

Introducción a la Verificación Formal

Clase 2 – 04/09/2025

¿Cuál es el tipo de printf?

`printf "%s\n" : string -> unit`

`printf "%d\n" : int -> unit`

`printf "%d + %d = %f\n" : int -> int -> float -> unit`

Entonces `printf : string -> ...?`

Tipado simple

$$\begin{array}{c} t ::= c \mid t \rightarrow t \end{array} \qquad \begin{array}{c} \text{T-ABS-ST} \\ \hline \Gamma, x : t \vdash e : s \\ \hline \Gamma \vdash (\lambda(x : t).e) : t \rightarrow s \end{array} \qquad \begin{array}{c} \text{T-APP-ST} \\ \hline \Gamma \vdash f : t \rightarrow s \quad \Gamma \vdash e : t \\ \hline \Gamma \vdash fe : s \end{array}$$

- Tipado en STLC
- Los tipos son *cerrados*: no dependen del contexto Γ
- Si $f : t \rightarrow s$, entonces f e debe tener tipo s

Sistema F: polimorfismo paramétrico

$$\begin{array}{c} t ::= X \mid c \mid t \rightarrow t \mid \forall X.t \\[10pt] \text{T-ABS} \quad \frac{\Gamma, x:t \vdash e : s}{\Gamma \vdash (\lambda(x:t).e) : t \rightarrow s} \quad \text{T-APP} \quad \frac{\Gamma \vdash f : t \rightarrow s \quad \Gamma \vdash e : t}{\Gamma \vdash fe : s} \\[10pt] \text{T-TABS} \quad \frac{\Gamma, X \vdash e : s}{\Gamma \vdash (\Lambda X.e) : \forall X.s} \quad \text{T-TAPP} \quad \frac{\Gamma \vdash e : \forall X.t \quad \Gamma \vdash A : \text{Type}}{\Gamma \vdash eA : t[A/X]} \end{array}$$

- Algo de “dependencia”: los términos pueden depender de tipos
 - a.k.a. polimorfismo, y en este caso es *paramétrico*
- Los tipos ahora *tienen variables ligadas*
 - Deben estar bien formados
- Dos “tipos” de aplicación y abstracción
 - Sintaxis distinguida

[Wadler 1989 – Theorems for Free!](#)

Tipado dependiente

Sintaxis uniforme para expresiones y tipos

$t ::= x \mid c \mid (x : t) \rightarrow t$

$(t \rightarrow t')$ es azúcar para $(x : t) \rightarrow t'$ con $x \notin FV(t')$

x puede aparecer en **s**

T-ABS-DEP

$\Gamma, x : t \vdash e : s$

$\Gamma \vdash (\lambda(x : t).e) : (x : t) \rightarrow s$

T-APP-DEP

$\Gamma \vdash f : (x : t) \rightarrow s \quad \Gamma \vdash e : t$

$\Gamma \vdash fe : s[e/x]$

Sustitución por el argumento real

- En tipado dependiente, los tipos pueden depender de valores
 - Ya vimos algunos ejemplos encubiertos:

```
val incr''': (x:nat) -> y:int{y = x+1}
let incr''' (x:nat) : y:int{y = x+1} = x+1
```

Polimorfismo en F*

- Como en otros lenguajes con tipos dependientes, las funciones polimórficas son simplemente funciones que toman un tipo como argumento

```
val id : (a:Type) -> a -> a
let id a x = x
```

```
let _ = assert (id int 42 == 42)
let _ = assert (id string "hola" == "hola")
```

```
let _ = assert (id Type string == string)
let _ = assert (id (id Type string) "hola" == "hola")
```

Al margen: terminología

$$\begin{aligned} & \Pi_{x:A} B \\ & (x : A) \rightarrow B \\ & \Pi(x : A) B(x) \end{aligned}$$

“Producto dependiente” (función dependiente, flecha dependiente, tipo Π)

$$\begin{aligned} \mathbf{Bool} \rightarrow A & \cong A \times A \\ \mathbf{Nat} \rightarrow A & \cong A \times A \times \cdots \times A \\ \Pi_{x:\mathbf{Nat}} A(x) & \cong A(0) \times A(1) \times \cdots \times A(n) \times \cdots \end{aligned}$$

“Suma dependiente” (par dependiente, tupla dependiente, tipo Σ)

$$\begin{aligned} \mathbf{Bool} \times A & \cong A + A \\ \mathbf{Nat} \times A & \cong A + A + \cdots + A \\ \Sigma_{x:\mathbf{Nat}} A(x) & \cong A(0) + A(1) + \cdots + A(n) + \cdots \end{aligned}$$

(demo)



