3.2 Programación de Pruebas

Isomorfismo de Curry-Howard

Construcción de Pruebas en Coq

- ¿Qué estamos haciendo cuando probamos un teorema en Coq?
  - **Construimos** un objeto, que es la **prueba** del teorema
- ¿En qué lenguaje está escrita esa prueba?
  - En **Cálculo Lambda!!!!**
  - Cada **enunciado lógico** se corresponde con un **tipo**
  - Cada **prueba** es un **objeto** del tipo correspondiente.

Isomorfismo de Curry-Howard

- Identificación de proposiciones con tipos

  $P : Prop$

  pensamos a P como el tipo cuyos objetos son las pruebas de P

- Identificación de pruebas con objetos

  $a : P$

  significa que a es una prueba de P

Cálculo Proposicional Minimal

Deducción Natural

- Proposiciones atómicas y de la forma $\alpha \to \beta$
- Juicios de la forma: $\Gamma \mid\!- \alpha$

  "α se deduce a partir del conjunto de hipótesis Γ"

  $\Gamma = [\alpha_1, \ldots, \alpha_n]$

Reglas:

$$\frac{}{\Gamma, \alpha \mid\!- \alpha}\ \text{ass}$$

$$\frac{\Gamma, \alpha \mid\!- \beta}{\Gamma \mid\!- \alpha \to \beta}\ {\to}\text{I}$$

$$\frac{\Gamma \mid\!- \alpha \to \beta \quad \Gamma \mid\!- \alpha}{\Gamma \mid\!- \beta}\ {\to}\text{E}$$

Deducción Natural en Coq

$[\alpha_1,\ldots,\alpha_n]$ |- α lo vemos escrito:

$$\frac{\begin{array}{l} H_1 : \alpha_1 \\ \vdots \\ H_n : \alpha_n \end{array}}{\alpha}$$

$$\frac{}{\Gamma, \alpha \;|\!\text{-}\; \alpha}\,\text{ass}$$

corresponde a **assumption**

$$\frac{\Gamma, \alpha \;|\!\text{-}\; \beta}{\Gamma \;|\!\text{-}\; \alpha \to \beta}\,\to\! I$$

corresponde a **intro**

$$\frac{\Gamma \;|\!\text{-}\; \alpha \to \beta \quad \Gamma \;|\!\text{-}\; \alpha}{\Gamma \;|\!\text{-}\; \beta}\,\to\! E$$

corresponde a **apply** o **cut**
(dependiendo de si $\alpha \to \beta$ está o no en Γ)

Cálculo λ simplemente tipado

sistema de tipos

Juicios de la forma: $\Gamma \mid\!- \; e: \alpha$

"la expresión e tiene tipo α bajo el contexto Γ"

$$\Gamma=[x_1:\alpha_1,\ldots, x_n:\alpha_n]$$

Reglas:

$$\frac{}{\Gamma, x:\alpha \mid\!- \; x: \alpha}\;\textbf{ctx}$$

$$\frac{\Gamma, x:\alpha \mid\!- \; e: \beta}{\Gamma \mid\!- \; \lambda x.e: \alpha \to \beta}\;\textbf{abs}$$

$$\frac{\Gamma \mid\!- \; e_1:\alpha \to \beta \quad \Gamma \mid\!- \; e_2: \alpha}{\Gamma \mid\!- \; (e_1\; e_2): \beta}\;\textbf{app}$$

comparemos...

