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

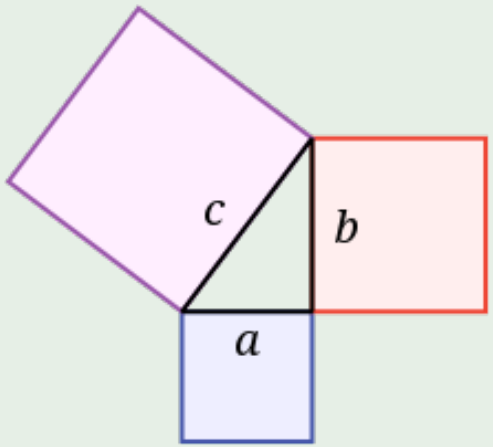
Repaso de lógica

- **Teorema**: Afirmación que puede ser *demostrada*.
- **Demostración** de un teorema:
  - *Argumento* que establece que el teorema es cierto
  - Usa *reglas de inferencia* a partir de *axiomas* y otros 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.

[1] Terrence Tao - Machine Assisted Proof

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).

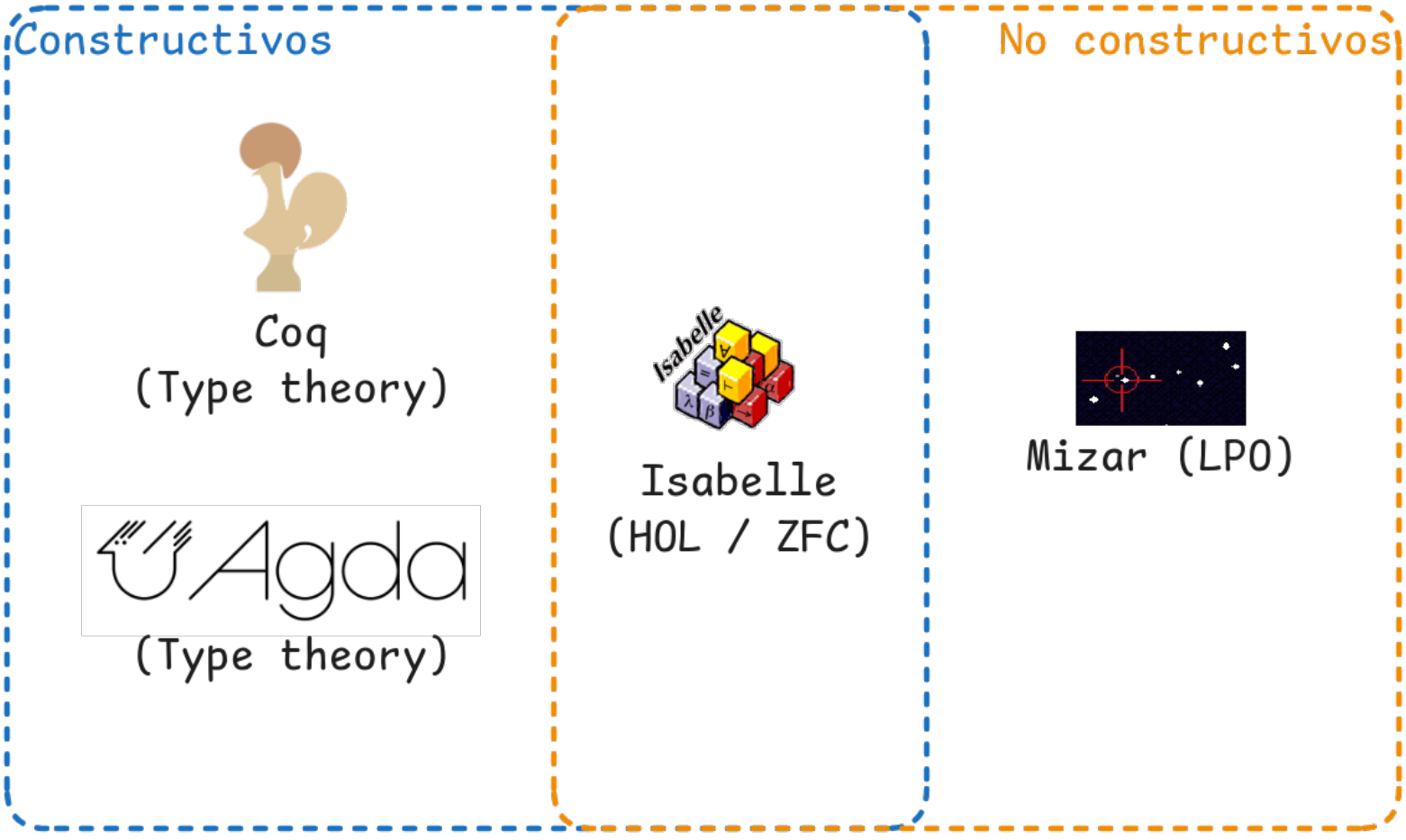
[1] Terrence Tao - Machine Assisted Proof

Asistentes de demostración

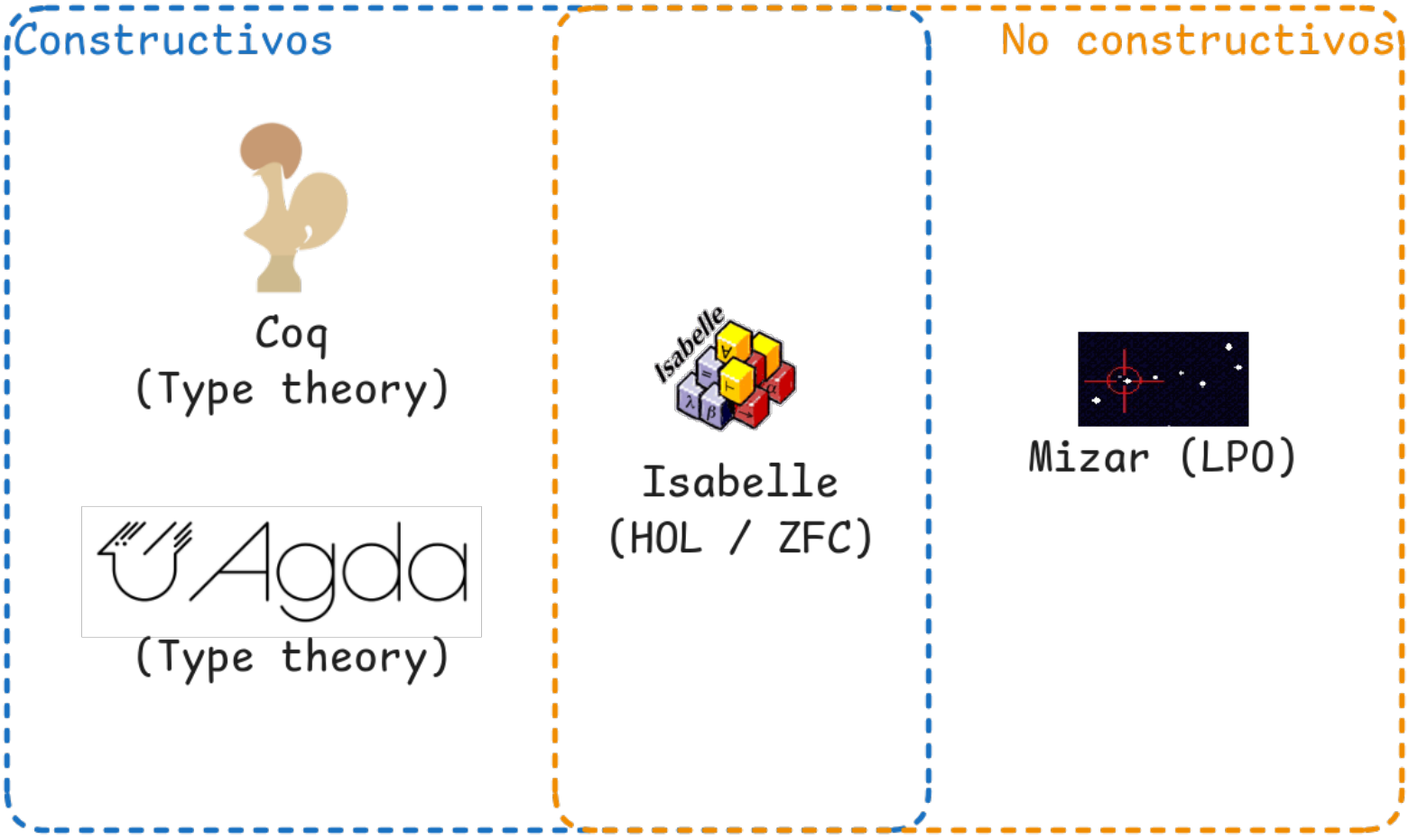
- Los **asistentes de demostración** son herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos usuales:
  - Formalización de teoremas matemáticos.
  - Verificación de programas.
- Ventajas:[1]
  - Facilitan la colaboración a gran escala (mediante la confianza 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

