Basic results on λ-calculus

Review: λ-calculus syntax

- Let's review the basic notions of λ-calculus from last time.
- If V is a set of variables the set of valid λ-terms Λ is described by the grammar

$$\Lambda = V \mid \Lambda\Lambda \mid (\lambda V.\Lambda)$$

- We think of

$$\lambda x.U$$

as a function that takes x to U (where x can appear in U).

Currying

- As a shorthand, we write

$$\lambda x.(\lambda y.U) \quad \text{as} \quad \lambda xy.\, U$$

- The idea (known as "currying") is that a function that returns a function can be thought of as a function of two variables.

Bound and free variables

- Any variable that appears next to the λ sign is called a **bound** variable

(technically the λ introduces a **binder**, and it is the occurrences in scope that are **bound**, but we lose nothing with this abuse of language)

- All other variables are called **free** variables.
- Thus in

$$\lambda xy.\, xyz$$

the variables x and y are bound, and z is free.

Bound variables in mathematics

- We can think of free variables as constants outside of our control.
- We regularly encounter bound variables in mathematics:

$$\int_0^1 f(x)\,dx \qquad \lim_{x\to 0} f(x) \qquad \sum_{i=1}^{n} a_i$$

- In each expression, the variable (x or i) is a placeholder — renaming it doesn't change the meaning.

α-conversion

- Renaming bound variables does not change the meaning of the λ-term:

$$\lambda x.\, xy \quad = \quad \lambda u.\, uy$$

- One can reuse bound variable names:

$$\lambda x.\, (\lambda x.\, x) \quad = \quad \lambda u.\, (\lambda x.\, x)$$

  Both represent a function that returns the identity.

- We identify λ-terms that differ only by renaming bound variables. This process is called α-**conversion**.

In theoretical treatments, we often assume all bound variables have distinct names (the **Barendregt convention**) to simplify proofs about substitution. In implementations, techniques like de Bruijn indices or unique identifier generation serve a similar purpose.

β-reduction

- The computational rule of λ-calculus is β-**reduction**, which takes any expression of the form

$$(\lambda x.N)\, M$$

and evaluates it to

$$N[x := M]$$

i.e., the λ-term with every *free* instance of x in N replaced by M.

- **IMPORTANT** : If you have something like A B C the computation is always left-associative, i.e

$$ABC = (AB)C$$
$$ABCD = ((AB)C)D$$

Encoding computation

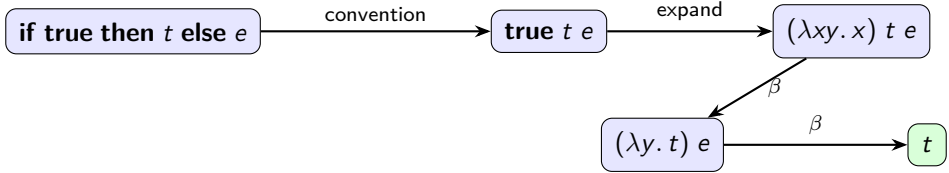
- λ-calculus consists of a syntax (valid expressions) and a single evaluation rule.
- To use it as a computational model, we first encode what we want to express:

$$\begin{aligned} \textbf{true} &= \lambda xy.\, x \\ \textbf{false} &= \lambda xy.\, y \\ \textbf{if } b \textbf{ then } t \textbf{ else } e &= b\, t\, e \end{aligned}$$

- The conditional simply applies b to both branches — the boolean selects which one to return.

