

PPA

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

Manuel Panichelli

Director: Pablo Barenbaum

Jurado: Verónica Becher, Miguel Pagano (FaMAF, UNC)
Departamento de Computación, FCEyN, UBA

Diciembre 2024

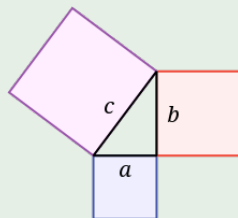
Introducción

Definiciones (Conceptos centrales)

- **Axioma:** Afirmación se declara *siempre válida* (sin demostración).
- **Teorema:** Afirmación que puede ser *demostrada*.
- **Demostración:**
 - *Argumento* que establece que un teorema es cierto.
 - Usa *reglas de inferencia* a partir de *axiomas* y otros teoremas.
 - Enmarcada en un *sistema deductivo*.

Ejemplo (Teorema de Pitágoras)

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



- **Sistema:** Geometría euclidiana.
- **Axioma:** Se puede dibujar una línea recta entre dos puntos.

- Herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos:
 - Formalización de teoremas matemáticos.
 - Verificación de programas.

¹Terence Tao - Machine Assisted Proof

- Herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos:
 - Formalización de teoremas matemáticos.
 - Verificación de programas.
- Ventajas:¹
 - Facilitan la colaboración a gran escala (mediante la confianza en el asistente).

¹Terence Tao - Machine Assisted Proof

- Herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos:
 - 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* (filtrando alucinaciones automáticamente).

¹Terence Tao - Machine Assisted Proof

Constructivos



Coq
(Type theory)



(Type theory)

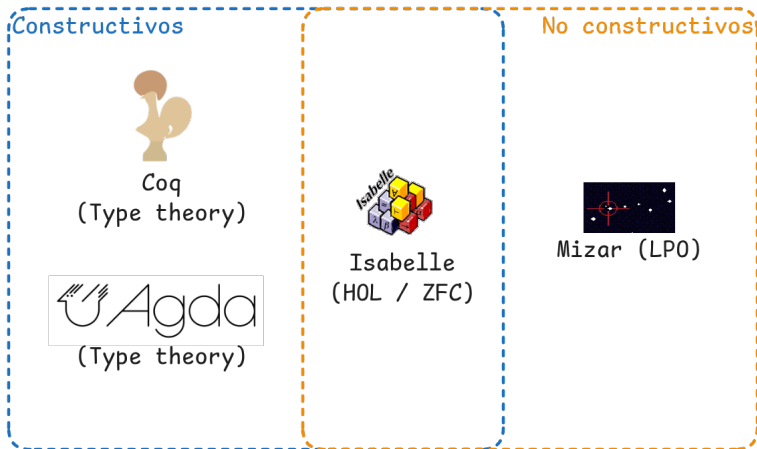


Isabelle
(HOL / ZFC)

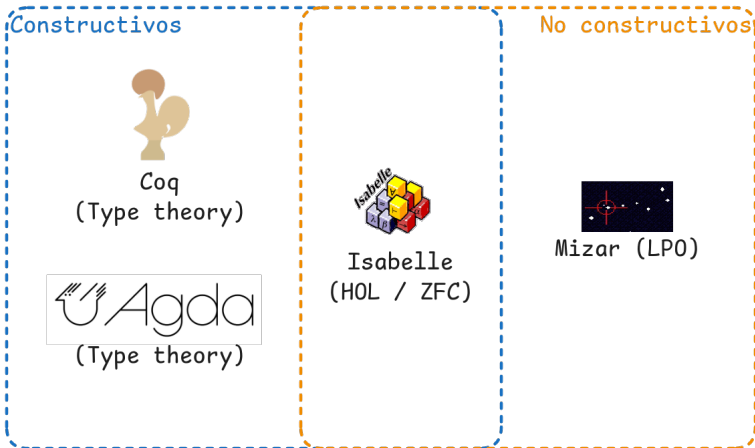
No constructivos



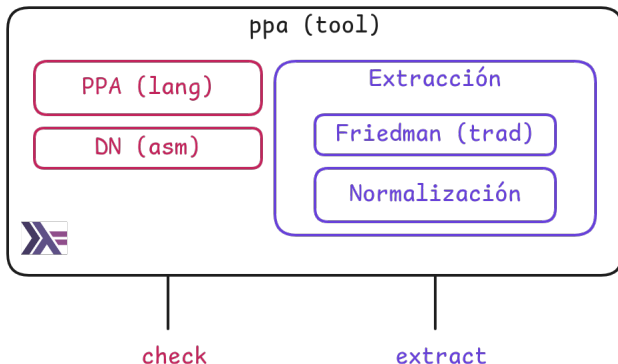
Mizar (LP0)



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



¿Dónde cae ppa?



Diseñamos e implementamos **ppa** (*Pani's Proof Assistant*). Partes:

- El lenguaje **PPA** para escribir demostraciones.
- Mecanismo de **extracción de testigos** de demostraciones no constructivas (**aporte principal**).

Definición (Axiomas)

- 1 Los alumnos que faltan a los exámenes, los reprueban.
- 2 Si se reprueba un final, se recursa la materia.

Ejemplo representación de demostraciones

Definición (Axiomas)

- 1 Los alumnos que faltan a los exámenes, los reprueban.
- 2 Si se reprueba un final, se recursa la materia.

Teorema

Si un alumno falta al final de una materia, entonces la recursa

Demostración.

