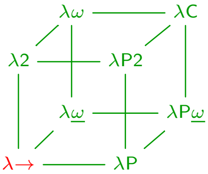
propositional logic & simple types

Freek Wiedijk

Type Theory & Coq
2024–2025
Radboud University Nijmegen

September 13, 2024



teaser: a proof term for a proof

$$\lambda x : a \to b \to c.\, \lambda y : a \to b.\, \lambda z : a.\, xz(yz)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a \to b \to c^x] \qquad [a^z]}{b \to c}\,E{\to} \qquad \dfrac{[a \to b^y] \qquad [a^z]}{b}\,E{\to}}{c}\,E{\to}}{a \to c}\,I[z]{\to}}{(a \to b) \to a \to c}\,I[y]{\to}}{(a \to b \to c) \to (a \to b) \to a \to c}\,I[x]{\to}$$

many systems

- one untyped lambda calculus

  (last week: recap)

- many typed lambda calculi = type theories

  - **Church-style**
    variables explicitly typed

    the systems in this course
    Coq!

  - **Curry-style**
    assigning types to untyped terms

    (in one of Herman's lectures)

- many logics

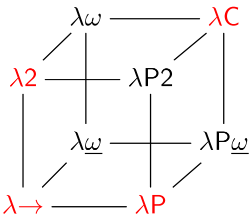
Curry-Howard correspondence

'propositions-as-types'

type systems	$\sim$	logics
datatypes	$\sim$	propositions
$A \times B$	$\sim$	$A \wedge B$
$A \to B$	$\sim$	$A \to B$
objects of a datatype	$\sim$	proofs of a proposition
'inhabitants'		'proof objects'
non-empty type	$\sim$	true proposition
empty type	$\sim$	false proposition
derivation in a type theory	$\sim$	derivation in a logic

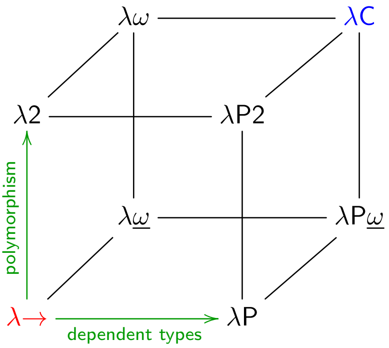
type systems in this course

logics	$\sim$	type systems		
propositional logic	$\sim$	$\lambda\!\to$	$=$	simply typed lambda calculus
predicate logic	$\sim$	λP	$=$	dependently typed lambda calculus
second order logic	$\sim$	$\lambda 2$	$=$	polymorphic lambda calculus
higher order logic	$\sim$	λC	$=$	CC $=$ Calculus of Constructions
'the logic of Coq'	$\sim$	CIC	$=$	Calculus of Inductive Constructions



the Barendregt cube
eight PTSs = pure type systems

(not explicitly in Femke's course notes)



- **natural deduction** $B_1, \ldots, B_m \vdash A$

  introduction and elimination rules for each connective

$$\frac{\ldots \vdash \ldots}{\ldots \vdash \cdots \otimes \cdots}\, I \qquad\qquad \frac{\ldots \vdash \cdots \otimes \cdots}{\ldots \vdash \ldots}\, E$$

  - Gentzen–Prawitz style: proof is a derivation tree
  - Jaśkowski–Fitch style: proof consists of nested boxes / flags

- **sequent calculus**: LK & LJ $B_1, \ldots, B_m \vdash A_1, \ldots, A_n$

  left and right rules for each connective

$$\frac{\ldots \vdash \ldots}{\cdots \otimes \cdots \vdash \ldots}\, L \qquad\qquad \frac{\ldots \vdash \ldots}{\ldots \vdash \cdots \otimes \cdots}\, R$$

