Simply Typed λ-Calculus

Akim Demaille akim@lrde.epita.fr

EPITA — École Pour l'Informatique et les Techniques Avancées

March 22, 2009

About these lecture notes

Many of these slides are largely inspired from Andrew D. Ker's lecture notes [2, 3]. Some slides are even straightforward copies from them.

Simply Typed λ-Calculus

Types

Types

Types first appeared with

- Curry (1934) for Combinatory Logic
- Church (1940)

Types are syntactic objects assigned to terms:

$$M : A \qquad M \text{ has type } A$$

For instance:

$$\mathsf{I} : A \to A$$

Types

Types first appeared with

- Curry (1934) for Combinatory Logic
- Church (1940)

Types are syntactic objects assigned to terms:

$$M : A \qquad M \text{ has type } A$$

For instance:

$$\mathsf{I} : A \to A$$

Untyped λ-calculus

λ-terms

Λ, set of λ-terms

$$\frac{}{x \in \Lambda}\, x \in \mathcal{V} \qquad \frac{M \in \Lambda \quad N \in \Lambda}{(MN) \in \Lambda} \qquad \frac{M \in \Lambda}{(\lambda x \cdot M) \in \Lambda}\, x \in \mathcal{V}$$

$\lambda\beta$

The $\lambda\beta$ Formal System

$$\frac{}{M = M} \qquad \frac{M = N}{N = M} \qquad \frac{M = N \quad N = L}{M = L}$$

$$\frac{M = M' \quad N = N'}{MN = M'N'} \qquad \frac{M = N}{\lambda x \cdot M = \lambda x \cdot N}$$

$$\frac{}{(\lambda x \cdot M)N = [N/x]M}$$

Properties of $\lambda\beta$

β-reduction is Church-Rosser.

Any term has (at most) a unique NF.

β-reduction is not normalizing.

Some terms have no NF (Ω).

Properties of $\lambda\beta$

β-reduction is Church-Rosser.

Any term has (at most) a unique NF.

β-reduction is not normalizing.

Some terms have no NF (Ω).

Properties of $\lambda\beta$

β-reduction is Church-Rosser.

Any term has (at most) a unique NF.

β-reduction is not normalizing.

Some terms have no NF (Ω).

Properties of $\lambda\beta$

β-reduction is Church-Rosser.

Any term has (at most) a unique NF.

β-reduction is not normalizing.

Some terms have no NF (Ω).

Paradoxes

Self application

What is the computational meaning of $\lambda x \cdot xx$?

- Stop considering anything can be applied to anything
- A function and its argument have different behaviors

Types
$\lambda^{\rightarrow}$: Type Assignments
Untyped λ-calculus
Paradoxes
Church vs. Curry

Self application

What is the computational meaning of $\lambda x \cdot xx$?

- Stop considering anything can be applied to anything
- A function and its argument have different behaviors

A. Demaille Simply Typed λ-Calculus 11 / 45

Church vs. Curry

1. Types
   - Untyped λ-calculus
   - Paradoxes
   - Church vs. Curry
2. $\lambda^{\rightarrow}$: Type Assignments

Simple Types

- A set of type variables
  $\alpha, \beta, \ldots$
- A symbol $\rightarrow$ for functions
  $\alpha \rightarrow \alpha$, $\alpha \rightarrow (\beta \rightarrow \gamma)$, $(\alpha \rightarrow \beta) \rightarrow \gamma$, ...
- Possibly constants for "primitive" types
  ι for integers, etc.

Types
$\lambda^{\rightarrow}$: Type Assignments
Untyped λ-calculus
Paradoxes
Church vs. Curry

Simple Types

- A set of type variables
  $\alpha, \beta, \ldots$
- A symbol $\rightarrow$ for functions
  $\alpha \rightarrow \alpha$, $\alpha \rightarrow (\beta \rightarrow \gamma)$, $(\alpha \rightarrow \beta) \rightarrow \gamma$, ...
- Possibly constants for "primitive" types
  ι for integers, etc.