- Asumo que falta. Quiero ver que recursa.
- Por (1), sabemos que si falta, entonces reprueba. Por lo tanto reprobó.
- Por (2), sabemos que si reprueba, entonces recursa. Por lo tanto recursó.



Ejemplo representación de demostraciones

Definición (Axiomas)

- 1 Los alumnos que faltan a los exámenes, los reprueban.
- 2 Si se reprueba un final, se recursa la materia.

Teorema

Si un alumno falta al final de una materia, entonces la recursa

Demostración.

- Asumo que falta. Quiero ver que recursa.
- Por (1), sabemos que si falta, entonces reprueba. Por lo tanto reprobó.
- Por (2), sabemos que si reprueba, entonces recursa. Por lo tanto recursó.



Problema: Es poco precisa. No se puede representar formalmente.

Deducción natural (DN)

Definición (Términos)

Los términos están dados por la gramática:

$$\begin{array}{ll} t ::= x & \text{(variables)} \\ \quad | f(t_1, \dots, t_n) & \text{(funciones)} \end{array}$$

Definición (Fórmulas)

Las fórmulas están dadas por la gramática:

$$\begin{array}{ll} A, B ::= p(t_1, \dots, t_n) & \text{(predicados)} \\ \quad | \perp \mid \top & \text{(falso y verdadero)} \\ \quad | A \wedge B \mid A \vee B & \text{(conjunción y disyunción)} \\ \quad | A \rightarrow B \mid \neg A & \text{(implicación y negación)} \\ \quad | \forall x.A \mid \exists x.A & \text{(cuantificador universal y existencial)} \end{array}$$

Reglas de inferencia

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

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

Definición (Reglas de inferencia)

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \text{I} \rightarrow$$

Modus ponens:

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \text{E} \rightarrow$$

Donde Γ es un **contexto de demostración** y \vdash la **relación de derivabilidad** o consecuencia.

Ejemplo (Teorema en DN)

Notamos:

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

Axiomas $F \rightarrow X$ y $X \rightarrow R$. Afirmamos $F \rightarrow R$.

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

Axiomas $F \rightarrow X$ y $X \rightarrow R$. Afirmamos $F \rightarrow R$.

Ejemplo (Demostración en DN)

$$\frac{\Gamma = (F \rightarrow X), (X \rightarrow R), F \vdash R}{(F \rightarrow X), (X \rightarrow R) \vdash F \rightarrow R} \mapsto$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

Axiomas $F \rightarrow X$ y $X \rightarrow R$. Afirmamos $F \rightarrow R$.

Ejemplo (Demostración en DN)

$$\frac{\frac{\Gamma = (F \rightarrow X), (X \rightarrow R), F \vdash R}{(F \rightarrow X), (X \rightarrow R) \vdash F \rightarrow R} \text{I} \rightarrow}{\vdash (F \rightarrow X), (X \rightarrow R), F \vdash R} \text{E} \rightarrow$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

Axiomas $F \rightarrow X$ y $X \rightarrow R$. Afirmamos $F \rightarrow R$.

Ejemplo (Demostración en DN)

$$\frac{\frac{\Gamma \vdash X \rightarrow R}{\Gamma = (F \rightarrow X), (X \rightarrow R), F \vdash R} \quad \Gamma \vdash X}{(F \rightarrow X), (X \rightarrow R) \vdash F \rightarrow R} \begin{matrix} E \rightarrow \\ I \rightarrow \end{matrix}$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

Axiomas $F \rightarrow X$ y $X \rightarrow R$. Afirmamos $F \rightarrow R$.

Ejemplo (Demostración en DN)

$$\frac{\frac{\overline{\Gamma \vdash X \rightarrow R} \text{ Ax} \quad \Gamma \vdash X}{\Gamma = (F \rightarrow X), (X \rightarrow R), F \vdash R} E \rightarrow}{(F \rightarrow X), (X \rightarrow R) \vdash F \rightarrow R} I \rightarrow$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

Axiomas $F \rightarrow X$ y $X \rightarrow R$. Afirmamos $F \rightarrow R$.

Ejemplo (Demostración en DN)

$$\frac{\frac{\frac{\Gamma \vdash X \rightarrow R}{\Gamma \vdash X \rightarrow R} \text{Ax} \quad \frac{\frac{\Gamma \vdash F \rightarrow X}{\Gamma \vdash F \rightarrow X} \text{Ax} \quad \frac{\Gamma \vdash F}{\Gamma \vdash F} \text{Ax}}{\Gamma \vdash X} \text{E} \rightarrow}{\frac{\Gamma \vdash (F \rightarrow X), (X \rightarrow R), F \vdash R}{(F \rightarrow X), (X \rightarrow R) \vdash F \rightarrow R} \text{E} \rightarrow} \text{I} \rightarrow$$

Otras reglas de inferencia

- $I\neg$, $E\neg$, $I\wedge$, $E\wedge_1$, $E\wedge_2$
- $I\vee_1$, $I\vee_2$, $E\vee$
- $I\forall$, $E\forall$, $I\exists$, $E\exists$
- $E\perp$, IT , LEM

Otras reglas de inferencia

- $I\neg$, $E\neg$, $I\wedge$, $E\wedge_1$, $E\wedge_2$
- $I\vee_1$, $I\vee_2$, $E\vee$
- $I\forall$, $E\forall$, $I\exists$, $E\exists$
- $E\perp$, IT , LEM

Reglas admisibles

No tenemos regla por ej. para *modus tollens*:

$$((A \rightarrow B) \wedge \neg B) \rightarrow \neg A$$

- Queremos un sistema lógico **minimal**, sin reglas **admisibles**.
- Se implementan como funciones o *macros*.