- **Hilbert-style** $\vdash A$

  axioms for each connective (and at most two rules)

defining a logic or type system

- **syntax**
  - terms
  - types
  - formulas
  - contexts
  - judgments

$$M, N ::= x \mid M\,N \mid \lambda x.\,M$$

- **rules**
  - logics: proof rules
  - type systems: typing rules

no rules for untyped lambda calculus

- **reduction**

$$(\lambda x.\,M)\,N \to_\beta M[x := N]$$

three propositional logics

- **minimal propositional logic**
  the logic that corresponds to simply typed lambda calculus
  only implication: $\to$
- **constructive propostional logic**
  all connectives: $\to$, $\wedge$, $\vee$, $\neg$, $\bot$, $\top$
- **classical propositional logic**

$$A \vee \neg A$$
$$\neg\neg A \to A$$

propositional logic: syntax

formulas

$$A, B ::= a \mid A \to B \mid A \wedge B \mid A \vee B \mid \neg A \mid \top \mid \bot$$

a, b, c, ...	atomic propositions later: propositional variables
A, B, C, ...	meta-variables for arbitrary formulas

implication associates to the right:

$$a \to b \to c \quad \text{means} \quad a \to (b \to c)$$

order of binding strength: $\neg > \wedge > \vee > \to$

$$a \vee \neg b \wedge c \to d \quad \text{means} \quad (a \vee ((\neg b) \wedge c)) \to d$$

the two rules of minimal propositional logic

the introduction and elimination rules of implication:

$$
\dfrac{\begin{array}{c}[A^x] \\ \vdots \\ B\end{array}}{A \to B}\, I[x]{\to}
\qquad\qquad
\dfrac{\begin{array}{c}\vdots \\ A \to B\end{array} \qquad \begin{array}{c}\vdots \\ A\end{array}}{B}\, E{\to}
$$

in the introduction rule the assumption $[A^x]$ may be used an arbitrary number of times: zero, one or more

in the elimination rule the proof of $A \to B$ is on the *left* of the proof of A

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a \to b \to c^{x}] \qquad [a^{z}]}{b \to c}\,E\!\to \qquad \dfrac{[a \to b^{y}] \qquad [a^{z}]}{b}\,E\!\to}{c}\,E\!\to}{a \to c}\,I[z]\!\to}{(a \to b) \to a \to c}\,I[y]\!\to}{\color{red}{(a \to b \to c) \to (a \to b) \to a \to c}}\,I[x]\!\to
$$

constructive propositional logic: the other rules

$$\dfrac{\begin{matrix}\vdots & & \vdots \\ A & & B\end{matrix}}{A \wedge B}\,{\color{red}I\wedge} \qquad \dfrac{\begin{matrix}\vdots \\ A \wedge B\end{matrix}}{A}\,{\color{red}El\wedge} \qquad \dfrac{\begin{matrix}\vdots \\ A \wedge B\end{matrix}}{B}\,{\color{red}Er\wedge}$$

$$\dfrac{\begin{matrix}\vdots \\ A\end{matrix}}{A \vee B}\,{\color{red}Il\vee} \qquad \dfrac{\begin{matrix}\vdots \\ B\end{matrix}}{A \vee B}\,{\color{red}Ir\vee} \qquad \dfrac{\begin{matrix}\vdots & & \vdots & & \vdots \\ A \vee B & & A \to C & & B \to C\end{matrix}}{C}\,{\color{red}E\vee}$$

$$\dfrac{\begin{matrix}[A^x] \\ \vdots \\ \bot\end{matrix}}{\neg A}\,{\color{red}I\neg} \qquad \dfrac{\begin{matrix}\vdots & & \vdots \\ \neg A & & A\end{matrix}}{\bot}\,{\color{red}E\neg} \qquad \dfrac{}{\top}\,{\color{red}I\top} \qquad \dfrac{\begin{matrix}\vdots \\ \bot\end{matrix}}{A}\,{\color{red}E\bot}$$

$${\color{green}\neg A := A \to \bot}$$

variants of elimination rules

- what Coq's `elim` tactic implements for conjunction:

$$\dfrac{\begin{matrix}\vdots \\ A \wedge B\end{matrix} \qquad \begin{matrix}\vdots \\ A \to B \to C\end{matrix}}{C}\; E\wedge$$