A. Demaille Simply Typed λ-Calculus 13 / 45

Types
$\lambda^{\rightarrow}$: Type Assignments
Untyped λ-calculus
Paradoxes
Church vs. Curry

Simple Types

- A set of type variables
  $\alpha, \beta, \ldots$
- A symbol $\rightarrow$ for functions
  $\alpha \rightarrow \alpha$, $\alpha \rightarrow (\beta \rightarrow \gamma)$, $(\alpha \rightarrow \beta) \rightarrow \gamma$, ...
- Possibly constants for "primitive" types
  ι for integers, etc.

A. Demaille Simply Typed λ-Calculus 13 / 45

Simple Types

By convention $\rightarrow$ is right-associative:

$$\alpha \rightarrow \beta \rightarrow \gamma = \alpha \rightarrow (\beta \rightarrow \gamma)$$

This matches the right-associativity of λ:

$$\lambda x \cdot \lambda y \cdot M = \lambda x \cdot (\lambda y \cdot M)$$

Types
$\lambda^{\rightarrow}$: Type Assignments
Untyped λ-calculus
Paradoxes
Church vs. Curry

Simple Types

By convention $\rightarrow$ is right-associative:

$$\alpha \rightarrow \beta \rightarrow \gamma = \alpha \rightarrow (\beta \rightarrow \gamma)$$

This matches the right-associativity of λ:

$$\lambda x \cdot \lambda y \cdot M = \lambda x \cdot (\lambda y \cdot M)$$

A. Demaille Simply Typed λ-Calculus 14 / 45

Alonzo Style, or Haskell Way?

Church: Typed λ-calculus

$$\frac{x : \alpha \quad x : \alpha}{\lambda x^{\alpha} \cdot x : \alpha \rightarrow \alpha}$$

Curry: λ-calculus with Types

$$\frac{x : \alpha \quad x : \alpha}{\lambda x \cdot x : \alpha \rightarrow \alpha}$$

$\lambda^{\rightarrow}$: Type Assignments

Types

1 Types

2 $\lambda^{\rightarrow}$: Type Assignments

- Types
- Type Deductions
- Subject Reduction Theorem
- Reducibility
- Typability

Simple Types

$\mathcal{TV}$ a set of type variables $\alpha, \beta, \ldots$

Simple Types

The set $\mathcal{T}$ of types $\sigma, \tau, \ldots$:

$$\frac{}{\alpha \in \mathcal{T}}\ \alpha \in \mathcal{TV} \qquad \frac{\sigma \in \mathcal{T} \quad \tau \in \mathcal{T}}{(\sigma \to \tau) \in \mathcal{T}}$$

Type Contexts

Statement

A statement $M : \sigma$ is a pair with $M \in \Lambda, \sigma \in \mathcal{T}$.
M is the subject, σ the predicate.

Type Context, Basis

A type context Γ is a finite set of statements over distinct variables $\{x_1 : \sigma_1, \ldots\}$.

Assignment

The variable x is assigned the type σ in Γ iff $x : \sigma \in \Gamma$.

Type Contexts

Statement

A statement $M : \sigma$ is a pair with $M \in \Lambda, \sigma \in \mathcal{T}$.
M is the subject, σ the predicate.

Type Context, Basis

A type context Γ is a finite set of statements over distinct variables $\{x_1 : \sigma_1, \ldots\}$.

Assignment

The variable x is assigned the type σ in Γ iff $x : \sigma \in \Gamma$.

Type Contexts

Statement

A statement $M : \sigma$ is a pair with $M \in \Lambda, \sigma \in \mathcal{T}$.
M is the subject, σ the predicate.

Type Context, Basis

A type context Γ is a finite set of statements over distinct variables $\{x_1 : \sigma_1, \ldots\}$.

Assignment

The variable x is assigned the type σ in Γ iff $x : \sigma \in \Gamma$.

Type Contexts

Type Context Restrictions

$\Gamma - x$ is the Γ with all assignment $x : \sigma$ removed.
$\Gamma \restriction M$ is $\Gamma - \mathrm{FV}(M)$.