PPA

Mizar \rightsquigarrow Isar (Isabelle) \rightsquigarrow *Mathematical Vernacular*²

Forma *natural* de representar demostraciones matemáticas. Ideas:

²De Freek Wiedijk

Mizar \rightsquigarrow Isar (Isabelle) \rightsquigarrow *Mathematical Vernacular*²

Forma *natural* de representar demostraciones matemáticas. Ideas:

- Deducción natural en estilo de *Fitch*.

²De Freek Wiedijk

Mizar \rightsquigarrow Isar (Isabelle) \rightsquigarrow *Mathematical Vernacular*²

Forma *natural* de representar demostraciones matemáticas. Ideas:

- Deducción natural en estilo de *Fitch*.
- Reglas de inferencia *declarativas*: Afirmar

$$A_1, \dots, A_n \vdash A$$

sin tener que demostrarlo a mano (automático).

²De Freek Wiedijk

Mizar \rightsquigarrow Isar (Isabelle) \rightsquigarrow *Mathematical Vernacular*²

Forma *natural* de representar demostraciones matemáticas. Ideas:

- Deducción natural en estilo de *Fitch*.
- Reglas de inferencia *declarativas*: Afirmar

$$A_1, \dots, A_n \vdash A$$

sin tener que demostrarlo a mano (automático).

- Sintaxis similar a un lenguaje de programación

²De Freek Wiedijk

Lenguaje PPA, inspirado en el Mathematical Vernacular.

Ejemplo demostración

```
1 axiom "ax1": forall A. forall E.  
2   falta(A, E) -> reprueba(A, E)  
3 axiom "ax2": forall A. forall M.  
4   reprueba(A, final(M)) -> recursa(A, M)
```

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

Ejemplo demostración

```
1 axiom "ax1": forall A. forall E.  
2   falta(A, E) -> reprueba(A, E)  
3 axiom "ax2": forall A. forall M.  
4   reprueba(A, final(M)) -> recursa(A, M)  
5  
6 theorem "falta_entonces_recura":  
7   forall A. forall M. falta(A, final(M)) -> recursa(A, M)  
8 proof
```


Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

Ejemplo demostración

```
1  axiom "ax1": forall A. forall E.  
2      falta(A, E) -> reprueba(A, E)  
3  axiom "ax2": forall A. forall M.  
4      reprueba(A, final(M)) -> recursa(A, M)  
5  
6  theorem "falta_entonces_recura":  
7      forall A. forall M. falta(A, final(M)) -> recursa(A, M)  
8  proof  
9      let A  
10     let M
```

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

Ejemplo demostración

```
1  axiom "ax1": forall A. forall E.  
2      falta(A, E) -> reprueba(A, E)  
3  axiom "ax2": forall A. forall M.  
4      reprueba(A, final(M)) -> recursa(A, M)  
5  
6  theorem "falta_entonces_recura":  
7      forall A. forall M. falta(A, final(M)) -> recursa(A, M)  
8  proof  
9      let A  
10     let M  
11     suppose "h1": falta(A, final(M))
```

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

Ejemplo demostración

```
1  axiom "ax1": forall A. forall E.  
2      falta(A, E) -> reprueba(A, E)  
3  axiom "ax2": forall A. forall M.  
4      reprueba(A, final(M)) -> recursa(A, M)  
5  
6  theorem "falta_entonces_recura":  
7      forall A. forall M. falta(A, final(M)) -> recursa(A, M)  
8  proof  
9      let A  
10     let M  
11     suppose "h1": falta(A, final(M))  
12     have "h2": reprueba(A, final(M)) by "ax1", "h1"
```

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

Ejemplo demostración

```

1  axiom "ax1": forall A. forall E.
2      falta(A, E) -> reprueba(A, E)
3  axiom "ax2": forall A. forall M.
4      reprueba(A, final(M)) -> recursa(A, M)
5
6  theorem "falta_entonces_recura":
7      forall A. forall M. falta(A, final(M)) -> recursa(A, M)
8  proof
9      let A
10     let M
11     suppose "h1": falta(A, final(M))
12     have "h2": reprueba(A, final(M)) by "ax1", "h1"
13     thus recursa(A, M) by "ax2", "h2"
14 end

```

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
IT	by
$E\perp$	by

Comandos y reglas de inferencia

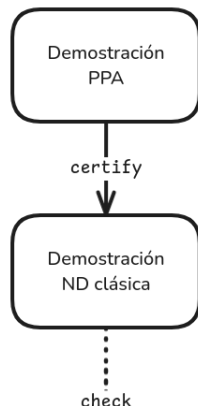
Regla	Comando
LEM	cases
Ax	by
$I\exists$	take
$E\exists$	consider
$I\forall$	let
$E\forall$	by
$I\vee_1$	by
$I\vee_2$	by
$E\vee$	cases

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
$E\perp$	by

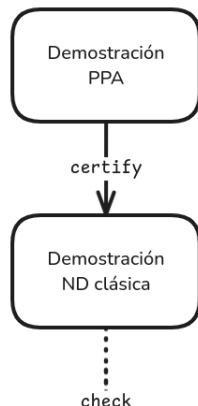
Adicionales:

