PPA

Un asistente de demostración para lógica de primer orden con extracción de testigos usando la traducción de Friedman

Manuel Panichelli

Deparatamento de Computación, FCEyN, UBA

Diciembre 2024

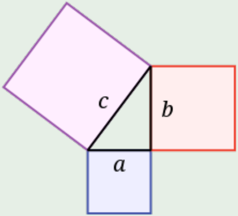
Introducción

- **Teorema**: Afirmación que puede ser *demostrada*.
- **Demostración** de un teorema: *Argumento lógico* que usa las *reglas de inferencia* de un *sistema deductivo* para establecer que el teorema es una *consecuencia lógica* de los *axiomas* y teoremas probados anteriormente.
- **Axiomas**: Afirmaciones que son siempre válidas (sin demostración).

Ejemplo de teorema

Ejemplo (Teorema de Pitágoras)

$$a^2 + b^2 = c^2.$$



- **Sistema**: Geometría euclidiana
- Un axioma: se puede dibujar una línea recta entre dos puntos

Asistentes de demostración

- Los **asistentes de demostración** son herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos usuales:
  - Formalización de teoremas matemáticos.
  - Verificación de programas.
- Ventajas:[1]
  - Facilitan la colaboración a gran escala (mediante la confianza en el asistente).
  - Habilitan generación automática de demostraciones con IA. Por ej. un *LLM* (como *ChatGPT*) suele devolver alucinaciones, que pueden ser filtradas automáticamente con un asistente.

[1] Terrence Tao - Machine Assisted Proof

Asistentes de demostración

Constructivos

No constructivos

Coq
(Type theory)

Agda
(Type theory)

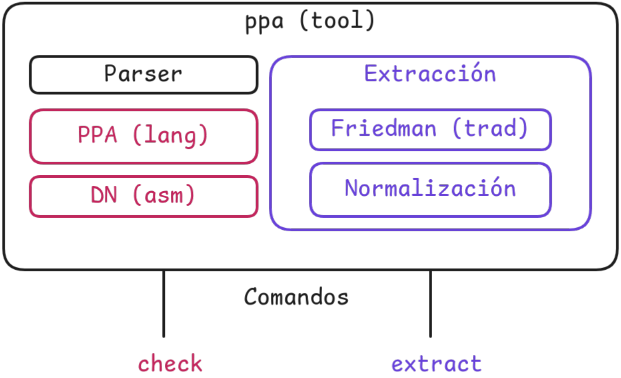
Isabelle
(HOL / ZFC)

Mizar (LPO)

Extracción de testigos

De una demo de $\exists x . p(x)$, encontrar t tq $p(t)$.

PPA



Diseñamos e implementamos en Haskell la herramienta ppa (*Pani's Proof Assistant*): un asistente de demostración para LPO **clásica**. Dos partes:

- El lenguaje **PPA** para escribir demostraciones.
- **Extracción de testigos** (Aporte principal).

Representación de demostraciones

¿Cómo representamos las demostraciones? Ejemplo:

- Tenemos dos premisas
  1. Los alumnos que faltan a los exámenes, los reprueban.
  2. Si se reprueba un final, se recursa la materia.
- A partir de ellas, podríamos demostrar que si un alumno falta a un final, entonces recursa la materia.

Representación de demostraciones

¿Cómo representamos las demostraciones? Ejemplo:

- Tenemos dos premisas
  1. Los alumnos que faltan a los exámenes, los reprueban.
  2. Si se reprueba un final, se recursa la materia.
- A partir de ellas, podríamos demostrar que si un alumno falta a un final, entonces recursa la materia.

Teorema

Si ((falta entonces reprueba) y (reprueba entonces recursa)) y falta, entonces recursa

Demostración.

- Asumo que falta. Quiero ver que recursa.
- Sabemos que si falta, entonces reprueba. Por lo tanto reprobó.
- Sabemos que si reprueba, entonces recursa. Por lo tanto recursó.

$\square$

Sistemas deductivos

- **Problema**: Poco precisa. No se puede representar rigurosamente.
- Necesitamos **sistemas deductivos**: sistemas lógicos formales usados para escribir demostraciones
- Usamos **deducción natural**
  - **Lenguaje formal**: lógica de primer orden.
  - **Reglas de inferencia**: Por ejemplo, *modus ponens* (si es cierto $A \to B$ y A, se puede concluir B) o *modus tollens* (si es cierto $A \to B$ y $\neg B$, se puede concluir $\neg A$)

Lógica de primer orden

Definición (Términos)

Los términos están dados por la gramática:

$$\begin{aligned} t ::= {} & x && \text{(variables)} \\ & \mid f(t_1, \ldots, t_n) && \text{(funciones)} \end{aligned}$$

Definición (Fórmulas)

Las fórmulas están dadas por la gramática:

$$\begin{aligned} A, B ::= {} & p(t_1, \ldots, t_n) && \text{(predicados)} \\ & \mid \bot \mid \top && \text{(falso o \textit{bottom} y verdadero o \textit{top})} \\ & \mid A \wedge B \mid A \vee B && \text{(conjunción y disyunción)} \\ & \mid A \rightarrow B \mid \neg A && \text{(implicación y negación)} \\ & \mid \forall x.A \mid \exists x.A && \text{(cuantificador universal y existencial)} \end{aligned}$$

Deducción natural

Ejemplo

Ejemplo (Demostración en DN)

Notamos:

- $X \equiv \mathrm{reprueba}(\mathit{juan}, \mathrm{final}(\mathit{logica}))$
- $R \equiv \mathrm{recursa}(\mathit{juan}, \mathit{logica})$
- $F \equiv \mathrm{falta}(\mathit{juan}, \mathrm{final}(\mathit{logica}))$

Queremos probar

$$\Big((F \to X) \wedge (X \to R)\Big) \to (F \to R)$$

Ejemplo

Ejemplo (Demostración en DN)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash (F \to X) \wedge (X \to R)}\,\text{Ax}}{\Gamma \vdash X \to R}\,\text{E}\wedge_1 \qquad \begin{array}{c}\Pi \\ \Gamma \vdash X\end{array}}{\Gamma = (F \to X) \wedge (X \to R),\ F \vdash R}\,\text{E}\!\to}{(F \to X) \wedge (X \to R) \vdash F \to R}\,\text{I}\!\to}{\vdash \Big((F \to X) \wedge (X \to R)\Big) \to (F \to R)}\,\text{I}\!\to$$

donde

$$\Pi = \dfrac{\dfrac{\dfrac{}{\Gamma \vdash (F \to X) \wedge (X \to R)}\,\text{Ax}}{\Gamma \vdash F \to X}\,\text{E}\wedge_2 \qquad \dfrac{}{\Gamma \vdash F}\,\text{Ax}}{\Gamma \vdash X}\,\text{E}\!\to$$

Deducción natural

Definición (Contexto de demostración)

Γ es un **contexto de demostración**, conjunto de fórmulas que se asumen válidas.
Notación: $\Gamma, \varphi = \Gamma \cup \{\varphi\}$

Deducción natural

Definición (Contexto de demostración)

Γ es un **contexto de demostración**, conjunto de fórmulas que se asumen válidas.
Notación: $\Gamma, \varphi = \Gamma \cup \{\varphi\}$

Definición (Relación de derivabilidad)

- $\vdash$ es la **relación de derivabilidad** definida a partir de las *reglas de inferencia*.
- Permite escribir juicios $\Gamma \vdash \varphi$.
- Decimos que φ es *derivable* a partir de Γ.
- Intuición: "*φ es una consecuencia de las suposiciones de Γ*"
- El juicio es cierto si en una cantidad finita de pasos podemos concluir φ a partir de las fórmulas de Γ, los axiomas y las reglas de inferencia.

Definición (Reglas de inferencia)

$$\frac{}{\Gamma, A \vdash A}\ \text{Ax} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \text{I}\wedge$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \text{I}\to \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \text{E}\wedge_1$$

$$\frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \text{E}\to \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \text{E}\wedge_2$$

Dos tipos para cada conectivo y cuantificador, dada una fórmula formada con un conectivo:

- **Introducción**: ¿Cómo la demuestro?
- **Eliminación**: ¿Cómo la uso para demostrar otra?