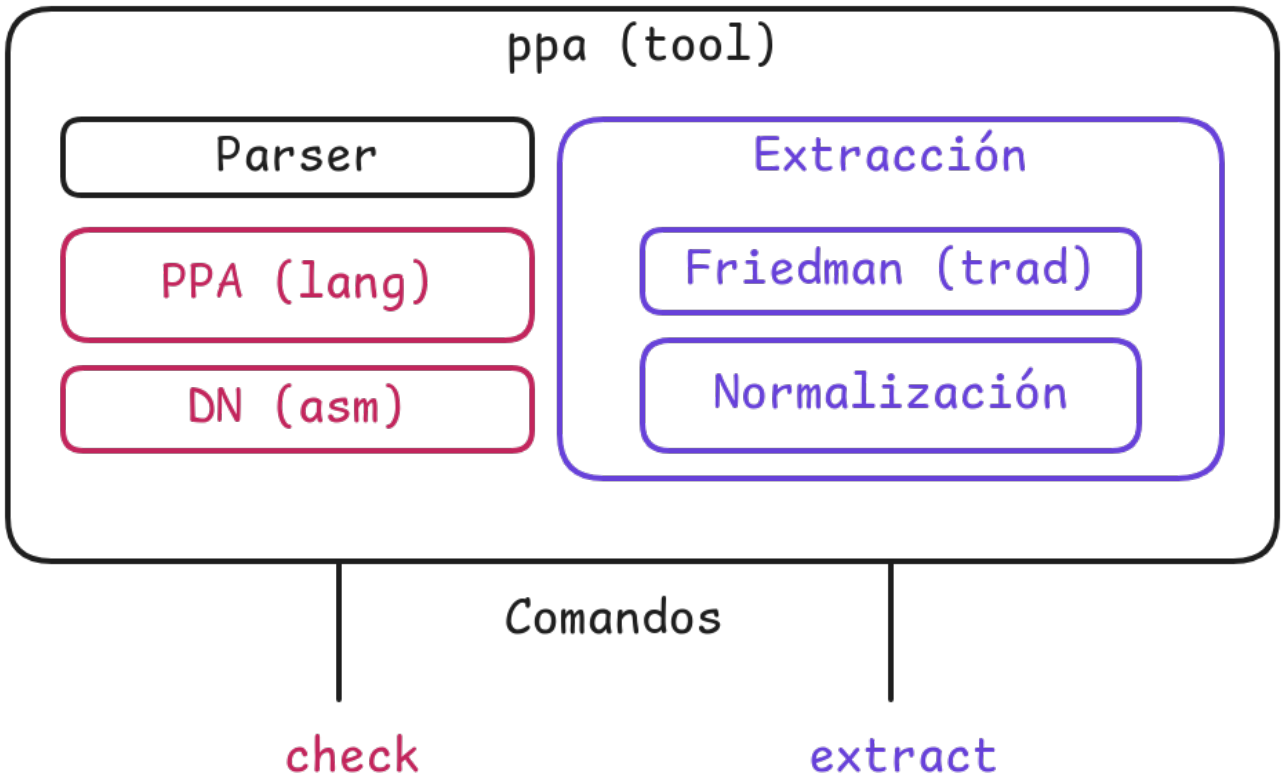


Asistentes de demostración



Extracción de testigos: De una demo de $\exists x.p(x)$, encontrar t tq $p(t)$.
Lógica constructiva = sencillo, no constructiva = complicado.

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.

□

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
    - *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:

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

Deducción natural

Ejemplo

Ejemplo (Demostración en DN)

Notamos:

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

Queremos probar

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

Ejemplo

Ejemplo (Demostración en DN)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash (F \to X) \wedge (X \to R)}\,\mathrm{Ax}}{\Gamma \vdash X \to R}\,\mathrm{E}\wedge_1 \qquad \Pi}{\dfrac{\Gamma = (F \to X) \wedge (X \to R),\ F \vdash R}{(F \to X) \wedge (X \to R) \vdash F \to R}\,\mathrm{I}\to}\,\mathrm{E}\to}{\vdash \Big((F \to X) \wedge (X \to R)\Big) \to (F \to R)}\,\mathrm{I}\to$$

donde

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

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 $\Gamma \vdash \varphi$. Intuición: "φ *es una consecuencia de las suposiciones de* Γ"
- 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.

Reglas de inferencia

Definición (Reglas de inferencia)

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

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

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

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

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

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

Alfa equivalencia

- Podemos usar $\exists x.p(x)$ y $\exists y.p(y)$ intercambiablemente.
- Son α-equivalentes (renombrando variables ligadas de forma apropiada, son iguales).

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)

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

Sustitución

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.

Eliminación de universal

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

Capturas

Evitamos automáticamente la **captura de variables**

$$(\forall y.p(x))\{x := y\} \neq \forall y.p(y) \qquad \text{(capturada)}$$
$$(\forall y.p(x))\{x := y\} = \forall z.p(y) \qquad \text{(renombrada)}$$

PPA

Mathematical Vernacular

$$\textit{Mathematical Vernacular}^2 = \text{Mizar} + \text{Isar (Isabelle)}$$

Forma *natural* de representar demostraciones matemáticas. Ideas:

- **Deducción natural en estilo de *Fitch***. Notación equivalente, demostraciones como listas de fórmulas en lugar de árboles.

[2] De Freek Wiedijk

Mathematical Vernacular

$$\textit{Mathematical Vernacular}^2 = \text{Mizar} + \text{Isar (Isabelle)}$$

Forma *natural* de representar demostraciones matemáticas. Ideas:

- **Deducción natural en estilo de *Fitch***. Notación equivalente, demostraciones como listas de fórmulas en lugar de árboles.
- **Reglas de inferencia *declarativas***: Afirmar

$$A_1, \ldots, A_n \vdash A$$

sin tener que demostrarlo a mano (automático).

[2] De Freek Wiedijk

Mathematical Vernacular

$$\textit{Mathematical Vernacular}^2 = \text{Mizar} + \text{Isar (Isabelle)}$$

Forma *natural* de representar demostraciones matemáticas. Ideas:

- **Deducción natural en estilo de *Fitch***. Notación equivalente, demostraciones como listas de fórmulas en lugar de árboles.
- **Reglas de inferencia *declarativas***: Afirmar

  $$A_1, \ldots, A_n \vdash A$$

  sin tener que demostrarlo a mano (automático).
- **Sintaxis similar a un lenguaje de programación** en lugar al lenguaje natural.