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

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



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



Certificado de demostraciones

El procedimiento de 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
```

$$\frac{\frac{h : p(v) \vdash p(v)}{h : p(v) \vdash \exists x.p(X)} \text{Ax}_h \mid \exists}{\vdash p(v) \rightarrow \exists x.p(X)} \mid \rightarrow_h$$

Certificado de demostraciones

El procedimiento de 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
```

$$\frac{\frac{\frac{}{h : p(v) \vdash p(v)}{h : p(v) \vdash \exists x.p(X)} \text{Ax}_h}{\vdash p(v) \rightarrow \exists x.p(X)} \text{I}\exists}{\vdash p(v) \rightarrow \exists x.p(X)} \text{I}\rightarrow_h$$

PImpI

```
{ hypAntecedent = "h"  
  , proofConsequent =
```

PExistsI

```
{ inst = TFun "v" []  
  , proofFormWithInst = PAx "h"  
}
```

```
}
```

Ejemplo sin cuantificadores (1/4)

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 (1/4)

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
```

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

$$\Gamma \vdash ((a \rightarrow b) \wedge a) \rightarrow b$$

tomando las hipótesis ax1, ax2 del **contexto**.

- ② **Razonamos por el absurdo:** Negamos la fórmula y buscamos una contradicción.

$$\Gamma, \neg[(a \rightarrow b) \wedge a \rightarrow b] \vdash \perp$$

Ejemplo sin cuantificadores (2/4)

- ② **Razonamos por el absurdo:** Negamos la fórmula y buscamos una contradicción.

$$\Gamma, \neg[((a \rightarrow b) \wedge a) \rightarrow b] \vdash \perp$$

Definición (Eliminación de doble negación)

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

Ejemplo sin cuantificadores (2/3)

3 La convertimos a **DNF** (*Forma Normal Disyuntiva*)

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

Conversión a DNF - Reglas admisibles

Reglas admisibles para conversión a DNF

Pasos base

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

$$\neg\perp \dashv\vdash \top$$

$$\neg\top \dashv\vdash \perp$$

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

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

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

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

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

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

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

Pasos recursivos de congruencia

(con $A \dashv\vdash A'$, $B \dashv\vdash B'$)

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

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

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

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

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

30 demostraciones

Ejemplo sin cuantificadores (3/3)

- 4 Demostramos una **contradicción** refutando cada cláusula

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



Definición (Reglas de inferencia)

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \text{E}\vee$$

$$\frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \perp} \text{E}\neg$$

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

Definición (Eliminación de universal)

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

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x.(p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x.(p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x.(p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

- ❸ No es refutable. $\exists \forall$ con $x := u$ una **meta-variable** fresca.

$$(p(u) \rightarrow q(u)) \wedge p(k) \wedge \neg q(k)$$

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

- ❸ No es refutable. $\exists \forall$ con $x := u$ una **meta-variable** fresca.

$$(p(u) \rightarrow q(u)) \wedge p(k) \wedge \neg q(k)$$

- ❹ Re-convertimos a DNF

$$(\neg p(u) \wedge p(k) \wedge \neg q(k)) \vee (q(u) \wedge p(k) \wedge \neg q(k))$$

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

- ❸ No es refutable. $\exists \forall$ con $x := u$ una **meta-variable** fresca.

$$(p(u) \rightarrow q(u)) \wedge p(k) \wedge \neg q(k)$$

- ❹ Re-convertimos a DNF

$$(\neg p(u) \wedge p(k) \wedge \neg q(k)) \vee (q(u) \wedge p(k) \wedge \neg q(k))$$

- ❺ Refutamos cada cláusula (**unificando**).

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

- ❸ No es refutable. $\exists \forall$ con $x := u$ una **meta-variable** fresca.

$$(p(u) \rightarrow q(u)) \wedge p(k) \wedge \neg q(k)$$

- ❹ Re-convertimos a DNF

$$(\neg p(u) \wedge p(k) \wedge \neg q(k)) \vee (q(u) \wedge p(k) \wedge \neg q(k))$$

- ❺ Refutamos cada cláusula (**unificando**).

- $\neg p(u) \wedge p(k) \wedge \neg q(k)$ tenemos $p(u) \doteq p(k)$ con $\{u := k\}$

Ejemplo con cuantificadores

- ❶ Supongamos que tenemos que resolver siguiente implicación

$$\begin{aligned} & \left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \\ & \equiv \neg \left[\left((\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \right) \rightarrow q(k) \right] \end{aligned}$$

- ❷ Convertimos a DNF (\forall es *opaco*)

$$(\forall x. (p(x) \rightarrow q(x))) \wedge p(k) \wedge \neg q(k)$$

- ❸ No es refutable. $\exists \forall$ con $x := u$ una **meta-variable** fresca.

$$(p(u) \rightarrow q(u)) \wedge p(k) \wedge \neg q(k)$$

- ❹ Re-convertimos a DNF

$$(\neg p(u) \wedge p(k) \wedge \neg q(k)) \vee (q(u) \wedge p(k) \wedge \neg q(k))$$

- ❺ Refutamos cada cláusula (**unificando**).

- $\neg p(u) \wedge p(k) \wedge \neg q(k)$ tenemos $p(u) \doteq p(k)$ con $\{u := k\}$
- $q(u) \wedge p(k) \wedge \neg q(k)$ tenemos $q(u) \doteq q(k)$ con $\{u := k\}$

Teniendo $\Gamma = \{h_1 : B_1, \dots, h_n : B_n\}$, para **thus** A **by** h_1, \dots, h_n :

- ➊ Buscamos las hipótesis en el contexto y armamos la fórmula
- ➋ **Razonamos por el absurdo**: Asumimos negación y demostramos una contradicción (\perp)
- ➌ Convertimos la fórmula a **DNF**.
- ➍ Demostramos una **contradicción** refutando cada cláusula
 - Contiene \perp o dos fórmulas opuestas ($a, \neg a$),
 - Eliminando universales **consecutivos** ($\neg p(k, t), \forall x. \forall y. p(x, y)$)

Teniendo $\Gamma = \{h_1 : B_1, \dots, h_n : B_n\}$, para **thus** A **by** h_1, \dots, h_n :

- ➊ Buscamos las hipótesis en el contexto y armamos la fórmula
- ➋ **Razonamos por el absurdo**: Asumimos negación y demostramos una contradicción (\perp)
- ➌ Convertimos la fórmula a **DNF**.
- ➍ Demostramos una **contradicción** refutando cada cláusula
 - Contiene \perp o dos fórmulas opuestas ($a, \neg a$),
 - Eliminando universales **consecutivos** ($\neg p(k, t), \forall x. \forall y. p(x, y)$)

Alcance

Completo para lógica proposicional y **heurístico** para primer orden.
(aceptable, la validez de LPO es indecidible por el *Teorema de Church*).

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

Descarga de conjunciones

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

Problema:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \text{I}\wedge$$

Descarga