Reglas de inferencia

Otras reglas de inferencia

- E$\bot$, I$\top$
- I$\neg$, E$\neg$
- I$\vee_1$, I$\vee_2$, E$\vee$
- I$\forall$, E$\forall$
- I$\exists$, E$\exists$
- LEM

Reglas admisibles

- Mencionamos *modus tollens* pero no aparece en las reglas de inferencia.
- Queremos un sistema lógico **minimal**: no agregamos las reglas **admisibles**, derivables a partir de las existentes.
- Se implementan como funciones o *macros*.

Lema (Modus tollens)

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash (A \to B) \wedge \neg B}\,Ax}{\Gamma \vdash \neg B}\,E\wedge_2 \qquad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash (A \to B) \wedge \neg B}\,Ax}{\Gamma \vdash A \to B}\,E\wedge_1 \qquad \dfrac{}{\Gamma \vdash A}\,Ax}{\Gamma \vdash B}\,E\to}{\Gamma = (A \to B) \wedge \neg B, A \vdash \bot}\,E\neg}{(A \to B) \wedge \neg B \vdash \neg A}\,I\neg}{\vdash (A \to B \wedge \neg B) \to \neg A}\,I\to
$$

Sustitución sin capturas

Definición (Sustitución)

Notamos como $A\{x := t\}$ a la sustitución de todas las ocurrencias libres de la variable x por el término t en la fórmula A.

$$\frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A\{x := t\}} \; \mathsf{E}\forall$$

Queremos evitar la **captura de variables**, por ejemplo

$$(\forall y.p(x))\{x := y\} \stackrel{?}{=} \forall y.p(y)$$

Lo evitamos **automáticamente**: cuando se encuentra con una captura, se renombra la variable ligada de forma que no ocurra

$$(\forall y.p(x))\{x := y\} = \forall z.p(y)$$

Alfa equivalencia

- Si tenemos una hipótesis $\exists x.p(x)$ queremos poder usarla para demostrar $\exists y.p(y)$.
- No son iguales, pero son α-**equivalentes**: si renombramos variables ligadas de forma apropiada, son iguales.

PPA

Mathematical vernacular

Hay una forma natural de representar demostraciones matemáticas[2]. Descubierta e implementada independientemente en Mizar, Isar (Isabelle), etc. Combinación de ideas:

- **Deducción natural en estilo de *Fitch***. Notación equivalente, demostraciones como listas de fórmulas en lugar de árboles.
- **Reglas de inferencia *declarativas***: una forma de afirmar que $A_1, \ldots, A_n \vdash A$ es válida, sin tener que demostrarlo a mano (automáticamente).
- **Sintaxis similar a un lenguaje de programación** en lugar al lenguaje natural.

[2] *Mathematical Vernacular* de Freek Wiedijk

PPA

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

```
axiom falta_reprueba: forall A . forall E .
    falta(A, E) -> reprueba(A, E)
axiom reprueba_recursa: forall A . forall M .
    reprueba(A, final(M)) -> recursa(A, M)
```

PPA

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

```
axiom falta_reprueba: forall A . forall E .
    falta(A, E) -> reprueba(A, E)
axiom reprueba_recursa: forall A . forall M .
    reprueba(A, final(M)) -> recursa(A, M)

theorem falta_entonces_recursa: forall A . forall M .
    falta(A, final(M)) -> recursa(A, M)
proof
```

PPA

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

```
axiom falta_reprueba: forall A . forall E .
   falta(A, E) -> reprueba(A, E)
axiom reprueba_recursa: forall A . forall M .
   reprueba(A, final(M)) -> recursa(A, M)

theorem falta_entonces_recursa: forall A . forall M .
   falta(A, final(M)) -> recursa(A, M)
proof
   let A
   let M
```

PPA

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

```
axiom falta_reprueba: forall A . forall E .
   falta(A, E) -> reprueba(A, E)
axiom reprueba_recursa: forall A . forall M .
   reprueba(A, final(M)) -> recursa(A, M)

theorem falta_entonces_recursa: forall A . forall M .
   falta(A, final(M)) -> recursa(A, M)
proof
   let A
   let M
   suppose falta: falta(A, final(M))
```

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

```
axiom falta_reprueba: forall A . forall E .
    falta(A, E) -> reprueba(A, E)
axiom reprueba_recursa: forall A . forall M .
    reprueba(A, final(M)) -> recursa(A, M)

theorem falta_entonces_recursa: forall A . forall M .
    falta(A, final(M)) -> recursa(A, M)
proof
    let A
    let M
    suppose falta: falta(A, final(M))
    have reprueba: reprueba(A, final(M)) by falta_reprueba, falta
```

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

```
axiom falta_reprueba: forall A . forall E .
   falta(A, E) -> reprueba(A, E)
axiom reprueba_recursa: forall A . forall M .
   reprueba(A, final(M)) -> recursa(A, M)

theorem falta_entonces_recursa: forall A . forall M .
   falta(A, final(M)) -> recursa(A, M)
proof
   let A
   let M
   suppose falta: falta(A, final(M))
   have reprueba: reprueba(A, final(M)) by falta_reprueba, falta
   thus recursa(A, M) by reprueba_recursa, reprueba
end
```

Programas

Un **programa** de PPA consiste en una lista de **declaraciones**, que pueden ser

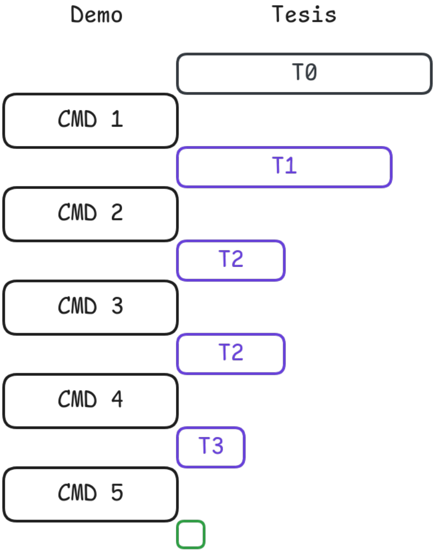
- **Axiomas**: fórmulas que se asumen válidas

```
axiom <name> : <form>
```

- **Teoremas**: fórmulas junto con sus demostraciones.

```
theorem <name> : <form>
proof
  <steps>
end
```

Demostraciones



- Lista de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.
- Tienen disponible un **contexto** con todas las hipótesis asumidas (axiomas) o demostradas (teoremas y comandos que demuestran hipótesis auxiliares).

by - El mecanismo principal de demostración

```
<form> by <h1>, ..., <hn>
```

- Afirma que la fórmula es una consecuencia lógica de las fórmulas que corresponden a las hipótesis provistas.
- Demostrado **automáticamente**: Por debajo usa un *solver* completo para lógica proposicional pero *heurístico* para primer orden.
- Dos comandos principales: **`thus`** y **`have`**.

Thus

```
thus <form> by <h1>, ..., <hn>
```

Si <form> es *parte* de la tesis, y el *solver* puede demostrar la implicación, lo demuestra automáticamente y lo descarga de la tesis.

Eliminación de implicación

```
axiom ax1: a -> b
axiom ax2: b -> c

theorem t1: a -> c
proof
   suppose a: a

   // La tesis ahora es c
   thus c by a, ax1, ax2
end
```

Eliminación de universal

```
axiom ax: forall X . f(X)

theorem t: f(n)
proof
   thus f(n) by ax
end
```

Have

```
have <name>: <form> by <h1>, ..., <hn>
```

Análogo a **`thus`**, pero introduce una afirmación *auxiliar* sin reducir la tesis, agregándola al contexto.

Have

```
have <name>: <form> by <h1>, ..., <hn>
```

Análogo a **`thus`**, pero introduce una afirmación *auxiliar* sin reducir la tesis, agregándola al contexto.

Hipótesis anterior implícita

Ambas pueden referirse a la hipótesis anterior con guión medio (`-`), y pueden hacerlo implícitamente usando **`hence`** y **`then`**.

