

# PPA

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

Manuel Panichelli

Departamento de Computación, FCEyN, UBA

Diciembre 2024

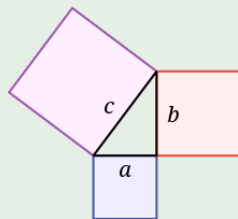
## Introducción

## Definiciones (Conceptos centrales)

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

- Los **asistentes de demostración** son herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos:
  - Formalización de teoremas matemáticos.
  - Verificación de programas.

---

<sup>1</sup>Terence Tao - Machine Assisted Proof

- Los **asistentes de demostración** son herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos:
  - Formalización de teoremas matemáticos.
  - Verificación de programas.
- Ventajas:<sup>1</sup>
  - Facilitan la colaboración a gran escala (mediante la confianza en el asistente).

---

<sup>1</sup>Terence Tao - Machine Assisted Proof

- Los **asistentes de demostración** son herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos:
  - Formalización de teoremas matemáticos.
  - Verificación de programas.
- Ventajas:<sup>1</sup>
  - 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.

---

<sup>1</sup>Terence Tao - Machine Assisted Proof

Constructivos



Coq  
(Type theory)



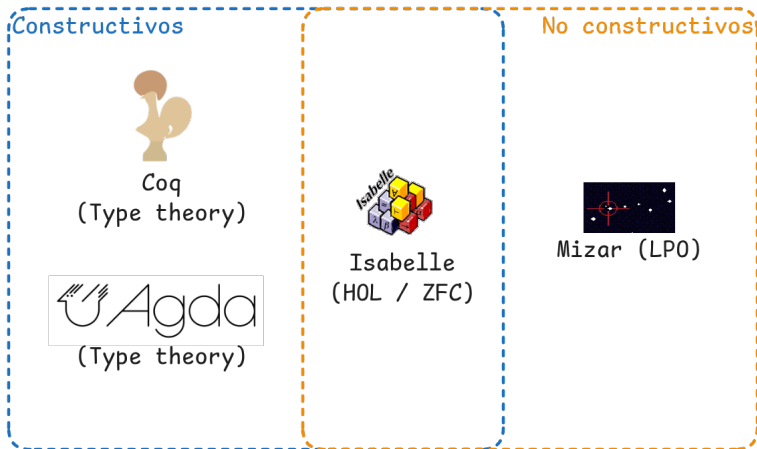
Isabelle  
(HOL / ZFC)

No constructivos

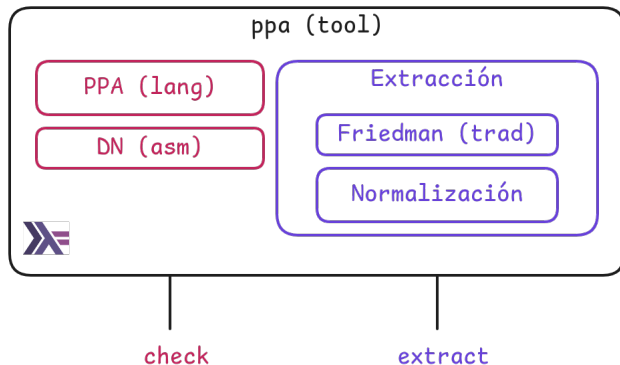


Mizar (LPO)





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



Diseñamos e implementamos en Haskell `ppa` (*Pani's Proof Assistant*). Dos 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:

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

# Reglas de inferencia

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

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

## Definición (Reglas de inferencia)

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

$$\frac{}{\Gamma, A \vdash A} \text{Ax}$$

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \text{E} \rightarrow \quad (\textit{modus ponens})$$

## Definiciones

$\Gamma$  es un contexto de demostración y  $\vdash$  la relación de derivabilidad.



