Introduction to Lambda Calculus

Theory of Formal Languages · Lecture 16

Prof. K.R. Chowdhary

What does “computable” mean?

1 Turing

Turing machine

A function is computable iff an idealized machine can compute it.

2 Gödel

General recursive functions

Smallest class containing constants and successor, closed under composition and recursion.

3 Church

Lambda calculus

A function is computable iff it can be written as a lambda term.

Church–Turing thesis: all three models compute exactly the same functions, and this class is taken to match “intuitive” computability.

Why λ-calculus matters

λ

Functions as terms

Functions are finite expressions, and any term can take itself as input.

λ

Purely syntactic

One rewrite rule (β-conversion) plus function definition. No machine details.

λ

Foundation for languages

Denotational semantics grew out of it, and compilers and type systems use it.

λ-calculus + extras

data types, I/O, side effects, updateable memory, objects

Lisp

Scheme

Haskell

ML

Syntax: three building blocks

x	**Variable**	A parameter or value.
$(\lambda x.M)$	**Abstraction**	A function definition. x becomes bound in M.
$(M\ N)$	**Application**	Function M applied to argument N.

Conventions

Application associates left: $P\ Q\ R = ((P\ Q)\ R)$ **Abstraction associates right:** $\lambda x\ y.\ x\ y\ x = \lambda x.(\lambda y.((x\ y)\ x))$

β-reduction: the one rule

$$(\lambda x.M)\ N \rightarrow M[x := N]$$

Substitute N for every free occurrence of x in the body M.

Normal form: a term with no subterm of the form (λx.M) N.

$(\lambda x.x)\ a \rightarrow_\beta a$

The identity function returns its argument.

$(\lambda x.y)\ a \rightarrow_\beta y$

x is unused, so a is discarded.

$(\lambda x.\ x+1)\ 5 \rightarrow_\beta 5 + 1 = 6$

Extended with arithmetic for illustration.

Currying and Church–Rosser

Multiple arguments: one at a time

$$(\lambda n.\lambda m.\ n^2+m)\ 3\ 5$$

↓

$$(\lambda m.\ 3^2+m)\ 5$$

↓

$$3^2 + 5 = 14$$

Evaluation order doesn't matter

$$(7+4) * (8+5*3)$$

Left first: $11 * (8+5*3) \rightarrow 11 * 23$

Right first: $(7+4) * (8+15) \rightarrow 11 * 23$

Both reach normal form: 253

Key takeaways

1. λ-calculus is a Turing-complete model built from variables, abstraction and application.
2. β-reduction is the single computation rule: substitute the argument into the body.
3. Church–Rosser: the normal form does not depend on evaluation order.
4. Functional languages like Lisp, Scheme, Haskell and ML extend the λ-calculus.