Comando	Alternativo	¿Reduce la tesis?
`thus`	**`hence`**	Sí
`have`	**`then`**	No

Comandos y reglas de inferencia

Regla	Comando
LEM	**`cases`**
Ax	**`by`**
$\mathrm{I}\exists$	**`take`**
$\mathrm{E}\exists$	**`consider`**
$\mathrm{I}\forall$	**`let`**
$\mathrm{E}\forall$	**`by`**
$\mathrm{I}\vee_1$	**`by`**
$\mathrm{I}\vee_2$	**`by`**
$\mathrm{E}\vee$	**`cases`**

Regla	Comando
$\mathrm{I}\wedge$	**`by`**
$\mathrm{E}\wedge_1$	**`by`**
$\mathrm{E}\wedge_2$	**`by`**
$\mathrm{I}\rightarrow$	**`suppose`**
$\mathrm{E}\rightarrow$	**`by`**
$\mathrm{I}\neg$	**`suppose`**
$\mathrm{E}\neg$	**`by`**
$\mathrm{I}\top$	**`by`**
$\mathrm{E}\bot$	**`by`**

Comandos adicionales

equivalently `<form>`

Permite reducir la tesis a una fórmula equivalente

claim `<name>: <form>`

Análogo a **have** pero con una sub-demostración.

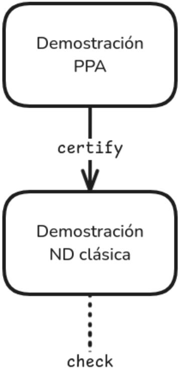
Esquema de claim

```
theorem t: <form1>
proof
   claim <name>: <form2>
   proof
      // Demostración de <form2>.
   end
   // Demostración de <form1> refiriéndose a <name>.
end
```

Certificador

Certificados

- Las demostraciones de PPA se *certifican* generando una demostración de deducción natural.
- No deberían generarse demostraciones erróneas, pero son chequeadas independientemente como mecanismo de *fallback*.



Criterio de de Bruijn

Un asistente de demostración cumple con el criterio de de Bruijn si satisface que sus demostraciones puedan ser chequeadas por un programa independiente, pequeño y confiable.

Contexto global

Se generan N demostraciones de deducción natural para cada programa, y se guardan en el *contexto global*. El chequeo se extiende a contextos.

```
axiom ax1: q
axiom ax2: q -> p
axiom ax3: p -> r

theorem t1: p
proof
    thus p by ax1, ax2
end

theorem t2: r
proof
    thus r by t1, ax3
end
```

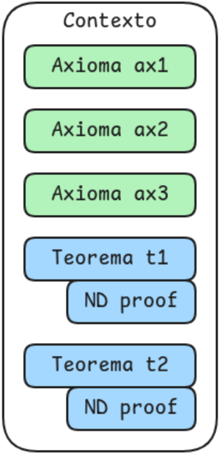


Figura: Contexto resultante de certificar un programa

Certificado de demostraciones

El certificado de una demostración es recursivo: se certifica cada comando, generando una demostración en deducción natural cuyas premisas son el certificado del resto de la demostración en PPA.

```
theorem t:
    p(v) -> exists X . p(X)
proof
    suppose h: p(v)
    take X := v
    thus p(v) by h
end
```

$$\cfrac{\cfrac{\cfrac{}{h : p(v) \vdash p(v)}\ \mathrm{Ax}_h}{h : p(v) \vdash \exists x.p(X)}\ \mathrm{I}\exists}{\vdash p(v) \to \exists x.p(X)}\ \mathrm{I}{\to}_h$$

Figura: Ejemplo de certificado generado para un programa

Contexto local

Cada demostración tiene un contexto local a ella con las hipótesis agregadas por ciertos comandos (**suppose**, **consider**, **have**, **claim**, etc.). Necesaria para obtener las fórmulas asociadas a las hipótesis en el **by**.

```
axiom ax1: p -> q
theorem t: (q -> r) -> p -> r
proof
   suppose h1: (q -> r)
   suppose h2: p
   then tq: q by ax1
   hence r by h1
end
```

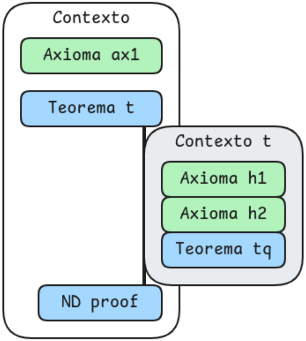


Figura: Ejemplo de contexto local

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \ldots, h_n : B_n\}$, para certificar
`thus A by h1, ..., hn:`

1. Buscamos las hipótesis en el contexto. Queremos demostrar

$$B_1 \wedge \ldots \wedge B_n \to A$$

2. **Razonamos por el absurdo**: Asumiendo la negación buscamos una contradicción

$$\begin{aligned} \neg(B_1 \wedge \ldots \wedge B_n \to A) &\equiv \neg(\neg(B_1 \wedge \ldots \wedge B_n) \vee A) \\ &\equiv B_1 \wedge \ldots \wedge B_n \wedge \neg A \end{aligned}$$

3. Convertimos la negación a forma normal disyuntiva (**DNF**)

$$(a_1 \wedge \ldots \wedge a_n) \vee \ldots \vee (b_1 \wedge \ldots \wedge b_m)$$

4. Buscamos una **contradicción** refutando cada cláusula individualmente. Será refutable si
   - Contiene $\perp$ o dos fórmulas opuestas $(a, \neg a)$,
   - Eliminando existenciales consecutivos y reiniciando el proceso, se consigue una refutación $(\neg p(k), \forall x.p(x))$

Ejemplo sin cuantificadores (1/2)

Tenemos el siguiente programa

```
axiom ax1: a -> b
axiom ax2: a
theorem t: b
proof
   thus b by ax1, ax2
end
```

1. Para certificar **thus** b **by** ax1, ax2 hay que generar una demostración para la implicación

$$\big((a \to b) \wedge a\big) \to b$$

2. Negamos la fórmula

$$\neg\big[\big((a \to b) \wedge a\big) \to b\big]$$

Ejemplo sin cuantificadores (2/2)

3. La convertimos a DNF

$$
\begin{array}{ll}
\neg[((a \to b) \land a) \to b] & \\
\equiv \neg[\neg((a \to b) \land a) \lor b] & (A \to B \equiv \neg A \lor B) \\
\equiv \neg\neg((a \to b) \land a) \land \neg b & (\neg(A \lor B) \equiv \neg A \land \neg B) \\
\equiv ((a \to b) \land a) \land \neg b & (\neg\neg A \equiv A) \\
\equiv (\neg a \lor b) \land a \land \neg b & (A \to B \equiv \neg A \lor B) \\
\equiv (\neg a \lor b) \land a \land \neg b & ((A \lor B) \land C \equiv (A \land C) \lor (B \land C)) \\
\equiv (\neg a \land a \land \neg b) \lor & \\
\quad (b \land a \land \neg b) &
\end{array}
$$

4. Refutamos cada cláusula

$$(\neg a \land a \land \neg b) \lor (b \land a \land \neg b)$$

Ejemplo con cuantificadores (1/3)

Tenemos el siguiente programa

```
axiom ax1: forall X . p(X) -> q(X)
axiom ax2: p(a)
theorem t: q(a)
proof
   thus q(a) by ax1, ax2
end
```

1. Para certificar **thus** `q(a)` **by** `ax1, ax2` hay que generar una demostración para la implicación

$$\Big((\forall x.(p(x) \rightarrow q(x))) \wedge p(a)\Big) \rightarrow q(a)$$

2. Negamos la fórmula

$$\neg\left[\Big((\forall x.(p(x) \rightarrow q(x))) \wedge p(a)\Big) \rightarrow q(a)\right]$$

Ejemplo con cuantificadores (2/3)

3. La convertimos a DNF

$$
\begin{aligned}
&\neg \left[\Big((\forall x.(p(x) \to q(x))) \land p(a) \Big) \to q(a) \right] \\
&\equiv \neg \left[\neg \Big((\forall x.(p(x) \to q(x))) \land p(a) \Big) \lor q(a) \right] \\
&\equiv \neg\neg \Big((\forall x.(p(x) \to q(x))) \land p(a) \Big) \land \neg q(a) \\
&\equiv (\forall x.(p(x) \to q(x))) \land p(a) \land \neg q(a)
\end{aligned}
$$

como a los ojos de DNF un $\forall$ es opaco, a pesar de que dentro tenga una implicación, la fórmula ya está en forma normal.

4. Buscamos una contradicción refutando cada cláusula. No hay forma encontrando literales opuestos o $\perp$, por ej. la cláusula $p(a)$ no es refutable.