Deducción Natural	Cálculo λ
$$\frac{}{\Gamma, \alpha \vdash \alpha}\ \textbf{ass}$$	$$\frac{}{\Gamma, x{:}\alpha \vdash x{:}\ \alpha}\ \textbf{ctx}$$
$$\frac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \rightarrow \beta}\ \rightarrow\textbf{I}$$	$$\frac{\Gamma, x{:}\alpha \vdash e{:}\ \beta}{\Gamma \vdash \lambda x.e{:}\ \alpha \rightarrow \beta}\ \textbf{abs}$$
$$\frac{\Gamma \vdash \alpha \rightarrow \beta \quad \Gamma \vdash \alpha}{\Gamma \vdash \beta}\ \rightarrow\textbf{E}$$	$$\frac{\Gamma \vdash e_1{:}\alpha \rightarrow \beta \quad \Gamma \vdash e_2{:}\ \alpha}{\Gamma \vdash (e_1\ e_2){:}\ \beta}\ \textbf{app}$$

Más similitudes: Reducciones

Cálculo λ

$$\dfrac{\dfrac{\Gamma, x{:}\alpha \mid\!- e{:}\ \beta}{\Gamma \mid\!- \lambda x.e{:}\ \alpha \to \beta}\,\text{abs} \qquad \Gamma \mid\!- a{:}\ \alpha}{\Gamma \mid\!- (\lambda x.e\ a){:}\ \beta}\,\text{app} \quad \overset{\beta}{\Longrightarrow} \quad \dfrac{\Gamma, x{:}\alpha \mid\!- e{:}\ \beta \qquad \Gamma \mid\!- a{:}\ \alpha}{\Gamma \mid\!- (\lambda x.e\ a) = e[x{:=}a] : \beta}$$

Deducción Natural

$$\dfrac{\dfrac{\Gamma, \alpha \mid\!- \beta}{\Gamma \mid\!- \alpha \to \beta}\,\to\!\text{I} \qquad \Gamma \mid\!- \alpha}{\Gamma \mid\!- \beta}\,\to\!\text{E} \quad \overset{\text{cut}}{\Longrightarrow} \quad \dfrac{\Gamma, \alpha \mid\!- \beta \qquad \Gamma \mid\!- \alpha}{\Gamma \mid\!- \beta}\,\text{cut}$$

Isomorfismo de Curry-Howard

Un poco de historia:

- En 1958 H. B. Curry observó que los axiomas del cálculo proposicional $(\alpha\to\beta\to\gamma)\to(\alpha\to\beta)\to\alpha\to\gamma$, $\alpha\to\beta\to\alpha$ y $\alpha\to\alpha$ se correspondían con los tipos de los combinadores S,K e I
- En 1965 W. Tait descubrió una correspondencia entre la eliminación de lemas en pruebas (cut-elimination) y la β-reducción en el cálculo λ.
- En 1969 W. A. Howard desarrolla una noción de construcción adecuada para representar las pruebas de la lógica intuicionista.

Cálculo de Predicados Minimal

Deducción Natural

- Proposiciones atómicas y de la forma $\alpha \rightarrow \beta$ o $\forall x \in A.\beta$
- Juicios de la forma: $\Gamma \mid\!- \alpha$

  "α se deduce a partir del conjunto de hipótesis y objetos Γ"

  $\Gamma = [x_1 \in A_1, \ldots, x_m \in A_m] \cup [\alpha_1, \ldots, \alpha_n]$

Reglas: ass, →I, →E **más:**

$$\frac{\Gamma, x \in A \mid\!- \beta}{\Gamma \mid\!- \forall x \in A.\ \beta}\ \forall I$$

$$\frac{\Gamma \mid\!- \forall x \in A.\ \beta \qquad \Gamma \mid\!- a \in A}{\Gamma \mid\!- \beta(a)}\ \forall E$$

En Coq...

$[x_1: A_1,.., x_m:A_m] \cup [\alpha_1,.., \alpha_n] \mid\!- \alpha$ **lo vemos escrito**

$$\begin{array}{c} x_1:A_1 \\ \vdots \\ x_m:A_m \\ H_1: \alpha_1 \\ \vdots \\ H_n:\alpha_n \\ \hline \alpha \end{array}$$

$$\frac{\Gamma, x \in A \mid\!- \beta}{\Gamma \mid\!- \forall x \in A.\ \beta}\ \forall\mathrm{I}$$

corresponde a **intro x**

$$\frac{\Gamma \mid\!- \forall x \in A.\ \beta \qquad \Gamma \mid\!- a \in A}{\Gamma \mid\!- \beta[a/x]}\ \forall\mathrm{E}$$

corresponde a **apply**

Cálculo λ con tipos dependientes

sistema de tipos

- Juicios de la forma: Γ |- e: α

"la expresión e tiene tipo α bajo el contexto Γ"