[2] De Freek Wiedijk

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

Ejemplo demostración

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

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

Ejemplo demostración

```
1  axiom falta_reprueba: forall A . forall E .
2     falta(A, E) -> reprueba(A, E)
3  axiom reprueba_recursa: forall A . forall M .
4     reprueba(A, final(M)) -> recursa(A, M)
5
6  theorem falta_entonces_recursa: forall A . forall M .
7     falta(A, final(M)) -> recursa(A, M)
8  proof
```

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

Ejemplo demostración

```
1  axiom falta_reprueba: forall A . forall E .
2     falta(A, E) -> reprueba(A, E)
3  axiom reprueba_recursa: forall A . forall M .
4     reprueba(A, final(M)) -> recursa(A, M)
5
6  theorem falta_entonces_recursa: forall A . forall M .
7     falta(A, final(M)) -> recursa(A, M)
8  proof
9     let A
10    let M
```

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

Ejemplo demostración

```
1  axiom falta_reprueba: forall A . forall E .
2     falta(A, E) -> reprueba(A, E)
3  axiom reprueba_recursa: forall A . forall M .
4     reprueba(A, final(M)) -> recursa(A, M)
5
6  theorem falta_entonces_recursa: forall A . forall M .
7     falta(A, final(M)) -> recursa(A, M)
8  proof
9     let A
10    let M
11    suppose falta: falta(A, final(M))
```

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

Ejemplo demostración

```
1  axiom falta_reprueba: forall A . forall E .
2     falta(A, E) -> reprueba(A, E)
3  axiom reprueba_recursa: forall A . forall M .
4     reprueba(A, final(M)) -> recursa(A, M)
5
6  theorem falta_entonces_recursa: forall A . forall M .
7     falta(A, final(M)) -> recursa(A, M)
8  proof
9     let A
10    let M
11    suppose falta: falta(A, final(M))
12    have reprueba: reprueba(A, final(M)) by falta_reprueba, falta
```

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

Ejemplo demostración

```
1  axiom falta_reprueba: forall A . forall E .
2     falta(A, E) -> reprueba(A, E)
3  axiom reprueba_recursa: forall A . forall M .
4     reprueba(A, final(M)) -> recursa(A, M)
5
6  theorem falta_entonces_recursa: forall A . forall M .
7     falta(A, final(M)) -> recursa(A, M)
8  proof
9     let A
10    let M
11    suppose falta: falta(A, final(M))
12    have reprueba: reprueba(A, final(M)) by falta_reprueba, falta
13    thus recursa(A, M) by reprueba_recursa, reprueba
14 end
```

by - El mecanismo principal de demostración

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

- Por debajo usa un *solver* completo para lógica proposicional pero *heurístico* para primer orden.
- Si puede, demuestra **automáticamente** que la fórmula es consecuencia lógica de la justificación.
- Toma las hipótesis del **contexto**: fórmulas asumidas (axiomas) o demostradas (teoremas y comandos que demuestran hipótesis auxiliares).

Comandos y reglas de inferencia

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

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

Comandos y reglas de inferencia

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

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

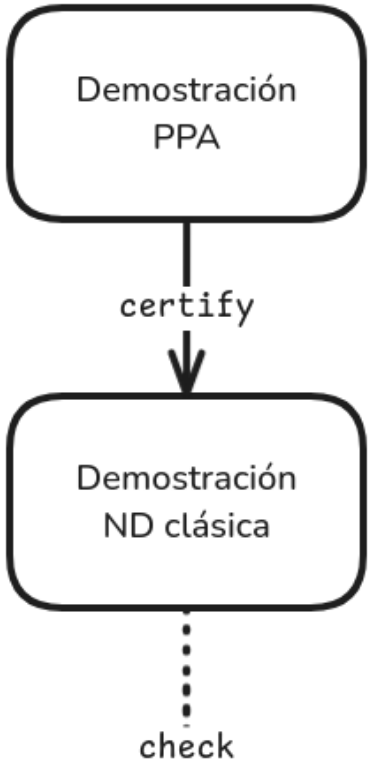
Adicionales:

- **`equivalently`**: Reduce la tesis a una fórmula equivalente.
- **`claim`**: Análogo a **`have`** pero con una sub-demostración.

Certificador

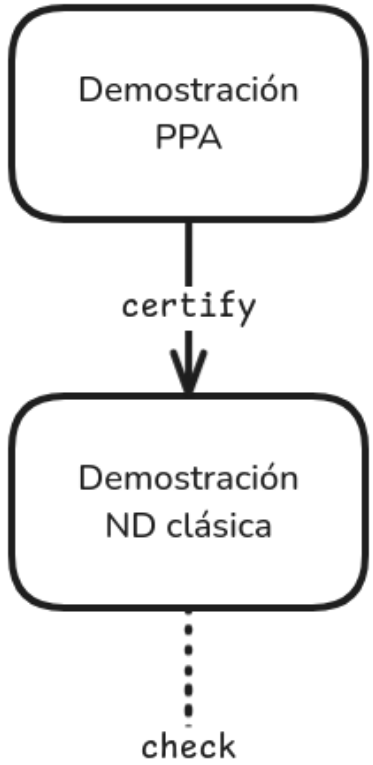
Certificados

- Las demostraciones de PPA se *certifican* generando una demostración de deducción natural.
- Evita confiar en la implementación del asistente.



Certificados

- Las demostraciones de PPA se *certifican* generando una demostración de deducción natural.
- Evita confiar en la implementación del asistente.



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.

Certificado de demostraciones

El certificado de una demostración es recursivo:

```
1  theorem t:
2     p(v) -> exists X . p(X)
3  proof
4     suppose h: p(v)
5     take X := v
6     thus p(v) by h
7  end
```

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

Figura: Ejemplo de certificado generado para un programa

Contexto global

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

```
1  axiom ax1: q
2  axiom ax2: q -> p
3  axiom ax3: p -> r
4
5  theorem t1: p
6  proof
7     thus p by ax1, ax2
8  end
9
10 theorem t2: r
11 proof
12    thus r by t1, ax3
13 end
```

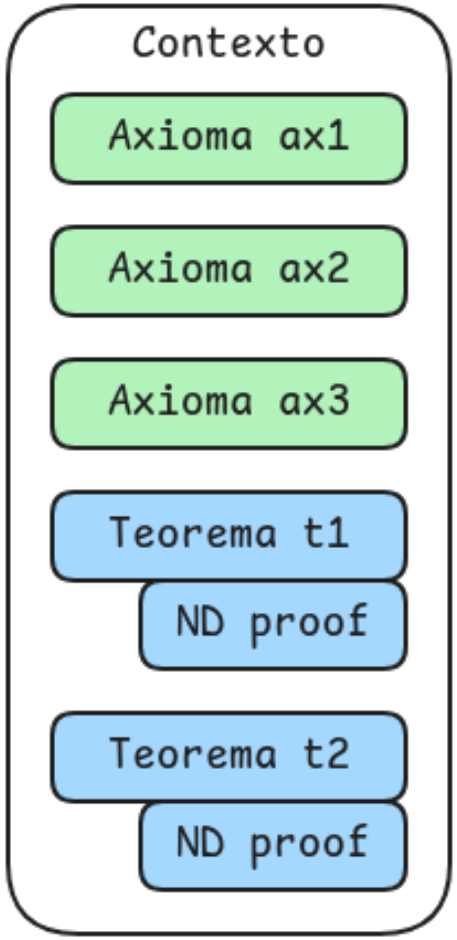


Figura: Contexto resultante de certificar un programa

Contexto local

Cada demostración tiene un contexto local a ella con las hipótesis agregadas por ciertos comandos (**suppose**, **consider**, **have**, **claim**, etc.).

```
1  axiom ax1: p -> q
2  theorem t: (q -> r) -> p -> r
3  proof
4     suppose h1: (q -> r)
5     suppose h2: p
6     then tq: q by ax1
7     hence r by h1
8  end
```

