overview

typing system for lambda P

decidability

Curry-Howard-De Bruijn isomorphism

typing system for lambdaP

$$\frac{}{\vdash * : \Box}\ \textit{start}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ \textit{var}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\ \textit{weak}$$

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : s}{\Gamma \vdash \Pi x{:}A \,.\, B : s}\ \textit{prod}$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x{:}A \,.\, B : s}{\Gamma \vdash \lambda x : A.\, M : \Pi x{:}A \,.\, B}\ \textit{abs}$$

$$\frac{\Gamma \vdash F : \Pi x{:}A \,.\, B \qquad \Gamma \vdash M : A}{\Gamma \vdash (F\ M) : B[x := M]}\ \textit{app}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}\ \textit{conv} \qquad \text{with } B =_\beta B'$$

examples

$\mathsf{nat} : *, n : \mathsf{nat} \vdash n : \mathsf{nat}$

$\mathsf{nat} : * \vdash \Pi n : \mathsf{nat}.\, \mathsf{nat} : *$

$\mathsf{nat} : * \vdash \lambda n : \mathsf{nat}.\, n : \Pi n : \mathsf{nat}.\, \mathsf{nat}$

decidability issues

- type inhabitation problem (TIP)
  $\Gamma \vdash ? : A$
  undecidable in lambda P

- type checking problem (TCP)
  $\Gamma \vdash P : A?$
  decidable in lambda P

- type synthesis problem (TSP) or typability problem
  $\Gamma \vdash P :?$
  decidable in lambda P

overview

typing system for lambda P

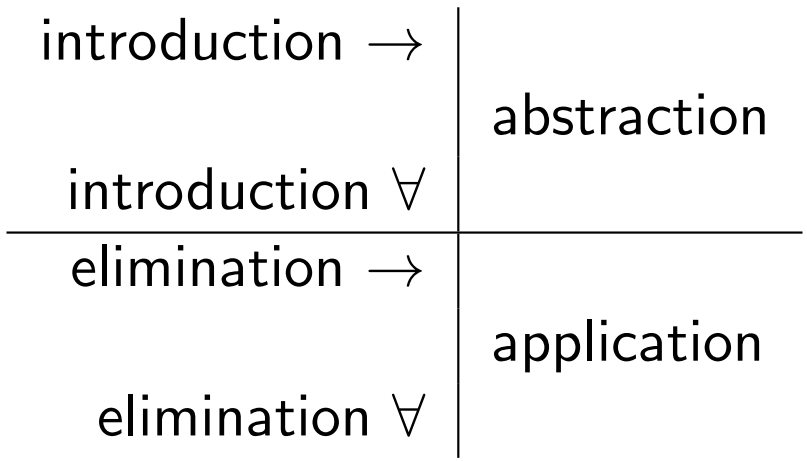
decidability

Curry-Howard-De Bruijn isomorphism

Curry-Howard-De Bruijn isomorphism

(a fragment of) λP corresponds to minimal pred1

Curry-Howard-De Bruijn isomorphism



minimal pred1: only implication and for all

formulas of minimal pred1 compared with prop1:

	prop1	pred1
a	$+$	$+$
$A \to B$	$+$	$+$
$\forall x.\, B$	$-$	$+$

minimal pred1: introduction rules

$$\frac{B}{A \to B}\ I\to \qquad \frac{B}{\forall x.\, B}\ I\forall$$

minimal pred1: elimination rules

$$\frac{A \to B \qquad A}{B}\ E\to \qquad \frac{\forall x.\, B}{B[x := M]}\ E\forall$$

minimal pred1 in λP: formulas

logic	λP	Coq
$P(M_1, \ldots, M_n)$	$PM_1 \ldots M_n : \star$	`P M1 ... Mn : Prop`
$A \to B$	$A \to B : \star$	`A -> B : Prop`
$\forall x.\, B$	$\Pi x : \mathsf{Terms}.\, B : \star$	`forall x: Terms, B : Prop`

with $P : \Pi x_1 : \mathsf{Terms}. \ldots \Pi x_n : \mathsf{Terms}. \star$
or $P : \mathsf{Terms} \to \ldots \to \mathsf{Terms} \to \star$

formulas use (algebraic) terms

minimal pred1 in λP: algebraic terms

logic	λP	Coq
	$\mathsf{Terms} : \star$	`Terms : Set`
x	$x : \mathsf{Terms}$	`x : Terms`
$f(M_1, \ldots, M_n)$	$f\ M_1\ \ldots\ M_n : \mathsf{Terms}$	`f M1 ... Mn : Terms`

with $f : \Pi x_1 : \mathsf{Terms}. \ldots \Pi x_n : \mathsf{Terms}. \mathsf{Terms}$
or $f : \mathsf{Terms} \to \ldots \to \mathsf{Terms} \to \mathsf{Terms}$