$\Gamma=[x_1{:}\alpha_1,\ldots, x_n{:}\alpha_n]$

Reglas: **ctx más:**

$$\frac{\Gamma, x{:}\alpha \mid\!- e{:}\ \beta}{\Gamma \mid\!- \lambda x^{\alpha}.e{:}\ (x{:}\alpha)\beta}\ \textbf{abs}$$

$$\frac{\Gamma \mid\!- e{:}\ (x{:}\alpha)\beta \qquad \Gamma \mid\!- a{:}\ \alpha}{\Gamma \mid\!- (e\ a){:}\ \beta[x:=a]}\ \textbf{app}$$

comparemos otra vez...

Deducción Natural	Cálculo λ
$$\frac{\Gamma, x \in A \mid\!- \beta}{\Gamma \mid\!- \forall x \in \alpha.\ \beta}\ \forall I$$	$$\frac{\Gamma, x{:}\alpha \mid\!- e{:}\ \beta}{\Gamma \mid\!- \lambda x^{\alpha}.e{:}\ (x{:}\alpha)\beta}\ \text{abs}$$
$$\frac{\Gamma \mid\!- \forall x \in A.\ \beta \quad \Gamma \mid\!- a \in A}{\Gamma \mid\!- \beta[a/x]}\ \forall E$$	$$\frac{\Gamma \mid\!- e{:}\ (x{:}\alpha)\beta \quad \Gamma \mid\!- a{:}\ \alpha}{\Gamma \mid\!- (e\ a){:}\ \beta[a/x]}\ \text{app}$$

Observaciones sobre los productos

La regla del producto nos sirve para representar tres tipos de funciones

$$\frac{\Gamma \vdash \alpha : \textbf{Set} \qquad \Gamma, x:\alpha \vdash \beta : \textbf{Set}}{\Gamma \vdash (x:\alpha)\beta : \textbf{Set}}\ \textbf{prod}$$

fun x:nat =>x : forall x:nat, nat (= nat→nat)

fun n:nat => diag n:forall n:nat, Mat n n

$$\frac{\Gamma \vdash \alpha : \textbf{Set} \qquad \Gamma, x:\alpha \vdash \beta : \textbf{Prop}}{\Gamma \vdash (x:\alpha)\beta : \textbf{Prop}}\ \textbf{prod}$$

fun x:nat => leS x: forall x:nat, Le x (S x)

Ax : forall x:nat, x=0 → ~∃y.x=Sy

$$\frac{\Gamma \vdash \alpha : \textbf{Prop} \qquad \Gamma, x:\alpha \vdash \beta : \textbf{Prop}}{\Gamma \vdash (x:\alpha)\beta : \textbf{Prop}}\ \textbf{prod}$$

fun (H:z=0) => Ax z H:

forall H:z=0, ~(∃y.z=Sy)

z=0→ ~(∃y.z=Sy)

Isomorfismo en Coq

- Cuando constuimos una **prueba** de un enunciado en Coq, estamos construyendo un **término** λ del tipo correspondiente al enunciado.

