CS3202: Logic, Specification and Verification

CS3202-LSV 2006–07

`cs3202.lec@cs.st-andrews.ac.uk`

Dr. James McKinna, Rm 1.03

Dr. Stéphane Lengrand, Rm. 1.02

Lecture 4 (20/02/2007):

Intuitionistic vs. classical logic

Decorating proof-trees with variables and terms

A typo in the tutorial sheet

- Prove? $\vdash ((A \Rightarrow B) \Rightarrow B) \Rightarrow A$
- Check whether it is a tautology. Conclusion?

A typo in the tutorial sheet

- Prove? $\vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
- Check whether it is a tautology. Conclusion?

Truthtable

-

A	B	$A \Rightarrow B$	$(A \Rightarrow B) \Rightarrow A$	$((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
T	T	T	T	T
T	F	F	T	T
F	T	T	F	T
F	F	T	F	T

Reasoning by contradiction

- $$\dfrac{\dfrac{\dfrac{\dfrac{[(A \Rightarrow B) \Rightarrow A] \qquad \dfrac{\dfrac{\dfrac{[A] \quad [\neg A]}{\bot}}{B}}{A \Rightarrow B}}{A} \qquad [\neg A]}{\bot}}{A}}{((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

Is this correct?

Reasoning by contradiction

- In fact we proved:

$$
\dfrac{\dfrac{\dfrac{\dfrac{[(A \Rightarrow B) \Rightarrow A] \qquad \dfrac{\dfrac{\dfrac{[A] \quad [\neg A]}{\bot}}{B}}{A \Rightarrow B}}{A} \qquad [\neg A]}{\bot}}{\neg\neg A}}{((A \Rightarrow B) \Rightarrow A) \Rightarrow \neg\neg A}
$$

Natural deduction: soundness & completeness

- Soundness: $\phi_1, \ldots, \phi_n \vdash \phi$ implies $\phi_1, \ldots, \phi_n \models \phi$.

  Completeness: $\phi_1, \ldots, \phi_n \models \phi$ implies $\phi_1, \ldots, \phi_n \vdash \phi$?

  No.

Classical Logic vs. Intuitionistic Logic

- Last lectures: *Intuitionistic Logic*

  Semantically *incomplete* (w.r.t. truthtables)

- *Classical logic* is obtained by adding:

Reasoning by contradiction

$$\frac{\begin{array}{c}[\neg A] \\ \vdots \\ \bot\end{array}}{A}$$

Elimination of Double Negation

$$\frac{\neg\neg A}{A}$$

Peirce's Law

$$\frac{(A \Rightarrow B) \Rightarrow A}{A}$$

Semantically *complete* (w.r.t. truthtables)

Proof-terms for proof-trees

Linking open/active leaves and variables

- We use *Variables* to keep track of the set of open/active leaves.
- In Coq: `Variable H:A.`
- Idea: Represent proofs with terms /

  Decorate inference rules & steps with terms and variable annotations
- Along proof tree: need to keep track of the link open/active leaves — variables in an *Environment* (a.k.a. *Context*):

  Environment = mapping from variables to wff (e.g. $x : A$ for $x \mapsto A$)
- In decorated trees: Nodes are labelled with *environment+term+wff*

$$\Gamma \vdash M : A$$

Typing, proofs...

- *Noooooo!* same symbol as in $\phi_1, \ldots, \phi_n \vdash \phi$!

  But... it will be the case (to be checked in each decorated rule later) that

  There exists an M & a (decorated) proof-tree of

  $$x_1 : \phi_1, \ldots, x_n : \phi_n \vdash M : \phi$$

  if & only if

  $$\phi_1, \ldots, \phi_n \vdash \phi$$

- Notion of *Typing*: $\Gamma \vdash M : A$ "M is of type A in the environment Γ"
  = "M represents a proof of A under the (decorated) assumptions Γ"

Natural deduction as a Typing System

- Use of an hypothesis:

$$\overline{\Gamma, x : A \vdash x : A}$$

On the l.-h. side, the *wff* A is declared to be available as an open assumption, decorated with x.

The *proof-tree* A (or rather, $\dfrac{A}{A}\,\mathsf{copy}$?), having the wff A as an open assumption, concludes A and is decorated by x in the environment $\Gamma, x : A$.

"copy" = "use" ?