Type Deductions

A Natural Presentation

Type derivations are trees built from the following nodes.

$$\frac{M : \sigma \rightarrow \tau \quad N : \sigma}{}$$

A Natural Presentation

Type derivations are trees built from the following nodes.

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau}$$

A Natural Presentation

Type derivations are trees built from the following nodes.

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau} \qquad \frac{\begin{array}{c} x : \sigma \\ \vdots \\ M : \tau \end{array}}{}$$

A Natural Presentation

Type derivations are trees built from the following nodes.

$$\frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau} \qquad \frac{\begin{array}{c} [x : \sigma] \\ \vdots \\ M : \tau \end{array}}{\lambda x \cdot M : \sigma \to \tau}$$

Type Statement

Type Statement

A statement $M : \sigma$ is derivable from the type context Γ,

$$\Gamma \vdash M : \sigma$$

if there is a derivation of $M : \sigma$ which all non-canceled assumptions are in Γ.

Type Statements

Prove

$$\vdash \lambda fx \cdot f(fx) : (\sigma \to \sigma) \to \sigma \to \sigma$$

Type Statements

Prove

$$\vdash \lambda fx \cdot f(fx) : (\sigma \to \sigma) \to \sigma \to \sigma$$

$$\dfrac{}{\lambda fx \cdot f(fx) : (\sigma \to \sigma) \to \sigma \to \sigma}$$

Type Statements

Prove

$$\vdash \lambda fx \cdot f(fx) : (\sigma \to \sigma) \to \sigma \to \sigma$$

$$\dfrac{\dfrac{\dfrac{[f : \sigma \to \sigma]^{(2)} \qquad \dfrac{[f : \sigma \to \sigma]^{(2)} \quad [x : \sigma]^{(1)}}{fx : \sigma}}{f(fx) : \sigma}}{\lambda x \cdot f(fx) : \sigma \to \sigma}\,(1)}{\lambda fx \cdot f(fx) : (\sigma \to \sigma) \to \sigma \to \sigma}\,(2)$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

$$\frac{}{\lambda xy \cdot x : \sigma \to \tau \to \sigma}$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

$$\frac{\dfrac{[x : \sigma]^{(1)}}{\lambda y \cdot x : \tau \to \sigma}}{\lambda xy \cdot x : \sigma \to \tau \to \sigma}\ (1)$$

Alternative Presentation of Type Derivations

Type Derivations

Type derivations are trees built from the following nodes.

$$\frac{}{\{x : \sigma\} \mapsto x : \sigma}$$

$$\frac{\Gamma \mapsto M : \sigma \to \tau \quad \Delta \mapsto N : \sigma}{\Gamma \cup \Delta \mapsto MN : \tau}\ \Gamma, \Delta \text{ consistent}$$

$$\frac{\Gamma \mapsto M : \tau}{\Gamma \setminus \{x : \sigma\} \mapsto \lambda x \cdot M : \sigma \to \tau}\ \Gamma, \{x : \sigma\} \text{ consistent}$$

$$\frac{\Gamma \mapsto M : \tau}{\Gamma, \Delta \vdash M : \tau}$$

Alternative Presentation of Type Derivations

Type Derivations

Type derivations are trees built from the following nodes.

$$\frac{}{\{x : \sigma\} \mapsto x : \sigma}$$

$$\frac{\Gamma \mapsto M : \sigma \to \tau \quad \Delta \mapsto N : \sigma}{\Gamma \cup \Delta \mapsto MN : \tau}\ \Gamma, \Delta \text{ consistent}$$

$$\frac{\Gamma \mapsto M : \tau}{\Gamma \setminus \{x : \sigma\} \mapsto \lambda x \cdot M : \sigma \to \tau}\ \Gamma, \{x : \sigma\} \text{ consistent}$$

$$\frac{\Gamma \mapsto M : \tau}{\Gamma, \Delta \vdash M : \tau}$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

$$\frac{}{\mapsto \lambda xy \cdot x : \sigma \to \tau \to \sigma}$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

$$\frac{\dfrac{\overline{\{x : \sigma\} \mapsto x : \sigma}}{\{x : \sigma\} \mapsto \lambda y \cdot x : \tau \to \sigma}}{\mapsto \lambda xy \cdot x : \sigma \to \tau \to \sigma}$$