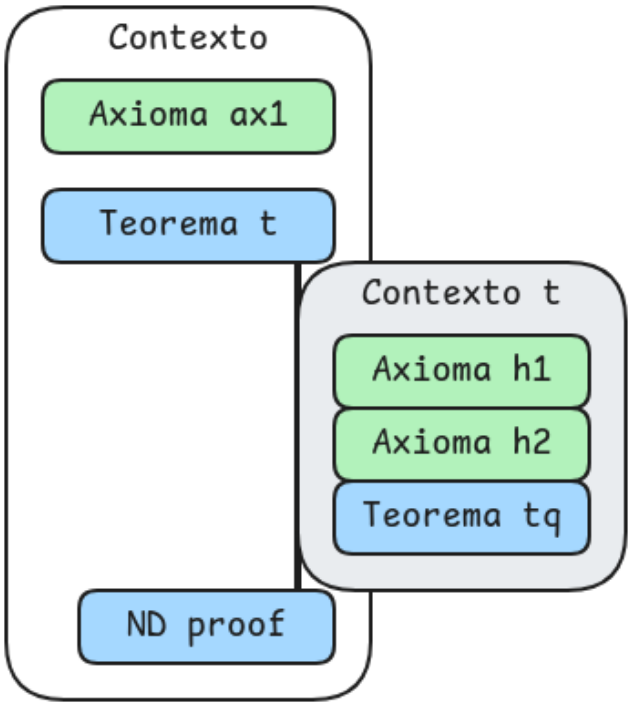


Figura: Ejemplo de contexto local

Certificado del by

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

1. Buscamos las hipótesis en el contexto. Queremos demostrar

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

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \ldots, h_n : B_n\}$, para **`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

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

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \ldots, h_n : B_n\}$, para `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**)

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

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \ldots, h_n : B_n\}$, para **`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

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \ldots, h_n : B_n\}$, para `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 $\bot$ o dos fórmulas opuestas $(a, \neg a)$,

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \ldots, h_n : B_n\}$, para `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 $\bot$ 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)

By sin cuantificadores

```
1  axiom ax1: a -> b
2  axiom ax2: a
3  theorem t: b
4  proof
5     thus b by ax1, ax2
6  end
```

Ejemplo sin cuantificadores (1/2)

By sin cuantificadores

```
1  axiom ax1: a -> b
2  axiom ax2: a
3  theorem t: b
4  proof
5     thus b by ax1, ax2
6  end
```

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

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

Ejemplo sin cuantificadores (1/2)

By sin cuantificadores

```
1  axiom ax1: a -> b
2  axiom ax2: a
3  theorem t: b
4  proof
5     thus b by ax1, ax2
6  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 y buscamos una contradicción.

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

Ejemplo sin cuantificadores (2/2)

3. La convertimos a DNF

$$
\begin{aligned}
&\neg[((a \to b) \wedge a) \to b] \\
&\equiv \neg[\neg((a \to b) \wedge a) \vee b] && (A \to B \equiv \neg A \vee B) \\
&\equiv \neg\neg((a \to b) \wedge a) \wedge \neg b && (\neg(A \vee B) \equiv \neg A \wedge \neg B) \\
&\equiv ((a \to b) \wedge a) \wedge \neg b && (\neg\neg A \equiv A) \\
&\equiv (\neg a \vee b) \wedge a \wedge \neg b && (A \to B \equiv \neg A \vee B) \\
&\equiv (\neg a \vee b) \wedge a \wedge \neg b && ((A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)) \\
&\equiv (\neg a \wedge a \wedge \neg b) \vee \\
&\quad (b \wedge a \wedge \neg b)
\end{aligned}
$$

Ejemplo sin cuantificadores (2/2)

3. La convertimos a DNF

$$
\begin{aligned}
&\neg[((a \to b) \wedge a) \to b] \\
&\equiv \neg[\neg((a \to b) \wedge a) \vee b] && (A \to B \equiv \neg A \vee B) \\
&\equiv \neg\neg((a \to b) \wedge a) \wedge \neg b && (\neg(A \vee B) \equiv \neg A \wedge \neg B) \\
&\equiv ((a \to b) \wedge a) \wedge \neg b && (\neg\neg A \equiv A) \\
&\equiv (\neg a \vee b) \wedge a \wedge \neg b && (A \to B \equiv \neg A \vee B) \\
&\equiv (\neg a \vee b) \wedge a \wedge \neg b && ((A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)) \\
&\equiv (\neg a \wedge a \wedge \neg b) \vee \\
&\quad (b \wedge a \wedge \neg b)
\end{aligned}
$$

4. Refutamos cada cláusula

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

Ejemplo con cuantificadores (1/3)

By con cuantificadores

```
1  axiom ax1: forall X . p(X) -> q(X)
2  axiom ax2: p(k)
3  theorem t: q(k)
4  proof
5     thus q(k) by ax1, ax2
6  end
```

Ejemplo con cuantificadores (1/3)

By con cuantificadores

```
1  axiom ax1: forall X . p(X) -> q(X)
2  axiom ax2: p(k)
3  theorem t: q(k)
4  proof
5     thus q(k) by ax1, ax2
6  end
```

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

$$\Big(\big(\forall x.(p(x) \to q(x))\big) \wedge p(k)\Big) \to q(k)$$

Ejemplo con cuantificadores (1/3)

By con cuantificadores

```
1  axiom ax1: forall X . p(X) -> q(X)
2  axiom ax2: p(k)
3  theorem t: q(k)
4  proof
5     thus q(k) by ax1, ax2
6  end
```

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

$$\Big(\big(\forall x.(p(x) \to q(x))\big) \wedge p(k) \Big) \to q(k)$$

2. Negamos la fórmula

$$\neg \left[\Big(\big(\forall x.(p(x) \to q(x))\big) \wedge p(k) \Big) \to q(k) \right]$$

Ejemplo con cuantificadores (2/3)

3. La convertimos a DNF

$$
\begin{aligned}
&\neg\Big[\Big((\forall x.(p(x) \to q(x))) \wedge p(k)\Big) \to q(k)\Big] \\
&\equiv \neg\Big[\neg\Big((\forall x.(p(x) \to q(x))) \wedge p(k)\Big) \vee q(k)\Big] \\
&\equiv \neg\neg\Big((\forall x.(p(x) \to q(x))) \wedge p(k)\Big) \wedge \neg q(k) \\
&\equiv (\forall x.(p(x) \to q(x))) \wedge p(k) \wedge \neg q(k)
\end{aligned}
$$

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.

Ejemplo con cuantificadores (2/3)

3. La convertimos a DNF

$$
\begin{aligned}
&\neg\Big[\Big((\forall x.(p(x) \to q(x))) \wedge p(k)\Big) \to q(k)\Big] \\
&\equiv \neg\Big[\neg\Big((\forall x.(p(x) \to q(x))) \wedge p(k)\Big) \vee q(k)\Big] \\
&\equiv \neg\neg\Big((\forall x.(p(x) \to q(x))) \wedge p(k)\Big) \wedge \neg q(k) \\
&\equiv (\forall x.(p(x) \to q(x))) \wedge p(k) \wedge \neg q(k)
\end{aligned}
$$

   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 $\bot$, por ej. la cláusula $p(a)$ no es refutable.

Ejemplo con cuantificadores (3/3)

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

$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$

Ejemplo con cuantificadores (3/3)

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

$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$

6. Convertimos a DNF

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

Ejemplo con cuantificadores (3/3)

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

$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$

6. Convertimos a DNF

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

7. Buscamos una contradicción refutando cada cláusula. Los literales opuestos tienen que *unificar* en lugar de ser iguales.

Ejemplo con cuantificadores (3/3)

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

$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$

6. Convertimos a DNF

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

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

Ejemplo con cuantificadores (3/3)

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

$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$

6. Convertimos a DNF

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

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

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 ($\mathsf{E}\neg\neg$).
- **Conversión a DNF**: mediante la implementación de un *sistema de reescritura*.
- **Contradicciones**: mediante la *regla admisible* $\mathsf{E}\wedge_{\varphi}$ + $\mathsf{E}\vee$ + $\mathsf{I}\neg$.
- **Eliminación de cuantificadores universales**: mediante unificación y $\mathsf{E}\forall$.

Razonamiento por el absurdo

Razonamiento por el absurdo en DNF

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

Razonamiento por el absurdo

Razonamiento por el absurdo en DNF

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

Teorema (DNeg Elim)

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

Teorema (cut)

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

Lema (Razonamiento por el absurdo)

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

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.

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

Conversión a DNF - Congruencias

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

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

Congruencias

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

Conversión a DNF - Congruencias

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

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

Congruencias

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

$\neg$ es contravariante

Para demostrar $\neg A \vdash \neg A'$ no necesitamos una demostración de $A \vdash A'$ (*covariante*), sino de $A' \vdash A$ (*contravariante*).

Conversión a DNF - Congruencias

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

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

Congruencias

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

$\neg$ es contravariante

Para demostrar $\neg A \vdash \neg A'$ no necesitamos una demostración de $A \vdash A'$ (*covariante*), sino de $A' \vdash A$ (*contravariante*).
$\Rightarrow$ 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

¡26 demostraciones!

Contradicciones

Ejemplo

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

donde

$$\Pi_L = \dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash \neg a \wedge a \wedge \neg b}\,\mathrm{Ax}}{\Gamma_1 \vdash \neg a}\,\mathrm{E}\wedge_{\neg a} \quad \dfrac{\dfrac{}{\Gamma_1 \vdash \neg a \wedge a \wedge \neg b}\,\mathrm{Ax}}{\Gamma_1 \vdash a}\,\mathrm{E}\wedge_{a}}{\Gamma_1 = \Gamma, b \wedge a \wedge \bot \vdash \bot}\,\mathrm{E}\neg$$

Contradicciones

Ejemplo

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

donde

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

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

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

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

