overview

prop1

simply typed lambda-calculus

Curry–Howard–De Bruijn isomorphism: dynamics

pred1
<table>
<thead>
<tr>
<th>Connective</th>
<th>Prop1 Rules</th>
<th>Coq Tactics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow$</td>
<td>$I[x] \rightarrow$</td>
<td>intro x</td>
</tr>
<tr>
<td>$E \rightarrow$</td>
<td>$E \bot$</td>
<td>elimtype False</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>$E \land$</td>
<td>elim x</td>
<td></td>
</tr>
<tr>
<td>$\lor$</td>
<td>$I \lor I \lor r$</td>
<td>left right</td>
</tr>
<tr>
<td>$E \lor$</td>
<td>elim x</td>
<td></td>
</tr>
</tbody>
</table>
prop1 and Coq

the connective or type constructor → is built-in

the connectives ⊥, ∧, ∨ are defined as inductive types
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
  \[ \neg\neg A \rightarrow A \]

- add Peirce’s law
  \[ ((A \rightarrow B) \rightarrow A) \rightarrow A \]
classical logic: examples of tautologies

assume the law of excluded middle

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

• Peirce’s Law
  \( ((A \rightarrow B) \rightarrow A) \rightarrow A \)
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 \( A \to B \) \( \sim \) 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 \( \bot \) \( \sim \) does not exist
overview: propositional logic

- **minimal logic (ML)**
  \[ ((((((A \rightarrow B) \rightarrow A) \rightarrow A) \rightarrow B) \rightarrow B) \]

- **ML + ⊥**
  \[ (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A) \]

- **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 λ-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 derivation: important questions

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
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 $\Gamma \vdash M : A$
a \lambda\text{-term} may be reduced or rewritten or evaluated
\( \beta \)-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
\(\beta\)-reduction: examples

\[
(\lambda p : \text{nat} \to \text{bool}. \lambda x : \text{nat}. p \ x) \text{ even } 3 \to_\beta ?
\]

\[
(\lambda f : \text{nat} \to \text{nat}. f \ 2) (\lambda x : \text{nat}. x) \to_\beta ?
\]

\[
(\lambda x : \text{nat}. f \ x \ x) \ 2 \to_\beta ?
\]

\[
(\lambda x : \text{nat}. 2) \ 3 \to_\beta ?
\]
\(\beta\)-reduction: normal form

\(\beta\)-redex:
subterm of the form \((\lambda x : A. M) N\)

normal form:
term without a \(\beta\)-redex
\(\beta\)-reduction: some theorems

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

If \( M \rightarrow^* P_1 \) \( M \rightarrow^* P_2 \) 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

\[ \text{ML} \sim \lambda \to \]

- formula $\sim$ type
- propositional variable $\sim$ type variable
- connective $\to$ $\sim$ type constructor $\to$

- proof $\sim$ term
- assumption $\sim$ term variable
- implication introduction $\sim$ abstraction
- implication elimination $\sim$ 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
Curry–Howard–De Bruijn isomorphism

- Proof normalization $\sim$ $\beta$-reduction
- Detour $\sim$ Redex
- Normalization step $\sim$ Reduction step
- Normal proof $\sim$ Normal form
\[ \lambda x : A. ((\lambda y : A. \lambda z : B. y) x) \]

% nu naast bewijs met detour naar normaalvorm
% type derivation van \( \lambda x : A. ((\lambda y : A. \lambda z : B. y) x) \)
% en dan de $\beta$-stap naar \( \lambda x : A. \lambda z : B. x \)
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
  \[ x \text{ in Terms} \]

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

  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)

\( a b c p q \)

\( A \rightarrow B \)

\( \bot \)

\( \top \)

\( A \wedge B \)

\( A \vee B \)
formulas: definition for pred1

\[ a(\ldots) \ b(\ldots) \ c(\ldots) \ p(\ldots) \ q(\ldots) \]

\[ A \rightarrow B \]

\[ \forall x. B \]

\[ \bot \]

\[ \top \]

\[ A \land B \]

\[ A \lor B \]

\[ \exists x. A \]
formulas: examples

in prop1:
\[ a \rightarrow a \]

in pred1:
\[ \forall x. a(x) \rightarrow a(x) \]
first order:
object

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

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