Type Statements

Prove

$$\vdash \lambda xy \cdot x : \sigma \to \tau \to \sigma$$

$$\frac{\dfrac{\overline{\{x : \sigma\} \mapsto x : \sigma}}{\{x : \sigma\} \mapsto \lambda y \cdot x : \tau \to \sigma}}{\mapsto \lambda xy \cdot x : \sigma \to \tau \to \sigma} \qquad \frac{\dfrac{[x : \sigma]^{(1)}}{\lambda y \cdot x : \tau \to \sigma}}{\lambda xy \cdot x : \sigma \to \tau \to \sigma}\,(1)$$

Type Statements

Type $\omega = \lambda x \cdot xx$.

Type Statements

Type $\omega = \lambda x \cdot xx$.

$$\dfrac{\vdots}{\mapsto \lambda x \cdot xx : \sigma}$$

Type Statements

Type $\omega = \lambda x \cdot xx$.

$$\dfrac{\dfrac{\vdots}{\{x : \sigma_1\} \mapsto xx : \sigma_2}}{\mapsto \lambda x \cdot xx : \sigma}\ \sigma = \sigma_1 \to \sigma_2$$

Type Statements

Type $\omega = \lambda x \cdot xx$.

$$\dfrac{\dfrac{\dfrac{\vdots}{\{x : \sigma_1\} \mapsto x : \tau \to \sigma_2} \qquad \dfrac{\vdots}{\{x : \sigma_1\} \mapsto x : \tau}}{\{x : \sigma_1\} \mapsto xx : \sigma_2}}{\mapsto \lambda x \cdot xx : \sigma}\ \sigma = \sigma_1 \to \sigma_2$$

Typability

Typability

A term M is typable if there exists a type σ such that $\vdash M : \sigma$.

- ω, Ω are not typable.
- S, K, I are typable.
- Y is not typable!

Types
$\lambda^{\to}$: Type Assignments
Types
Type Deductions
Subject Reduction Theorem
Reducibility
Typability

Typability

Typability

A term M is typable if there exists a type σ such that $\vdash M : \sigma$.

- ω, Ω are not typable.
- S, K, I are typable.
- Y is not typable!

A. Demaille Simply Typed λ-Calculus 29 / 45

Types
$\lambda^{\to}$: Type Assignments
Types
Type Deductions
Subject Reduction Theorem
Reducibility
Typability

Typability

Typability

A term M is typable if there exists a type σ such that $\vdash M : \sigma$.

- ω, Ω are not typable.
- S, K, I are typable.
- Y is not typable!

A. Demaille Simply Typed λ-Calculus 29 / 45

Types
$\lambda^{\to}$: Type Assignments
Types
Type Deductions
Subject Reduction Theorem
Reducibility
Typability

Typability

Typability

A term M is typable if there exists a type σ such that $\vdash M : \sigma$.

- ω, Ω are not typable.
- S, K, I are typable.
- Y is not typable!

A. Demaille Simply Typed λ-Calculus 29 / 45

Subject Construction Lemma I

Consider the derivation π for $M : \sigma$.

- If $M = x$, then $\Gamma = \{x : \sigma\}$ and

$$\pi = \frac{}{\{x : \sigma\} \mapsto x : \sigma}$$

- If $M = NL$, then

$$\pi = \frac{\Gamma \upharpoonright N \mapsto N : \tau \rightarrow \sigma \quad \Gamma \upharpoonright L \mapsto L : \tau}{\Gamma \mapsto NL : \sigma}$$

Subject Construction Lemma I

Consider the derivation π for $M : \sigma$.