```
1  axiom "ax1": a & e
2  axiom "ax2": b & c
3  axiom "ax3": d
4  theorem "and discharge":
5      (a & b) & ((c & d) & e)
6  proof
7      thus a & e by "ax1"
8      thus d by "ax3"
9      thus b & c by "ax2"
10 end
```

Descarga de conjunciones

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

- Reordena la conjunción (tratando como conjunto).

Descarga

$$(a \wedge e) \wedge (b \wedge c \wedge d)$$

```
1  axiom "ax1": a & e
2  axiom "ax2": b & c
3  axiom "ax3": d
4  theorem "and discharge":
5      (a & b) & ((c & d) & e)
6  proof
7      thus a & e by "ax1"
8      thus d by "ax3"
9      thus b & c by "ax2"
10 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 "ax1": a & e
2 axiom "ax2": b & c
3 axiom "ax3": d
4 theorem "and discharge":
5     (a & b) & ((c & d) & e)
6 proof
7     thus a & e by "ax1"
8     thus d by "ax3"
9     thus b & c by "ax2"
10 end
```

- Reordena la conjunción (tratando como conjunto).

$$(a \wedge e) \wedge (b \wedge c \wedge d)$$

- Demuestra la equivalencia con **equivalently** (usa mismo solver que el **by**)

$$(a \wedge e) \wedge (b \wedge c \wedge d) \\ \rightarrow (a \wedge b) \wedge ((c \wedge d) \wedge e)$$

Descarga de conjunciones

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

Descarga

```
1 axiom "ax1": a & e
2 axiom "ax2": b & c
3 axiom "ax3": d
4 theorem "and discharge":
5     (a & b) & ((c & d) & e)
6 proof
7     thus a & e by "ax1"
8     thus d by "ax3"
9     thus b & c by "ax2"
10 end
```

- Reordena la conjunción (tratando como conjunto).

$$(a \wedge e) \wedge (b \wedge c \wedge d)$$

- Demuestra la equivalencia con **equivalently** (usa mismo solver que el **by**)

$$(a \wedge e) \wedge (b \wedge c \wedge d) \\ \rightarrow (a \wedge b) \wedge ((c \wedge d) \wedge e)$$

- **by** es completo para proposicional \Rightarrow resuelve asociatividad, conmutatividad e idempotencia (repetidos)

Extracción de testigos

Extracción simple

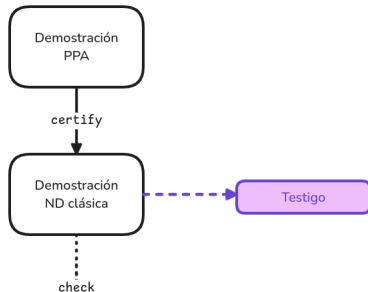
```
1 axiom ax: es_bajo(juan)
2 theorem t: exists Alguien . es_bajo(Alguien)
3 proof
4   take Alguien := juan
5   thus es_bajo(juan) by "ax"
6 end
```

Extracción simple