Ejemplo con cuantificadores (3/3)

5. Probamos eliminando $\forall x.(p(x) \rightarrow q(x))$. Reemplazamos x por una meta-variable fresca u.

$$(p(\mathrm{u}) \rightarrow q(\mathrm{u})) \land p(a) \land \neg q(a)$$

6. Convertimos a DNF

$$\begin{aligned}
&(p(\mathrm{u}) \rightarrow q(\mathrm{u})) \land p(a) \land \neg q(a) \\
&\equiv (\neg p(\mathrm{u}) \lor q(\mathrm{u})) \land p(a) \land \neg q(a) \\
&\equiv ((\neg p(\mathrm{u}) \land p(a)) \lor (q(\mathrm{u}) \land p(a))) \land \neg q(a) \\
&\equiv (\neg p(\mathrm{u}) \land p(a) \land \neg q(a)) \lor \\
&\quad\ (q(\mathrm{u}) \land p(a) \land \neg q(a))
\end{aligned}$$

7. Buscamos una contradicción refutando cada cláusula. Los literales opuestos tienen que *unificar* en lugar de ser iguales.
   - $\neg p(\mathrm{u}) \land p(a) \land \neg q(a)$ tenemos $p(\mathrm{u}) \doteq p(a)$ con $\{\mathrm{u} := a\}$
   - $q(\mathrm{u}) \land p(a) \land \neg q(a)$ tenemos $q(\mathrm{u}) \doteq q(a)$ con $\{\mathrm{u} := a\}$

Desafío

¡Hay que generar una demostración en deducción natural!

Pasos

- **Razonamiento por el absurdo**: mediante las *reglas admisibles* cut y eliminación de la doble negación (E$\neg\neg$).
- **Conversión a DNF**: mediante la implementación de un *sistema de reescritura*.
- **Contradicciones**: mediante la *regla admisible* E$\wedge_\varphi$ + E$\vee$ + I$\neg$.
- **Eliminación de cuantificadores universales**: mediante unificación y E$\forall$.

Razonamiento por el absurdo

$$\vdash B_1 \land \ldots \land B_n \to A \overset{?}{\leadsto} \neg(B_1 \land \ldots \land B_n \to A) \vdash \bot$$

Teorema (DNeg Elim)

$$\dfrac{}{\neg\neg A \vdash A}\, E\neg\neg$$

Teorema (cut)

$$\dfrac{\Gamma, B \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A}\, cut$$

Lema (Razonamiento por el absurdo)

$$\dfrac{\dfrac{\begin{matrix}\vdots \\ \Gamma, \neg A \vdash \bot\end{matrix}}{\Gamma \vdash \neg\neg A}\, I\neg}{\Gamma \vdash A}\, cut,\ E\neg\neg$$

Conversión a DNF

Implementamos una traducción mediante el siguiente sistema de reescritura. **Algoritmo**: reescribir de a un paso hasta que no cambie (clausura de Kleene)

$\neg\neg a \rightsquigarrow a$	eliminación de $\neg\neg$
$\neg\bot \rightsquigarrow \top$	
$\neg\top \rightsquigarrow \bot$	
$a \to b \rightsquigarrow \neg a \vee b$	definición de implicación
$\neg(a \vee b) \rightsquigarrow \neg a \wedge \neg b$	distributiva de $\neg$ sobre $\wedge$
$\neg(a \wedge b) \rightsquigarrow \neg a \vee \neg b$	distributiva de $\neg$ sobre $\vee$
$(a \vee b) \wedge c \rightsquigarrow (a \wedge c) \vee (b \wedge c)$	distributiva de $\wedge$ sobre $\vee$ (der)
$c \wedge (a \vee b) \rightsquigarrow (c \wedge a) \vee (c \wedge b)$	distributiva de $\wedge$ sobre $\vee$ (izq)
$a \vee (b \vee c) \rightsquigarrow (a \vee b) \vee c$	asociatividad de $\vee$
$a \wedge (b \wedge c) \rightsquigarrow (a \wedge b) \wedge c$	asociatividad de $\wedge$

Conversión a DNF - Congruencias

Para reescribir una sub-fórmula (trivial sintácticamente), hay que demostrar las congruencias de los conectivos.

$$a \vee \neg(b \vee c) \rightsquigarrow a \vee (\neg b \wedge \neg c)$$

Congruencias

$$A \vdash A' \Rightarrow A \wedge B \vdash A' \wedge B$$
$$A \vdash A' \Rightarrow A \vee B \vdash A' \vee B$$
$$A' \vdash A \Rightarrow \neg A \vdash \neg A'$$

$\neg$ es contravariante

Para demostrar $\neg A \vdash \neg A'$ no necesitamos una demostración de $A \vdash A'$, sino de $A' \vdash A$.
$\Rightarrow$ para todas las reescrituras, incluso las congruencias, tenemos que demostrarlas en ambos sentidos.

Conversión a DNF - Reglas admisibles

Reglas admisibles para conversión a DNF

Pasos base

$$\neg\neg a \dashv\vdash a$$

$$\neg\bot \dashv\vdash \top$$

$$\neg\top \dashv\vdash \bot$$

$$a \rightarrow b \dashv\vdash \neg a \vee b$$

$$\neg(a \vee b) \dashv\vdash \neg a \wedge \neg b$$

$$\neg(a \wedge b) \dashv\vdash \neg a \vee \neg b$$

$$(a \vee b) \wedge c \dashv\vdash (a \wedge c) \vee (b \wedge c)$$

$$c \wedge (a \vee b) \dashv\vdash (c \wedge a) \vee (c \wedge b)$$

$$a \vee (b \vee c) \dashv\vdash (a \vee b) \vee c$$

$$a \wedge (b \wedge c) \dashv\vdash (a \wedge b) \wedge c$$

Pasos recursivos de congruencia (con $A \dashv\vdash A'$)

$$A \wedge B \dashv\vdash A' \wedge B$$

$$A \vee B \dashv\vdash A' \vee B$$

$$\neg A \dashv\vdash \neg A'$$

Contradicciones

Ejemplo

$$
\dfrac{\dfrac{}{(\neg a \wedge a \wedge \neg b)}\text{Ax} \quad\quad \Pi_L \quad\quad \dfrac{\dfrac{}{\Gamma_1 \vdash b \wedge a \wedge \bot}\text{Ax}}{\Gamma, b \wedge a \wedge \bot \vdash \bot}\text{E}\wedge_\bot}{\Gamma \vdash \;\vee (b \wedge a \wedge \bot) \qquad \Gamma, \neg a \wedge a \wedge \neg b \vdash \bot \qquad \Gamma = (\neg a \wedge a \wedge \neg b) \vee (b \wedge a \wedge \bot) \vdash \bot}\text{E}\vee
$$

donde

$$
\Pi_L = \dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash \neg a \wedge a \wedge \neg b}\text{Ax}}{\Gamma_1 \vdash \neg a}\text{E}\wedge_{\neg a} \qquad \dfrac{\dfrac{}{\Gamma_1 \vdash \neg a \wedge a \wedge \neg b}\text{Ax}}{\Gamma_1 \vdash a}\text{E}\wedge_a}{\Gamma_1 = \Gamma, b \wedge a \wedge \bot \vdash \bot}\text{E}\neg
$$

Lema (Regla admisible $\text{E}\wedge_\varphi$)

$$
\dfrac{\Gamma \vdash \varphi_1 \wedge \ldots \wedge \varphi_i \wedge \ldots \wedge \varphi_n \qquad n \in \mathbb{N}}{\Gamma \vdash \varphi_i} E\wedge_{\varphi_i}
$$

Alcance y limitaciones del by

- **Completo** para lógica proposicional y **heurístico** para primer orden.
- Esto es aceptable, la validez de LPO es indecidible (Teorema de Church).
- Elimina los $\forall$ consecutivos de a lo sumo una hipótesis. Pero le faltan más cosas.

Ejemplo de falla en eliminación

```
axiom ax1: forall X . p(X) -> q(X)
axiom ax2: forall X . p(X)
theorem t: q(a)
proof
   thus q(a) by ax1, ax2
end
```

Descarga de conjunciones

Si la tesis es una conjunción, se puede probar un subconjunto de ella y se reduce el resto.

