TERM ASSIGNMENT

A "term assignment" for Natural Deduction

[Church 1930s
Curry 1950s   Howard 60s
Prawitz, '65 , Martin-Löf]

Propositions $\cong$ Types

Proofs $\cong$ Programs

$$\begin{array}{c} \pi \\ \Gamma \vdash p \end{array} \quad \cong \quad \Gamma \vdash e : \tau$$

Γ — set of propositions

Label each element of Γ by a distinct variable.

Helps us track which hypothesis is being used (and where) in a proof.

- Absurdity $\perp$ can't have a proof
- A proof of $p_1 \wedge p_2$ is a proof of p_1 and a proof of p_2
- A proof of $p_1 \vee p_2$ is a proof of p_1 or a proof of p_2, with a marker saying which
- A proof of $p_1 \rightarrow p_2$ is a (function) method that takes a proof of p_1, and (possibly using it) returns a proof of p_2.
- A proof of $\neg p$ is a proof that says that a proof of p would be absurd.

(HYP) = VAR

$$\frac{}{\Gamma \vdash x : p} \quad x : p \in \Gamma$$

$(\to I)$ = ABS

$$\frac{\Gamma, x : p_1 \vdash e : p_2}{\Gamma \vdash \lambda x . e : p_1 \to p_2}$$

$(\to E)$ = APP

$$\frac{\Gamma \vdash e_1 : p_1 \to p_2 \qquad \Gamma \vdash e_2 : p_1}{\Gamma \vdash (e_1\ e_2) : p_2}$$

$(\top\text{-}I)$

$$\frac{}{\Gamma \vdash () : \top}$$

$(\wedge I)$

$$\frac{\Gamma \vdash e_1 : p_1 \qquad \Gamma \vdash e_2 : p_2}{\Gamma \vdash (e_1, e_2) : p_1 \wedge p_2}$$

$$(\wedge E_\ell) \quad \frac{\Gamma \vdash e : p_1 \wedge p_2}{\Gamma \vdash \mathrm{proj}_1^{(2)}\, e : p_1}$$

$$(\wedge E_r) \quad \frac{\Gamma \vdash e : p_1 \wedge p_2}{\Gamma \vdash \mathrm{proj}_2^{(2)}\, e : p_2}$$

$$(\vee I_\ell) \quad \frac{\Gamma \vdash e : p_1}{\Gamma \vdash \mathrm{inl}(e) : p_1 \vee p_2}$$

$$(\vee I_r) \quad \frac{\Gamma \vdash e : p_2}{\Gamma \vdash \mathrm{inr}(e) : p_1 \vee p_2}$$

$$(\vee E) \quad \frac{\Gamma \vdash e_0 : p_1 \vee p_2 \qquad \Gamma, x : p_1 \vdash e_1 : q \qquad \Gamma, y : p_2 \vdash e_2 : q}{\Gamma \vdash \underline{\mathrm{case}}\ e_0\ \underline{\mathrm{of}}\ \mathrm{inl}(x) \to e_1 \parallel \ \ldots : q}$$

$\mathrm{inr}(y) \to e_2$

Will not (yet) give you the rules for negation ...
the model of computation is not easy to understand
("continuations" GRIFFIN'90)

A functional language

$$
\begin{aligned}
e \in \mathcal{E}xp ::= \; & x && \text{variable} \\
| \; & \lambda x . e && \text{function} \\
| \; & (e_1 \; e_2) && \text{application} \\
| \; & () \\
| \; & (e_1 , e_2) && \text{pair} \\
| \; & proj_i^{(2)} \; e && i = 1, 2 \text{ projection} \\
| \; & inl(e) && \text{constr left} \\
| \; & inr(e) && \text{constr right} \\
| \; & \underline{case} \; e_0 \; \underline{of} \; inl(x) \to e_1 \mid inr(y) \to e_2 && \text{case/match/switch}
\end{aligned}
$$

free and bound variables

(free, binding, bound occurrences)

focussing on the first sub-language

x , $\lambda x.e$, $(e_1\ e_2)$

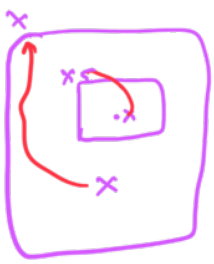
binding occurrence of x

scope of binding of x.