```
1  axiom ax1: forall X . p(X) -> q(X)
2  axiom ax2: forall X . p(X)
3  theorem t: q(a)
4  proof
5     thus q(a) by ax1, ax2
6  end
```

Descarga de conjunciones

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

Descarga

```
1  axiom "a": a
2  axiom "b": b
3  axiom "c": c
4  axiom "d": d
5  axiom "e": e
6  theorem "and discharge":
7     (a & b) & ((c & d) & e)
8  proof
9     thus a & e by "a", "e"
10    thus d by "d"
11    thus b & c by "b", "c"
12 end
```

Descarga de conjunciones

(TODO: Agregar esto)

Extracción de testigos

Extracción simple

Extracción simple

```
1  axiom ax: es_bajo(juan)
2  theorem t: exists Alguien . es_bajo(Alguien)
3  proof
4     take Alguien := juan
5     thus es_bajo(juan) by ax
6  end
```

Extracción simple

Extracción simple

```
1  axiom ax: es_bajo(juan)
2  theorem t: exists Alguien . es_bajo(Alguien)
3  proof
4     take Alguien := juan
5     thus es_bajo(juan) by ax
6  end
```

take <var> := <term>

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

Extracción indirecta con instanciación

Extracción con instanciación

```
1  axiom cero_min: forall N . cero < N
2  axiom lt_leq: forall N . forall M . M < N -> M <= N
3
4  theorem todo_numero_tiene_lt: forall N. exists M . M < N
5  proof
6     let N
7     take M := cero
8     thus cero < N by cero_min
9  end
10
11 theorem todo_numero_tiene_leq: forall N. exists M . M <= N
12 proof
13    let N'
14    consider Min st h: Min < N' by todo_numero_tiene_lt
15    take M := Min
16    thus Min <= N' by h, lt_leq
17 end
```

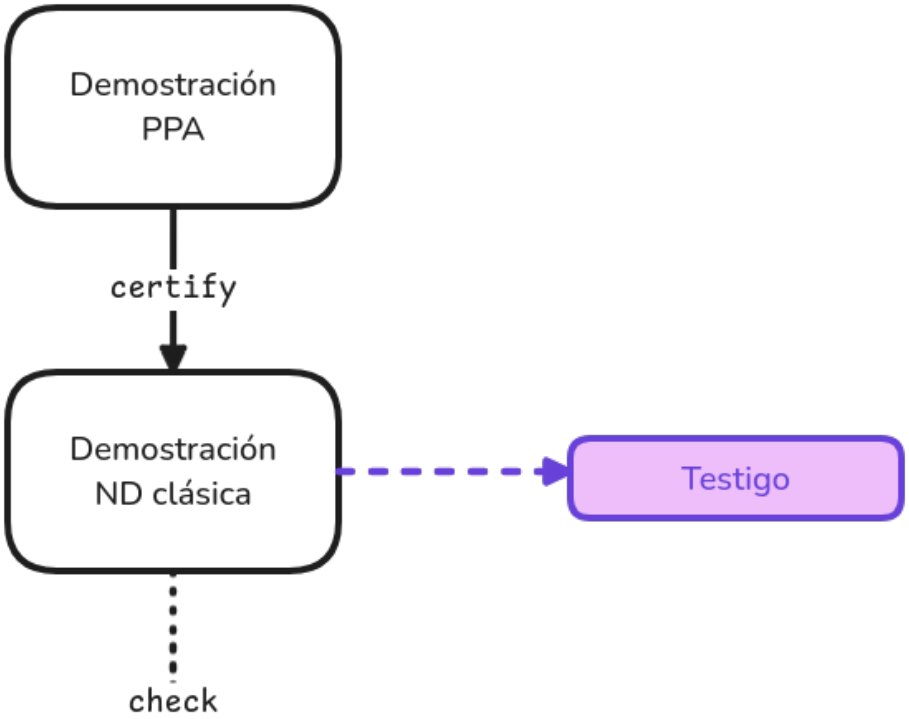
Extracción por el absurdo

Extracción por el absurdo