## Ejemplo (Teorema en DN)

Notamos:

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

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:

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

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:

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

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

## Ejemplo (Demostración en DN)

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

# Reglas de inferencia

## Otras reglas de inferencia

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

## Otras reglas de inferencia

- $I\neg$ ,  $E\neg$ ,  $I\wedge$
- $IV_1$ ,  $IV_2$ ,  $EV$
- $I\forall$ ,  $E\forall$ ,  $I\exists$ ,  $E\exists$
- $E\perp$ ,  $IT$ ,  $LEM$

No tenemos regla por ej. para *modus tollens*:  $(A \rightarrow B) \wedge \neg B \rightarrow \neg A$

- Queremos un sistema lógico **minimal**: no agregamos las reglas **admisibles**, derivables a partir de las existentes.
- Se implementan como funciones o *macros*.



# Reglas de inferencia

## Otras reglas de inferencia

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

No tenemos regla por ej. para *modus tollens*:  $(A \rightarrow B) \wedge \neg B \rightarrow \neg A$

- Queremos un sistema lógico **minimal**: no agregamos las reglas **admisibles**, derivables a partir de las existentes.
- Se implementan como funciones o *macros*.

## Alfa equivalencia

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

## Eliminación de universal

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

## Eliminación de universal

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

## Definición (Sustitución)

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

## Eliminación de universal

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

## Definición (Sustitución)

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

## Capturas

Evitamos automáticamente la **captura de variables** (renombrando a fórmula  $\alpha$ -equivalente tq no ocurra)

$$(\forall y.p(\mathbf{x}, y))\{x := y\} \neq \forall y.p(\mathbf{y}, y) \quad (\text{capturada})$$

$$(\forall y.p(\mathbf{x}, y))\{x := y\} = \forall \mathbf{z}.p(\mathbf{y}, \mathbf{z}) \quad (\text{renombrada})$$

PPA

Mizar  $\rightsquigarrow$  Isar (Isabelle)  $\rightsquigarrow$  *Mathematical Vernacular*<sup>2</sup>

Forma *natural* de representar demostraciones matemáticas. Ideas:

---

<sup>2</sup>De Freek Wiedijk

Mizar  $\rightsquigarrow$  Isar (Isabelle)  $\rightsquigarrow$  *Mathematical Vernacular*<sup>2</sup>

Forma *natural* de representar demostraciones matemáticas. Ideas:

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

---

<sup>2</sup>De Freek Wiedijk

Mizar  $\rightsquigarrow$  Isar (Isabelle)  $\rightsquigarrow$  *Mathematical Vernacular*<sup>2</sup>

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

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

sin tener que demostrarlo a mano (automático).

---

<sup>2</sup>De Freek Wiedijk



Mizar  $\rightsquigarrow$  Isar (Isabelle)  $\rightsquigarrow$  *Mathematical Vernacular*<sup>2</sup>

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

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

sin tener que demostrarlo a mano (automático).

- **Sintaxis similar a un lenguaje de programación** en lugar al lenguaje natural.

---

<sup>2</sup>De Freek Wiedijk

*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 "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*. Demostraciones son listas de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla.

### 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": forall A . forall M .  
7   falta(A, final(M)) -> recursa(A, M)  
8 proof
```

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 "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": forall A . forall M .
7      falta(A, final(M)) -> recursa(A, M)
8  proof
9      let A
10     let M
```

*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 "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": 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))
```

*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 "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": 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 "ax1", "falta"