Verifying the encoding

We must verify our encoding behaves as intended:



Similarly: **false** $t\ e = (\lambda xy.\, y)\, t\, e \to_\beta (\lambda y.\, y)\, e \to_\beta e$ ✓

Semantics vs syntax

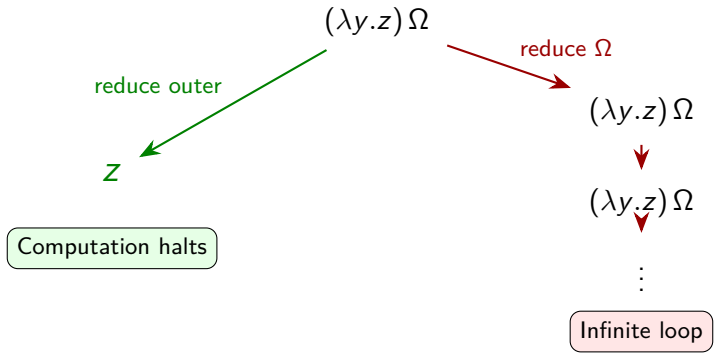
- We see here a fundamental concern of logic: the consistency between **semantics** (intended meaning) and **syntax** (formal rules). This connection will resurface when we study the relationship between λ-calculus and logic.
- For our boolean encoding, it's easy to verify the match.
- But for more complex systems, there might be a mismatch we fail to see — this would render our computations meaningless.

The consistency problem

- A basic worry: perhaps λ-calculus is itself inconsistent, and evaluating in different orders gives different results.
- If so, it wouldn't matter what our conventions are — the entire system would be useless!
- And this worry is justified...

An important example

Let $\Omega := (\lambda x.\, x\, x)(\lambda x.\, x\, x)$. Note that reducing Ω yields Ω again.



Different reduction orders give different behaviors! Can we still trust the results?

Two key questions

This example raises two fundamental questions:

1. **When does computation conclude?**
   - We need a notion of "final answer"
2. **Is the final answer unique?**
   - If different reduction paths could yield different final answers, the calculus would be useless

Today we'll answer both questions.

Toward a notion of "final answer"

- We will now define a **normal form** for λ-terms — a state that can reasonably be considered the final result of a computation.
- For this definition we need the notion of a **subterm**.
- For this purpose it is useful to realize that λ-terms can be decomposed as trees.

λ-terms as trees

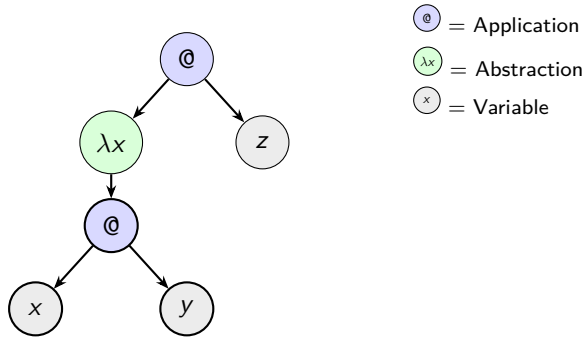
Every λ-term can be represented as a tree:

- **Variables** x are leaves
- **Applications** $M\,N$ are binary nodes with M and N as children
- **Abstractions** $\lambda x.M$ are unary nodes with M as child

A **subterm** is any term corresponding to a subtree.

Tree decomposition: example

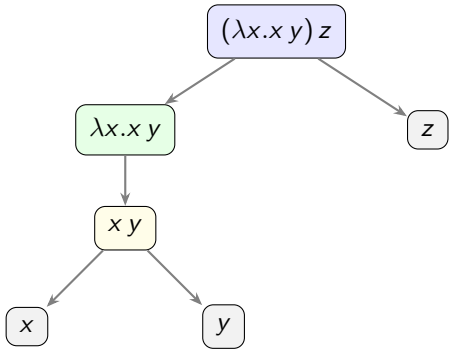
The term $(\lambda x.x\,y)\,z$ as a tree:



This is a **redex**: the root is an application whose left child is a λ-abstraction, i.e., a redex is any λ-term of the form $(\lambda x.N)\,M$ (a *red*ucible *ex*pression).

Subterms

The **subterms** of $(\lambda x.x\ y)\ z$ are all the subtrees:

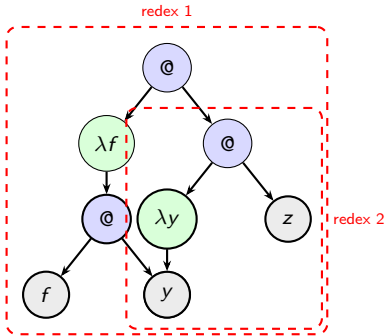


6 subterms total:

1. $(\lambda x.x\ y)\ z$
2. $\lambda x.x\ y$
3. z
4. $x\ y$
5. x
6. y

Finding redexes in a term

A term is in β-**normal form** if no subterm is a redex.