**AND NOW FOR SOMETHING
COMPLETELY DIFFERENT**

Lógica formal

- Presentación “a la Gentzen”
 - Hipótesis “a descargar”
- Reglas de introducción y eliminación
- Una prueba es una derivación correcta de una proposición
- “Formal”: las pruebas pueden chequearse mecánicamente

$$\begin{array}{c}
 \Rightarrow\text{-INTRO} \\
 \frac{A}{\vdots} \\
 \frac{B}{A \Rightarrow B}
 \end{array}
 \qquad
 \begin{array}{c}
 \Rightarrow\text{-ELIM} \\
 \frac{A \Rightarrow B \quad A}{B}
 \end{array}$$

$$\begin{array}{c}
 \wedge\text{-INTRO} \\
 \frac{A \quad B}{A \wedge B}
 \end{array}
 \qquad
 \begin{array}{c}
 \wedge\text{-ELIM-L} \\
 \frac{A \wedge B}{A}
 \end{array}
 \qquad
 \begin{array}{c}
 \wedge\text{-ELIM-R} \\
 \frac{A \wedge B}{B}
 \end{array}$$

Lógica formal - Secuentes

- Secuentes: explicitan las hipótesis
- Prop de Eliminación de cortes:
 - Corte: introducción seguida de eliminación para un mismo conectivo
 - Toda prueba puede simplificarse a una prueba sin cortes
- Todo corte puede reducirse, eso es trivial
- ¿El proceso termina?
 - Sí, Gentzen demostró que si se toman los cortes más externos, siempre termina

$$\begin{array}{c}
 \text{Ax} \\
 \hline
 \Gamma, A, \Gamma' \vdash A
 \end{array}
 \quad
 \begin{array}{c}
 \Rightarrow\text{-INTRO} \\
 \Gamma, A \vdash B \\
 \hline
 \Gamma \vdash A \Rightarrow B
 \end{array}$$

$$\begin{array}{c}
 \Rightarrow\text{-ELIM} \\
 \Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A \\
 \hline
 \Gamma \vdash B
 \end{array}$$

$$\begin{array}{c}
 \wedge\text{-INTRO} \\
 \Gamma \vdash A \quad \Gamma \vdash B \\
 \hline
 \Gamma \vdash A \wedge B
 \end{array}
 \quad
 \begin{array}{c}
 \wedge\text{-ELIM-L} \\
 \Gamma \vdash A \wedge B \\
 \hline
 \Gamma \vdash A
 \end{array}
 \quad
 \begin{array}{c}
 \wedge\text{-ELIM-R} \\
 \Gamma \vdash A \wedge B \\
 \hline
 \Gamma \vdash B
 \end{array}$$

$$\begin{array}{c}
 [\Gamma, A \vdash A] \\
 \vdots \\
 \hline
 \Gamma, A \vdash^1 B
 \end{array}
 \quad
 \Gamma \vdash^2 A$$

$$\frac{\Gamma, A \vdash^1 B}{\Gamma \vdash A \Rightarrow B} \quad \Gamma \vdash^2 A \quad \rightsquigarrow \quad \frac{[\Gamma \vdash^2 A]}{\Gamma \vdash^1 B}$$

$$\frac{\text{Ax}}{\overline{\Gamma, A, \Gamma' \vdash A}} \quad \frac{\begin{array}{c} \Rightarrow\text{-INTRO} \\ \Gamma, A \vdash B \end{array}}{\overline{\Gamma \vdash A \Rightarrow B}}$$

$$\frac{\begin{array}{c} \Rightarrow\text{-ELIM} \\ \Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A \end{array}}{\Gamma \vdash B}$$

$$\frac{\begin{array}{c} \wedge\text{-INTRO} \\ \Gamma \vdash A \quad \Gamma \vdash B \end{array}}{\Gamma \vdash A \wedge B} \quad \frac{\begin{array}{c} \wedge\text{-ELIM-L} \\ \Gamma \vdash A \wedge B \end{array}}{\Gamma \vdash A} \quad \frac{\begin{array}{c} \wedge\text{-ELIM-R} \\ \Gamma \vdash A \wedge B \end{array}}{\Gamma \vdash B}$$

$$\frac{\begin{array}{c} \vee\text{-INTRO-L} \\ \Gamma \vdash A \end{array}}{\Gamma \vdash A \vee B} \quad \frac{\begin{array}{c} \vee\text{-INTRO-R} \\ \Gamma \vdash B \end{array}}{\Gamma \vdash A \vee B} \quad \frac{\begin{array}{c} \vee\text{-ELIM} \\ \Gamma \vdash A \vee B \end{array} \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

$$\frac{}{\Gamma \vdash \top} \quad \frac{\begin{array}{c} \perp\text{-ELIM} \\ \Gamma \vdash \perp \end{array}}{\overline{\Gamma \vdash A}}$$

(PD: Consistencia)

- ¿Cómo se demuestra que la lógica es *consistente*?
 - i.e. tiene al menos una prop P tal que **no** ocurre $\vdash P$
- 1. Se demuestra que la lógica tiene la propiedad de eliminación de cortes
 - Esta es la parte difícil, en general por mostrar que el procedimiento termina
- 2. No hay pruebas sin cortes de Falso
 - Una prueba sin cortes tiene que terminar con una introducción
 - Entonces esto es trivial, porque falso no tiene introducciones.