introduction rules correspond to abstraction

$$\frac{B}{A \to B}\ I{\to} \qquad \frac{B}{\forall x.\, B}\ I{\forall}$$

correspond to

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

$$\frac{\Gamma, x : A \vdash M : A \qquad \Gamma \vdash \Pi : A.\, B : s}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B}$$

in Coq: intro

elimination rules correspond to application

$$\frac{A \to B \qquad A}{B}\ E\to \qquad \frac{\forall x.\, B}{B[x := M]}\ E\forall$$

correspond to

$$\frac{F : A \to B \qquad M : A}{F\ M : B}$$

$$\frac{F : \Pi x{:}A\,.\,B \qquad M : A}{F\ M : B[x := M]}$$

in Coq: apply

logics and type theory

1st-order minimal propositional logic $\leftrightarrow$
simple type theory

1st-order minimal predicate logic $\leftrightarrow$
dependent type theory

2nd-order minimal propositional logic $\leftrightarrow$
polymorphic type theory

overview

formulas of prop2

terminology

proofs of prop2

classical prop2

impredicative definitions of logical connectives

overview

- formulas of prop2
- terminology
- proofs of prop2
- classical prop2
- impredicative definitions of logical connectives

formulas of prop1 (already seen)

$a\ b\ c\ p\ q$

$A \to B$

$\bot$

$\top$

$A \wedge B$

$A \vee B$

formulas of pred1 (already seen)

(using terms)

$a(\ldots)$ $b(\ldots)$ $c(\ldots)$ $p(\ldots)$ $q(\ldots)$

$A \to B$

$\forall x.\, A$

$\bot$

$\top$

$A \wedge B$

$A \vee B$

$\exists x.\, A$

formulas of prop2 (new)

$a\ b\ c\ p\ q$

$A \to B$

$\forall a.\, A$

$\bot$

$\top$

$A \wedge B$

$A \vee B$

$\exists a.\, A$

examples

in prop1:
$a \to a$

in pred1:
$\forall x.\, a(x) \to a(x)$

in prop2:
$\forall a.\, a \to a$

for every proposition, that proposition implies itself

overview

- formulas of prop2
- terminology
- proofs of prop2
- classical prop2
- impredicative definitions of logical connectives

higher-order

first order:
object

second order:
set of first-order objects
predicate on objects
function from objects to objects

third order:
set of second-order objects
predicate on predicates on objects
functions from second order objects

higher-order logic

first-order:
quantification over variables of order 1
$a \to a$
$\forall x.\, a(x) \to a(x)$

second-order:
quantification over variables of order 2
$\forall a.\, a \to a$
$\forall a.\, \forall x.\, a(x) \to a(x)$
$\forall f.\, \forall x.\, a(f(x)) \to a(f(x))$

third-order:
quantification over variables of order 3
$\forall b.\, \forall f.\, b(f) \to \forall x.\, a(f(x))$

quantify over predicates gives pred2
same without terms gives prop2

second-order predicate logic: example

induction principle for natural numbers

$$\forall a(a(0) \to (\forall m.\, a(m) \to a(S(m))) \to \forall n.\, a(n))$$

m	1st order variable
n	1st order variable
0	1st order constant
a	2nd order variable
S	2nd order constant

second-order predicate logic: example

there is a sorting function

$\exists f : \text{natlist} \to \text{natlist}.\, \forall l : \text{natlist}.\, \text{sorted}(f(l)) \wedge \text{permutation}(l, f(l)$

f	2nd order variable
l	1st order variable
sorted	2nd order constant
permutation	2nd order constant

examples

prop2	pred2
$\forall a.\, a \to a$	$\forall p.\, \forall x.\, p(x) \to p(x)$
prop1	**pred1**
$a \to a$	$\forall x.\, p(x) \to p(x)$

overview

- formulas of prop2
- terminology
- proofs of prop2
- classical prop2
- impredicative definitions of logical connectives

proof rules for prop2

introduction rules	elimination rules
$I\top$	
	$E\bot$
$I[x] \to$	$E \to$
$I\wedge$	$El\wedge$, $Er\wedge$
$Il\vee$, $Ir\vee$	$E\vee$
$I\forall$	$E\forall$
$I\exists$	$E\exists$

universal quantification for prop2

$\forall$ introduction:

$$\frac{A}{\forall a.\, A}\ I\forall$$

variable condition: a not free in any open assumption
check: variable does not occur in any of the available assumptions