```
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
```

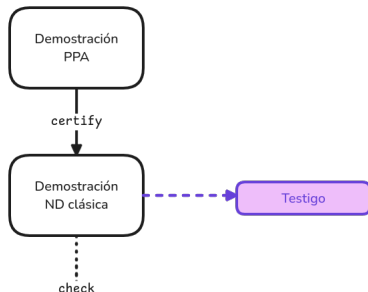
- No es tan simple como buscar un **take** (\exists).
- Útil: Para demostraciones *indirectas* o sin **take** .

- Buscamos un mecanismo general para extraer testigos de los certificados.



- Buscamos un mecanismo general para extraer testigos de los certificados.
- Pero la lógica clásica **no es constructiva**, por LEM:

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



Ejemplo (Fórmula sin demostración constructiva)

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

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

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 razonando por casos con LEM de $C \vee \neg C$

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



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 razonando por casos con LEM de $C \vee \neg C$

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



¡No nos dice explícitamente si $y = 1$ o $y = 0$! No es *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 razonando por casos con LEM de $C \vee \neg C$

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



¡No nos dice explícitamente si $y = 1$ o $y = 0$! No es *constructiva*.

¿Entonces por qué lógica clásica?

- Permite razonar por el absurdo, con $E\neg\neg \equiv \text{LEM}$.
- Existen fórmulas que admiten *solo demostraciones no constructivas* (i.e. clásicas) Ejemplo: $\neg(A \wedge B) \rightarrow \neg A \vee B$ usa LEM.

lógica intuicionista = lógica clásica – LEM

Características:

³Ni principios de razonamiento equivalentes, como $E_{\neg\neg}$

lógica intuicionista = lógica clásica – LEM

Características:

- No tiene LEM³, entonces siempre es constructiva.

³Ni principios de razonamiento equivalentes, como $E_{\neg\neg}$

lógica intuicionista = lógica clásica – LEM

Características:

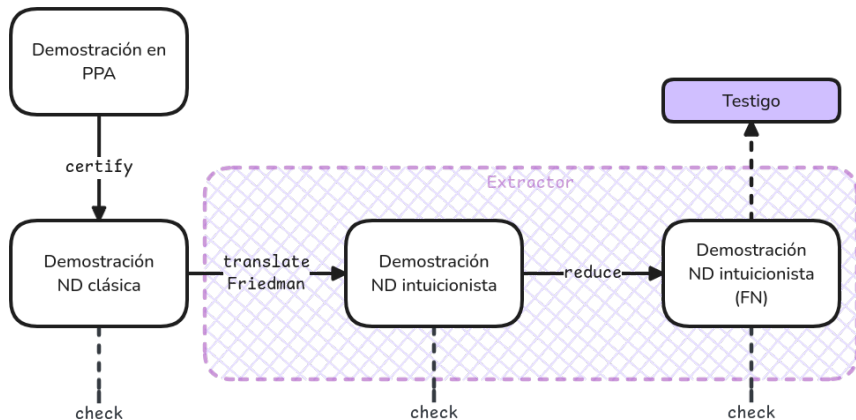
- No tiene LEM³, entonces siempre es constructiva.
- Proceso de normalización con *forma normal* buena, una demostración de un \exists comienza con $I\exists$

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

- Siempre permite extracción de testigos.

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

Estrategia de extracción indirecta



Traducción de Friedman

Traducción de doble negación

- Queremos *embeber* lógica clásica a intuicionista (no son equivalentes)
- Traducción de **doble negación**: método general.
- **Intuición**: “agregar una doble negación a todo”.
- En clásica son equivalentes ($E_{\neg\neg} \equiv \text{LEM}$) pero en intuicionista es más débil.

Traducción de doble negación

- Queremos *embeber* lógica clásica a intuicionista (no son equivalentes)
- Traducción de **doble negación**: método general.
- **Intuición**: “agregar una doble negación a todo”.
- En clásica son equivalentes ($E_{\neg\neg} \equiv \text{LEM}$) pero en intuicionista es más débil.

Teorema

$$\begin{array}{ccc} \Pi & & \Pi^N \\ \Gamma \vdash_c A & \rightsquigarrow & \Gamma^N \vdash_I A^N \end{array}$$

Traducción de doble negación

- Queremos *embeber* lógica clásica a intuicionista (no son equivalentes)
- Traducción de **doble negación**: método general.
- **Intuición**: “agregar una doble negación a todo”.
- En clásica son equivalentes ($E_{\neg\neg} \equiv \text{LEM}$) pero en intuicionista es más débil.

Teorema

$$\frac{\Pi}{\Gamma \vdash_c A} \rightsquigarrow \frac{\Pi^N}{\Gamma^N \vdash_I A^N}$$

Problema: Necesitamos la misma fórmula

$$(\exists x.A)^N = \neg \forall x. \neg \neg A$$

Teorema (Traducción de Friedman)

Si tenemos

$$\Pi \triangleright \Gamma \vdash_C \psi$$

Podemos generar una nueva demostración Σ tal que

$$\Sigma \triangleright \Gamma \vdash_I \psi$$

Si ψ y Γ cumplen con ciertas restricciones.

Traducción de doble negación relativizada

Definición (Negación relativizada)

Podemos ver a $\neg A \equiv A \rightarrow \perp$. Definimos $\neg_R A \equiv A \rightarrow R$

Traducción de doble negación relativizada

Definición (Negación relativizada)