```

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 "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": 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 "ax1", "falta"
13     thus recursa(A, M) by "ax2", "reprueba"
14 end

```

# Comandos y reglas de inferencia

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

Regla	Comando
$I\wedge$	<b>by</b>
$E\wedge_1$	<b>by</b>
$E\wedge_2$	<b>by</b>
$I\rightarrow$	<b>suppose</b>
$E\rightarrow$	<b>by</b>
$I\neg$	<b>suppose</b>
$E\neg$	<b>by</b>
$IT$	<b>by</b>
$E\perp$	<b>by</b>



# Comandos y reglas de inferencia

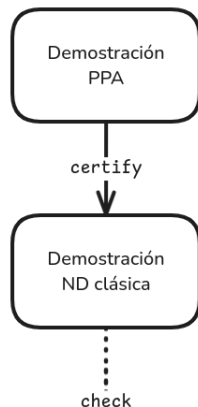
Regla	Comando
LEM	<b>cases</b>
$Ax$	<b>by</b>
$I\exists$	<b>take</b>
$E\exists$	<b>consider</b>
$I\forall$	<b>let</b>
$E\forall$	<b>by</b>
$I\vee_1$	<b>by</b>
$I\vee_2$	<b>by</b>
$E\vee$	<b>cases</b>

Regla	Comando
$I\wedge$	<b>by</b>
$E\wedge_1$	<b>by</b>
$E\wedge_2$	<b>by</b>
$I\rightarrow$	<b>suppose</b>
$E\rightarrow$	<b>by</b>
$I\neg$	<b>suppose</b>
$E\neg$	<b>by</b>
$I\top$	<b>by</b>
$E\perp$	<b>by</b>

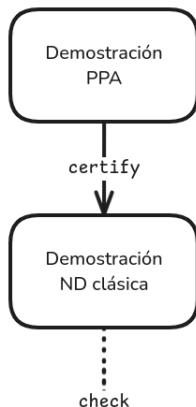
Adicionales:

- **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** (sus demostraciones pueden ser chequeadas por un programa independiente)



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
```

$$\frac{\frac{\overline{h : p(v) \vdash p(v)} \text{ } \textcolor{red}{Ax_h}}{h : p(v) \vdash \exists x.p(X)} \text{ } \exists}{\vdash p(v) \rightarrow \exists x.p(X)} \text{ } \mapsto_h$$

**Figura:** Ejemplo de certificado generado para un programa

## by - El mecanismo principal de demostración

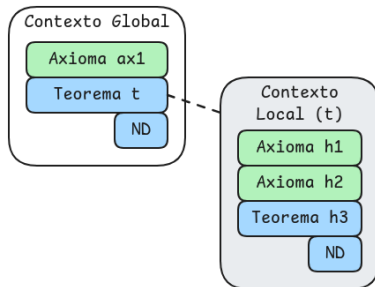
```
thus <form> by <h1>, ..., <hn>  
have <name>: <form> by <h1>, ..., <hn>
```

- Si puede, demuestra **automáticamente** que la fórmula es consecuencia lógica de la justificación.
- Por debajo usa un *solver heurístico* para primer orden.

## by - El mecanismo principal de demostración

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

- Si puede, demuestra **automáticamente** que la fórmula es consecuencia lógica de la justificación.
- Por debajo usa un *solver heurístico* para primer orden.
- Toma las hipótesis del **contexto** local o global: fórmulas asumidas o demostradas.



## 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

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



## Ejemplo sin cuantificadores (2/4)

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

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

## 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)

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

### Definición (Introducción de negación)

$$\frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} I_{\neg}$$

$$\frac{\Gamma, \neg A \vdash \perp}{\Gamma \vdash \neg\neg A} I_{\neg}$$

## Ejemplo sin cuantificadores (2/3)

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

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

# 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*)

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

## 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.  $\text{E}\forall$  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  $\{u := k\}$

# Certificador del by

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
- ➋ **Razonamos por el absurdo**: Asumiendo la negación buscamos una contradicción.
- ➌ Convertimos la fórmula a **DNF**.
- ➍ Buscamos una **contradicción** refutando cada cláusula individualmente. Será refutable si
  - Contiene  $\perp$  o dos fórmulas opuestas  $(a, \neg a)$ ,
  - Eliminando universales **consecutivos** y reiniciando el proceso, se consigue una refutación  $(\neg p(k, t), \forall x. \forall y. p(x, y))$

## Alcance

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

# 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 "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

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 "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

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
```

- Reordena la conjunción (tratando como conjunto).

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

- Demuestra la equivalencia con **equivalently** (por abajo, 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 "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
```

- Reordena la conjunción (tratando como conjunto).