x "free" if not in the scope of a binding occurrence of x

x bound if in the scope of a binding occurrence of x

scope inside a scope



Can rename systematically the binding & corresponding bound occurrences w/o changing meaning ...
"α- conversion".

$$fv(x) = \{x\}$$

$$fv((e_1\ e_2)) = fv(e_1) \cup fv(e_2)$$

$$fv(\lambda x.e) = fv(e) - \{x\}$$

Recall

Substitution in the presence of binders $\lambda\_.\_$

$[x := e_2]e_1$ Substitute e_2 for all free occurrences of x in e_1

- Should replace all and only free occurrences of x in e_1 by e_2
- Should not "capture" any free z in e_2 by a binding $\lambda z.\_$ in e_1

(free in e_2 should stay free.)

Case-wise specification ($[x := e_2]e_1$ written as $e_1[e_2/x]$)

	e_1	$e_1[e_2/x]$	
(a)	x	e_2	
(b)	y	y	$y \not\equiv x$
(c)	$(e_{11} \ \ e_{12})$	$((e_{11}[e_2/x]) \ (e_{12}[e_2/x]))$	
(d)	$\lambda x.e_{11}$	$\lambda x.e_{11}$	
(e)	$\lambda y.e_{11}$	$\lambda y.(e_{11}[e_2/x])$	$y \not\equiv x$ and $(x \notin fv(e_{11})$ or $y \notin fv(e_2))$
(f)	$\lambda y.e_{11}$	$\lambda z.((e_{11}[z/y])[e_2/x])$	$y \not\equiv x$ and $x \in fv(e_{11})$ and $y \in fv(e_2)$; $z \notin fv(e_{11})$, $z \notin fv(e_2)$ "z fresh"

"Execution"

$$(\beta)\quad \overline{((\lambda x.e_1)\ e_2) \ \triangleright_{1\beta}\ e_1[e_2/x]}$$

$\circledast$

$$(\text{op})\quad \frac{e_1 \ \triangleright_{1\beta}\ e_1'}{(e_1\ e_2)\ \triangleright_{1\beta}\ (e_1'\ e_2)}$$

$$(\text{arg})\quad \frac{e_2 \ \triangleright_{1\beta}\ e_2'}{(e_1\ e_2)\ \triangleright_{1\beta}\ (e_1\ e_2')}$$

$\circledast\circledast$

$$(\xi)\quad \frac{e \ \triangleright_{1\beta}\ e'}{\lambda x.e\ \triangleright_{1\beta}\ \lambda x.e'}$$

$\boxed{\ast}$

- Most programming language (all?) <u>never</u> use (ξ) rule. $\boxed{\cancel{\xi}}$

- Call by value (SML, OCaml, LISP, ...) put restrictions on (β) and (arg) rules. (β_v), (arg_v)

- Call by name (Haskell, Miranda, ...) do not use (arg) rule

- No restriction: Theoretical λ-calculus

SUBJECT REDUCTION THEOREM

If $\Gamma \vdash e : p$

and $e \rhd_{1\beta} e'$

then $\Gamma \vdash e' : p$

Type does not change during evaluation (execution)

SUBSTITUTION LEMMA (TYPING)

If $\Gamma, x : p_1 \vdash e_1 : p_2$

and $\Gamma \vdash e_2 : p_1$

then $\Gamma \vdash e_1[e_2/x] : p_2$

Proof of Substitution Lemma:

By induction on structure of e_1

- $e_1 \equiv x$

$\Gamma, x : p_1 \vdash x : p_2 \qquad (\therefore p_1 \equiv p_2)$

Defn of subst. $\quad e_1[e_2/x] \equiv e_2$

But we are given $\Gamma \vdash e_2 : p_2$

$\cdot \quad e_1 \equiv y \quad \neq \wedge$

$\Gamma, x:\phi_1 \vdash y:\phi_2$ $\quad y:\phi_2 \in \Gamma$ $\quad x \neq y$

Defn of subst

$e_1[e_2/x] \equiv y$

$\Gamma \vdash y:\phi_2$

$\cdot \; e_1 \equiv (e_{11} \quad e_{12})$

$\Gamma, x:\phi_1 \vdash (e_{11} \; e_{12}) : \phi_2$

why?