Natural deduction as a Typing System

- $\Rightarrow$-introduction:

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x : A.M) : A \Rightarrow B}$$

NB: In Coq's concrete syntax

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\texttt{fun}\ \ x : A => M) : A \rightarrow B}$$

In fact, no need for `imp_i`:

`imp_i A B (fun x:A=>M)` is *`fun x:A=>M`*

This is **(Creation of) unnamed function** (see next slide)

Natural deduction as a Typing System

- COQ's definitions: `Let myfavoritename := myterm.`
  or `Definition myfavoritename := myterm.`
- Hence, think of `Let myfunction := fun x:A => M.` as

  ```
  myfunction (x:A){
  ...
  M
  ...
  }
  ```

  It is of type `A->B` (if `M` is of type `B`)

  `x` is only available in the body `M` of the function.

  `A` is an assumption temporarily made for the (sub-)proof `M` of `B`.

Natural deduction as a Typing System

- $\Rightarrow$-elimination (a.k.a. *Modus Ponens*):

$$\frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

Again, no need for `imp_e`: *`imp_e A B M N`* is *`M N`*

This is **Function application**

Natural deduction as a Typing System

- $\vee$-introduction:

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \texttt{or\_introl}\ M : A \vee B} \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash \texttt{or\_intror}\ M : A \vee B}$$

Again, no need for `or_il`: *or_il A B M* is *or_introl M* and similarly for `or_ir`.

This is a **Cast** in a union type, with a tag to remember which side of the union a term comes from.

Natural deduction as a Typing System

- $\vee$-elimination:

$$\frac{\Gamma \vdash M : A \vee B \qquad \Gamma, x : A \vdash N : C \qquad \Gamma, y : B \vdash P : C}{\Gamma \vdash \begin{array}{l}\texttt{match } M \texttt{ with} \\ \texttt{or\_introl } x \texttt{ => } N \\ \texttt{|or\_intror } y \texttt{ => } P \\ \texttt{end}\end{array} : C}$$

This is a **Case analysis** on the tag specifying which part of the union M comes from.

`or_e A B M (fun x:A=>N) (fun y:A=>P)`

is

```
match M with
or_introl x => N
|or_intror y => P
end
```

.

Natural deduction as a Typing System

- $\wedge$-introduction:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \texttt{conj}\ M\ N : A \wedge B} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \texttt{pair}\ M\ N : A * B}$$

$\texttt{pair} M N$ abbreviated as (M, N)

Again, no need for `and_i`: *`and_i A B M N`* is *`conj M N`*

This is a **Pair** / 2-component **Structure**.

Natural deduction as a Typing System

- $\bigwedge$-elimination:

$$\frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \begin{array}{l}\texttt{match } M \texttt{ with}\\ \texttt{conj } x\, y \texttt{ => } x\\ \texttt{end}\end{array} : A} \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \begin{array}{l}\texttt{match } M \texttt{ with}\\ \texttt{conj } x\, y \texttt{ => } y\\ \texttt{end}\end{array} : B}$$

This is the **Access to one component of a Pair / binary structure**.

Natural deduction as a Typing System

- Point raised: Type "Inference"
- With these rules, it is true that

  Given Γ, M, there is *at most one* type A s.t. there is a decorated proof-tree concluding $\Gamma \vdash M : A$.

  (& there is an algorithm to find it: run by `Check` command in Coq)

- Exercise: check this fact in all the decorated rules.

  In particular, it relies on indicating ": A" in $\longrightarrow$-intro rule ($\lambda x : A.M$)

  Otherwise, how would you find the type of $\lambda x.x$?

  Exercise: why is it not needed in the discharges of $\vee$-elim?

Natural deduction: the actual rules for $\perp$ and $\neg$

- So far, no computational interpretation of $\perp$ and $\neg$:

- $\perp$-introduction: none

  $\perp$-elimination: $\dfrac{\perp}{A}$

- $\neg$-introduction: $\dfrac{\begin{array}{c}[A] \\ \vdots \\ \perp\end{array}}{\neg A}$

  $\neg$-elimination: $\dfrac{A \qquad \neg A}{\perp}$

Natural deduction as a Typing System *for λ-calculus*

- Implicational fragment ($\Rightarrow$ as the only connective), in *intuitionistic logic*. Syntax obtained:

$$M, N, P, \ldots ::= x \mid \lambda x : A.M \mid M\ N$$

is the *λ-calculus* (Church, 30's).

Set of variables decorating open assumptions (k.a. **Free variables**):

$$\begin{aligned}
\mathsf{FV}(x) &= x \\
\mathsf{FV}(M\ N) &= \mathsf{FV}(M) \cup \mathsf{FV}(N) \\
\mathsf{FV}(\lambda x : A.M) &= \mathsf{FV}(M) \setminus \{x\}
\end{aligned}$$

Full lecture on it later.

Questions?