Descarga

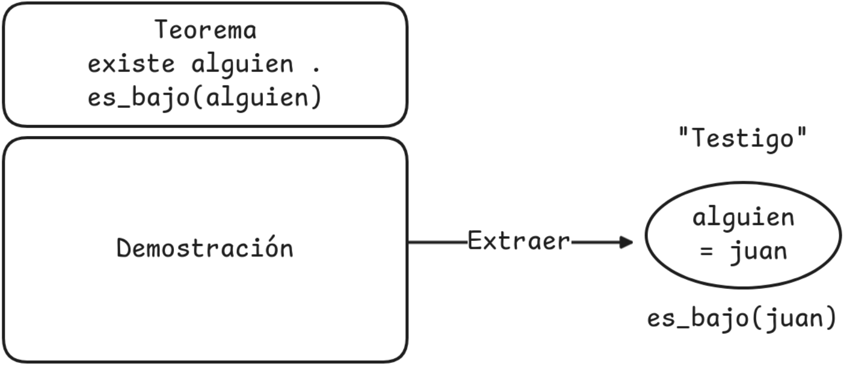
```
axiom "a": a
axiom "b": b
axiom "c": c
axiom "d": d
axiom "e": e
theorem "and discharge":
   (a & b) & ((c & d) & e)
proof
   thus a & e by "a", "e"
   thus d by "d"
   thus b & c by "b", "c"
end
```

Descarga de conjunciones

(TODO: Agregar esto)

Extracción de testigos

Intuición del problema



Extracción simple

Extracción simple

```
axiom ax: es_bajo(juan)
theorem t: exists Alguien . es_bajo(Alguien)
proof
   take Alguien := juan
   thus es_bajo(juan) by ax
end
```

Extracción con instanciación

```
axiom cero_min: forall N . cero <= N
theorem todo_numero_tiene_menor:
    forall N. exists M . M <= N
proof
    let N
    take M := cero
    thus <=(cero, N) by cero_min
end
```

Extraemos un testigo `cero` y podemos instanciar `N` en lo que sea, por ej. la siguiente fórmula es demostrable: `cero <= 60`

Extracción indirecta

Extracción indirecta

```
axiom ax1: no_es_alto(juan)
axiom ax2: forall X. no_es_alto(X) -> es_bajo(X)

theorem t1: exists X. no_es_alto(X)
proof
   take X := juan
   thus no_es_alto(juan) by ax1
end

theorem t2: exists X. es_bajo(X)
proof
   consider Y st h: no_es_alto(Y) by t1
   take X := Y
   hence es_bajo(Y) by ax2
end
```

Extracción **indirecta** de `theorem` `t2` nos da el testigo `juan`.

Extracción por el absurdo

Extracción por el absurdo

```
axiom juanEsBajo: bajo(juan)

theorem noTodoElMundoEsAlto: ~forall X. ~bajo(X)
proof
   suppose todosSonAltos: forall X. ~bajo(X)
   thus false by juanEsBajo, todosSonAltos
end

theorem hayAlguienBajo: exists X. bajo(X)
```

- En general $\exists x.\varphi \equiv \neg\forall x.\neg\varphi$.
- Sin **take** ($\mathrm{I}\exists$) explícito, igual podemos extraer el testigo a partir del **theorem** `hayAlguienBajo`: `juan`.
- La implementación no es tan directa como buscar un $\mathrm{I}\exists$

Buscamos un mecanismo general tal que,

Mecanismo de extracción de testigos

A partir de una demostración en PPA para una fórmula de la forma

$$\forall x_0 \ldots \forall x_n \exists y . \varphi(x_0, \ldots, x_n, y),$$

1. la certifique generando una demostración en deducción natural **clásica**
2. a partir de ella extraiga u tal que, para $t_0, \ldots, t_n$ cuales quiera, valga

$$\varphi(t_0, \ldots, t_n, u)$$

Extracción de testigos

Buscamos un mecanismo general tal que,

Mecanismo de extracción de testigos

A partir de una demostración en PPA para una fórmula de la forma

$$\forall x_0 \ldots \forall x_n \exists y . \varphi(x_0, \ldots, x_n, y),$$

1. la certifique generando una demostración en deducción natural **clásica**
2. a partir de ella extraiga u tal que, para $t_0, \ldots, t_n$ cuales quiera, valga

$$\varphi(t_0, \ldots, t_n, u)$$

Lógica clásica

La lógica clásica **no es constructiva**, por LEM:

$$\frac{}{\Gamma \vdash A \vee \neg A}\ \mathsf{LEM}$$

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

Sea C algo indecidible (tipo HALT), queremos ver que vale

$$\exists y.(y = 1 \wedge C) \vee (y = 0 \wedge \neg C)$$

podemos demostrarlo por LEM, sabemos que vale $C \vee \neg C$

- Supongamos que vale C. Tomo $y = 1$.
- Supongamos que vale $\neg C$. Tomo $y = 0$. $\square$

¡No nos dice cual es cierto! No es *constructiva*. No tenemos forma de saber si es cierto C o $\neg C$ (indecidible).

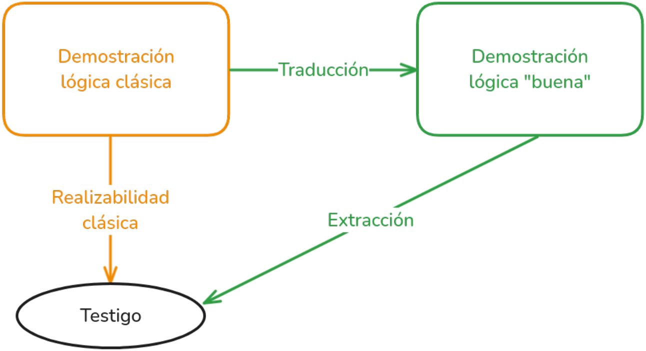
Teorema

Existen dos números irracionales a y b tales que a^b es racional.

¿Entonces por qué lógica clásica?

- Existen fórmulas que admiten demostraciones constructivas y no constructivas, y otras *solo no constructivas* (i.e. clásicas).

Clases de estrategias de extracción



Clases de estrategias de extracción de demostraciones en lógica clásica:

- **Directas**: Extraer directamente de demostraciones clásicas. Técnicas de *realizabilidad clásica* (Semánticas de λ-cálculos clásicos).
- **Indirectas**: Convertir la demostración a una lógica que se porte mejor y extraer de ahí.

$$\textbf{lógica intuicionista} = \text{lógica clásica} - \text{LEM}$$

Características:

- No tiene LEM[3], entonces siempre es constructiva.
- Noción de *forma normal* buena: una demostración de un $\exists$ debería comenzar con $\mathsf{I}\exists$:

$$\frac{\Gamma \vdash A\{x := t\}}{\Gamma \vdash \exists x.A}\ \mathsf{I}\exists$$

- Proceso de normalización análogo a reducción de λ-cálculo (su semántica operacional), visto desde el isomorfismo Curry-Howard.

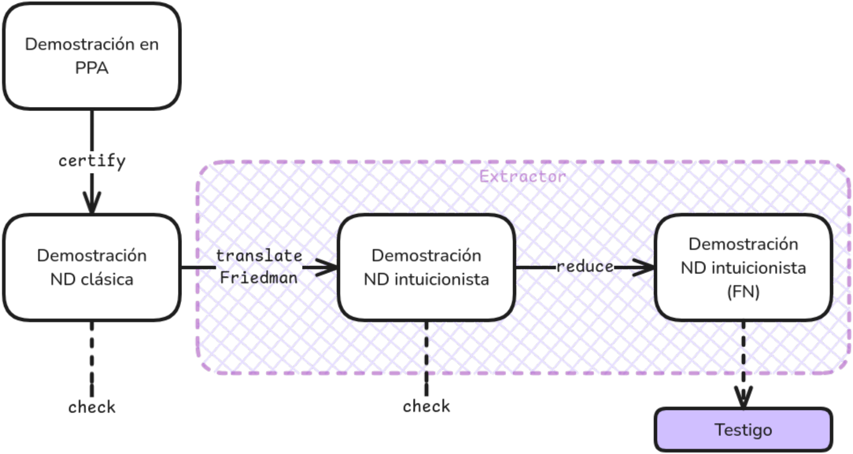
[3] Ni principios de razonamiento equivalentes, como $\mathsf{E}\neg\neg$

Traducción de Friedman

La **traducción de Friedman** permite embeber la lógica clásica en la intuicionista, para demostraciones de *algunas* fórmulas: de la clase Π_2, de la forma

$$\forall y_1 \ldots \forall y_n . \exists x . \varphi(x, y_1, \ldots, y_n)$$

Estrategia de extracción indirecta



Traducción de doble negación relativizada

Definición (Traducción de doble negación relativizada)