$\not\vdash \bot$

Correspondencia Curry-Howard

- a.k.a. “propositions as types”

$$\begin{array}{c}
 \text{Ax} \\
 \hline
 \Gamma, A, \Gamma' \vdash A \\
 \\
 \text{\(\Rightarrow\)-INTRO} \\
 \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \\
 \\
 \text{\(\Rightarrow\)-ELIM} \\
 \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \\
 \\
 \text{\(\wedge\)-INTRO} \\
 \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \\
 \\
 \text{\(\wedge\)-ELIM-L} \\
 \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \\
 \\
 \text{\(\wedge\)-ELIM-R} \\
 \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{Ax} \\
 \hline
 \Gamma, x : A, \Gamma' \vdash x : A \\
 \\
 \text{\(\rightarrow\)-INTRO} \\
 \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash (\lambda x. e) : A \rightarrow B} \\
 \\
 \text{\(\rightarrow\)-ELIM} \\
 \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash e : A}{\Gamma \vdash fe : B} \\
 \\
 \text{\(\times\)-INTRO} \\
 \frac{\Gamma \vdash x : A \quad \Gamma \vdash y : B}{\Gamma \vdash (x, y) : A \times B} \\
 \\
 \text{\(\times\)-ELIM-L} \\
 \frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \mathbf{fst} \, p : A} \\
 \\
 \text{\(\times\)-ELIM-R} \\
 \frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \mathbf{snd} \, p : B}
 \end{array}$$

Relacionado: interpretación Brouwer-Heyting-Kolmogorov de lógica intuicionista

Curry-Howard

- Tomamos una proposición (tipo) P como cierta si existe un habitante de P
- Las pruebas **son programas**, reducen y tienen contenido computacional
 - Una prueba de $A \vee B$ decide cuál caso vale
 - Cuando tengamos existenciales: son programas que computan un testigo

$$\frac{\frac{\overline{p : A \times B \vdash p : A \times B}}{p : A \times B \vdash \mathbf{snd} \, p : B} \quad \frac{\overline{p : A \times B \vdash p : A \times B}}{p : A \times B \vdash \mathbf{fst} \, p : A}}{p : A \times B \vdash (\mathbf{snd} \, p, \mathbf{fst} \, p) : B \times A}$$

$$\vdash \lambda p. (\mathbf{snd} \, p, \mathbf{fst} \, p) : A \times B \rightarrow B \times A$$

$$\frac{\frac{\overline{A \times B \vdash A \times B}}{A \times B \vdash B} \quad \frac{\overline{A \times B \vdash A \times B}}{A \times B \vdash A}}{A \times B \vdash B \times A}$$

$$\vdash A \times B \rightarrow B \times A$$

$$\frac{\frac{\overline{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash B} \quad \frac{\overline{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash A}}{A \wedge B \vdash B \wedge A}$$

$$\vdash A \wedge B \implies B \wedge A$$

Curry-Howard

- El término codifica la prueba
 - Si el tipado es dirigido por sintaxis, es inmediato construir un tipado para un término dado
 - Chequear una prueba es fácil = chequear el tipo de un término es fácil
 - Encontrar una prueba es difícil = encontrar un habitante de un tipo es difícil
- Es un principio general, hay una correspondencia para cada lógica
 - STLC \approx Lógica proposicional
 - System F \approx Lógica de segundo orden
 - Tipado dependiente (tipos dependen de valores) \approx Lógica de predicados
 - Tipado dependiente + polimorfismo \approx lógica de alto orden
 - Y más...