Podemos ver a $\neg A \equiv A \rightarrow \perp$. Definimos $\neg_R A \equiv A \rightarrow R$

Definición (Traducción de doble negación relativizada)

$$\perp^{\neg\neg} = R$$

$$A^{\neg\neg} = \neg_R \neg_R A \quad \text{con } A \text{ atómica}$$

$$(\neg A)^{\neg\neg} = \neg_R A^{\neg\neg}$$

$$(A \wedge B)^{\neg\neg} = A^{\neg\neg} \wedge B^{\neg\neg}$$

$$(A \vee B)^{\neg\neg} = \neg_R (\neg_R A^{\neg\neg} \wedge \neg_R B^{\neg\neg})$$

$$(A \rightarrow B)^{\neg\neg} = A^{\neg\neg} \rightarrow B^{\neg\neg}$$

$$(\forall x. A)^{\neg\neg} = \forall x. A^{\neg\neg}$$

$$(\exists x. A)^{\neg\neg} = \neg_R \forall x. \neg_R A^{\neg\neg}$$

Funcionamiento de traducción de Friedman

Partiendo de

$$\Pi \triangleright \Gamma \vdash_C \psi$$

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

Funcionamiento de traducción de Friedman

Partiendo de

$$\Pi \triangleright \Gamma \vdash_C \psi$$

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

- 1 Aplicar traducción de doble negación relativizada (recursivamente a fórmula y demostración) tomando " $R = \psi$ ".

$$\Pi^{\neg\neg} \triangleright \Gamma^{\neg\neg} \vdash_I \psi^{\neg\neg}.$$

Funcionamiento de traducción de Friedman

Partiendo de

$$\Pi \triangleright \Gamma \vdash_C \psi$$

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

- 1 Aplicar traducción de doble negación relativizada (recursivamente a fórmula y demostración) tomando " $R = \psi$ ".

$$\Pi^{\neg\neg} \triangleright \Gamma^{\neg\neg} \vdash_I \psi^{\neg\neg}.$$

- 2 Usarla para demostrar la fórmula original.

Restricción: ψ debe ser Π_2 con φ **conjuntiva**.

$$\Sigma \triangleright \Gamma^{\neg\neg} \vdash_I \forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n).$$

Funcionamiento de traducción de Friedman

Partiendo de

$$\Pi \triangleright \Gamma \vdash_C \psi$$

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

- 1 Aplicar traducción de doble negación relativizada (recursivamente a fórmula y demostración) tomando " $R = \psi$ ".

$$\Pi^{\neg\neg} \triangleright \Gamma^{\neg\neg} \vdash_I \psi^{\neg\neg}.$$

- 2 Usarla para demostrar la fórmula original.

Restricción: ψ debe ser Π_2 con φ **conjuntiva**.

$$\Sigma \triangleright \Gamma^{\neg\neg} \vdash_I \forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n).$$

- 3 Mantener el contexto (reemplazando Ax por $A \vdash_I A^{\neg\neg}$)

Restricción: Axiomas (Γ) deben ser **F-fórmulas**.

$$\Sigma' \triangleright \Gamma \vdash_I \forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n). \quad \square$$

Tipos de fórmulas

Definición (Gramática de fórmulas)

(atómicas) $A ::= \perp \mid \top \mid p(t_1, \dots, t_n)$

(F-fórmulas) $F ::= A$

$\mid F \wedge F \mid F \vee F$

$\mid \forall x.F \mid \exists x.F$

$\mid C \rightarrow F \mid \neg C$

(conjuntivas) $C ::= A \mid C \wedge C$

Lema

Sea F una F-fórmula. Vale $F \vdash_I F^{\neg\neg}$.

Lema

Sea C una fórmula conjuntiva. Vale $\neg_R C \vdash_I \neg_R C^{\neg\neg}$.

Normalización

Motivación: evitar “*desvíos superfluos*”.

Ejemplo

$$\frac{\frac{\frac{\Gamma \vdash A}{Ax_{h_1}} \quad \frac{\Gamma \vdash B}{Ax_{h_2}}}{\Gamma \vdash A \wedge B} I\wedge}{\Gamma \vdash A} E\wedge_1$$

Motivación: evitar “desvíos superfluos”.

Ejemplo

$$\frac{\frac{\frac{\Gamma \vdash A}{Ax_{h_1}} \quad \frac{\Gamma \vdash B}{Ax_{h_2}}}{\Gamma \vdash A \wedge B} I\wedge}{\Gamma \vdash A} E\wedge_1 \rightsquigarrow \frac{\Gamma \vdash A}{Ax_{h_1}}$$

Donde

$$\Gamma = \{h_1 : A, h_2 : B\}$$

Motivación: evitar “desvíos superfluos”.

Ejemplo

$$\frac{\frac{\frac{\Gamma \vdash A}{Ax_{h_1}} \quad \frac{\Gamma \vdash B}{Ax_{h_2}}}{\Gamma \vdash A \wedge B} I\wedge}{\Gamma \vdash A} E\wedge_1 \rightsquigarrow \frac{\Gamma \vdash A}{Ax_{h_1}}$$

Donde

$$\Gamma = \{h_1 : A, h_2 : B\}$$

Definición (Reducción de conjunción)

$$\frac{\frac{\frac{\Pi_1}{\Gamma \vdash A_1} \quad \frac{\Pi_2}{\Gamma \vdash A_2}}{\Gamma \vdash A_1 \wedge A_2} I\wedge}{\Gamma \vdash A_i} E\wedge_i \rightsquigarrow \frac{\Pi_i}{\Gamma \vdash A_i}$$

