

logical verification lecture 13

2011 05 20

overview

big projects in proof checking

- **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
- **verificard** by the München theorem proving group using Isabelle

Curry-Howard-De Bruijn isomorphism

formulas as types

proofs as terms

Curry-Howard-De Bruijn isomorphism

prop1 $\sim \lambda \rightarrow$

pred1 $\sim \lambda P$

prop2 $\sim \lambda 2$

logic and lambda calculus

formulas

types

proofs

terms

proof rules

typing rules

provability

inhabitation

proof checking

type checking

logic

$\text{prop1} \subset \text{pred1}$

$\text{prop1} \subset \text{prop2}$

$\text{minimal} \subset \text{intuitionistic} \subset \text{classical}$

formulas

	prop1	pred1	prop2
a	+	+	+
$A \rightarrow B$	+	+	+
$\forall x. B$	-	+	-
$\forall a. B$	-	-	+

formulas in Coq

a

a

$A \rightarrow B$

$A \rightarrow B$

$\forall x. B$

forall x :Terms, B

$\forall a. B$

forall a :Prop, B

proof rules

introduction rules and elimination rules

proofs in minimal logic

	prop1	pred1	prop2
$I[x] \rightarrow$	+	+	+
$E \rightarrow$	+	+	+
$I\forall$ (terms)	-	+	-
$E\forall$ (terms)	-	+	-
$I\forall$ (prop)	-	-	+
$E\forall$ (prop)	-	-	+

proofs in intuitionistic logic

	prop1	pred1	prop2
$I\wedge$	+	+	+
$E\wedge$	+	+	+
$I\vee$	+	+	+
$E\vee$	+	+	+
$E\perp$	+	+	+

proofs in intuitionistic logic

	prop1	pred1	prop2
$I\exists$ (terms)	-	+	-
$E\exists$ (terms)	-	+	-
$I\exists$ (prop)	-	-	+
$E\exists$ (prop)	-	-	+

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 $\text{prop1} \sim \lambda \rightarrow$:
 $A \rightarrow (A \rightarrow B) \rightarrow B$
- in $\text{pred1} \sim \lambda P$:
 $(\forall x. P(x) \rightarrow Q(x)) \rightarrow P(M) \rightarrow Q(M)$
- in $\text{prop2} \sim \lambda 2$:
 $a \rightarrow \forall b. ((a \rightarrow b) \rightarrow b)$

formulas as types

a

$A \rightarrow B$

$\forall x. B$

$\forall a. B$

a

$\Pi x:A. B$

$\Pi x:\text{Terms}. B$

$\Pi a:*. B$

proofs as terms

proof rules correspond to typing rules

implication introduction and abstraction

$$\frac{B}{A \rightarrow B}$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash A \rightarrow B : *}{\Gamma \vdash \lambda x : A. b : A \rightarrow B}$$

prop1 ~ $\lambda \rightarrow$
pred1 ~ λP
prop2 ~ $\lambda 2$

implication elimination and application

$$\frac{A \rightarrow B \quad A}{B}$$

$$\frac{\Gamma \vdash P : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash (P a) : B}$$

prop1 ~ $\lambda \rightarrow$
pred1 ~ λP
prop2 ~ $\lambda 2$

quantification introduction and abstraction

$$\frac{B}{\forall x. B}$$

$$\frac{\Gamma, x : \text{Terms} \vdash b : B \quad \Gamma \vdash \Pi a : \text{Terms} . B : *}{\Gamma \vdash \lambda a : \text{Terms} . b : \Pi a : \text{Terms} . B}$$

$$\text{pred1} \sim \lambda P$$

quantification elimination and application

$$\frac{\forall x. B}{B[x := M]}$$

$$\frac{\Gamma \vdash P : \Pi x:\text{Terms}. B \quad \Gamma \vdash M : \text{Terms}}{\Gamma \vdash (P M) : B[x := M]}$$

$$\text{pred1} \sim \lambda P$$

quantification elimination and application

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

$$\frac{\Gamma \vdash P : \Pi a : * . B \quad \Gamma \vdash A : *}{\Gamma \vdash (P A) : B[a := A]}$$

prop2 ~ λ2

lambda calculus

$\lambda \rightarrow \subset \lambda P$

$\lambda \rightarrow \subset \lambda 2$

inductive definitions: data-types and predicates

the lambda cube

terms and simple types

$$A ::= a \mid A \rightarrow A$$
$$M ::= x \mid \lambda x : A. M \mid (M M)$$

terms and types general

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

term normalization

β -reduction rule:

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

β -reduction step:

application of the rule in a context

β -reduction:

a sequence of β -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

decidability issues

- type checking problem (TCP)
 $\Gamma \vdash P : A?$
- type inhabitation problem (TIP)
 $\Gamma \vdash ? : A$
- type synthesis problem (TSP) or typability problem
 $\Gamma \vdash P : ?$

decidability

	$\lambda \rightarrow$	λP	$\lambda 2$
TCP	+	+	+
TIP	+	-	-
TSP	+	-	-

inductive datatype: example

```
Inductive natlist : Set :=  
  natnil : natlist  
| natcons : nat -> natlist -> natlist.
```

typing following the product rule for λ^{\rightarrow} :

```
nat -> natlist -> natlist : Set
```

family of inductive datatypes: example

```
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 λP :

```
nat -> Set : Type
```

family of inductive datatypes: example

```
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
```