Sea $\neg_R A \equiv A \to R$, se define la traducción de doble negación relativizada:

$$
\begin{aligned}
\bot^{\neg\neg} &= \bot \\
A^{\neg\neg} &= \neg_R \neg_R A \quad \text{con } A \text{ atómica} \\
(\neg A)^{\neg\neg} &= \neg_R A^{\neg\neg} \\
(A \wedge B)^{\neg\neg} &= A^{\neg\neg} \wedge B^{\neg\neg} \\
(A \vee B)^{\neg\neg} &= \neg_R(\neg_R A^{\neg\neg} \wedge \neg_R B^{\neg\neg}) \\
(A \to B)^{\neg\neg} &= A^{\neg\neg} \to B^{\neg\neg} \\
(\forall x.A)^{\neg\neg} &= \forall x.A^{\neg\neg} \\
(\exists x.A)^{\neg\neg} &= \neg_R \forall x.\neg_R A^{\neg\neg}
\end{aligned}
$$

Teorema

Si $\Pi \rhd \Gamma \vdash_C A$, *luego* $\Pi^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I A^{\neg\neg}$

Traducción de doble negación relativizada

Definición (Traducción de doble negación relativizada)

Sea $\neg_R A \equiv A \to R$, se define la traducción de doble negación relativizada:

$$
\begin{aligned}
\bot^{\neg\neg} &= \bot \\
A^{\neg\neg} &= \neg_R \neg_R A \quad \text{con } A \text{ atómica} \\
(\neg A)^{\neg\neg} &= \neg_R A^{\neg\neg} \\
(A \wedge B)^{\neg\neg} &= A^{\neg\neg} \wedge B^{\neg\neg} \\
(A \vee B)^{\neg\neg} &= \neg_R(\neg_R A^{\neg\neg} \wedge \neg_R B^{\neg\neg}) \\
(A \to B)^{\neg\neg} &= A^{\neg\neg} \to B^{\neg\neg} \\
(\forall x.A)^{\neg\neg} &= \forall x.A^{\neg\neg} \\
(\exists x.A)^{\neg\neg} &= \neg_R \forall x.\neg_R A^{\neg\neg}
\end{aligned}
$$

Teorema

Si $\Pi \rhd \Gamma \vdash_C A$, *luego* $\Pi^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I A^{\neg\neg}$

El truco de Friedman

Teorema (Traducción de Friedman)

Sea Π una demostración clásica de

$$\forall y_1 \ldots \forall y_n . \exists x . \varphi(x, y_1, \ldots, y_n),$$

y φ una fórmula ***conjuntiva****. Podemos generar una demostración intuicionista de la misma fórmula.*

Definición (Fórmulas conjuntivas)

Generadas por

$$A ::= \bot \mid \top \mid p(t_1, \ldots, t_n) \mid A \wedge A$$

El truco de Friedman

Ejemplo sin $\forall$ (similar, omitiendo detalles técnicos).

Lema (Traducción de Friedman simplificada)

Sea φ una fórmula conjuntiva. Si tenemos

$$\Gamma \vdash_C \exists x.\varphi,$$

podemos generar una demostración intuicionista de la misma fórmula

$$\Gamma^{\neg\neg} \vdash_I \exists x.\varphi.$$

El truco de Friedman

Demostración.

Aplicando la traducción tomando $R = \exists x.\varphi$, tenemos que

$$(\Pi \rhd \Gamma \vdash_C \exists x.\varphi)^{\neg\neg} \Leftrightarrow \Pi^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I \neg_R \forall x.\neg_R \varphi^{\neg\neg}$$

Luego,

$$\dfrac{\begin{matrix}\Pi^{\neg\neg} \\ \Gamma^{\neg\neg} \vdash_I \neg_R \forall x \neg_R \varphi^{\neg\neg}\end{matrix} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma^{\neg\neg}, \varphi \vdash_I \varphi}\;\text{Ax}}{\Gamma^{\neg\neg}, \varphi \vdash_I R = \exists x.\varphi}\;\text{I}\exists}{\Gamma^{\neg\neg} \vdash_I \neg_R \varphi}\;\text{I}\!\to}{\Gamma^{\neg\neg} \vdash_I \neg_R \varphi^{\neg\neg}}\;\text{cut, I}(\neg_R \cdot \neg\neg)}{\Gamma^{\neg\neg} \vdash_I \forall x \neg_R \varphi^{\neg\neg}}\;\text{I}\forall}{\Gamma^{\neg\neg} \vdash_I \exists x.\varphi}\;\text{E}\!\to$$

$\square$

Introducción de negación relativizada

Lema (Introducción de $\neg_R$)

Si A es conjuntiva, entonces vale $\neg_R A \vdash_I \neg_R A^{\neg\neg}$ y lo notamos con la regla admisible $I(\neg_R \cdot \neg\neg)$.

Demostración.

Por inducción estructural en la fórmula. Intuición:

- Atómicas trivial. $\neg_R A \vdash_I \neg_R A^{\neg\neg} \iff \neg_R A \vdash_I \neg_R \neg_R \neg_R A$ sale con *eliminación de triple negación*.
- En lógica intuicionista, el $\neg$ contiene "poca información". Son más difíciles las demostraciones como $\neg_R(A \wedge B) \vdash_I \neg_R(A \wedge B)^{\neg\neg}$
- En lógica clásica requeriría LEM.

$\square$

Traducción de demostraciones

Teorema

Si $\Pi \rhd \Gamma \vdash_C A$, *luego* $\Pi^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I A^{\neg\neg}$

Demostración.

Inducción estructural sobre la demostración. **Estrategia**: traducimos recursivamente las partes de Π y las usamos para construir una nueva demostración de $A^{\neg\neg}$.

- I$\wedge$, E$\wedge_1$, E$\wedge_2$, I$\to$, E$\to$, I$\vee_1$, I$\vee_2$, I$\forall$, E$\forall$, I$\neg$, E$\neg$, I$\top$, Ax, I$\exists$ fáciles.
- LEM interesante.
- E$\bot$ inducción estructural sobre la fórmula.
- E$\vee$ y E$\exists$ son análogos y requieren un truco: usar la eliminación de la doble negación. No vale E$\neg\neg$ pero si E$\neg_R\neg_R$ (probado por inducción estructural sobre la fórmula).

$\square$

Traducción de introducción de conjunción

Lema (Traducción de I$\wedge$)

$$\dfrac{\begin{array}{c}\Pi_A \\ \Gamma \vdash_I A\end{array} \qquad \begin{array}{c}\Pi_B \\ \Gamma \vdash_I B\end{array}}{\Gamma \vdash_I A \wedge B}\ I\wedge$$

Dada una aparición de la regla $I\wedge$, es posible traducirla generando una demostración de $(A \wedge B)^{\neg\neg} = A^{\neg\neg} \wedge B^{\neg\neg}$.

Traducción de introducción de conjunción

Demostración.

Por hipótesis inductiva, tenemos que

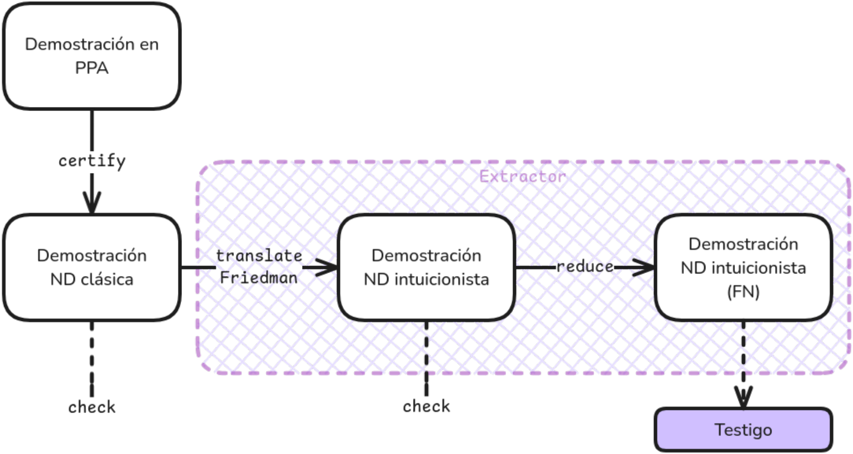
$$\Pi_A^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I A^{\neg\neg}$$
$$\Pi_B^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I B^{\neg\neg}$$