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

- Demuestra la equivalencia con **equivalently** (por abajo, 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 indirecta con instanciación

```
1 axiom padre_es_padre: forall A. es_padre(A, padre(A))
2 theorem todos_tienen_padre: forall Q. exists P. es_padre(Q, P)
3 proof
4   let Q
5   take P := padre(Q)
6   thus es_padre(Q, padre(Q)) by "padre_es_padre"
7 end
```

# Extracción indirecta con instanciación

```
1  axiom padre_es_padre: forall A. es_padre(A, padre(A))
2  theorem todos_tienen_padre: forall Q. exists P. es_padre(Q, P)
3  proof
4      let Q
5      take P := padre(Q)
6      thus es_padre(Q, padre(Q)) by "padre_es_padre"
7  end
8
9  axiom def_abuelo: forall P. forall Q. forall R.
10     (es_padre(P, Q) & es_padre(Q, R)) <=> es_abuelo(P, R)
11  theorem todos_tienen_abuelo: forall A. exists B. es_abuelo(A, B)
```

# Extracción indirecta con instanciación

```
1  axiom padre_es_padre: forall A. es_padre(A, padre(A))
2  theorem todos_tienen_padre: forall Q. exists P. es_padre(Q, P)
3  proof
4      let Q
5      take P := padre(Q)
6      thus es_padre(Q, padre(Q)) by "padre_es_padre"
7  end
8
9  axiom def_abuelo: forall P. forall Q. forall R.
10     (es_padre(P, Q) & es_padre(Q, R)) <-> es_abuelo(P, R)
11  theorem todos_tienen_abuelo: forall A. exists B. es_abuelo(A, B)
12  proof
13      let A
14      consider X st "h1": es_padre(A, X) by "todos_tienen_padre"
15      consider Y st "h2": es_padre(X, Y) by "todos_tienen_padre"
16      take B := Y
17      thus es_abuelo(A, Y) by "h1", "h2", "def_abuelo"
18  end
```

# Extracción indirecta

Para extraer de

```
theorem todos_tienen_abuelo: forall A. exists B. es_abuelo(A, B)
```

Usando ppa,

```
$ ppa extract parientes.ppa \  
  --theorem todos_tienen_abuelo \  
  --terms nacho
```

Running program... OK!

Translating... OK!

Checking translated... OK!

Extracted witness: padre(padre(nacho))

of formula: es\_abuelo(nacho, padre(padre(nacho)))

# 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)
```



# 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  $\neg\forall x.\neg\varphi \equiv \exists x.\varphi$ .
- Sin **take** ( $\exists$ ) explícito, igual podemos extraer el testigo a partir del **theorem** hayAlguienBajo: juan.

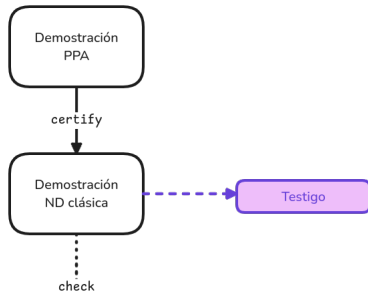
# Extracción por el absurdo

## 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)
```

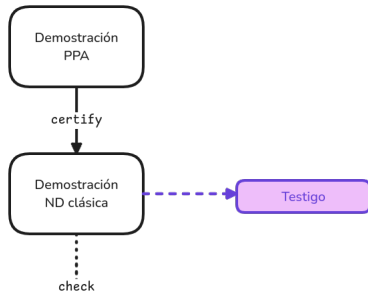
- En general  $\neg\forall x.\neg\varphi \equiv \exists x.\varphi$ .
- Sin **take** ( $\exists$ ) explícito, igual podemos extraer el testigo a partir del **theorem** hayAlguienBajo: juan.
- La implementación no es tan directa como buscar un  $\exists$  en el árbol de la demostración.