$\Gamma, x:\phi_1 \vdash e_{11} : \phi_{21} \to \phi_2$

$\Gamma, x:\phi_1 \vdash e_{12} : \phi_{21}$

Defn of subst

$e_1[e_2/x] \equiv ((e_{11}[e_2/x]) \quad (e_{12}[e_2/x]))$

$\Gamma \vdash e_{11}[e_2/x] : \phi_{21} \to \phi_2$

$\Gamma \vdash e_{12}[e_2/x] : \phi_{21}$

$\Gamma \vdash e_{12}[e_2/x] : \phi_{21}$

$$\frac{}{\Gamma \vdash \left(\left(e_{11}[e_2/x]\right) \left(e_{12}[e_2/x]\right) \right) : \phi_2} (\to E)$$

- e_1 is $\lambda x. e_{11}$

$$\Gamma, x:\phi_1 \vdash \lambda x. e_{11} : \phi_{21} \to \phi_{22}$$

why?

$$\Gamma, x:\phi_1, x:\phi_{21} \vdash \underbrace{e_{11} : \phi_{22}}_{\text{any free } x \text{ here}}$$

is

By defn of subst.

$$\lambda x. e_{11} [e_2/x] \text{ is } \lambda x. e_{11}$$

$$\frac{\Gamma, x:\phi_{21} \vdash e_{11} : \phi_{22}}{\Gamma \vdash \lambda x. e_{11} : \phi_{21} \to \phi_{22}}$$

- e_1 is $\lambda y. e_{11}$ with $y \not\equiv x$.

$$\Gamma, x : \phi_1 \vdash \lambda y . e_{11} : \phi_{21} \to \phi_{22}$$

$$\Gamma, x : \phi_1, y : \phi_{21} \vdash e_{11} : \phi_{22}$$

by defn of subst

$$\boxed{(\lambda y . e_{11})[e_2/x] \equiv \lambda y . (e_{11}[e_2/x])}$$

$y \not\equiv x$, $x \notin fv(e_{11})$

or $\underline{y \notin fv(e_2)}$

$$(\to I)\ \frac{\Gamma, y : \phi_{21} \vdash e_{11}[e_2/x] : \phi_{22}}{\Gamma \vdash \lambda y . (e_{11}[e_2/x]) : \phi_{21} \to \phi_{22}}$$

(TRY OUR LEVEL BEST TO AVOID CASE (f))

- e_1 is $\lambda y . e_{11}$ with $x \not\equiv y$

  $y \in fv(e_2)$

  $x \in fv(e_{11})$

$$\Gamma, x : \phi_1 \vdash \lambda y . e_{11} : \phi_{21} \to \phi_{22}$$

$$\Gamma, x : \phi_1, y : \phi_{21} \vdash e_{11} : \phi_{22}$$

By defn of subst

$$\lambda y. e_{11} [e_2/x] \equiv \lambda z. ((e_{11} [z/y]) [e_2/x])$$

Claim: $\Gamma, x:\phi_1, z:\phi_{21} \vdash e_{11}[z/y] : \phi_{22}$

$$\frac{\Gamma, z:\phi_{21} \vdash ((e_{11}[z/y]) [e_2/x]) : \phi_{22}}{\Gamma \vdash \lambda z. ((e_{11}[z/y]) [e_2/x]) : \phi_{21} \to \phi_{22}}$$

