Normalization for Typed Lambda Calculi

Lecture 1: Simple Type Theory and Weak Normalization

H. Geuvers

Radboud University
Nijmegen, NL

ISR 2026
July 2026

Outline

Simple Type Theory

Properties of $\lambda\!\rightarrow$

Weak normalization for Simple Type Theory

Simple type theory

Theory of simple types: $\lambda\!\!\to$ or simple type theory, STT. Just arrow types

$$\mathsf{Typ} := \mathsf{TVar} \mid (\mathsf{Typ} \to \mathsf{Typ})$$

- Examples: $(\alpha \to \beta) \to \alpha$, $(\alpha \to \beta) \to ((\beta \to \gamma) \to (\alpha \to \gamma))$
- Brackets associate to the right and outside brackets are omitted:
  $(\alpha \to \beta) \to (\beta \to \gamma) \to \alpha \to \gamma$
- Types are denoted by $A, B, \ldots$.

Simple type theory à la Church

Formulation with contexts to declare the free variables:

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$$

is a context, usually denoted by Γ.

DEFINITION The derivation rules of $\lambda\!\to$ (à la Church) are as follows. This gives an inductive definition of the derivable judgments $\Gamma \vdash M : A$.

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B} \qquad \frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x : A.\, P : A \to B}$$

$\Gamma \vdash_{\lambda\to} M : A$ if there is a derivation using these rules with conclusion $\Gamma \vdash M : A$

Examples of typable terms

$$\begin{array}{rcl} \lambda x : A.\, \lambda y : B.\, x & : & A \to B \to A \\ \lambda x : A \to B.\, \lambda y : B \to C.\, \lambda z : A.\, y\,(x\,z) & : & (A \to B) \to (B \to C) \to A \to C \\ \lambda x : A.\, \lambda y : (B \to A) \to A.\, y(\lambda z : B.\, x) & : & A \to ((B \to A) \to A) \to A \end{array}$$

Not for every type there is a closed term of that type:

$(\alpha \to \alpha) \to \alpha$ is not inhabited

That is: there is no term M such that

$$\vdash M : (\alpha \to \alpha) \to \alpha.$$

Typed Terms versus Type Assignment

- With typed terms also called typing à la Church, we have terms with type information in the λ-abstraction

$$\lambda x : A.\, x \;\; : \;\; A \to A$$

  - Terms have unique types,
  - The type is directly computed from the type info in the variables.
- With typed assignment also called typing à la Curry, we assign types to untyped λ-terms

$$\lambda x.\, x : A \to A$$

  - Terms do not have unique types,
  - A principal type can be computed using unification.

Simple Type Theory à la Church and à la Curry

$\lambda{\to}$ (à la Church):

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \qquad \frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x{:}\,A.\,P : A \to B}$$

$\lambda{\to}$ (à la Curry):

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \qquad \frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x.\,P : A \to B}$$

Examples

- Typed Terms (Church style):

  $$\lambda x : \alpha.\, \lambda y : (\beta \to \alpha) \to \alpha.\, y(\lambda z : \beta.\, x)$$

  has only the type $\alpha \to ((\beta \to \alpha) \to \alpha) \to \alpha$
- Type Assignment (Curry):

  $$\lambda x.\, \lambda y.\, y(\lambda z.\, x)$$

  can be assigned the types
  - $\alpha \to ((\beta \to \alpha) \to \alpha) \to \alpha$
  - $\alpha \to ((\beta \to \alpha) \to \gamma) \to \gamma$
  - $(\alpha \to \alpha) \to ((\beta \to \alpha \to \alpha) \to \gamma) \to \gamma$
  - $\ldots$

  with $\alpha \to ((\beta \to \alpha) \to \gamma) \to \gamma$ being the principal type

Church vs. Curry typing

- The Curry formulation is especially interesting for (functional) programming:
  - you want to write as little type information as possible;
  - let the compiler infer the types for you.
- The Church formulation is especially interesting for proof checking:
  - terms are created interactively;
  - if one adds dependent types, the type structure is so intricate that type inference is undecidable (if you start from an untyped term).

Typical problems one would like to have an algorithm for

$\Gamma \vdash M : A?$	Type Checking Problem	TCP
$\Gamma \vdash M : ?$	Type Synthesis Problem	TSP
$\Gamma \vdash ? : A$	Type Inhabitation Problem	TIP

For $\lambda\!\rightarrow$, all these problems are decidable, both for Curry style and for Church style.
For Curry style TSP is usually called the Type Assignment Problem, TAP.

