overview

simply typed lambda-calculus
Curry–Howard–De Bruijn isomorphism: dynamics

prop1

prop1 and Coq

<table>
<thead>
<tr>
<th>connective</th>
<th>prop1 rules</th>
<th>Coq tactics</th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>$I[x] \to$</td>
<td>intro x</td>
</tr>
<tr>
<td></td>
<td>$E \to$</td>
<td>apply</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$E \bot$</td>
<td>elimtype False</td>
</tr>
<tr>
<td>$\land$</td>
<td>$I \land$</td>
<td>split</td>
</tr>
<tr>
<td></td>
<td>$E \land$</td>
<td>elim x</td>
</tr>
<tr>
<td>$\lor$</td>
<td>$I \lor$</td>
<td>left right</td>
</tr>
<tr>
<td></td>
<td>$E \lor$</td>
<td>elim x</td>
</tr>
</tbody>
</table>
classical logic

start with intuitionistic logic
add a classical axiom

from intuitionistic to classical logic

• add the law of excluded middle
  \( A \lor \neg A \)

• add the double negation rule
  \( \neg \neg A \to A \)

• add Peirce’s law
  \( ((A \to B) \to A) \to A \)

classical logic: examples of tautologies

assume the law of excluded middle

• double negation
  \( \neg \neg A \to A \)

• Peirce’s Law
  \( ((A \to B) \to A) \to A \)

constructive versus classical logic

constructive point of view:
main issue: provability
does the formula have a proof?

classical point of view:
main issue: truth
is the formula true?
constructive point of view (background)

Brouwer–Heyting–Kolmogorov interpretation

- proof of $A \rightarrow B$  $\sim$ function that maps proofs of $A$ to proofs of $B$
- proof of $A \land B$  $\sim$ pair of a proof of $A$ and a proof of $B$
- proof of $A \lor B$  $\sim$ either a proof of $A$ or a proof of $B$
- proof of $\bot$  does not exist

overview: propositional logic

- minimal logic (ML)
  $$(((A \rightarrow B) \rightarrow A) \rightarrow B) \rightarrow B)$$
- ML + $\bot$
  $$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$$
- intuitionistic logic
  $$A \lor B \rightarrow (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C$$
- classical logic
  $$A \lor \neg A$$

decidability

is a given proposition $A$ a tautology?

- decidable for intuitionistic prop1 using Heyting algebras
- decidable for classical prop1 using truth tables

what is the complexity of the decision procedures?

- using Heyting algebras: $2^{2^n}$ for formula of length $n$
- Statman: can be done in polynomial space
- using truth tables: $2^n$ with $n$ the number of propositional variables

simply typed $\lambda$-calculus: statics

- grammar for simple types
- grammar for terms (depends on grammar for types)
- typing system for deriving $\Gamma \vdash M : A$
  (in environment $\Gamma$ the term $M$ lives in type $A$)
- this corresponds to minimal prop1
Type Checking Problem (TCP) or typability problem
\[ \Gamma \vdash M : A \]

Type Synthesis Problem (TSP)
\[ \Gamma \vdash M : ? \]

Type Inhabitation Problem (TIP)
\[ \Gamma \vdash ? : A \]

for \( \lambda \rightarrow \) all decidable

for more difficult logics TIP becomes undecidable

**term normalization: \( \beta \)-reduction**

A \( \lambda \)-term may be reduced or rewritten or evaluated

**type derivation: some lemmas**

- **uniqueness of types**
  If \( \Gamma \vdash M : A \) and \( \Gamma \vdash M : B \), then \( A = B \)
  (NB: not for Curry style)

- **substitution property**
  If \( \Gamma, x : A, \Gamma' \vdash M : B \) and \( \Gamma \vdash P : A \), then \( \Gamma, \Gamma' \vdash M[x := P] : B \)

- **thinning or weakening**
  If \( \Gamma \vdash M : A \) and \( \Gamma \subseteq \Gamma' \), then \( \Gamma' \vdash M : A \)

- **strengthening**
  If \( \Gamma, x : B \vdash M : A \) and \( x \) is not free in \( M \), then \( \Gamma \vdash M : A \)