Luego, podemos generar una demostración de $A^{\neg\neg} \wedge B^{\neg\neg}$

$$\dfrac{\begin{array}{c}\Pi_A^{\neg\neg}\\ \Gamma^{\neg\neg} \vdash_I A^{\neg\neg}\end{array} \qquad \begin{array}{c}\Pi_B^{\neg\neg}\\ \Gamma^{\neg\neg} \vdash_I B^{\neg\neg}\end{array}}{\Gamma^{\neg\neg} \vdash_I A^{\neg\neg} \wedge B^{\neg\neg}}\ \mathrm{I}\wedge$$

$\square$

Repaso - estrategia de extracción indirecta



Normalización

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathsf{Ax}}{\vdash A \to A}\,\mathsf{I}{\to} \qquad \dfrac{\dfrac{}{B \vdash B}\,\mathsf{Ax}}{\vdash B \to B}\,\mathsf{I}{\to}}{\vdash (A \to A) \land (B \to B)}\,\mathsf{I}\land}{\vdash A \to A}\,\mathsf{E}\land_1 \quad \rightsquigarrow \quad \dfrac{\dfrac{}{A \vdash A}\,\mathsf{Ax}}{\vdash A \to A}\,\mathsf{I}{\to}$$

Normalización

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{Ax}}{\vdash A \to A}\,\mathrm{I}{\to} \qquad \dfrac{\dfrac{}{B \vdash B}\,\mathrm{Ax}}{\vdash B \to B}\,\mathrm{I}{\to}}{\dfrac{\vdash (A \to A) \wedge (B \to B)}{\vdash A \to A}\,\color{red}{\mathrm{E}\wedge_1}}\,\color{red}{\mathrm{I}\wedge} \quad \rightsquigarrow \quad \dfrac{\dfrac{}{A \vdash A}\,\mathrm{Ax}}{\vdash A \to A}\,\mathrm{I}{\to}$$

- Se van a ver todos de esa forma: Una **eliminación** demostrada inmediatamente por su **introducción** correspondiente.
- Ejemplo: $\mathrm{E}\wedge_1$ demostrada por $\mathrm{I}\wedge$.
- Idea: Simplificarlos sucesivamente hasta que no haya más y esté en **forma normal**.

Curry Howard

- **Isomorfismo Curry-Howard**: correspondencia entre demostraciones en deducción natural y términos de λ-cálculo.
- Normalización de demostraciones corresponde a semántica de λ-cálculo

Ejemplo

Conjunciones como el tipo de las tuplas, y las eliminaciones como proyecciones.

$$\pi_1(\langle M_1, M_2 \rangle) \leadsto M_1$$
$$\pi_2(\langle M_1, M_2 \rangle) \leadsto M_2$$

$$\dfrac{\dfrac{\begin{matrix}\Pi_1 \\ \Gamma \vdash A_1\end{matrix} \qquad \begin{matrix}\Pi_2 \\ \Gamma \vdash A_2\end{matrix}}{\Gamma \vdash A_1 \wedge A_2}\ \mathrm{I}\wedge}{\Gamma \vdash A_i}\ \mathrm{E}\wedge_i \qquad \leadsto \qquad \begin{matrix}\Pi_i \\ \Gamma \vdash A_i\end{matrix}$$

Normalización de implicación

$$\dfrac{\dfrac{\begin{array}{c}\Pi_B \\ \Gamma, h : A \vdash B\end{array}}{\Gamma \vdash A \to B}\ \mathrm{I}{\to}_h \qquad \begin{array}{c}\Pi_A \\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ \mathrm{E}{\to}$$

- Primer idea: $\Pi_B \triangleright \Gamma \vdash B$

Normalización de implicación

$$\dfrac{\dfrac{\begin{array}{c}\Pi_B\\ \Gamma, h : A \vdash B\end{array}}{\Gamma \vdash A \to B}\ \mathrm{I}{\to}_h \qquad \begin{array}{c}\Pi_A\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ \mathrm{E}{\to}$$

- Primer idea: $\xcancel{\Pi_B \rhd \Gamma \vdash B}$
- ¡**no sería correcto**! La demostración Π_B requiere la hipótesis $h : A$, que no necesariamente está en Γ, es agregada por $\mathrm{I}{\to}_h$
- Correcto: usar Π_B, pero *sustituyendo* todas las ocurrencias de la hipótesis h por la demostración Π_A.
- Es necesaria una noción de sustitución de hipótesis por demostraciones (sin capturas).

Normalización de implicación

$$\dfrac{\dfrac{\begin{array}{c}\Pi_B\\ \Gamma, h : A \vdash B\end{array}}{\Gamma \vdash A \to B}\ \mathrm{I}{\to}_h \qquad \begin{array}{c}\Pi_A\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B}\ \mathrm{E}{\to} \quad \rightsquigarrow \quad \begin{array}{c}\Pi_B\{h := \Pi_A\}\\ \Gamma \vdash B\end{array}$$

- Primer idea: $\cancel{\Pi_B \triangleright \Gamma \vdash B}$
- ¡**no sería correcto**! La demostración Π_B requiere la hipótesis $h : A$, que no necesariamente está en Γ, es agregada por $\mathrm{I}{\to}_h$
- Correcto: usar Π_B, pero *sustituyendo* todas las ocurrencias de la hipótesis h por la demostración Π_A.
- Es necesaria una noción de sustitución de hipótesis por demostraciones (sin capturas).

Reglas de reducción

Además, hay reglas para

- E$\exists$ con I$\exists$,
- E$\forall$ con I$\forall$,
- E$\neg$ con I$\neg$,
- E$\vee$ con I$\vee$

Algoritmo de reducción

- Original: Similar a DNF, reducir de a 1 paso sucesivamente hasta que sea irreducible.
- Problema: Congruencias se reducían de a un paso. Muy lento (demostraciones muy grandes)

$$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \mathsf{I}\wedge$$
$$\vdots$$
$$\Pi$$

reducíamos de a un paso a la vez $A \rightsquigarrow A_1 \rightsquigarrow A_2 \rightsquigarrow \ldots \rightsquigarrow A^*$ hasta llegar a A^* irreducible y recién ahí aplicamos mismo para B. En cada paso se recorría todo el árbol.

Dos tipos de estrategias:

- Un paso
- Muchos pasos
    - **Gross Knuth**: reduce en muchos pasos todos los sub-términos posibles al mismo tiempo.

En un solo paso, reducimos

$$\dfrac{\begin{array}{c} \dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\, \mathrm{I}\wedge \\ \vdots \\ \Pi \end{array}}{} \quad \rightsquigarrow \quad \begin{array}{c} \dfrac{\Gamma \vdash A^* \qquad \Gamma \vdash B^*}{\Gamma \vdash A \wedge B}\, \mathrm{I}\wedge \\ \vdots \\ \Pi \end{array}$$

Limitaciones

- **Incompleta**: no contempla *reducciones permutativas* (mezclando introducciones y eliminaciones de conectivos distintos).
  - *Mejora*: Implementarlas.
- **Ineficiente**: en cada paso reinicia la búsqueda de todos los focos de evaluación.
  - *Mejora*: Usar una máquina abstracta que implemente reducción a forma normal, Crégut para reducción *call-by-name* fuerte o la máquina de Biernacka para reducción *call-by-need* fuerte.

Programa con falla de extracción

```
axiom ax_1: roba(tuco) | mata(tuco)
axiom ax_2: forall X . roba(X) -> criminal(X)
axiom ax_3: forall X . mata(X) -> criminal(X)

theorem t: exists X . criminal(X)
proof
    take X := tuco
    cases by ax_1
        case roba(tuco)
            hence criminal(tuco)
                by ax_2

        case mata(tuco)
            hence criminal(tuco)
                by ax_3
    end
end
```

Certifica el programa generando una demostración que en lugar de comenzar con $I\exists$, comienza con $E\vee$ y en cada rama introduce el existencial dos veces, con el mismo término

Problema con axiomas

Lema (Traducción de Friedman simplificada)

Sea φ una fórmula conjuntiva. Si tenemos

$$\Gamma \vdash_C \exists x.\varphi,$$

podemos generar una demostración intuicionista de la misma fórmula

$$\Gamma^{\neg\neg} \vdash_I \exists x.\varphi.$$

Problema: la demostración normalizada no puede comenzar con I$\exists$

$$\neg_R \neg_R p(v) \vdash_I \exists x.p(x)$$

Nos gustaría

$$p(v) \vdash_I \exists x.p(x)$$

F-fórmulas

Definición (F-fórmulas)

B es una *F-fórmula* si está generada por:

$$
\begin{aligned}
B ::= {} & p(t_1, \ldots, t_n) \mid \bot \mid \top \\
& \mid B \wedge B \mid B \vee B \\
& \mid \forall x.B \mid \exists x.B \\
& \mid A \to B \\
& \mid \neg A
\end{aligned}
$$

Donde A son fórmulas *conjuntivas*

Lema (Introducción de la traducción $\neg\neg$)

Si B es una F-fórmula, vale $B \vdash_I B^{\neg\neg}$.