- If $M = x$, then $\Gamma = \{x : \sigma\}$ and

$$\pi = \frac{}{\{x : \sigma\} \mapsto x : \sigma}$$

- If $M = NL$, then

$$\pi = \frac{\Gamma \upharpoonright N \mapsto N : \tau \rightarrow \sigma \quad \Gamma \upharpoonright L \mapsto L : \tau}{\Gamma \mapsto NL : \sigma}$$

Subject Construction Lemma I

Consider the derivation π for $M : \sigma$.

- If $M = \lambda x \cdot N$, then $\sigma = \sigma_1 \rightarrow \sigma_2$ and
  - If $x \in \mathrm{FV}(N)$

$$\pi = \frac{\Gamma \cup \{x : \sigma_1\} \mapsto N : \sigma_2}{\Gamma \mapsto \lambda x \cdot N : \sigma_1 \rightarrow \sigma_2}$$

  - If $x \notin \mathrm{FV}(N)$

$$\pi = \frac{\Gamma \mapsto N : \sigma_2}{\Gamma \mapsto \lambda x \cdot N : \sigma_1 \rightarrow \sigma_2}$$

Subject Construction Lemma I

Consider the derivation π for $M : \sigma$.

- If $M = \lambda x \cdot N$, then $\sigma = \sigma_1 \rightarrow \sigma_2$ and
  - If $x \in \mathrm{FV}(N)$

$$\pi = \frac{\Gamma \cup \{x : \sigma_1\} \mapsto N : \sigma_2}{\Gamma \mapsto \lambda x \cdot N : \sigma_1 \rightarrow \sigma_2}$$

  - If $x \notin \mathrm{FV}(N)$

$$\pi = \frac{\Gamma \mapsto N : \sigma_2}{\Gamma \mapsto \lambda x \cdot N : \sigma_1 \rightarrow \sigma_2}$$

Subject Construction Lemma I

Consider the derivation π for $M : \sigma$.

- If $M = \lambda x \cdot N$, then $\sigma = \sigma_1 \rightarrow \sigma_2$ and
  - If $x \in \mathrm{FV}(N)$

$$\pi = \frac{\Gamma \cup \{x : \sigma_1\} \mapsto N : \sigma_2}{\Gamma \mapsto \lambda x \cdot N : \sigma_1 \rightarrow \sigma_2}$$

  - If $x \notin \mathrm{FV}(N)$

$$\pi = \frac{\Gamma \mapsto N : \sigma_2}{\Gamma \mapsto \lambda x \cdot N : \sigma_1 \rightarrow \sigma_2}$$

Derivations are not Unique

They are for β-normal forms.

Subject Reduction Theorem

1. Types
2. $\lambda^{\rightarrow}$: Type Assignments
   - Types
   - Type Deductions
   - Subject Reduction Theorem
   - Reducibility
   - Typability

Conversions and Types

α-Invariance

If $\Gamma \mapsto M : \sigma$ and $M \equiv_\alpha N$ then $\Gamma \mapsto N : \sigma$.

Substitution

If

- $\Gamma \cup \{x : \tau\} \vdash M : \sigma$,
- $\Delta \vdash N : \tau$,
- Γ, Δ consistent,
- $x \notin \mathrm{FV}(N)$

then

$$\Gamma \cup \Delta \vdash [N/x]M : \sigma$$

Conversions and Types

α-Invariance

If $\Gamma \mapsto M : \sigma$ and $M \equiv_\alpha N$ then $\Gamma \mapsto N : \sigma$.

Substitution

If

- $\Gamma \cup \{x : \tau\} \vdash M : \sigma$,
- $\Delta \vdash N : \tau$,
- Γ, Δ consistent,
- $x \notin \mathrm{FV}(N)$

then

$$\Gamma \cup \Delta \vdash [N/x]M : \sigma$$

Conversions and Types

α-Invariance

If $\Gamma \mapsto M : \sigma$ and $M \equiv_\alpha N$ then $\Gamma \mapsto N : \sigma$.

