It follows by Corollary 4.3 that $x \neq_\beta \lambda y.xy$. $\square$

## 4.3 Preliminary remarks on the proof of the Church-Rosser Theorem

Consider any binary relation $\to$ on a set, and let $\twoheadrightarrow$ be its reflexive transitive closure. Consider the following three properties of such relations:

(a)
$$\begin{tikzcd}
 & M \arrow[dl, ->>] \arrow[dr, ->>] & \\
N \arrow[dr, ->>, dotted] & & P \arrow[dl, ->>, dotted] \\
 & Z &
\end{tikzcd}$$

(b)
$$\begin{tikzcd}
 & M \arrow[dl] \arrow[dr] & \\
N \arrow[dr, ->>, dotted] & & P \arrow[dl, ->>, dotted] \\
 & Z &
\end{tikzcd}$$

(c)
$$\begin{tikzcd}
 & M \arrow[dl] \arrow[dr] & \\
N \arrow[dr, dotted] & & P \arrow[dl, dotted] \\
 & Z &
\end{tikzcd}$$

Each of these properties states that for all $M, N, P$, if the solid arrows exist, then there exists $Z$ such that the dotted arrows exist. The only difference between (a), (b), and (c) is the difference between where $\to$ and $\twoheadrightarrow$ are used.

Property (a) is the Church-Rosser property. Property (c) is called the diamond property (because the diagram is shaped like a diamond).

A naive attempt to prove the Church-Rosser Theorem might proceed as follows: First, prove that the relation $\to_\beta$ satisfies property (b) (this is relatively easy to prove); then use an inductive argument to conclude that it also satisfies property (a).

Unfortunately, this does not work: the reason is that in general, property (b) does not imply property (a)! An example of a relation that satisfies property (b) but not

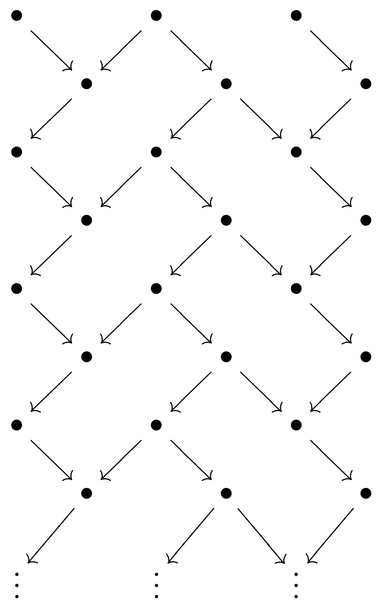

Figure 2: An example of a relation that satisfies property (b), but not property (a)

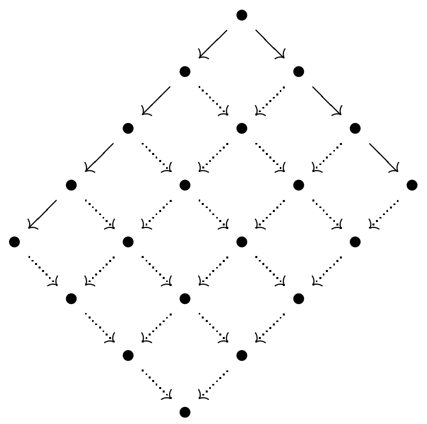

Figure 3: Proof that property (c) implies property (a)

property (a) is shown in Figure 2. In other words, a proof of property (b) is not sufficient in order to prove property (a).

On the other hand, property (c), the diamond property, *does* imply property (a). This is very easy to prove by induction, and the proof is illustrated in Figure 3. But unfortunately, $\beta$-reduction does not satisfy property (c), so again we are stuck.

To summarize, we are faced with the following dilemma:

- $\beta$-reduction satisfies property (b), but property (b) does not imply property (a).
- Property (c) implies property (a), but $\beta$-reduction does not satisfy property (c).

On the other hand, it seems hopeless to prove property (a) directly. In the next section, we will solve this dilemma by defining yet another reduction relation $\rhd$, with the following properties:

- $\rhd$ satisfies property (c), and
- the transitive closure of $\rhd$ is the same as that of $\to_\beta$ (or $\to_{\beta\eta}$).

## 4.4 Proof of the Church-Rosser Theorem

In this section, we will prove the Church-Rosser Theorem for $\beta\eta$-reduction. The proof for $\beta$-reduction (without $\eta$) is very similar, and in fact slighly simpler, so we omit it here. The proof presented here is due to Tait and Martin-Löf. We begin by defining a new relation $M \rhd M'$ on terms, called *parallel one-step reduction*. We define $\rhd$ to be the smallest relation satisfying

$$
\begin{array}{ll}
(1) & \dfrac{}{x \rhd x} \\[2ex]
(2) & \dfrac{P \rhd P' \qquad N \rhd N'}{PN \rhd P'N'} \\[2ex]
(3) & \dfrac{N \rhd N'}{\lambda x.N \rhd \lambda x.N'} \\[2ex]
(4) & \dfrac{Q \rhd Q' \qquad N \rhd N'}{(\lambda x.Q)N \rhd Q'[N'/x]} \\[2ex]
(5) & \dfrac{P \rhd P',\ \text{where } x \notin FV(P)}{\lambda x.Px \rhd P'}.
\end{array}
$$