Manteniendo el contexto

- Después de la traducción, se reemplaza cada axioma
- Los axiomas deben ser F-fórmulas.

F-fórmulas

$$
\begin{aligned}
A &::= \bot \mid \top \mid p(t_1, \ldots, t_n) \\
F &::= A \\
&\quad \mid F \wedge F \mid F \vee F \\
&\quad \mid \forall x.F \mid \exists x.F \\
&\quad \mid C \rightarrow F \mid \neg C \\
C &::= A \mid C \wedge C
\end{aligned}
$$

A: Atómicas F: F-fórmulas C: Fórmulas conjuntivas

Harrop

$$
\begin{aligned}
A &::= \bot \mid \top \mid p(t_1, \ldots, t_n) \\
G &::= A \\
&\quad \mid G \wedge G \mid G \vee G \\
&\quad \mid \forall x.G \mid \exists x.G \\
&\quad \mid H \rightarrow G \\
H &::= A \mid H \wedge H \\
&\quad \mid \forall x.H \\
&\quad \mid G \rightarrow A
\end{aligned}
$$

G: G-fórmulas H: Harrop Hereditarias

Harrop

F-fórmulas

$$
\begin{aligned}
A &::= \bot \mid \top \mid p(t_1, \ldots, t_n) \\
F &::= A \\
&\quad \mid F \wedge F \mid F \vee F \\
&\quad \mid \forall x.F \mid \exists x.F \\
&\quad \mid C \rightarrow F \mid \neg C \\
C &::= A \mid C \wedge C
\end{aligned}
$$

A: Atómicas F: F-fórmulas C: Fórmulas conjuntivas

Harrop

$$
\begin{aligned}
A &::= \bot \mid \top \mid p(t_1, \ldots, t_n) \\
G &::= A \\
&\quad \mid G \wedge G \mid G \vee G \\
&\quad \mid \forall x.G \mid \exists x.G \\
&\quad \mid H \rightarrow G \\
H &::= A \mid H \wedge H \\
&\quad \mid \forall x.H \\
&\quad \mid G \rightarrow A
\end{aligned}
$$

G: G-fórmulas H: Harrop Hereditarias

Mecanismo de extracción de testigos

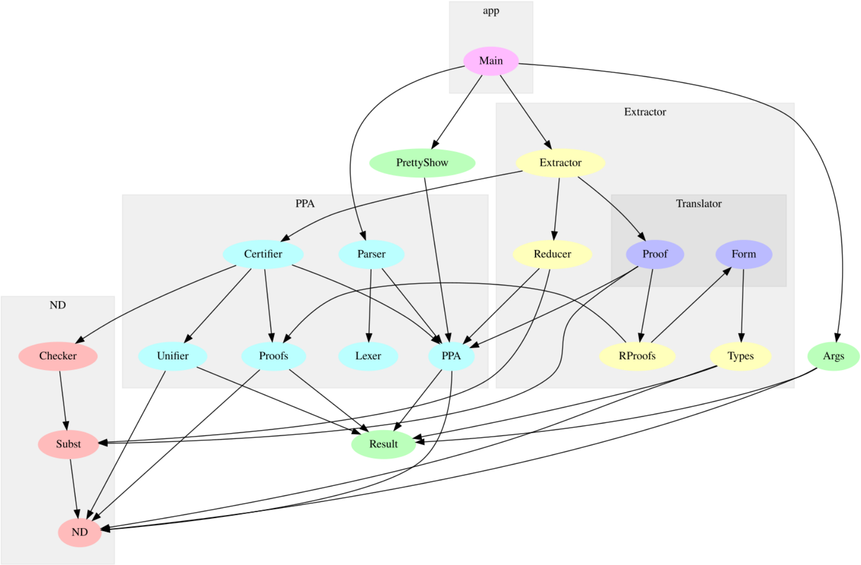
-

Dificultades

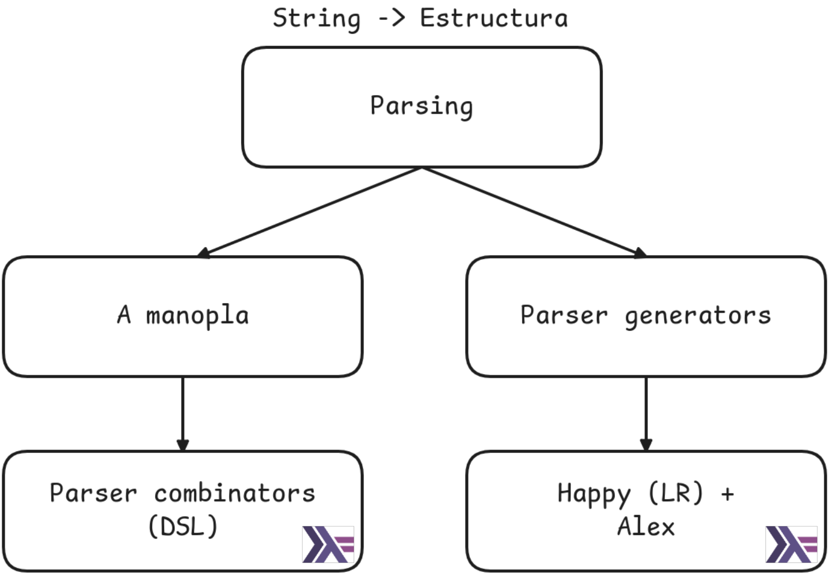
- *Integration hell*
- Encontré poca bibliografía de Friedman. Ni hablar que hable de generación de demostraciones en deducción natural.

Detalles de implementación

La herramienta ppa



Parser y lexer



Implementación

- Implementado en Haskell
- 330 tests
- X? LoC

Trabajo futuro

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.
- Refinar fórmulas conjuntivas. Profundizar vínculo con Harrop.

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.
- Refinar fórmulas conjuntivas. Profundizar vínculo con Harrop.
- Sofisticar reducción de demostraciones: hacer completa (reglas permutativas) y más eficiente (implementando máquina abstracta).

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.
- Refinar fórmulas conjuntivas. Profundizar vínculo con Harrop.
- Sofisticar reducción de demostraciones: hacer completa (reglas permutativas) y más eficiente (implementando máquina abstracta).
- Mejorar PPA como lenguaje de programación: módulos, importar archivos, biblioteca estándar

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.
- Refinar fórmulas conjuntivas. Profundizar vínculo con Harrop.
- Sofisticar reducción de demostraciones: hacer completa (reglas permutativas) y más eficiente (implementando máquina abstracta).
- Mejorar PPA como lenguaje de programación: módulos, importar archivos, biblioteca estándar
- Extender PPA con tipos (usando LPO *many-sorted* con géneros)

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.
- Refinar fórmulas conjuntivas. Profundizar vínculo con Harrop.
- Sofisticar reducción de demostraciones: hacer completa (reglas permutativas) y más eficiente (implementando máquina abstracta).
- Mejorar PPA como lenguaje de programación: módulos, importar archivos, biblioteca estándar
- Extender PPA con tipos (usando LPO *many-sorted* con géneros)
- Modelar de forma nativa inducción (segundo orden) e igualdad

Trabajo futuro

- Sofisticar el *solver heurístico* del **by** (recursivo, eliminar más de una hipótesis).
- Extender traducción de Friedman a más de un existencial.
- Refinar fórmulas conjuntivas. Profundizar vínculo con Harrop.
- Sofisticar reducción de demostraciones: hacer completa (reglas permutativas) y más eficiente (implementando máquina abstracta).
- Mejorar PPA como lenguaje de programación: módulos, importar archivos, biblioteca estándar
- Extender PPA con tipos (usando LPO *many-sorted* con géneros)
- Modelar de forma nativa inducción (segundo orden) e igualdad
- Mejorar reporte de errores (muy bajo nivel)

Fin

- QR con la página
- Preguntas