the type checking problem (TCP)

input: \( \Gamma \vdash M : N \)?

output:
true yes, the typing judgement is derivable
false no, the typing judgement is not derivable

(usually) decidable

the proof checking problem

input: is candidate-proof \( P \) a proof of formula \( A \)?

output:
true yes, \( P \) is a proof of \( A \)
false no, \( P \) is not a proof of \( A \)

corresponds to type checking

the type synthesis problem (TSP)

input: \( \Gamma \vdash M : ? \)

output: yes and term \( N \) typing judgment is derivable
no typing judgment is not derivable

generally decidable
the type inhabitation problem (TIP)

input: $\Gamma \vdash ? : N$

output: yes and term $M$ typing judgment is derivable
no typing judgment is not derivable

generally undecidable (but decidable in $\lambda \rightarrow$)

the provability problem

input: is there a proof of formula $A$?
corresponds to type inhabitation problem

decidability issues

• type inhabitation problem (TIP)
  $\Gamma \vdash ? : A$
  decidable in $\lambda \rightarrow$, undecidable in $\lambda P$ and in $\lambda 2$

• type checking problem (TCP)
  $\Gamma \vdash P : A$?
  decidable in $\lambda \rightarrow$, in $\lambda P$, in $\lambda 2$

• type synthesis problem (TSP) or typability problem
  $\Gamma \vdash P : ?$
  decidable in $\lambda \rightarrow$, in $\lambda P$, in $\lambda 2$

reduction theory

subject reduction
types are preserved under computation
if $\Gamma \vdash M : A$ and $M \rightarrow_{\beta} M'$ then $\Gamma \vdash M' : A$

unique normal forms
result of a computation is unique via confluence:
if $M \rightarrow_{\beta} N$ and $M \rightarrow_{\beta} P$,
then there exists $Q$ such that
$N \rightarrow Q$ and $P \rightarrow Q$

termination (strong normalization)
all computations eventually end in a normal form
big projects in interactive proving (a selection)

- **CompCert** by Xavier Leroy et al. using Coq
- **four colour theorem** by Georges Gonthier using Coq
- **Flyspeck project** by Thomas Hales et al. using HOL Light and Isabelle/HOL
- **L4.verified** by the Trustworthy Systems Research @ Data61 using Isabelle/HOL

Curry–Howard–De Bruijn isomorphism

formulas as types

proofs as terms

Curry–Howard–De Bruijn isomorphism

logic and $\lambda$-calculus

<table>
<thead>
<tr>
<th>Prop</th>
<th>$\sim$</th>
</tr>
</thead>
<tbody>
<tr>
<td>prop1</td>
<td>$\lambda \to$</td>
</tr>
<tr>
<td>pred1</td>
<td>$\lambda P$</td>
</tr>
<tr>
<td>prop2</td>
<td>$\lambda 2$</td>
</tr>
</tbody>
</table>
logic

\[
\text{prop}_1 \subseteq \text{pred}_1
\]

\[
\text{prop}_1 \subseteq \text{prop}_2
\]

\[
\text{minimal} \subseteq \text{intuitionistic} \subseteq \text{classical}
\]

formulas

\[
\begin{array}{cccc}
\text{prop}_1 & \text{pred}_1 & \text{prop}_2 \\
+ & + & + \\
+ & + & + \\
- & + & - \\
- & - & + \\
\end{array}
\]

formulas in Coq

\[
\begin{array}{c}
a & a \\
A \rightarrow B & A \rightarrow B \\
\forall x. B & \text{forall } x:\text{Terms}, B \\
\forall a. B & \text{forall } a:\text{Prop}, B \\
\end{array}
\]

proof rules

introduction rules and elimination rules
proofs in minimal logic

<table>
<thead>
<tr>
<th>prop1</th>
<th>pred1</th>
<th>prop2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I[x] \rightarrow )</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( E \rightarrow )</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( I \forall \text{ (terms)} )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( E \forall \text{ (terms)} )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( I \forall \text{ (prop)} )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( E \forall \text{ (prop)} )</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