$\forall$ elimination:

$$\frac{\forall a.\, A}{A[a := B]}\ E\forall$$

existential quantification for prop2

$\exists$ introduction:

$$\frac{A[a := B]}{\exists a.\, A}\ I\exists$$

$\exists$ elimination:

$$\frac{\exists a.\, A \qquad \forall a.\, A \to B}{B}\ E\exists$$

variable condition: a not free in B

check: variable does not occur in the conclusion

examples of tautologies

- $(\forall b.\, b) \to a$
- $a \to \forall b.\, (b \to a)$
- $a \to \forall b.\, ((a \to b) \to b)$
- $(\exists b.\, a) \to a$
- $\exists b\, ((a \to b) \vee (b \to a))$

examples of non-tautological formulas

- $a \to (\forall a.\, a)$
- $p(x) \to (\forall x.\, p(x))$
- $(\exists a.\, a) \to a$
- $\forall a., \forall b.\, (a \to b) \vee (b \to a)$
  (classical logic needed)

minimal prop2: detour

introduction rule for a connective

immediately followed by an

elimination rule for the same connective

elimination of an implication detour (as in prop1)

$$\dfrac{\dfrac{\vdots}{B}}{\dfrac{A \to B}{B}}\, I[x] \to \qquad \dfrac{\vdots}{A}\, E \to$$

is replaced by

$$\dfrac{\vdots}{B}$$

where every occurrence of the assumption A^x is replaced by the proof

$$\dfrac{\vdots}{A}$$

elimination of an universal quantification detour

(similar to pred1)

$$\dfrac{\dfrac{B}{\forall a.\, B}\, I\forall}{B[a := A]}\, E\forall \quad \to \quad B[a := A]$$

everywhere a is replaced by A

overview

- formulas of prop2
- terminology
- proofs of prop2
- classical prop2
- impredicative definitions of logical connectives

classical prop2

all formulas and proof rules of intuitionistic prop2
plus a classical axiom from

- $\forall a.\, a \vee \neg a$
- $\forall a.\, \neg\neg a \to a$
- $\forall a.\, \forall b.\, ((a \to b) \to a) \to a$

example of a tautology:
$\forall a.\, \forall b.\, ((a \to b) \vee (b \to a))$

expressivity of classical prop2

- $\forall a.\, B$ is equivalent to $B[a := \bot] \wedge B[a := \top]$
- $\exists a.\, B$ is equivalent to $B[a := \bot] \vee B[a := \top]$

so classical prop1 is as expressive as classical prop2

in particular classical prop2 is decidable

however intuitionistic prop2 is not decidable

overview

formulas of prop2

terminology

proofs of prop2

classical prop2

impredicative definitions of logical connectives

schema

polymorphic lambda calculus

Curry-Howard-de Bruijn isomorphism

schema

polymorphic lambda calculus

Curry-Howard-de Bruijn isomorphism

polymorphic identity

identity on nat:
$\lambda x : \mathsf{nat}.\, x : \mathsf{nat} \to \mathsf{nat}$

identity on bool:
$\lambda x : \mathsf{bool}.\, x : \mathsf{bool} \to \mathsf{bool}$

polymorphic identity:
$\lambda a : * \,.\, \lambda x : a.\, x : \Pi a{:} * \,.\, a \to a$

in polymorphic λ-calculus,
we can abstract over variables a in $*$
that is in Coq: variables in `Set` or `Prop`

instantiation of polymorphic types

by application and β-reduction

application:

$$\frac{\lambda a : * \,.\, \lambda x : a.\, x : \Pi a{:} * \,.\, a \to a \qquad \mathsf{nat} : *}{(\lambda a : * \,.\, \lambda x : a.\, x)\,\mathsf{nat} : \mathsf{nat} \to \mathsf{nat}}$$

beta-reduction:

$(\lambda a : * \,.\, \lambda x : a.\, x)\,\mathsf{nat} \to_\beta \lambda x : \mathsf{nat}.\, x$

polymorphic iteration

iteration on nat:

$\lambda f : \mathsf{nat} \to \mathsf{nat}.\, \lambda x : \mathsf{nat}.\, f\,(f\,x)$

iteration on bool:

$\lambda f : \mathsf{bool} \to \mathsf{bool}.\, \lambda x : \mathsf{bool}.\, f\,(f\,x)$

polymorphic iteration:

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

lambda2: examples

$$\lambda a : \mathsf{Set}.\ \lambda x : a.\ x$$
$$\lambda a : \mathsf{Prop}.\ \lambda x : a.\ x$$
$$\lambda a : \star\ .\ \lambda x : a.\ x$$

$$\lambda a : \mathsf{Set}.\ \lambda x : a.\ \lambda b : \mathsf{Set}.\ \lambda y : a \to b.\ (y\ x)$$
$$\lambda a : \mathsf{Prop}.\ \lambda x : a.\ \lambda b : \mathsf{Prop}.\ \lambda y : a \to b.\ (y\ x)$$
$$\lambda a : \star\ .\ \lambda x : a.\ \lambda b : \star\ .\ \lambda y : a \to b.\ (y\ x)$$