Substitution

If

- $\Gamma \cup \{x : \tau\} \vdash M : \sigma$,
- $\Delta \vdash N : \tau$,
- Γ, Δ consistent,
- $x \notin \mathrm{FV}(N)$

then

$$\Gamma \cup \Delta \vdash [N/x]M : \sigma$$

Subject Reduction Theorem

Subject Reduction Theorem

If $\Gamma \vdash M : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash N : \sigma$.

What about the converse?

Subject Expansion

If $\Gamma \vdash N : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash M : \sigma$.

Subject Reduction Theorem

Subject Reduction Theorem

If $\Gamma \vdash M : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash N : \sigma$.

What about the converse?

Subject Expansion

If $\Gamma \vdash N : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash M : \sigma$.

Subject Reduction Theorem

Subject Reduction Theorem

If $\Gamma \vdash M : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash N : \sigma$.

What about the converse?

Subject Expansion

If $\Gamma \vdash N : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash M : \sigma$.

Subject Reduction Theorem

Subject Reduction Theorem

If $\Gamma \vdash M : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash N : \sigma$.

What about the converse?

Subject Expansion

If $\Gamma \vdash N : \sigma$ and $M \rightarrow_\beta N$ then $\Gamma \vdash M : \sigma$.

Ω is not typable, but $\mathsf{K I} \Omega \rightarrow \mathsf{I}$.

Reducibility

Types and Normalization

$\lambda^{\rightarrow}$ is Strongly Normalizing

All typable terms are β-strongly normalizing.

Typability

1. Types
2. $\lambda^{\rightarrow}$: Type Assignments
   - Types
   - Type Deductions
   - Subject Reduction Theorem
   - Reducibility
   - Typability

$\vdash M : \sigma$ suffices

Note that with $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$

$$\Gamma \vdash M : \tau$$

is equivalent to

$$\vdash \lambda x_1 \ldots x_n \cdot M : \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \tau$$

Decidability of Type Assignment

Type checking Given M, σ, does $\vdash M : \sigma$?

$$\vdash M : \sigma?$$

Typability Given M, does there exist σ such that $\vdash M : \sigma$?

$$\vdash M :?$$

Inhabitation Given σ, does there exist M such that $\vdash M : \sigma$?

$$\vdash ? : \sigma$$

Decidability of Type Assignment

Type Checking is Decidable

It is decidable whether a statement of $\lambda^{\rightarrow}$is provable.

Typability is Decidable

It is decidable whether a term of $\lambda^{\rightarrow}$has a type.

Decidability of Type Assignment

Type Checking is Decidable

It is decidable whether a statement of $\lambda^{\rightarrow}$is provable.

Typability is Decidable

It is decidable whether a term of $\lambda^{\rightarrow}$has a type.

Types are not Unique...

Typable terms do not have unique types.

$$\dfrac{\dfrac{\dfrac{}{\{x:\sigma\} \mapsto x:\sigma}}{\{x:\sigma\} \mapsto \lambda y \cdot x : \tau \rightarrow \sigma}}{\mapsto \mathsf{K} : \sigma \rightarrow \tau \rightarrow \sigma}$$

is valid for *any* σ, τ, including $\sigma = \sigma' \rightarrow \sigma'', \tau = (\tau' \rightarrow \tau'') \rightarrow \tau'$ etc.

Types are not Unique...

but some are more Unique than others ...

All the types of K are instances of $\sigma \rightarrow \tau \rightarrow \sigma$.

Bibliography Notes

[2] Complete and readable lecture notes on λ-calculus. Uses conventions different from ours.

[3] Additional information, including slides.

[1] A classical introduction to λ-calculus.

Bibliography I

Henk Barendregt and Erik Barendsen.
Introduction to lambda calculus, March 2000.
http://www.cs.ru.nl/~erikb/onderwijs/T3/materiaal/lambda.pdf.

Andrew D. Ker.
Lambda calculus and types, May 2005.
http://web.comlab.ox.ac.uk/oucl/work/andrew.ker/lambda-calculus-notes-full-v3.pdf.

Andrew D. Ker.
Lambda calculus notes, May 2005.
http://web.comlab.ox.ac.uk/oucl/work/andrew.ker/.