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

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



## 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*.

# Demostración no constructiva

## Ejemplo (Fórmula sin demostración constructiva)

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

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

podemos demostrarlo 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:  $\neg(A \wedge B) \rightarrow \neg A \vee B$  solo es válido en lógica clásica.



**lógica intuicionista** = lógica clásica – LEM

Características:

---

<sup>3</sup>Ni principios de razonamiento equivalentes, como  $E_{\neg\neg}$

**lógica intuicionista** = lógica clásica – LEM

Características:

- No tiene LEM<sup>3</sup>, entonces siempre es constructiva.

---

<sup>3</sup>Ni principios de razonamiento equivalentes, como  $E_{\neg\neg}$

**lógica intuicionista** = lógica clásica – LEM

Características:

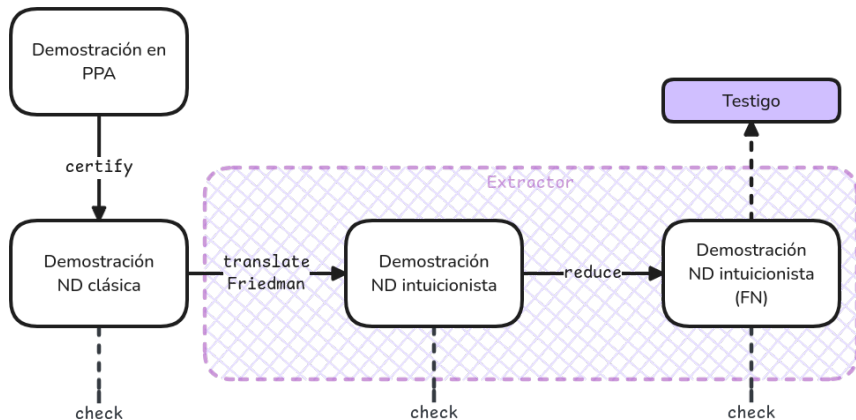
- No tiene LEM<sup>3</sup>, entonces siempre es constructiva.
- Siempre permite hacer extracción de testigos: proceso de normalización con *forma normal* buena, una demostración de un  $\exists$  debería comenzar con  $I\exists$  y de ahí sacás el testigo.

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

---

<sup>3</sup>Ni principios de razonamiento equivalentes, como  $E\neg\neg$

# Estrategia de extracción indirecta



## Normalización

**Motivación:** evitar “desvíos superfluos”.

## Ejemplo

$$\frac{\frac{\frac{\overline{\Gamma \vdash A} \text{Ax}_{h_1}}{\Gamma = h_1 : A, h_2 : B \vdash A \wedge B} \text{I}\wedge}{h_1 : A, h_2 : B \vdash A} \text{E}\wedge_1}{h_1 : A, h_2 : B \vdash A} \text{E}\wedge_1$$

**Motivación:** evitar “desvíos superfluos”.

## Ejemplo

$$\frac{\frac{\frac{\overline{\Gamma \vdash A}^{Ax_{h_1}}}{\Gamma = h_1 : A, h_2 : B \vdash A \wedge B} \quad \frac{\overline{\Gamma \vdash B}^{Ax_{h_2}}}{\Gamma = h_1 : A, h_2 : B \vdash A \wedge B}}{h_1 : A, h_2 : B \vdash A} \begin{matrix} I\wedge \\ E\wedge_1 \end{matrix} \quad \rightsquigarrow \quad \frac{}{h_1 : A, h_2 : B \vdash A}^{Ax_{h_1}}$$

**Motivación:** evitar “desvíos superfluos”.

## Ejemplo

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

## Definición (Reducción de conjunción)

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



# Normalización o reducción

**Motivación:** evitar “desvíos superfluos”.

## Ejemplo

$$\frac{\frac{\frac{}{\Gamma \vdash A} \text{Ax}_{h_1} \quad \frac{}{\Gamma \vdash B} \text{Ax}_{h_2}}{\Gamma = h_1 : A, h_2 : B \vdash A \wedge B} \text{I}\wedge}{h_1 : A, h_2 : B \vdash A} \text{E}\wedge_1 \quad \rightsquigarrow \quad \frac{}{h_1 : A, h_2 : B \vdash A} \text{Ax}_{h_1}$$