The term $(\lambda f.f\,x)\,((\lambda y.y)\,z)$ has **two redexes** and is **not** in β-normal form.

We think of β-normal forms as the final outcome of a computation — once we reach this stage, the computation must stop.

Examples

- The following are in β-normal form

$$xy \ , \ x(\lambda u.u)z$$

The second example is in normal form because it is interpreted as

$$(x(\lambda u.u))z$$

and $x(\lambda u.u)$ has no further reduction.

- The following is not in β-normal form

$$x((\lambda u.u)z) \ , \ (\lambda u.xu)v$$

The second example is obvious, but the first is not in β-normal form because we have indicated with paranthesis that we should view $x((\lambda u.u)z)$ as MN with $M = x$ and $N = (\lambda u.u)z$ and N is a redex that further reduces to z.

Can we always reach a normal form?

- We now have a notion of "final answer" — the β-normal form.
- But can we always reach it? And if so, is it unique?
- To address these questions precisely, we need some notation.

Notation

- If N is obtained from M by a **single** β-reduction step:

$$M \to_\beta N$$

- If N can be obtained from M by a (finite) **chain** of β-reductions:

$$M \to_\beta M_1 \to_\beta M_2 \to_\beta \cdots \to_\beta N$$

  then we write $M \twoheadrightarrow_\beta N$ (multi-step reduction).

- We think of β-normal forms as final answers — once reached, computation must stop.

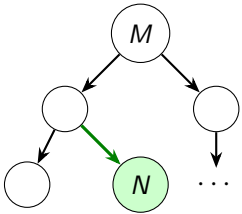
Normalizable terms

- A λ-term M is **weakly normalizable** if there exists some reduction path to a normal form.
- A λ-term M is **strongly normalizable** if every reduction path terminates (no infinite paths exist).
- Strongly normalizable $\Rightarrow$ weakly normalizable (but not conversely).

Weak vs strong normalization

Weakly Normalizable

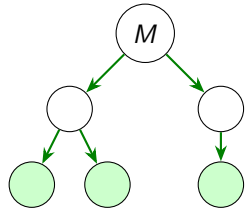
$\exists$ a path to normal form



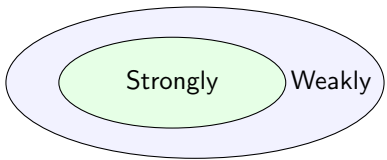
At least one path terminates.

Strongly Normalizable

All paths terminate



No infinite reduction paths.



Strongly normalizable $\subsetneq$ Weakly normalizable

β-equivalence

- Calculations don't have to go forward only.
- Two terms M and N are **β-equivalent**, written $M =_\beta N$, if there is a chain where each step is either $\to_\beta$ or $\leftarrow_\beta$:

$$M := M_0 \to_\beta M_1 \leftarrow_\beta M_2 \to_\beta M_3 \to_\beta M_4 := N$$

- A λ-term M **has a β-normal form** if $M =_\beta N$ for some N in β-normal form.

Today's main results

Theorem (Main Theorem)

1. *If $M =_\beta N$ for some N in β-normal form, then $M \twoheadrightarrow_\beta N$.*
2. *For any λ-term M, there is at most one β-normal form.*

Significance:

- Normal forms can always be reached by forward calculation
- The result is unique (if it exists)
- If no normal form exists, the computation loops forever