```
1  axiom juanEsBajo: bajo(juan)
2
3  theorem noTodoElMundoEsAlto: ~forall X. ~bajo(X)
4  proof
5     suppose todosSonAltos: forall X. ~bajo(X)
6     thus false by juanEsBajo, todosSonAltos
7  end
8
9  theorem hayAlguienBajo: exists X. bajo(X)
```

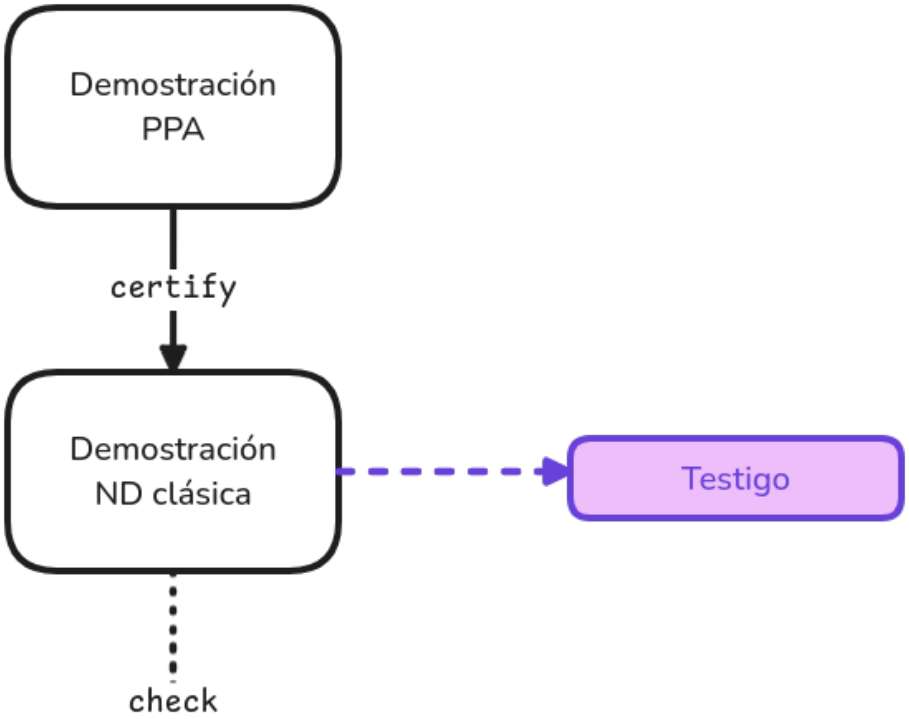
- En general $\exists x.\varphi \equiv \neg\forall x.\neg\varphi$.
- Sin **take** ($I\exists$) explícito, igual podemos extraer el testigo a partir del **theorem** hayAlguienBajo: juan.
- La implementación no es tan directa como buscar un $I\exists$ en el árbol de la demostración.

Lógica clásica



- Buscamos un mecanismo general que nos permita extraer testigos a partir de demostraciones en **deducción natural clásica**

Lógica clásica



- Buscamos un mecanismo general que nos permita extraer testigos a partir de demostraciones en **deducción natural clásica**
- Pero la lógica clásica **no es constructiva**, por LEM:

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

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

$$\exists y.(y = 1 \land C) \lor (y = 0 \land \neg C)$$

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

$$\exists y.(y = 1 \land C) \lor (y = 0 \land \neg C)$$

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

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

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

$$\exists y.(y = 1 \land C) \lor (y = 0 \land \neg C)$$

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

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

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

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

$$\exists y.(y = 1 \land C) \lor (y = 0 \land \neg C)$$

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

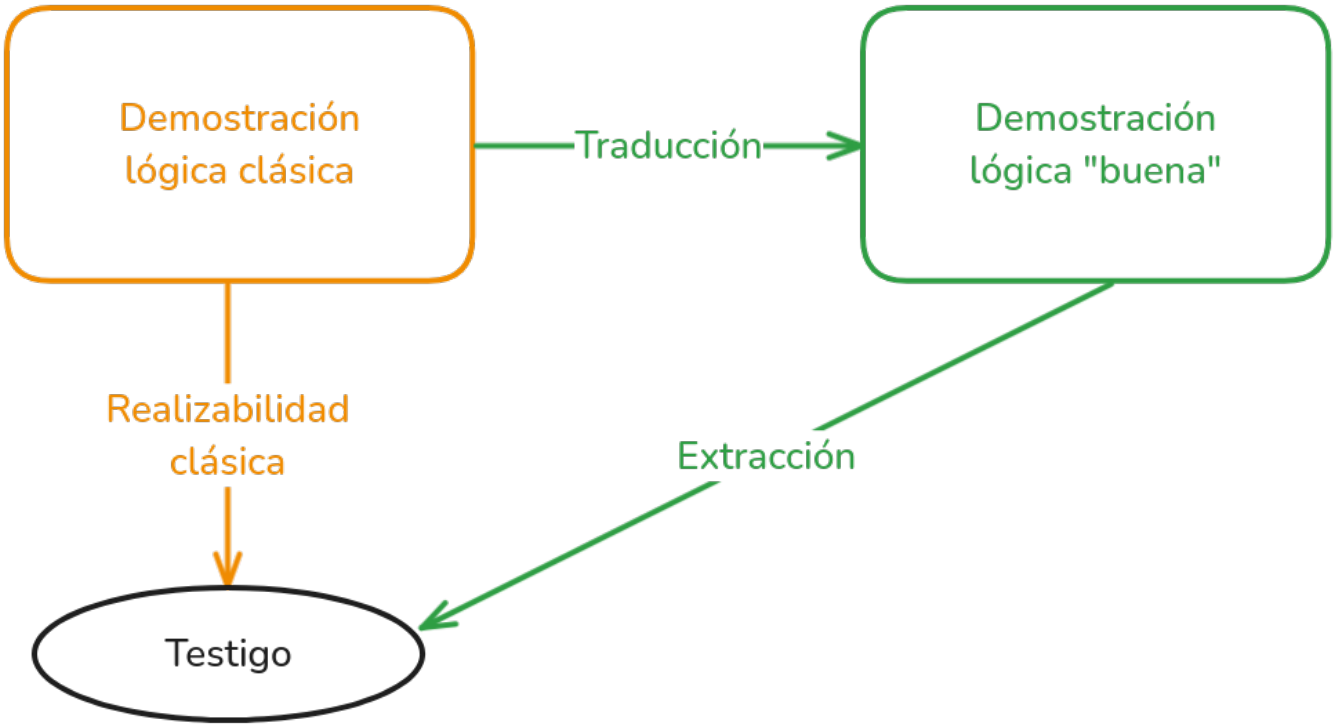
- Supongamos que vale C. Tomo $y = 1$.
- Supongamos que vale $\neg C$. Tomo $y = 0$. □

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

¿Entonces por qué lógica clásica?

- Existen fórmulas que admiten demostraciones constructivas y no constructivas, y otras *solo no constructivas* (i.e. clásicas).
- Además permite razonar por el absurdo, con E$\neg\neg \equiv$ LEM

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.

Lógica intuicionista

lógica intuicionista = lógica clásica − LEM

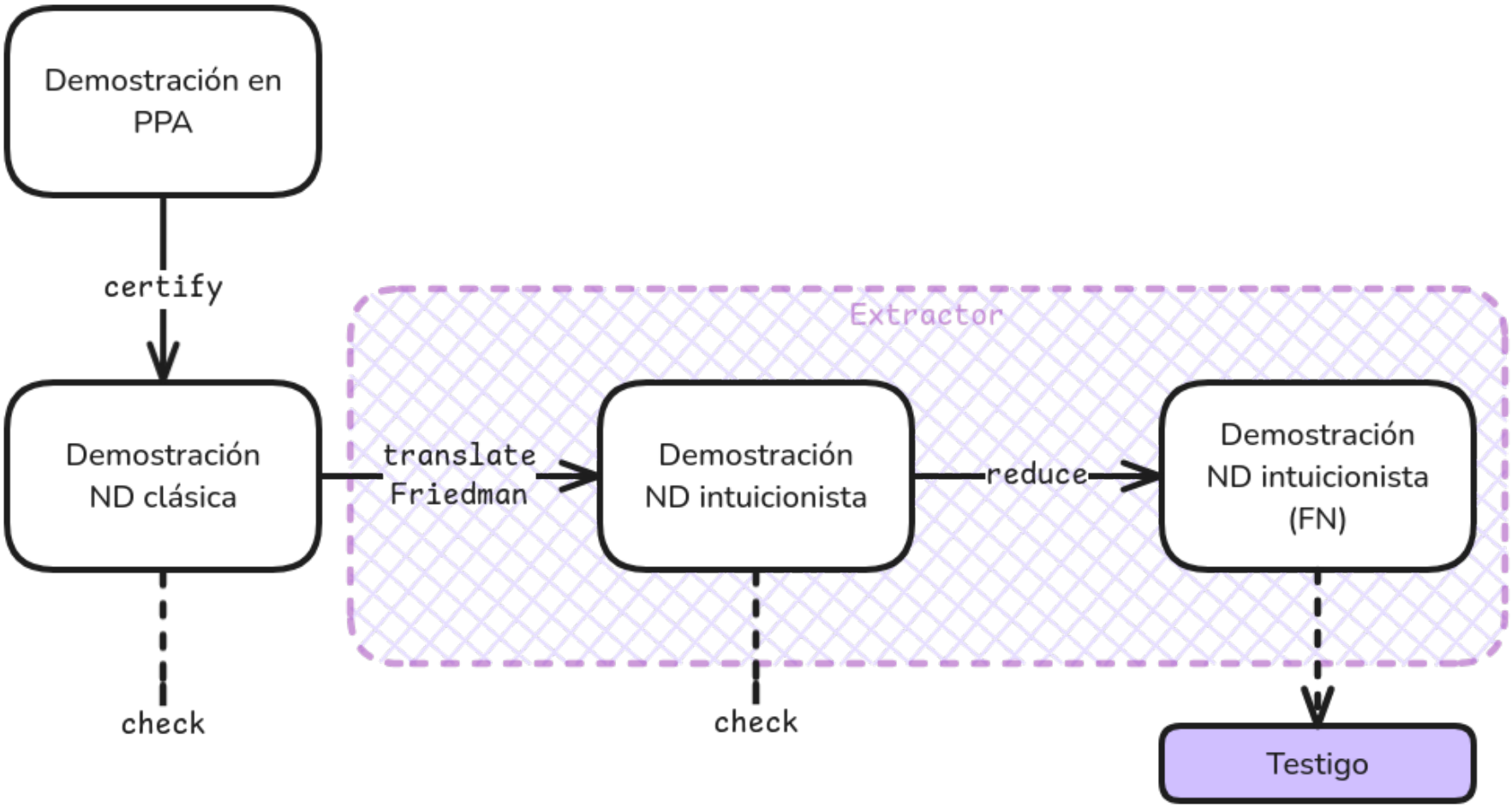
Características:

- No tiene LEM[3], entonces siempre es constructiva.
- Siempre permite hacer extracción de testigos: proceso de normalización con *forma normal* buena, una demostración de un $\exists$ debería comenzar con I$\exists$ y de ahí sacás el testigo.

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

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

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

Definición (Fórmulas conjuntivas)

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

Teorema (Traducción de Friedman)

*Sea φ una fórmula **conjuntiva**. Si tenemos*

$$\Pi \rhd \Gamma \vdash_C \forall y_1 \ldots \forall y_n . \exists x . \varphi(x, y_1, \ldots, y_n),$$

podemos generar una demostración intuicionista de la misma fórmula.

El truco de Friedman

Definición (Fórmulas conjuntivas)

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

Teorema (Traducción de Friedman)

*Sea φ una fórmula **conjuntiva**. Si tenemos*

$$\Pi \rhd \Gamma \vdash_C \forall y_1 \ldots \forall y_n . \exists x . \varphi(x, y_1, \ldots, y_n),$$

podemos generar una demostración intuicionista de la misma fórmula.

Lema (Traducción de Friedman simplificada)

Sea φ una fórmula conjuntiva. Si tenemos $\Pi \rhd \Gamma \vdash_C \exists x . \varphi$, podemos generar una demostración intuicionista de la misma fórmula.

El truco de Friedman

Demostración.

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

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

Luego,

El truco de Friedman

Demostración.

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

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

Luego,

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

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 $I(\neg_R \cdot \neg\neg)$.

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. Para predicados,

$$\neg_R A \vdash_I \neg_R A^{\neg\neg} \iff \neg_R A \vdash_I \neg_R \neg_R \neg_R A$$