- often the disjunction elimination rule is:

$$\dfrac{\begin{matrix} \\ \vdots \\ A \vee B\end{matrix} \qquad \begin{matrix}[A^x] \\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B^y] \\ \vdots \\ C\end{matrix}}{C}\; E[x,y]\vee$$

not what Coq's `elim` tactic implements for disjunction

example proof beyond minimal propositional logic

$$
\dfrac{[a \vee b^x] \qquad \dfrac{\dfrac{[a^y]}{b \vee a}\, Ir\vee}{a \to b \vee a}\, I[y]{\to} \qquad \dfrac{\dfrac{[b^z]}{b \vee a}\, Il\vee}{b \to b \vee a}\, I[z]{\to}}{\dfrac{b \vee a}{a \vee b \to b \vee a}\, I[x]{\to}}\, E\vee
$$

alternative style: explicit assumption lists

syntax

$$A, B ::= a \mid A \to B \mid \ldots \qquad \text{formulas}$$

$$\Gamma ::= \cdot \mid \Gamma, A \qquad \text{assumption lists}$$

$$\mathcal{J} ::= \Gamma \vdash A \qquad \text{sequents}$$

we do not write the dot and the comma after the dot
still natural deduction, *not* sequent calculus

rules

$$\frac{}{\Gamma \vdash A}\,\text{ass} \qquad \text{for } A \in \Gamma$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\,I{\to} \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\,E{\to} \qquad \ldots$$

the example proofs with explicit assumption lists

$\Gamma_1 := a \to b \to c,\ a \to b,\ a$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash a \to b \to c}\,\text{ass} \quad \dfrac{}{\Gamma_1 \vdash a}\,\text{ass}}{\Gamma_1 \vdash b \to c}\,E{\to} \qquad \dfrac{\dfrac{}{\Gamma_1 \vdash a \to b}\,\text{ass} \quad \dfrac{}{\Gamma_1 \vdash a}\,\text{ass}}{\Gamma_1 \vdash b}\,E{\to}}{\Gamma_1 \vdash c}\,E{\to}}{a \to b \to c,\ a \to b \vdash a \to c}\,I{\to}}{a \to b \to c \vdash (a \to b) \to a \to c}\,I{\to}}{\vdash (a \to b \to c) \to (a \to b) \to a \to c}\,I{\to}
$$

$$
\dfrac{\dfrac{\dfrac{}{a \vee b \vdash a \vee b}\,\text{ass} \quad \dfrac{\dfrac{\dfrac{}{a \vee b,\ a \vdash a}\,\text{ass}}{a \vee b,\ a \vdash b \vee a}\,Ir{\vee}}{a \vee b \vdash a \to b \vee a}\,I{\to} \quad \dfrac{\dfrac{\dfrac{}{a \vee b,\ b \vdash b}\,\text{ass}}{a \vee b,\ b \vdash b \vee a}\,Il{\vee}}{a \vee b \vdash b \to b \vee a}\,I{\to}}{a \vee b \vdash b \vee a}\,E{\vee}}{\vdash a \vee b \to b \vee a}\,I{\to}
$$

the S combinator

untyped
typed: Curry-style

$$\lambda xyz.\, xz(yz)$$

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

typed: Church-style

$$\lambda x : a \to b \to c.\, \lambda y : a \to b.\, \lambda z : a.\, xz(yz)$$

Coq syntax

```
fun x : a -> b -> c => fun y : a -> b => fun z : a =>
  x z (y z)
```

```
fun (x : a -> b -> c) (y : a -> b) (z : a) => x z (y z)
```

BHK interpretation

Brouwer–Heyting–Kolmogorov
$\sim$ Curry-Howard correspondence
constructive 'meaning' of the logical connectives

connection between proofs and lambda calculus

proof of $A \rightarrow B$	$=$	function from proofs of A to proofs of B
proof of $A \wedge B$	$=$	pair of a proof of A and a proof of B
proof of $A \vee B$	$=$	either a proof of A, or a proof of B
proof of $\neg A$	$=$	proof of $A \rightarrow \bot$
proof of $\top$		the object I
proof of $\bot$		*does not exist*

proof of $a \wedge b \rightarrow b \wedge a$ is

a function that inputs a pair $\langle x, y \rangle$, and returns $\langle y, x \rangle$
with x a proof of a and y a proof of b