The Church-Rosser theorem

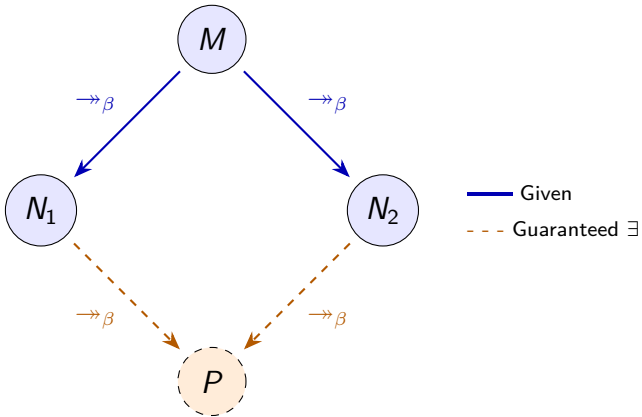
Our main tool is the **Church-Rosser theorem** (also called **confluence**).

Theorem (Church-Rosser)

If $M \twoheadrightarrow_\beta N_1$ *and* $M \twoheadrightarrow_\beta N_2$*, then there exists* P *such that* $N_1 \twoheadrightarrow_\beta P$ *and* $N_2 \twoheadrightarrow_\beta P$*.*

Intuition: Reduction is like water flowing downhill — different paths may diverge, but they can always rejoin. No matter which redexes you choose to reduce, you can never paint yourself into a corner.

Church-Rosser: the diamond property



We will prove Church-Rosser next time. For now, let's use it.

Result 1: Uniqueness of normal forms

Theorem

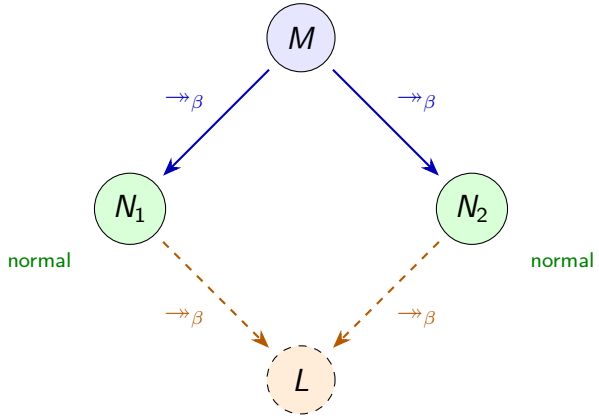
If $M \twoheadrightarrow_\beta N_1$ *and* $M \twoheadrightarrow_\beta N_2$ *with* N_1, N_2 *in* β*-normal form, then* $N_1 = N_2$.

Proof: By Church-Rosser, $\exists L$ with $N_1 \twoheadrightarrow_\beta L$ and $N_2 \twoheadrightarrow_\beta L$.

But N_1 is in normal form, so $N_1 = L$. Similarly $N_2 = L$.

Therefore $N_1 = N_2$. $\square$

Uniqueness: diagram



N_1, N_2 normal $\Rightarrow N_1 = L = N_2$

Result 2: Forward calculation

Theorem

If $M =_\beta N$ and N is in β-normal form, then $M \twoheadrightarrow_\beta N$.

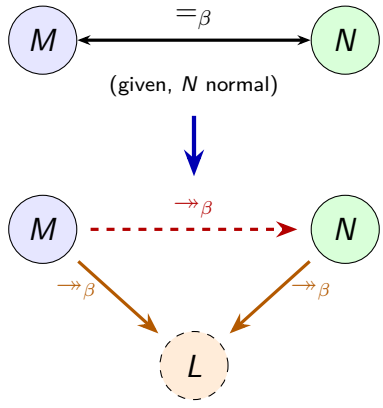
To prove this, we need an auxiliary lemma:

Lemma (Auxiliary)

If $M =_\beta N$, then $\exists L$ such that $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$.

Proof of theorem (assuming lemma): Given $M =_\beta N$ with N normal, the lemma gives us L with $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$. Since N is normal, $N = L$. Thus $M \twoheadrightarrow_\beta N$. $\square$

Summary: from equivalence to reduction

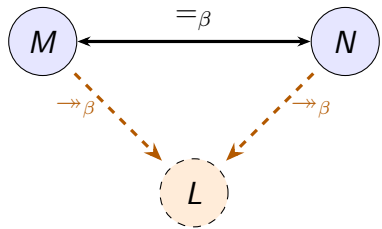


Lemma gives L. But N normal $\Rightarrow N = L \Rightarrow M \twoheadrightarrow_\beta N$

Proving the auxiliary lemma

Lemma

If $M =_\beta N$, then $\exists L$ such that $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$.