Definición (Otras reglas)

Además, hay reglas para simplificar

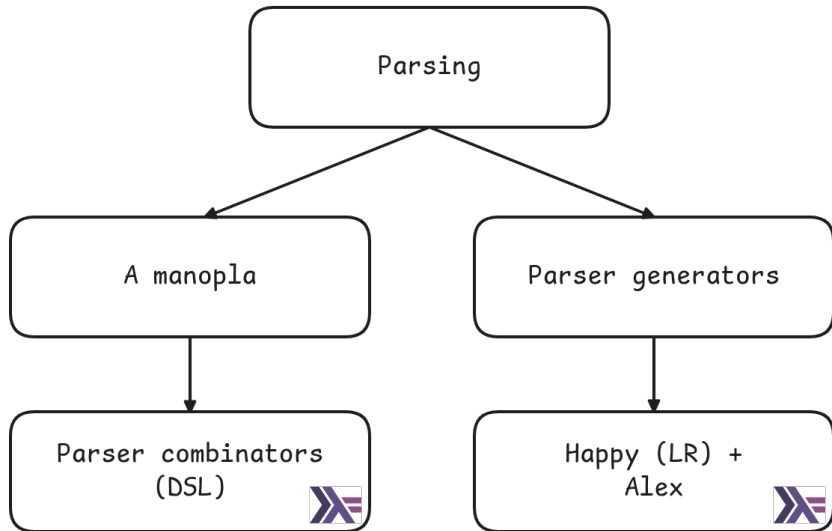
- $E\exists$ con $I\exists$, $E\forall$ con $I\forall$.
- $E\neg$ con $I\neg$, $E\vee$ con $I\vee$.
- $E\rightarrow$ con $I\rightarrow$.

Idea del algoritmo

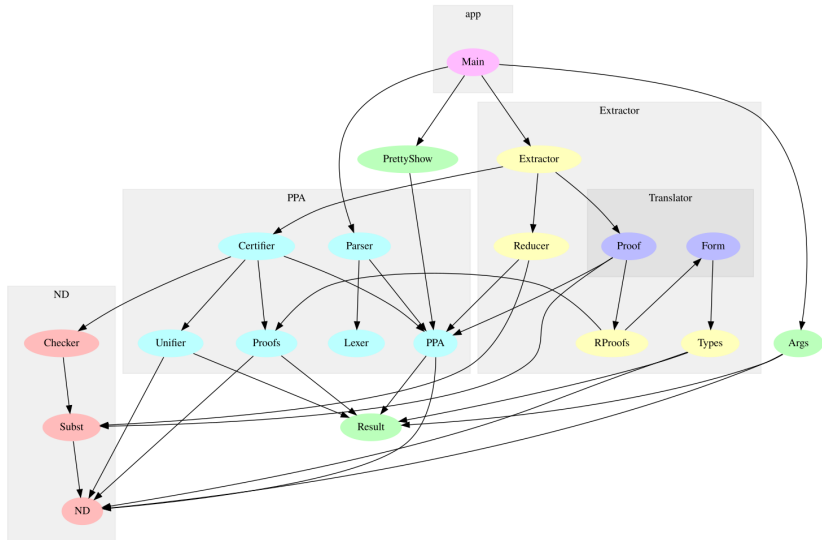
Aplicar las reglas de reducción sucesivamente hasta que no se pueda y la demostración esté en *forma normal*.

Detalles de implementación

String -> Estructura



La herramienta ppa



Haskell, 19 módulos con 330 tests

Conclusiones

Conclusiones - PPA

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.

Conclusiones - PPA

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.

Conclusiones - PPA

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.
 - Demostraciones generadas automáticamente difíciles de *debuggear*.

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.
 - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.
 - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras
 - Permitir importar archivos, implementar biblioteca estándar.

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.
 - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras
 - Permitir importar archivos, implementar biblioteca estándar.
 - Extender PPA con tipos (usando LPO *many-sorted* con géneros).

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.
 - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras
 - Permitir importar archivos, implementar biblioteca estándar.
 - Extender PPA con tipos (usando LPO *many-sorted* con géneros).
 - Modelar de forma nativa inducción (segundo orden) e igualdad.

- Diseñamos e implementamos ppa: un asistente de demostración, junto con el lenguaje PPA.
- Los programas se **certifican** generando demostraciones en *deducción natural*.
- Mecanismo heurístico de demostración automática: **by**.
Extensión: Hacerlo recursivo permitiendo eliminar los universales de más de una hipótesis.
- Desafíos
 - Haskell.
 - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras
 - Permitir importar archivos, implementar 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).

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío:** Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
 - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
 - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.
- Reducción

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
 - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.
- Reducción
 - Solo contempla introducciones y eliminaciones del mismo conectivo.

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
 - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.
- Reducción
 - Solo contempla introducciones y eliminaciones del mismo conectivo.
 - **Incompleta**: no contempla *reducciones permutativas*. *Mejora*: Implementarlas.

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
 - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.
- Reducción
 - Solo contempla introducciones y eliminaciones del mismo conectivo.
 - **Incompleta**: no contempla *reducciones permutativas*. *Mejora*: Implementarlas.
 - Hay algunas demostraciones que no se van a poder reducir a una forma normal útil. Ej: **cases** (EV).

Conclusiones - Extracción de testigos

Implementamos un mecanismo de **extracción de testigos**: composición de traducción de Friedman y reducción de ND intuicionista.

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
 - **Extensión**: A más de un \exists .
 - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
 - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.
- Reducción
 - Solo contempla introducciones y eliminaciones del mismo conectivo.
 - **Incompleta**: no contempla *reducciones permutativas*. *Mejora*: Implementarlas.
 - Hay algunas demostraciones que no se van a poder reducir a una forma normal útil. Ej: **cases** (EV).
 - **Ineficiente**: en cada paso reinicia la búsqueda de todos los focos de evaluación. *Mejora*: Usar una *máquina abstracta*.

¡Gracias!



github.com/mnPanic/tesis