$$\lambda p : a \wedge b.\, \langle \pi_2 p, \pi_1 p \rangle$$

different names for the same type theory

simply typed lambda calculus

$\parallel$

simple type theory = STT

$\parallel$

$\lambda{\rightarrow}$

3 typing rules

$\nparallel$

7 typing rules

same set of well-typed terms

simply typed lambda calculus: syntax and rules

syntax

$$A, B ::= a \mid A \to B \qquad \text{types}$$

$$M, N ::= x \mid M\,N \mid \lambda x : A.\, M \qquad \text{preterms}$$

$$\Gamma ::= \cdot \mid \Gamma, x : A \qquad \text{contexts}$$

$$\mathcal{J} ::= \Gamma \vdash M : A \qquad \text{judgments}$$

rules

$$\frac{}{\Gamma \vdash x : A} \quad \text{for } (x : A) \in \Gamma$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x : A.\, M) : A \to B} \qquad \frac{\Gamma \vdash F : A \to B \qquad \Gamma \vdash M : A}{\Gamma \vdash F\,M : B}$$

$\Gamma \vdash M : A$ is derivable $\implies$ M is well-typed

well-typed preterms are called terms

example type derivation

$$\lambda f : a \to b.\, \lambda x : a.\, f x$$

$$\dfrac{\dfrac{\overline{f : a \to b,\, x : a \vdash f : a \to b} \qquad \overline{f : a \to b,\, x : a \vdash x : a}}{\dfrac{f : a \to b,\, x : a \vdash f x : b}{f : a \to b \vdash (\lambda x : a.\, f x) : a \to b}}}{\vdash (\lambda f : a \to b.\, \lambda x : a.\, f x) : (a \to b) \to a \to b}$$

$$\overline{\Gamma \vdash x : A} \qquad \dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x : A.\, M) : A \to B} \qquad \dfrac{\Gamma \vdash F : A \to B \qquad \Gamma \vdash M : A}{\Gamma \vdash F M : B}$$

proofs versus type derivations: type derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f : a \to b,\ x : a \vdash f : a \to b} \quad \dfrac{}{f : a \to b,\ x : a \vdash x : a}}{f : a \to b,\ x : a \vdash f x : b}}{f : a \to b \vdash (\lambda x : a.\ f x) : a \to b}}{\vdash (\lambda f : a \to b.\ \lambda x : a.\ f x) : (a \to b) \to a \to b}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a \to b,\ a \vdash a \to b}\,\mathsf{ass} \quad \dfrac{}{a \to b,\ a \vdash a}\,\mathsf{ass}}{a \to b,\ a \vdash b}\,E{\to}}{a \to b \vdash a \to b}\,I{\to}}{\vdash (a \to b) \to a \to b}\,I{\to}$$

$$\dfrac{\dfrac{\dfrac{[a \to b^{f}] \qquad [a^{x}]}{b}\; E{\to}}{a \to b}\; I[x]{\to}}{(a \to b) \to a \to b}\; I[f]{\to}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a \to b,\, a \vdash a \to b}\;\textsf{ass} \qquad \dfrac{}{a \to b,\, a \vdash a}\;\textsf{ass}}{a \to b,\, a \vdash b}\; E{\to}}{a \to b \vdash a \to b}\; I{\to}}{\vdash (a \to b) \to a \to b}\; I{\to}$$

proofs versus type derivations: proof term

$$\dfrac{\dfrac{\dfrac{[a \to b^{f}] \quad [a^{x}]}{b}\,E{\to}}{a \to b}\,I[x]{\to}}{(a \to b) \to a \to b}\,I[f]{\to}$$

$$\lambda f : a \to b.\, \lambda x : a.\, f x$$

Curry-Howard correspondence

propositions		types
implications	$\longleftrightarrow$	function types
proof rules		proof terms
introduction rules	$\longleftrightarrow$	lambda abstraction
elimination rules	$\longleftrightarrow$	function application
assumption rule	$\longleftrightarrow$	variables

how to read proof terms

$$M := \ (\lambda f : a \to b.\, \lambda x : a.\, f x) \ : \ (a \to b) \to a \to b$$

this proof term M is a function
that takes as its first argument a proof of $a \to b$ called f
and as its second argument a proof of a called x

and then maps those to a proof of b

the argument f is itself a function that
maps proofs of a to proofs of b
(conform the BHK-interpretation)

x is an inhabitant of the type a
which corresponds to the proposition a

to get a proof of b, the function f is applied to x
and the result of that application then is the result of M