Resuming the main theorem

PROOF: By induction on structure of reason why $e \longrightarrow e'$

Assume $e \overset{\pi}{\triangleright_{1\beta}} e'$

Structure of π

$\cup$

Base case $\quad ht(\pi) = 0$

$\therefore \quad (\beta) \dfrac{}{e \ \rhd_{1\beta} \ e'}$

$\therefore \quad e \equiv ((\lambda x. e_1) \ e_2)$

$e' \equiv e_1[e_2/x]$

$$
\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma, x : \phi_1 \vdash e_1 : \phi_2}}{\Gamma \vdash \lambda x. e_1 : \phi_1 \to \phi}\,(\to I) \qquad \dfrac{\pi_2}{\Gamma \vdash e_2 : \phi_1}}{\Gamma \vdash ((\lambda x. e_1) \ e_2) : \phi}\,(\to E)
$$

By SUBSTITUTION LEMMA

$\Gamma \vdash e_1[e_2/x] : \phi_2$

Induction Hypothesis

Assume that if

$$\begin{array}{c} \hat{\pi} \\ \hat{e} \ \triangleright_{1\beta} \ \hat{e}' \end{array} \qquad ht(\hat{\pi}) \leq k$$

and $\hat{\Gamma} \vdash \hat{e} : \hat{p}$

then $\hat{\Gamma} \vdash \hat{e}' : \hat{p}$

Induction Step.

e is $(e_1 \ e_2)$

$$\begin{array}{c} \pi_1 \\ \end{array}$$

$$(\text{arg}) \ \dfrac{e_2 \ \triangleright_{1\beta} \ e_2'}{(e_1 \ e_2) \ \triangleright_{1\beta} \ \underbrace{(e_1 \ e_2')}_{e'}}$$

$$\dfrac{\Gamma \vdash (e_1 \ e_2) : p}{\begin{array}{c} \Gamma \vdash e_1 : p_1 \to p \\ \\ \Gamma \vdash e_2 : p_1 \end{array}}$$

I.H on π_1

$$\hat{\Gamma} = \Gamma$$
$$\hat{e} = e_2$$
$$\hat{e}' = e_2'$$
$$\hat{p} = p_1$$

$$\Gamma \vdash e_2' : p_1$$

$$(\rightarrow E)\ \frac{\Gamma \vdash e_1 : p_1 \rightarrow p \qquad \Gamma \vdash e_2' : p_1}{\Gamma \vdash (e_1\ e_2') : p}$$

- e is $(e_1\ e_2)$

$$\text{(arg)}\ \frac{\begin{array}{c}\pi_1\\ e_1 \triangleright_{1\beta} e_1'\end{array}}{(e_1\ e_2) \triangleright_{1\beta} \underbrace{(e_1'\ e_2)}_{e'}}$$

$$\Gamma \vdash (e_1\ e_2) : p$$

$$\left\{\begin{aligned} &\Gamma \vdash e_1 : p_1 \to p \\ &\Gamma \vdash e_2 : p_1 \end{aligned}\right.$$

I.H on π_1

$$\begin{aligned} \hat{\Gamma} &= \Gamma \\ \hat{e} &= e_1 \\ \hat{e}' &= \hat{e}_1' \\ \hat{p} &= p_1 \to p \end{aligned}$$

$$\Gamma \vdash e_1' : p_1 \to p$$

$$(\to E)\ \frac{\Gamma \vdash e_1' : p_1 \to p \qquad \Gamma \vdash e_2 : p_1}{\Gamma \vdash (e_1'\ e_2) : p}$$

- e is $\lambda x. e_1$

$$\pi_1$$

$$(\xi)\ \frac{e_1 \triangleright_{1\beta} e_1'}{\lambda x.e_1 \triangleright_{1\beta} \lambda x.e_1'}$$

$$\Gamma \vdash \lambda x.e_1 : \phi_1 \to \phi_2$$

$$\Gamma,\ x : \phi_1 \vdash e_1 : \phi_2$$

By I.H on π_1 with

$$\hat{\Gamma} = \Gamma,\ x : \phi_1$$

$$\hat{e} = e_1$$

$$\hat{e}' = e_1'$$

$$\hat{\phi} = \phi_2$$

we get

$$\Gamma,\ x : \phi_1 \vdash e_1' : \phi_2$$