Proof: By induction on the length of the $=_\beta$ chain.

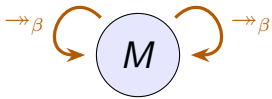
The conversion chain

$$M \longrightarrow M_1 \longleftarrow M_2 \quad \cdots \quad M_k \longrightarrow N$$

- Each arrow is either $\to_\beta$ or ${}_\beta\!\leftarrow$
- Chain length $= k + 1$ (number of steps)

Base Case: Chain Length = 0

$M = N$ (identical terms)



$L = M = N$

(0 reduction steps each)

Take $L = M = N$. Both reduce to L in zero steps. ✓

Inductive Step: Chain Length $= n$

Assume true for chains of length $< n$.

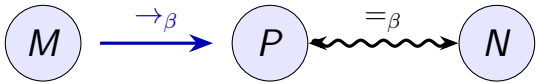
The first step is either:

Case 1:
$M \to_\beta P$
(forward step)

Case 2:
$M \,{}_\beta\!\leftarrow P$
(backward step)

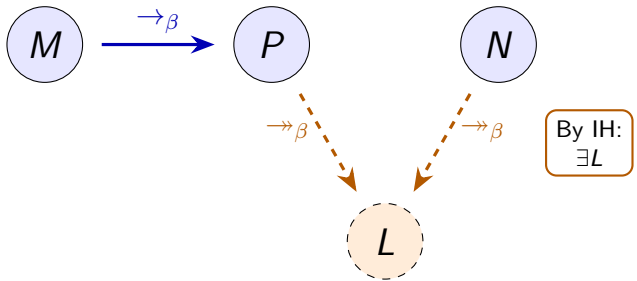
In both cases, $P =_\beta N$ with a shorter chain (length $n - 1$).

Case 1: $M \to_\beta P$ — Setup

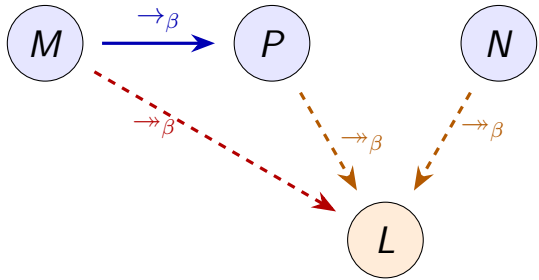


The chain $P =_\beta N$ has length $n - 1$.

Case 1: $M \to_\beta P$ — Apply IH

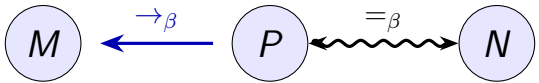


Case 1: $M \to_\beta P$ — Conclusion



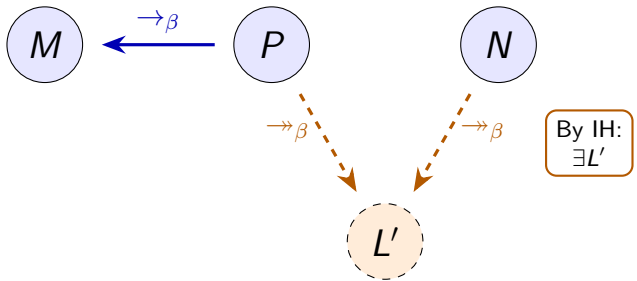
Since $M \to_\beta P \twoheadrightarrow_\beta L$, we have $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$. ✓

Case 2: $M \ {}_{\beta}\!\leftarrow P$ — Setup

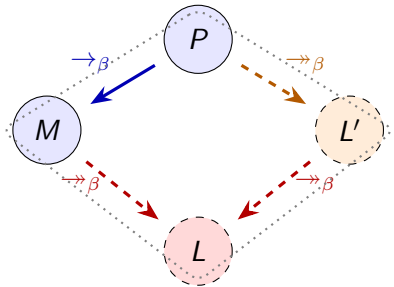


The chain $P =_\beta N$ has length $n - 1$.