Coq

the example

```
Parameter a b : Prop.

Lemma one :
  (a -> b) -> a -> b.
intros f x.
apply f.
apply x.
Qed.

Check one.
Print one.
```

```
one = fun (f : a -> b) (x : a) => f x
     : (a -> b) -> a -> b
```

$$\dfrac{\dfrac{\dfrac{[a \to b^f] \qquad [a^x]}{b}\,E{\to}}{a \to b}\,I[x]{\to}}{(a \to b) \to a \to b}\,I[f]{\to}$$

$$(\lambda f : a \to b.\, \lambda x : a.\, f x) : (a \to b) \to a \to b$$

example with disjunction

```
Parameter a b : Prop.

Lemma two :
  a \/ b -> b \/ a.
intros [y|z].
- right.
  apply y.
- left.
  apply z.
Qed.
```

$$\dfrac{[a \vee b^x] \qquad \dfrac{\dfrac{[a^y]}{b \vee a}\,Ir\vee}{a \to b \vee a}\,I[y]{\to} \qquad \dfrac{\dfrac{[b^z]}{b \vee a}\,Il\vee}{b \to b \vee a}\,I[z]{\to}}{\dfrac{b \vee a}{a \vee b \to b \vee a}\,I[x]{\to}}\,E\vee$$

example with conjunction

```
Parameter a b : Prop.

Lemma three :
  a /\ b -> b /\ a.
intros [y z].
split.
- apply z.
- apply y.
Qed.
```

$$
\dfrac{[a \wedge b^x] \qquad \dfrac{\dfrac{\dfrac{[b^z] \qquad [a^y]}{b \wedge a}\, I\wedge}{b \to b \wedge a}\, I[z]{\to}}{a \to b \to b \wedge a}\, I[y]{\to}}{\dfrac{b \wedge a}{a \wedge b \to b \wedge a}\, I[x]{\to}}\, E\wedge
$$

tactics for proof rules

$I{\to}$ $I\neg$	`intro` `intros`
$E{\to}$ $E\neg$	`apply`
ass	`exact` `apply`
$E\wedge$ $E\vee$ $E\bot$	`elim` `destruct` `intros` with pattern
$I\wedge$	`split`
$Il\vee$ $Ir\vee$	`left` `right`
$I\top$	`apply I` `I : True`

bullets

structure tactic scripts according to subgoals

related bullets need to match:

-	--	---	----	etc.
+	++	+++	++++	etc.
*	**	***	****	etc.

intro patterns

only works with `intros`, not with `intro`

the pattern needs to match the shape of assumptions

goal	tactic
$A \to G$	`intros HA.`
$A \to B \to G$	`intros HA HB.`
$A \land B \to G$	`intros [HA HB].`
$A \lor B \to G$	`intros [HA|HB].`
$A \lor (B \land C) \to D \to G$	`intros [HA|[HB HC]] HD.`

apply

the number n of antecedents of H may be zero
H not only a variable, may be an arbitrary term

$$H : A_1 \to \cdots \to A_n \to B$$

$$\text{goal } B \xrightarrow{\texttt{apply H}} \text{new goals } A_1, \ldots, A_n$$

$$\cfrac{\cfrac{\cfrac{\cfrac{[A_1 \to \cdots \to A_n \to B^H] \qquad A_1}{A_2 \to \cdots \to A_n \to B}\,E{\to} \qquad A_2}{\ddots}\,E{\to}}{A_{n-1} \to A_n \to B \qquad A_{n-1}}}{\cfrac{A_n \to B \qquad A_n}{B}\,E{\to}}\,E{\to}$$

elim

$$H : A_1 \to \cdots \to A_n \to B \otimes C$$

`elim` and `destruct` eliminate the connective $\otimes$
which leaves the goal unchanged

but also generate extra subgoals $A_1, \ldots, A_n$

summary

logic	type theory	Coq
introduction rules	lambda abstraction	`intro`
elimination rules	function application	`apply`
introduction rules	(in three weeks)	(various)
elimination rules	(in three weeks)	`elim`

for more about the tactics
read the Coq manual!

table of contents

contents