- TCP and TSP are equivalent: To solve $M\,N : A$, one has to solve $N :?$
- For Curry systems, TCP and TSP soon become undecidable beyond $\lambda\!\rightarrow$.
- TIP is undecidable for most extensions of $\lambda\!\rightarrow$, as it corresponds to provability in some logic. For $\lambda\!\rightarrow$, TIP is PSPACE complete.

Meta-theory for $\lambda\!\rightarrow$

THEOREM

- For $\lambda\!\rightarrow$ à la Church: Uniqueness of types
  If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A = B$.
- Subject Reduction or Closure
  If $\Gamma \vdash M : A$ and $M \rightarrow_{\beta\eta} N$, then $\Gamma \vdash N : A$.
- Strong Normalization
  If $\Gamma \vdash M : A$, then all $\beta\eta$-reductions from M terminate.

Extension with other connectives

Adding product types $\times$ to $\lambda{\rightarrow}$. (Proposition logic with conjunction $\wedge$.)

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_1 M : A} \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_2 M : B} \qquad \frac{\Gamma \vdash P : A \quad \Gamma \vdash Q : B}{\Gamma \vdash \langle P, Q \rangle : A \times B}$$

With reduction rules

$$\begin{array}{rcl} \pi_1 \langle P, Q \rangle & \rightarrow & P \\ \pi_2 \langle P, Q \rangle & \rightarrow & Q \end{array}$$

Similar rules can be given for sum-types $A + B$, corresponding to disjunction $A \vee B$.

Normalization of β

- Normal form A term M is in NF if there is no reduction step from M: $\neg\exists P(M \to_\beta P)$
- Weak Normalization A term M is WN if there is a reduction $M \to_\beta M_1 \to_\beta M_2 \to_\beta \ldots \to_\beta M_n$ with $M_n \in$ NF.
- Strong Normalization A term M is SN if there are no infinite reductions starting from M $\neg\exists(P_i)_{i\in\mathbb{N}}(M = P_0 \to_\beta P_1 \to_\beta P_2 \to_\beta \ldots)$.

We can give inductive definitions of NF, WN and SN.

Recap: relation between NF, WN and SN

Intermezzo: different definitions of "strong normalization"

M is SN if there are no infinite reductions starting from M
$\neg\exists(P_i)_{i\in\mathbb{N}}(M = P_0 \to_\beta P_1 \to_\beta P_2 \to_\beta \ldots)$
$\iff$ (classically) all β-reductions from M lead to a normal form
$\overset{??}{\iff}$ there is an upperbound k on the length of β-reductions from M.

Define $M \in \mathrm{SN}'$ as

$$\exists k \forall n, \forall P_1, \ldots, P_n (M = P_1 \to \ldots \to P_n) \implies n < k$$

- $M \in \mathrm{SN}' \implies M \in \mathrm{SN}$. (clearly, if there is a bound, then there is no infinite reduction)
- $M \in \mathrm{SN} \implies M \in \mathrm{SN}'$??
  Not in general for rewriting systems, but it holds for λ-calculus, because reduction in λ-calculus is finitely branching.

Proving (weak/strong) normalization of β

SN (or WN) for $\lambda\!\to$ cannot be proved by induction on the derivation

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B}$$

IH: M is SN and N is SN. So $M\,N$ is SN ??

No, e.g. $M = \lambda x.\, x\,x$, $N = \lambda x.\, x\,x$

Similarly for WN, the immediate induction proof fails.

We need an "induction loading": prove a stronger property that implies SN

Normalization of β for $\lambda\!\rightarrow$

Note:

- Terms may get larger under reduction
  $(\lambda f.\, \lambda x.\, f\,(f\,x))\,P \rightarrow_\beta \lambda x.\, P\,(P\,x)$
- Redexes may get multiplied under reduction.
  $(\lambda f.\lambda x.f\,(f\,x))((\lambda y.M)\,Q) \rightarrow_\beta \lambda x.((\lambda y.M)\,Q)\,(((\lambda y.M) Q)\,x)$
- New redexes may be created under reduction.
  $(\lambda f.\, \lambda x.\, f\,(f\,x))\,(\lambda y.\, N) \rightarrow_\beta \lambda x.\,(\lambda y.\, N)\,((\lambda y.\, N)\,x)$

First: Weak Normalization

- Weak Normalization: there is a reduction sequence that terminates,
- Strong Normalization: all reduction sequences terminate.