$$\begin{array}{c}
 \frac{}{p : A \times B \vdash p : A \times B} \quad \frac{}{p : A \times B \vdash p : A \times B} \\
 \frac{}{p : A \times B \vdash \mathbf{snd} \, p : B} \quad \frac{}{p : A \times B \vdash \mathbf{fst} \, p : A} \\
 \hline
 p : A \times B \vdash (\mathbf{snd} \, p, \mathbf{fst} \, p) : B \times A \\
 \hline
 \vdash \lambda p. (\mathbf{snd} \, p, \mathbf{fst} \, p) : A \times B \rightarrow B \times A
 \end{array}$$

$$\begin{array}{c}
 \frac{}{A \times B \vdash A \times B} \quad \frac{}{A \times B \vdash A \times B} \\
 \frac{}{A \times B \vdash B} \quad \frac{}{A \times B \vdash A} \\
 \hline
 A \times B \vdash B \times A \\
 \hline
 \vdash A \times B \rightarrow B \times A
 \end{array}$$

$$\begin{array}{c}
 \frac{}{A \wedge B \vdash A \wedge B} \quad \frac{}{A \wedge B \vdash A \wedge B} \\
 \frac{}{A \wedge B \vdash B} \quad \frac{}{A \wedge B \vdash A} \\
 \hline
 A \wedge B \vdash B \wedge A \\
 \hline
 \vdash A \wedge B \implies B \wedge A
 \end{array}$$

Correspondencia Curry-Howard (2)

- La correspondencia se extiende a la simplificación de pruebas
 - Tipo \approx proposición o predicado
 - Prueba \approx programa/algoritmo
 - “cut-elimination” de la implicancia \approx beta-reducción
 - “cut-elimination” de pares \approx reducción de proyecciones ($\text{fst } (x,y) \rightsquigarrow x$)
 - Prueba en forma normal \approx término en forma normal
 - La lógica tiene eliminación de cortes \approx el lenguaje es débilmente normalizante (en general... hay casos borde)

$$\frac{
 \begin{array}{c}
 [\Gamma, A \vdash A] \\
 \vdots \\
 \hline
 \Gamma, A \vdash^1 B
 \end{array}
 \quad
 \Gamma \vdash^2 A
 }{
 \Gamma \vdash A \implies B \quad \Gamma \vdash^2 A
 }
 \rightsquigarrow
 \frac{
 \begin{array}{c}
 [\Gamma \vdash^2 A] \\
 \vdots \\
 \hline
 \Gamma \vdash^1 B
 \end{array}
 }{
 \Gamma \vdash B
 }$$

["Propositions as Types" by Philip Wadler - YouTube](#)

Intuicionismo / constructivismo

- Para capturar la lógica clásica, nos falta una regla importante

$$\overline{\Gamma \vdash A \vee \neg A}$$

- En lógica intuicionista, se rechaza este axioma
 - Implicaría un procedimiento de decisión para cada prop A (en lógicas suf. expresivas contradice a Gödel y Turing a la vez)
 - No tiene interpretación computacional
 - Rompe propiedades como $(\vdash A \vee B) \implies (\vdash A) \vee (\vdash B)$
- Hay más axiomas clásicos, muchos son equivalentes
 - Notoriamente la eliminación de doble negación: $\neg\neg A \implies A$
 - Otros: ver práctica

Prueba no constructivas

Veamos un ejemplo. *Existen irracionales a y b , tales que a^b es racional.*

Demostración (por el absurdo): consideremos $\sqrt{2}^{\sqrt{2}}$. Por el principio del tercero excluido, este número será racional o irracional. En el primer caso, basta tomar $a = b = \sqrt{2}$; en el segundo caso, basta tomar $a = \sqrt{2}^{\sqrt{2}}$ y $b = \sqrt{2}$, pues $\sqrt{2}^{\sqrt{2}^{\sqrt{2}}} = 2$.

La demostración es impecable, pero no sabemos si $\sqrt{2}^{\sqrt{2}}$ es racional o no.

De la charla “Continuaciones: La Venganza del Goto” de Guido Macchi, JCC 2007

(demo)

Pureza / totalidad

- En un lenguaje de programación, ¿por qué no podemos demostrar cualquier P mediante recursion?

```
let rec prueba () : p = prueba ()  
let prueba () : p = let x = 1/0 in ...  
let prueba () : p = raise “☺”
```

- En la presencia de **efectos** (no terminación, parcialidad, excepciones, etc), la correspondencia se rompe. En general, sólo vale para lenguajes puros
- En F* tenemos definiciones puras y *efectuosas*
 - El sistema de efectos se encarga de separar las efectuosas del fragmento puro
 - Implica chequear terminación, completitud de pattern match, etc...

Tareas

- Completar Clase02.fst
- Leer capítulos 3 y 6
- Otras fuentes:
 - ["Propositions as Types" by Philip Wadler - YouTube](#)
 - [Cayenne – a language with dependent types \(Augustsson 1998\)](#)
 - El tipo de printf: [printf* \(fstarlang.github.io\)](#)
 - [Continuations and Logic.pdf \(cmu.edu\)](#)
 - [Guido Macchi – Continuaciones la Venganza del Goto \(JCC 2007\)](#)