A brief explanation of Coq Principles

Iscas

2017-11-29

Consider :

$$I = \lambda x.\ x$$
$$K = \lambda x.\ \lambda y.\ x$$
$$S = \lambda x.\ \lambda y.\ \lambda z.\ x\ z\ (y\ z)$$

are 3 generators of combinatory logic

Typed combinatory logic - Propositional logic

Terms

$I : \alpha \to \alpha$

$K: \alpha \to (\beta \to \alpha)$

$S: (\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$

$M: \alpha \to \beta,\ \ N: \alpha \ \vdash \ MN: \beta$

Types

??

$A \to A$

$A \to (B \to A)$

$A \to (B \to C)) \to ((A \to B) \to (A \to C))$

$A,\ \ A \to B \vdash B$

Formulas

Typed combinatory logic - Propositional logic

Terms

$I : \alpha \to \alpha$

$K : \alpha \to (\beta \to \alpha)$

$S : (\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$

$M : \alpha \to \beta, \; N : \alpha \vdash MN : \beta$

Types

Proofs

$A \to A$

$A \to (B \to A)$

$A \to (B \to C)) \to ((A \to B) \to (A \to C))$

$A, \; A \to B \vdash B$

Formulas

Typed combinatory logic - Propositional logic

Terms

Types

$I : \alpha \rightarrow \alpha$

$K: \alpha \rightarrow (\beta \rightarrow \alpha)$

$S: (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$

$M: \alpha \rightarrow \beta,\ \ N: \alpha \ \vdash MN: \beta$

Proofs

Formulas

$p: A \rightarrow A$

$q: A \rightarrow (B \rightarrow A)$

$h: A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

$h: A,\ \ p: A \rightarrow B \vdash p\ h: B$

Typed combinatory logic - Propositional logic

Programs

$I : \alpha \rightarrow \alpha$

$K: \alpha \rightarrow (\beta \rightarrow \alpha)$

$S: (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$

$M: \alpha \rightarrow \beta,\ \ N: \alpha \ \vdash MN: \beta$

Types

Proofs

$p: A \rightarrow A$

$q: A \rightarrow (B \rightarrow A)$

$h: A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

$h: A,\ \ p: A \rightarrow B \vdash p\, h: B$

Formulas

Curry-Howard correspondence

terms, programs — proofs

types ⟷ formulas

functions — theorems, lemmas

hydrodynamics ⟷ electricity

typed lambda-calculus

Intuitionistic implicational natural deduction	Lambda calculus type assignment rules
$\dfrac{}{\Gamma_1, \alpha, \Gamma_2 \vdash \alpha}\text{Ax}$	$\dfrac{}{\Gamma_1, x : \alpha, \Gamma_2 \vdash x : \alpha}$
$\dfrac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \rightarrow \beta} \rightarrow I$	$\dfrac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x . t : \alpha \rightarrow \beta}$
$\dfrac{\Gamma \vdash \alpha \rightarrow \beta \qquad \Gamma \vdash \alpha}{\Gamma \vdash \beta} \rightarrow E$	$\dfrac{\Gamma \vdash t : \alpha \rightarrow \beta \qquad \Gamma \vdash u : \alpha}{\Gamma \vdash t\, u : \beta}$

a theory of types

functions are terms, higher-order proofs

formulas may depend on terms, types are value dependent

proof may depend on types

full hierarchy of dependent types

consistency relies on well-foundation

↔ **termination**

extension with inductive data structures

a sequent calculus

h_1: A_1,   h_2: A_2, … h_n: A_n ⊢ B

written in Coq

h_1: A_1
h_2: A_2
…
h_n: A_n
===========
B

and by using hypotheses, lemmas and theorems, one finds

p: B

manipulating environments

h_1: A_1
h_2: A_2
…
h_n: A_n
===========
A → B

h_1: A_1
h_2: A_2
…
h_n: A_n
h_n+1: A
===========
B

h_1: A_1
h_2: A_2
…
h_n: A_n
===========
forall x:A , B

h_1: A_1
h_2: A_2
…
h_n: A_n
x: A
===========
B

apply theorems

```
h_1: A_1
h_2: A → B
...
h_n: A_n
===========
B
```

apply h_2 (←)

```
h_1: A_1
h_2: A → B
...
h_n: A_n
===========
A
```

```
h_1: A_1
h_2: A → C
...
h_n: A
===========
B
```

apply h_2 in h_n (→)

```
h_1: A_1
h_2: A → C
...
h_n: C
===========
B
```

apply equalities

h_1: A_1
h_2: A = B
...
h_n: A_n
===========
B

rewrite h_2

h_1: A_1
h_2: A = B
...
h_n: A_n
===========
A

case analysis

h_1: A_1
h_2: A = B
...
h_n: A_n
===========
B

rewrite h_2

h_1: A_1
h_2: A = B
...
h_n: A_n
===========
A

examples of Coq proofs