## Definición (Reducción de conjunción)

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

**Idea:** Simplificarlos sucesivamente hasta que no haya más y esté en **forma normal**.

## Definición (Normalización de implicación)

$$\frac{\frac{\frac{\Pi_B}{\Gamma, h : A \vdash B}}{\Gamma \vdash A \rightarrow B} \mid \rightarrow_h \quad \frac{\Pi_A}{\Gamma \vdash A} \text{E} \rightarrow}{\Gamma \vdash B} \rightsquigarrow \frac{\Pi_B}{\Gamma \vdash B}$$

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

# Normalización de implicación

## Definición (Normalización de implicación)

$$\frac{\frac{\frac{\Pi_B}{\Gamma, h : A \vdash B} \mid \rightarrow_h}{\Gamma \vdash A \rightarrow B} \quad \frac{\Pi_A}{\Gamma \vdash A} \quad E \rightarrow}{\Gamma \vdash B} \rightsquigarrow \frac{\Pi_B \{h := \Pi_A\}}{\Gamma \vdash B}$$

- Primer idea:  ~~$\Pi_B \triangleright \Gamma \vdash B$~~
- $\Pi_B$  requiere  $h : A$ , agregada por  $\mid \rightarrow_h$
- Correcto: usar  $\Pi_B$ , pero *sustituyendo* todas las ocurrencias de la hipótesis  $h$  por la demostración  $\Pi_A$  (sin capturas).

# Normalización de implicación

## Definición (Normalización de implicación)

$$\frac{\frac{\frac{\Pi_B}{\Gamma, h : A \vdash B} \quad \Gamma \vdash A \rightarrow B}{\Gamma \vdash B} \quad \frac{\Pi_A}{\Gamma \vdash A} \quad E \rightarrow}{\Gamma \vdash B} \rightsquigarrow \frac{\Pi_B \{h := \Pi_A\}}{\Gamma \vdash B}$$

- Primer idea:  ~~$\Pi_B \triangleright \Gamma \vdash B$~~
- $\Pi_B$  requiere  $h : A$ , agregada por  $I \rightarrow_h$
- Correcto: usar  $\Pi_B$ , pero *sustituyendo* todas las ocurrencias de la hipótesis  $h$  por la demostración  $\Pi_A$  (sin capturas).

## 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  $I\vee$ .

# Algoritmo de reducción

- **Algoritmo:** Reducir sucesivamente hasta que sea irreducible.
- **Estrategias de reducción:** en un paso o muchos pasos.
- *Gross-Knuth*: reduce en muchos pasos todos los sub-términos posibles al mismo tiempo.

En un solo paso,

$$\begin{array}{c} \Pi_A \quad \Pi_B \\ \vdots \\ \Pi \end{array}$$

# Algoritmo de reducción

- **Algoritmo:** Reducir sucesivamente hasta que sea irreducible.
- **Estrategias de reducción:** en un paso o muchos pasos.
- *Gross-Knuth*: reduce en muchos pasos todos los sub-términos posibles al mismo tiempo.

En un solo paso,

$$\begin{array}{ccccc} \Pi_A & & \Pi_B & & \Pi_A^* & & \Pi_B^* \\ & & \vdots & & & & \\ & & \Pi & & \begin{pmatrix} \Pi_A & \rightsquigarrow^* & \Pi_A^* \\ \Pi_B & \rightsquigarrow^* & \Pi_B^* \end{pmatrix} & & \Pi \end{array}$$

## 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

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

Problema: Necesitamos la misma fórmula

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

## Teorema (Traducción de Friedman)

Sea  $\varphi$  una fórmula **conjuntiva** y todas las fórmulas de  $\Gamma$  sean **F-fórmulas**.  
Si tenemos

$$\Pi \triangleright \Gamma \vdash_{\mathcal{C}} \forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n),$$

Podemos generar una nueva demostración  $\Sigma$  tal que

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

Se demuestra en **deducción natural** (para reducir).