Case 2: $M \ {}_{\beta}\!\leftarrow P$ — Apply IH



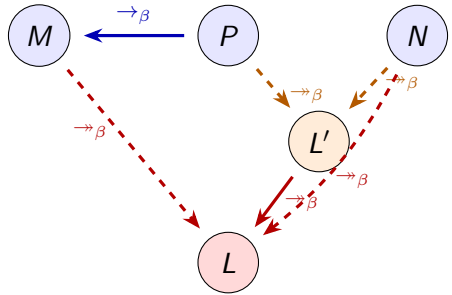
Case 2: $M \ {}_{\beta}\!\leftarrow P$ — Apply Church-Rosser



By CR:
$\exists L$

Apply Church-Rosser to $P \rightarrow_\beta M$ and $P \twoheadrightarrow_\beta L'$.

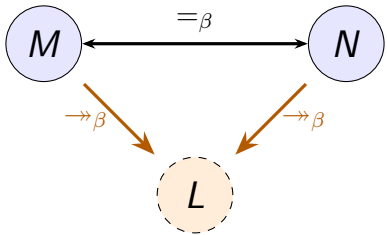
Case 2: $M \ {}_{\beta}\!\leftarrow P$ — Conclusion



$M \twoheadrightarrow_\beta L$ by CR. $N \twoheadrightarrow_\beta L' \twoheadrightarrow_\beta L$. ✓

Proof complete

By induction, for all $M =_\beta N$, there exists L with $M \twoheadrightarrow_\beta L$ and $N \twoheadrightarrow_\beta L$.



□

Summary of results

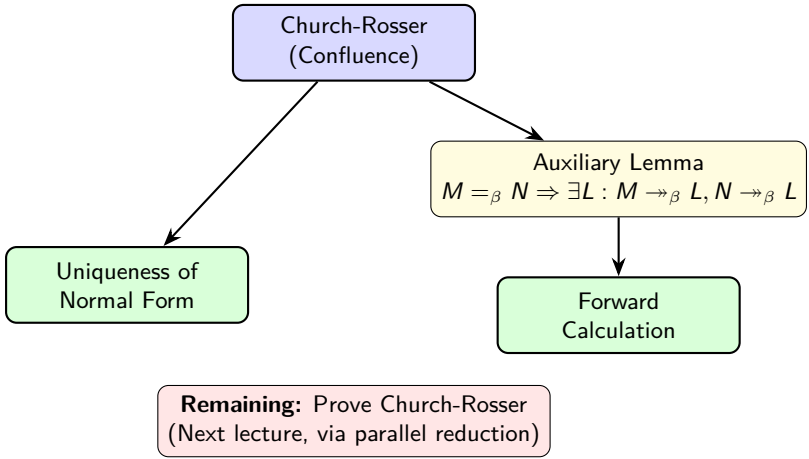
Theorem (Main Results)

1. ***Uniqueness:*** *Any λ-term has at most one β-normal form.*
2. ***Forward calculation:*** *If $M =_\beta N$ with N normal, then $M \twoheadrightarrow_\beta N$.*

Consequences:

- β-normal forms are well-defined "final answers"
- We can always compute forward to reach them
- Different reduction strategies may take different paths, but the destination is unique

Logical dependencies



Next time

Proving the Church-Rosser theorem

The proof uses a clever technique:

1. Define **parallel reduction** $\Rrightarrow_\beta$ (reduce multiple redexes simultaneously)
2. Show parallel reduction satisfies the **diamond property** directly
3. Show $\twoheadrightarrow_\beta$ is the transitive closure of $\Rrightarrow_\beta$
4. Conclude Church-Rosser for $\twoheadrightarrow_\beta$

This is one of the most elegant proofs in the theory of computation!