**\( \beta \)-reduction: definitions**

- **\( \beta \)-reduction rule:**
  \[ (\lambda x : A. M) N \rightarrow_\beta M[x := N] \]

- **\( \beta \)-reduction step:**
  Application of the rule in a context (a bigger term)

- **\( \beta \)-reduction**
  A sequence of \( \beta \)-reduction steps
\[ (\lambda p : \text{nat} \to \text{bool} \lambda x : \text{nat}. p x) \text{ even } 3 \to_\beta ? \]
\[ (\lambda f : \text{nat} \to \text{nat} \, 2) (\lambda x : \text{nat}. x) \to_\beta ? \]
\[ (\lambda x : \text{nat}. f x x) \, 2 \to_\beta ? \]
\[ (\lambda x : \text{nat}. 2) \, 3 \to_\beta ? \]

**β-reduction: some theorems**

subject reduction  
- types are preserved under computation  
  - if \( \Gamma \vdash M : A \) and \( M \to_\beta M' \), then \( \Gamma \vdash M' : A \)

unique normal forms  
- result of a computation is unique  
  - if \( M \to_\beta^* P_1 \) and \( M \to_\beta^* P_2 \) with \( P_1 \) and \( P_2 \) normal forms, then \( P_1 \equiv P_2 \)

termination  
- all computations terminate  
  - there is no infinite \( \beta \)-reduction sequence

**Curry–Howard–De Bruijn isomorphism: statics**

| ML | \( \to \)  
| formula | type  
| propositional variable | type variable  
| connective | type constructor | \( \to \)  

| proof | term  
| assumption | term variable  
| implication introduction | abstraction  
| implication elimination | application
what about the dynamic part?

is term normalization related to proof normalization?

yes: and this is the dynamic part of the Curry–Howard isomorphism

example

\[ \lambda x : A. ((\lambda y : A. \lambda z : B. y) x) \]

Curry–Howard–De Bruijn isomorphism

- proof normalization \( \sim \) \( \beta \)-reduction
- detour \( \sim \) redex
- normalization step \( \sim \) reduction step
- normal proof \( \sim \) normal form

first-order predicate logic (pred1)

- predicate:
  - atomic formula is built from a predicate and zero one or more terms

- first-order:
  - no quantification over formulas or predicates
first-order predicate logic (pred1)

syntax:
- terms (new compared to prop1)
- formulas
- judgments

proof rules:
- introduction rules
- elimination rules

terms: example

domain: \( \text{nat} = \{0, 1, 2, 3, \ldots\} \)

functions: addition, division

terms: 3, 5, 3 + 5

terms: definition

- a variable is a term
  \( x \) in Terms

- applying a function symbol to the right number of terms yields a term
  if \( M_1, \ldots, M_n \) in Terms then
  \( f(M_1, \ldots, M_n) \) in Terms

this is singly-sorted; we could work in a multi-sorted setting

predicates: example

- \( P(n) \) meaning \( n \) is a prime number
- \( E(n) \) meaning \( n \) is even
- \( Q(n, m) \) meaning \( n \) divides \( m \)
formulas: definition for prop1 (already seen)

\[ a \, b \, c \, p \, q \]
\[ A \rightarrow B \]
\[ \bot \]
\[ \top \]
\[ A \land B \]
\[ A \lor B \]

formulas: definition for pred1

\[ a(\ldots) \, b(\ldots) \, c(\ldots) \, p(\ldots) \, q(\ldots) \]
\[ A \rightarrow B \]
\[ \forall x. B \]
\[ \bot \]
\[ \top \]
\[ A \land B \]
\[ A \lor B \]
\[ \exists x. A \]

terminology: first-order

first order:
object

second order:
set of first-order objects
predicate on objects

first-order logic:
quantification over variables of order 1
\[ a \rightarrow a \]
\[ \forall x. a(x) \rightarrow a(x) \]

formulas: examples

in prop1:
\[ a \rightarrow a \]
in pred1:
\[ \forall x. a(x) \rightarrow a(x) \]