λ-Calculus

Normal forms & evaluation strategies

Normal forms

Redex: Left hand side of a reduction rule.

A λ-expression with no redex is said to be in **normal form.**
It is in **weak head normal form (WHNF)** if it has a λ-abstraction (or constructor or partially applied built in function in Haskell) at the top level.

Computation: Reduction of a λ-expression to normal form (or WHNF)

Church-Rosser Theorem: The λ-calculus is **confluent**, i.e. (informally) the order of reduction steps doesn't matter, in particular any two reductions to normal form will reach equivalent normal forms. (However, some evaluation strategies—see below—might not find it at all or take a long time to do so.)

Evaluation strategies

Applicative order	:= right-most/inner-most redex first
Call-by-value	:= like applicative order, but no reductions inside λ-abstractions
Normal order	:= left-most/outer-most redex first
Call-by-name	:= like normal order, but stop at WHNF (e.g. no reductions inside λ-abstractions, etc.)
Call-by-need	:= call-by-name + memoization
Lazy evaluation	:= An implementation of call-by-need with **thunks** used by Haskell. A **thunk** is an object in heap memory storing the state of the delayed evaluation of a function.