  sale con *eliminación de triple negación*.
- Conjunción tiene algunos trucos.

Traducción de demostraciones

Teorema

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

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).

Traducción de introducción de conjunción

Lema (Traducción de I∧)

$$\dfrac{\begin{array}{cc}\Pi_A & \Pi_B \\ \Gamma \vdash_I A & \Gamma \vdash_I B\end{array}}{\Gamma \vdash_I A \wedge B}\, I\wedge$$

Es posible demostrar $(A \wedge B)^{\neg\neg} = A^{\neg\neg} \wedge B^{\neg\neg}$.

Demostración.

Usando la HI: $\Pi_A^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I A^{\neg\neg}$ y $\Pi_B^{\neg\neg} \rhd \Gamma^{\neg\neg} \vdash_I B^{\neg\neg}$, generamos

$$\dfrac{\begin{array}{cc}\Pi_A^{\neg\neg} & \Pi_B^{\neg\neg} \\ \Gamma^{\neg\neg} \vdash_I A^{\neg\neg} & \Gamma^{\neg\neg} \vdash_I B^{\neg\neg}\end{array}}{\Gamma^{\neg\neg} \vdash_I A^{\neg\neg} \wedge B^{\neg\neg}}\, I\wedge$$

□

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$\exists$

$$p(v)^{\neg\neg} \vdash_I \exists x.p(x) \iff \neg_R\neg_R p(v) \vdash_I \exists x.p(x)$$

Nos gustaría *mantener el contexto original*: $p(v) \vdash_I \exists x.p(x)$

Manteniendo el contexto

Luego de la traducción, antes de reducir, reemplazamos cada cita (Ax) de un axioma $h : \varphi^{\neg\neg}$ por la demostración $\varphi \vdash_I \varphi^{\neg\neg}$.

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

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

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: Fórmulas atómicas
- F: F-fórmulas
- C: Fórmulas conjuntivas

F-fórmulas

F-fórmulas

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

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

Fórmulas de Harrop

$$
\begin{aligned}
G &::= A \\
&\quad \mid G \wedge G \mid G \vee G \\
&\quad \mid \forall x.G \mid \exists x.G \\
&\quad \mid H \to G \\
H &::= A \mid H \wedge H \\
&\quad \mid \forall x.H \\
&\quad \mid G \to A
\end{aligned}
$$

- G: G-fórmulas
- H: Fórmulas Harrop Hereditarias
- Generalización de cláusulas de Horn, usadas para realizabilidad

Normalización

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Normalización

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathsf{Ax}}{\vdash A \to A}\,\mathsf{I}\to \quad \dfrac{\dfrac{}{B \vdash B}\,\mathsf{Ax}}{\vdash B \to B}\,\mathsf{I}\to}{\dfrac{\vdash (A \to A) \wedge (B \to B)}{\vdash A \to A}\,\mathsf{E}\wedge_1}\,\mathsf{I}\wedge \quad \rightsquigarrow \quad \dfrac{\dfrac{}{A \vdash A}\,\mathsf{Ax}}{\vdash A \to A}\,\mathsf{I}\to$$

- Se van a ver todos de esa forma: Una **eliminación** demostrada inmediatamente por su **introducción** correspondiente.
- Ejemplo: $\mathsf{E}\wedge_1$ demostrada por $\mathsf{I}\wedge$.
- 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.

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

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

Normalización de implicación

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

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

Normalización de implicación

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

- Primer idea: ~~$\Pi_B \triangleright \Gamma \vdash B$~~
- Π_B requiere $h : A$, agregada por $\mathsf{I}{\to}_h$
- Correcto: usar Π_B, pero *sustituyendo* todas las ocurrencias de la hipótesis h por la demostración Π_A (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

Idea original: reducir en un paso sucesivamente hasta que sea irreducible.

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

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.
Problema: Muy lento

Estrategia de reducción

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

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

Limitaciones

- **Incompleta**: no contempla *reducciones permutativas* (mezclando introducciones y eliminaciones de conectivos distintos).

Limitaciones

- **Incompleta**: no contempla *reducciones permutativas* (mezclando introducciones y eliminaciones de conectivos distintos).
  - *Mejora*: Implementarlas.

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.

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.

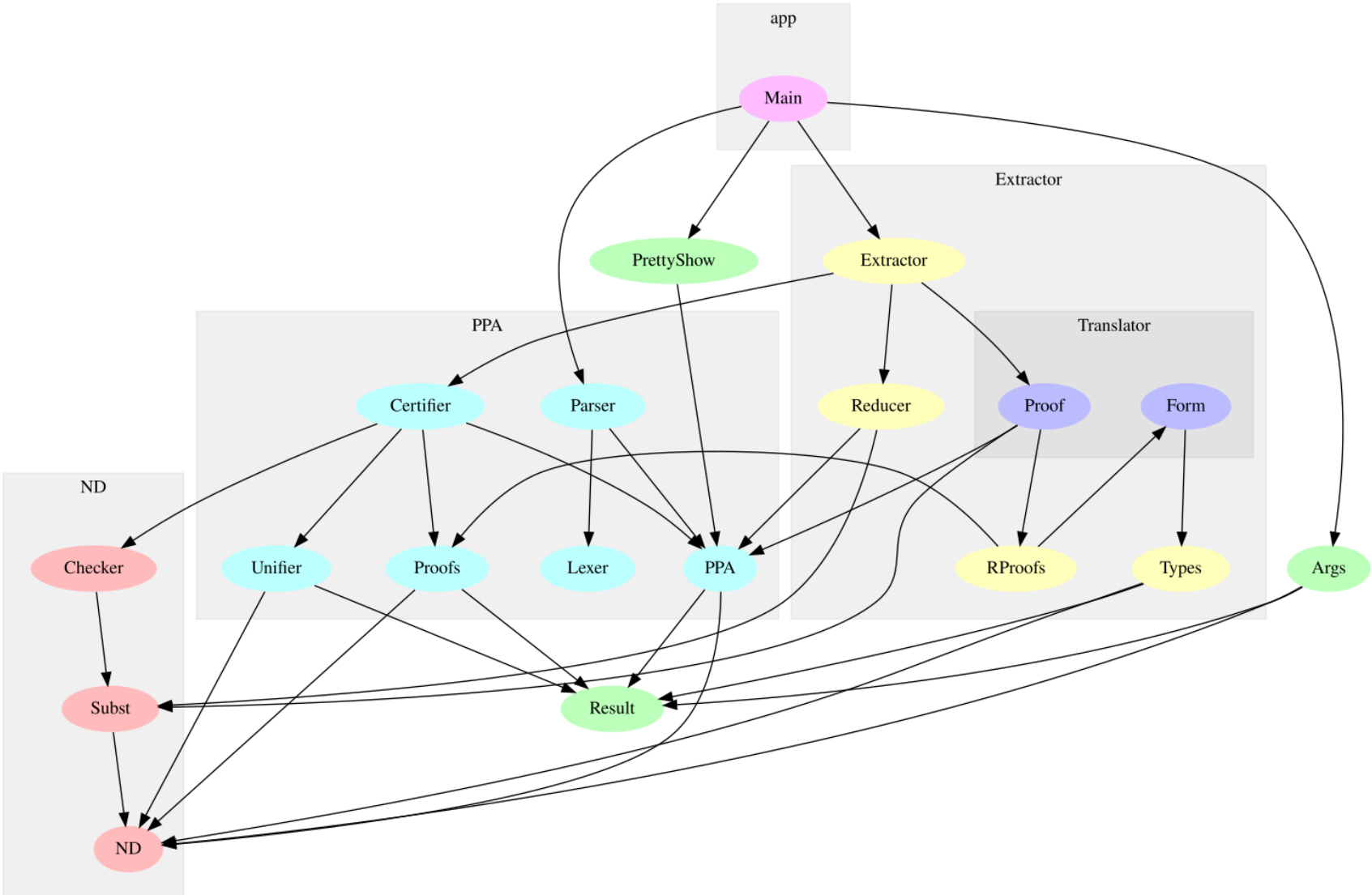
Programa con falla de extracción