proofs in intuitionistic logic

<table>
<thead>
<tr>
<th>prop1</th>
<th>pred1</th>
<th>prop2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I \land )</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( E \land )</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( I \lor )</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( E \lor )</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>( I \bot )</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

proofs in intuitionistic logic

<table>
<thead>
<tr>
<th>prop1</th>
<th>pred1</th>
<th>prop2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I \exists \text{ (terms)} )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( E \exists \text{ (terms)} )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( I \exists \text{ (prop)} )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( E \exists \text{ (prop)} )</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

detour

introduction rule for a connective
immediately followed by an
elimination rule for the same connective
logic: possible questions

- tautology
- correctness
- detours
- encodings in prop2

examples

- in prop1 \(\sim\lambda\rightarrow:\)
  \[A \rightarrow (A \rightarrow B) \rightarrow B\]

- in pred1 \(\sim\lambda P:\)
  \[(\forall x. P(x) \rightarrow Q(x)) \rightarrow P(M) \rightarrow Q(M)\]

- in prop2 \(\sim\lambda 2:\)
  \[a \rightarrow \forall b. ((a \rightarrow b) \rightarrow b)\]

formulas as types

<table>
<thead>
<tr>
<th>(a)</th>
<th>(A \rightarrow B)</th>
<th>(A \rightarrow B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Pi x : A. B)</td>
<td>(\Pi x : A. B)</td>
<td>(\Pi x : A. B)</td>
</tr>
<tr>
<td>(\forall x. B)</td>
<td>(\forall x. B)</td>
<td>(\forall x. B)</td>
</tr>
<tr>
<td>(\forall a. B)</td>
<td>(\forall a. B)</td>
<td>(\forall a. B)</td>
</tr>
</tbody>
</table>

proofs as terms

proof rules correspond to typing rules
quantification elimination and application

\[
\forall a, B \quad \frac{\Gamma \vdash P : \Pi a : \ast B}{\Gamma \vdash (P A) : B[a := A]}
\]

\[
\lambda \to \subset \lambda P
\]

\[
\lambda \to \subset \lambda 2
\]

prop2 \sim \lambda 2

inductive definitions: datatypes and predicates

the lambda cube

terms and simple types

\[
A ::= a \mid A \to A
\]

\[
M ::= x \mid \lambda x : A. M \mid (M M)
\]
terms and types general

$$M ::= x \mid \Pi x : M \mid \lambda x : M \mid (M M)$$

term normalization

$\beta$-reduction rule:

$$(\lambda x : A. M) N \rightarrow_\beta M[x := N]$$

$\beta$-reduction step:

application of the rule in a context

$\beta$-reduction:

a sequence of $\beta$-reduction steps

three product rules

$\lambda \rightarrow, \lambda P, \lambda 2$:

\[
\frac{
\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : *
}{\Gamma \vdash \Pi x : A. B : *}
\]

only $\lambda P$:

\[
\frac{
\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : \square
}{\Gamma \vdash \Pi x : A. B : \square}
\]

only $\lambda 2$:

\[
\frac{
\Gamma \vdash A : \square \quad \Gamma, x : A \vdash B : *
}{\Gamma \vdash \Pi x : A. B : *}
\]

lambda: possible questions

- typing derivation (for $\lambda \rightarrow$)
- terms
- typing rules
- inhabitants
- encodings
Inductive natlist : Set :=
  natnil : natlist
| natcons : nat -> natlist -> natlist.

typing following the product rule for $\lambda^+$:

nat -> natlist -> natlist : Set

Inductive natlist_dep : nat -> Set :=
  | nil_dep : natlist_dep 0
  | cons_dep : forall n : nat,
            nat -> natlist_dep n -> natlist_dep (S n).

typing following the product rule for $\lambda P$:

nat -> Set : Type

Inductive polylist (X : Set) : Set :=
  polynil : polylist X
| polycons : X -> polylist X -> polylist X.

typing following the product rule for $\lambda 2$:

forall X : Set, polylist X : Type