- La situación general es de la forma:

$$\frac{\Gamma_1}{?_1 : \alpha_1} \quad \cdots\cdots\cdots \quad \frac{\Gamma_n}{?_n : \alpha_n}$$

- **Tácticas**: constructoras de términos

Construcción de pruebas en Coq

$$\frac{\begin{array}{l}\Gamma \\ H: \alpha\end{array}}{?: \alpha}$$

assumption:

corresponde a la prueba **H:α**

$$\frac{\Gamma}{?: \alpha\to\beta}$$

intro H:

corresponde a la prueba **[H:α] ?$_1$**

donde ?$_1$ será la prueba de β corresp. a:

$$\frac{\begin{array}{l}\Gamma \\ H: \alpha\end{array}}{?_1: \beta}$$

$$\frac{\begin{array}{l}\Gamma \\ H: \alpha\to \beta\end{array}}{?: \beta}$$

apply H:

corresponde a la prueba **(H ?$_1$)**

donde ?$_1$ será la prueba de α corresp. a:

$$\frac{\begin{array}{l}\Gamma \\ H: \alpha\to\beta\end{array}}{?_1: \alpha}$$

Construcción de pruebas en Coq (cont.)

$$\frac{\Gamma}{?: \alpha}$$

cut β:
corresponde a la prueba **$(?_1\ ?_2)$**

donde $?_1$ y $?_2$ serán las pruebas de $\beta\to\alpha$ y β correspondientes a:

$$\frac{\Gamma}{?_1: \beta\to\alpha} \quad \text{y} \quad \frac{\Gamma}{?_2: \beta}$$

$$\frac{\Gamma}{?: (x:\alpha)\beta}$$

intro x:
corresponde a la prueba **$[x:\alpha]\ ?_1$**
donde $?_1$ será la prueba de β corresp. a:

$$\frac{\Gamma \quad x: \alpha}{?_1: \beta}$$

ver que es *exactamente* la misma explicación que para el caso $\alpha\to\beta$

Construcción de pruebas en Coq (cont.)

$$\frac{\Gamma \quad H : (x_1:\alpha_1)..(x_n:\alpha_n)\beta}{?: \gamma}$$

apply H:

corresponde a la prueba **$(H\ x_1\theta\ \ x_2\theta)$**

Donde θ es la sustitución que unifica a β con γ.

Además, $x_1\theta:\alpha_1\theta... x_n\theta:\alpha_n\theta$ deberán ser consecuencias de Γ

Coq chequea: $\dfrac{\Gamma}{x_1\theta:\alpha_1\theta}$ ... $\dfrac{\Gamma}{x_n\theta:\alpha_n\theta}$

Ejemplo:

$$\frac{\Gamma \quad H:(x:nat)(y:(P\ x))(Q\ x\ y)}{?: (Q\ 0\ a)}$$

apply H

(H 0 a): (Q 0 a)

$\dfrac{\Gamma}{\mathbf{0:nat}}$ $\dfrac{\Gamma}{\mathbf{a:(P\ 0)}}$

Programando Pruebas

- Los resultados ya probados y las hipótesis pueden pensarse como objetos de ciertos tipos (en general, son funciones)
  - H1: $A \rightarrow B$
  - Lema2: A
- Estas funciones pueden aplicarse a argumentos, que a su vez pueden ser pruebas de resultados o a otras hipótesis. De esta forma, podemos utilizar las pruebas como objetos de un lenguaje funcional
  - (H1 Lema2): B

Programando Pruebas
Ejemplos

$$\frac{H1: A \rightarrow B \quad H2: A}{B}$$

exact (H1 H2)

Probado!!

$$\frac{H1: A \rightarrow B \rightarrow C \quad H2: A \quad H3: B}{C}$$

exact ((H1 H2) H3)

Probado!!

$$\frac{H1: A \quad H2: A \rightarrow B \quad H3: B \rightarrow C}{C}$$

exact (H3 (H2 H1))

Probado!!

Programando Pruebas
Ejemplos (cont.)

H1: (x:A)(B x)
a: A

(B a)

exact (H1 a)

Probado!!

H1: $A \rightarrow$(x:B)(C x)
H2: A
z: B

(C z)

exact ((H1 H2) z)

Probado!!

H1: (x:A) $B \rightarrow C$
H2: B
z: A

C

exact ((H1 z) H2)

Probado!!

Tácticas para ver pruebas

- Show Proof: muestra el término λ correspondiente a la prueba que se está armando

- Show Tree: muestra la prueba como en el sistema de Deducción Natural