Redex creation

General property for (untyped) λ-calculus:
There are three ways in which a "new" β-redex can be created.

- Creation

$$(\lambda x.\ \ldots x\, P \ldots)\,(\lambda y.\, Q) \to_\beta \ldots (\lambda y.\, Q)\, P \ldots$$

- Multiplication

$$(\lambda x.\ \ldots x \ldots x \ldots)\,((\lambda y.\, Q)\, P) \to_\beta \ldots (\lambda y.\, Q)\, P \ldots (\lambda y.\, Q)\, P \ldots$$

- Identity

$$(\lambda x.\, x)\,(\lambda y.\, Q)\, P \to_\beta (\lambda y.\, Q)\, P$$

The height of a type and of a redex

The present proof of WN for $\lambda\!\rightarrow$ is originally from Turing, first published by Gandy (1980).

Definition.
The height (or order) of a type $h(A)$ is defined by

- $h(\alpha) := 0$
- $h(A_1 \to \ldots \to A_n \to \alpha) := \max(h(A_1), \ldots, h(A_n)) + 1.$

NB [Exercise] This is the same as defining

- $h(A \to B) := \max(h(A) + 1, h(B)).$

Definition.
The height of a redex $(\lambda x : A.P)Q$ is the height of the type of $\lambda x : A.P$

A measure on typed terms

DEFINITION.

We give a measure m to the terms by defining $m(N) := (h(N), \#N)$ with

- $h(N) =$ the maximum height of a redex in N,
- $\#N =$ the number of redexes of height $h(N)$ in N.

The measures of terms are ordered lexicographically:

$$(h_1, m) <_\ell (h_2, n) \text{ iff } h_1 < h_2 \text{ or } (h_1 = h_2 \text{ and } m < n).$$

Example:

$$(\lambda f : \alpha \to \alpha.\ \lambda y : \alpha.\ \mathbf{I}_1\, f\, (f\, (\mathbf{I}_2\, y)))\, (\mathbf{I}_3\, \mathbf{I}_4)$$

Weak Normalization proof (I)

Lemma.

- Every typed term N that is not in normal form contains a innermost redex of maximal height.
  This is a redex R of maximal height that does not contain another redex of maximal height.
- If $N \to_\beta N'$ by contracting an innermost redex of maximal height, we have $m(N) >_\ell m(N')$.

Proof

Here we only prove the second point. The crucial point is that if we contract an innermost redex of maximal height, we do not create a new redex of maximal height.
To verify this, we have to consider the three ways in which a new β-redexes can be created and prove that the measure decreases.

Weak Normalization proof (II)

Continuation of the proof, analysing the three cases for "redex creation".

- Creation

$$(\lambda x : A.\ \ldots x\, P \ldots)(\lambda y : B.\, Q) \rightarrow_\beta \ldots (\lambda y : B.\, Q) P \ldots$$

- Multiplication

$$(\lambda x : A.\ \ldots x \ldots x \ldots)((\lambda y : B.\, Q) P) \rightarrow_\beta \ldots (\lambda y : B.\, Q) P \ldots (\lambda y : B.\, Q) P$$

- Identity

$$(\lambda x : A.\, x)(\lambda y : B.\, Q) P \rightarrow_\beta (\lambda y : B.\, Q) P$$

Weak Normalization proof (III)

THEOREM. Weak Normalization
If P is a typable term in $\lambda\!\rightarrow$, then there is a terminating reduction starting from P.

PROOF.
Pick R, an innermost redex of maximal height in P.
Contract redex R, to obtain P_1, so $P \rightarrow_\beta P_1$.
We have $m(P) >_\ell m(P_1)$ and we can continue this process, obtaining $P \rightarrow_\beta P_1 \rightarrow_\beta P_2 \rightarrow_\beta \ldots$ with

$$m(P) >_\ell m(P_1) >_\ell m(P_2) >_\ell \ldots .$$

As there are no infinitely decreasing $<_\ell$ sequences, this sequence is finite and then we have arrived at a normal form P_n. $\square$.

Questions?