```
1  axiom ax_1: roba(tuco) | mata(tuco)
2  axiom ax_2: forall X . roba(X) -> criminal(X)
3  axiom ax_3: forall X . mata(X) -> criminal(X)
4
5  theorem t: exists X . criminal(X)
6  proof
7     take X := tuco
8     cases by ax_1
9        case roba(tuco)
10          hence criminal(tuco)
11             by ax_2
12
13       case mata(tuco)
14          hence criminal(tuco)
15             by ax_3
16    end
17 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

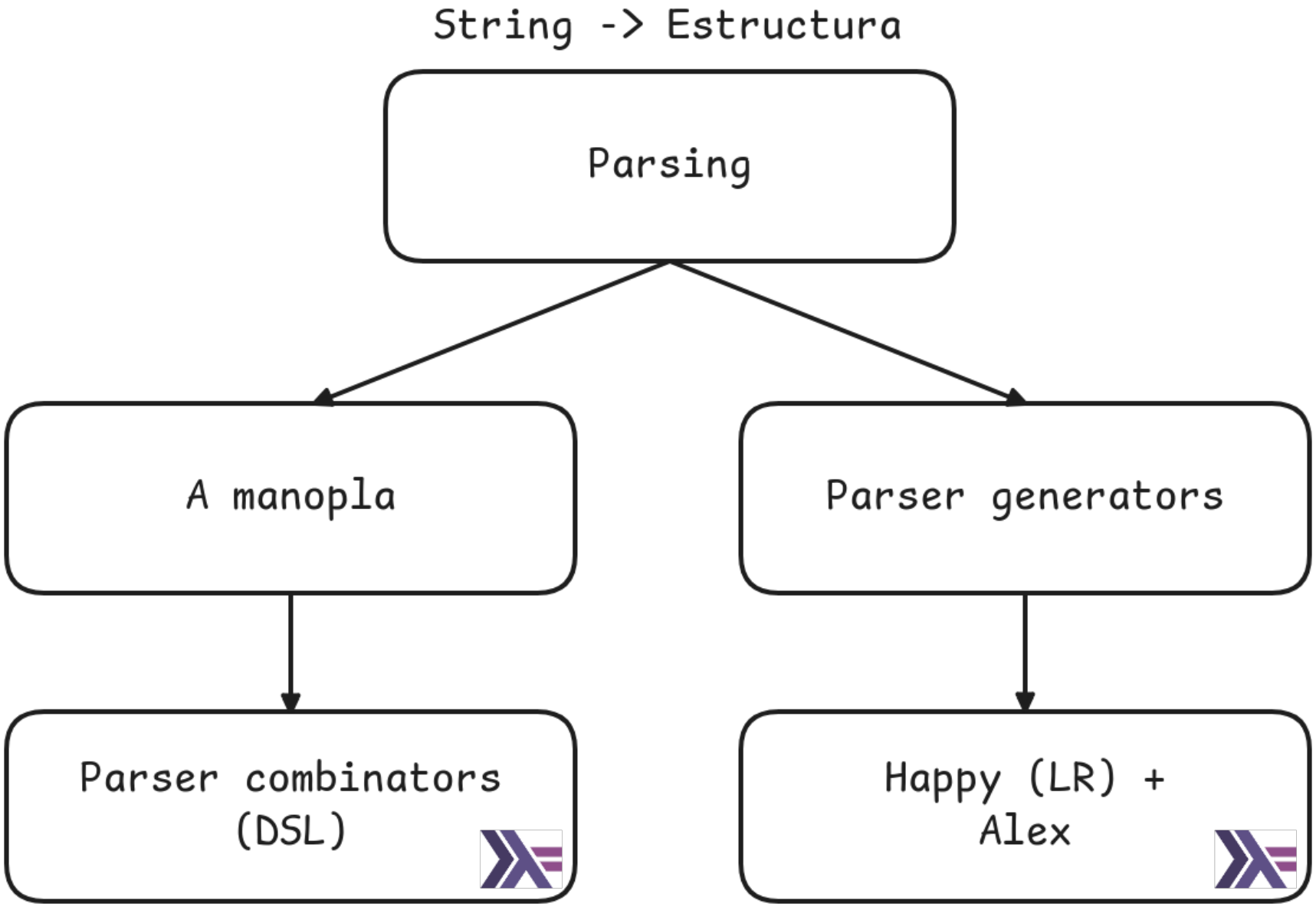
Detalles de implementación

La herramienta ppa



Haskell, 19 módulos con 330 tests

Parser y lexer



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