$$\lambda a : \mathsf{Set}.\ \lambda b : \mathsf{Set}.\ \lambda x : a \to b.\ \lambda y : a.\ (x\ y)$$
$$\lambda a : \mathsf{Prop}.\ \lambda b : \mathsf{Prop}.\ \lambda x : a \to b.\ \lambda y : a.\ (x\ y)$$
$$\lambda a : \star\ .\ \lambda b : \star\ .\ \lambda x : a \to b.\ \lambda y : a.\ (x\ y)$$

slogans

simply typed λ-calculus
terms depending on terms
$\lambda x.\, f\, x$

polymorphic λ-calculus terms depending on types $\lambda a : \star.\, \lambda x.\, x$
dependently typed λ-calculus types depending on terms
$\lambda n : \mathsf{nat}.\, \mathsf{prime}\, n$

lambda2: pseudo-terms

sorts	lambda abstraction
$*$, $\Box$	$\lambda x : A.\ N$
variables	product
$x, y, z, \ldots$	$\Pi x{:}A\,.\,N$
	function application
	$F\ N$

typing system for lambda2

selects the terms from the pseudo-terms

used to derive judgements of the form
$\Gamma \vdash M : N$

M is a term if we can derive
$\Gamma \vdash M : A$ or $\Gamma \vdash N : M$

for every kind of term there is a rule

typing system for lambda2: start and var

axiom:

$$\frac{}{\vdash * : \Box}$$

variable:

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$$

typing system for lambda2: app, abs, product

application:

$$\frac{\Gamma \vdash F : \Pi x{:}A \,.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash F\ N : B[x := N]}$$

abstraction:

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x{:}A \,.\, B : \star/\Box}{\Gamma \vdash \lambda x : A.\ M : \Pi x{:}A \,.\, B}$$

product:

$$\frac{\Gamma \vdash A : \star/\Box \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x{:}A \,.\, B : *}$$

typing system for lambda2: weaking and conversion

weakening:

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : \star/\Box}{\Gamma, x : C \vdash A : B}$$

conversion:

$$\frac{\Gamma \vdash A : B' \qquad \Gamma \vdash B : \star/\Box}{\Gamma \vdash A : B} \qquad \text{with } B =_\beta B'$$

the three product rules

all systems:

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x{:}A \,.\, B : *}$$

only in λP (to derive nat $\to \star : \Box$):

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : \Box}{\Gamma \vdash \Pi x{:}A \,.\, B : \Box}$$

only in $\lambda 2$ (to derive $\Pi a : \star.\, a \to a : \star$):

$$\frac{\Gamma \vdash A : \Box \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x{:}A \,.\, B : *}$$

schema

polymorphic lambda calculus

Curry-Howard-de Bruijn isomorphism

Curry-Howard-de Bruijn isomorphism

$$\begin{array}{lcl} \text{prop1} & \sim & \lambda \rightarrow \\ \text{pred1} & \sim & \lambda P \\ \text{prop2} & \sim & \lambda 2 \end{array}$$

introduction rules correspond to abstraction

$$\frac{B}{A \to B}\ I\to \qquad \frac{B}{\forall a.\, B}\ I\forall$$

correspond to

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

$$\frac{\Gamma, a : \star \vdash M : B \qquad \Gamma \vdash \Pi a : \star.\, B : \star}{\Gamma \vdash \lambda a : \star.\, M : \Pi a : \star.\, B}$$

in Coq: intro

elimination rules correspond to application

$$\frac{A \to B \qquad A}{B}\ E\to \qquad \frac{\forall a.\, B}{B[a := A]}\ E\forall$$

correspond to

$$\frac{F : A \to B \qquad M : A}{F\ M : B}$$

$$\frac{F : \Pi a{:}\star\ .\ B \qquad A : \star}{F\ A : B[a := A]}$$

in Coq: apply

examples

$(\forall b.\, b) \to a$

$a \to \forall b.\, (b \to a)$ (cf paradox)

$a \to \forall b.\, ((a \to b) \to b)$

more examples

$$\lambda a : \star \,.\, \lambda x : a.\, x \;:\; \Pi a{:}\star \,.\, \Pi x{:}a \,.\, a$$

$$\lambda a : \star \,.\, \lambda x : a.\, \lambda b : \star \,.\, \lambda y : a \to b.\, (y\, x) \;:\; \Pi a{:}\star \,.\, \Pi x{:}a \,.\, \Pi b{:}\star \,.\, \Pi y{:}a \to b \,.$$

$$\lambda a : \star \,.\, \lambda b : \star \,.\, \lambda x : a \to b.\, \lambda y : a.\, (x\, y) \;:\; \Pi a{:}\star \,.\, \Pi b{:}\star \,.\, \Pi x{:}a \to b \,.\, \Pi y{:}a \,.$$