# 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 intuicionista. Pasos:

# Funcionamiento de traducción de Friedman

Partiendo de

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

Queremos demostrar la misma fórmula en 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 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:**  $\psi$  debe ser  $\Pi_2$  con  $\varphi$  **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 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:**  $\psi$  debe ser  $\Pi_2$  con  $\varphi$  **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 ( $\Gamma$ ) 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

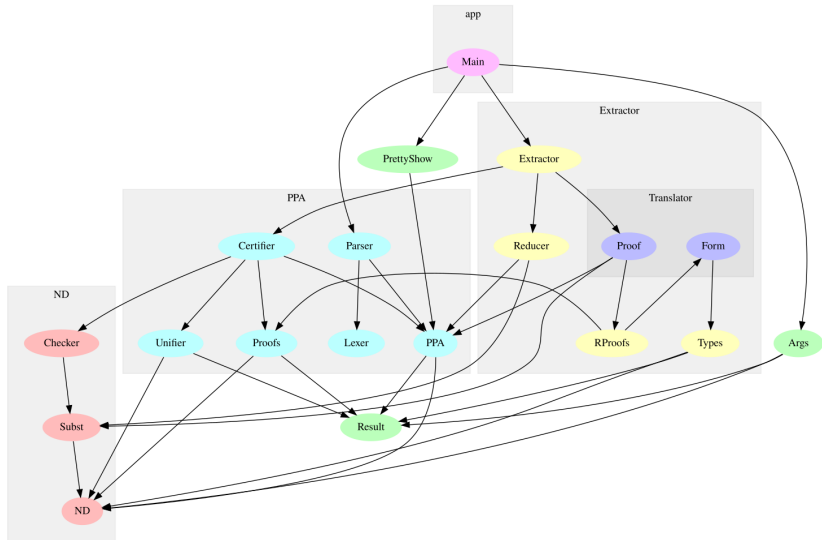
*Sea  $F$  una F-fórmula. Vale  $F \vdash_I F^{\neg\neg}$ .*

## Lema

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

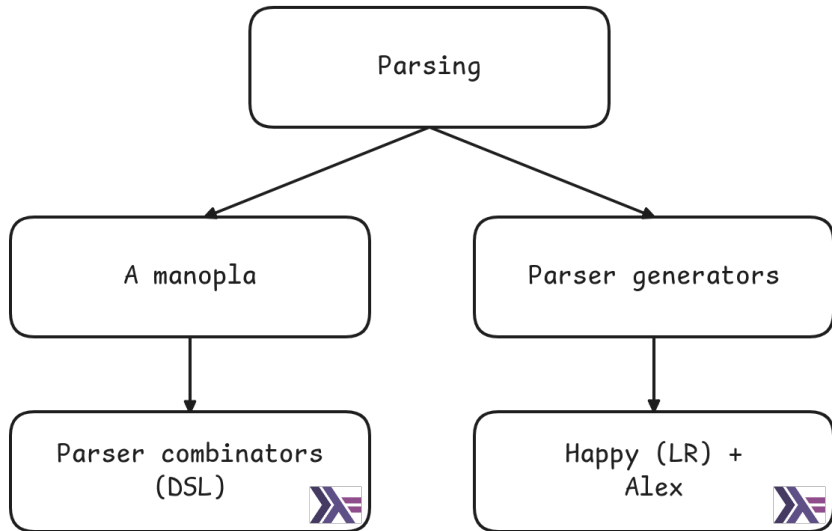
## Detalles de implementación

# La herramienta ppa



Haskell, 19 módulos con 330 tests

String -> Estructura



## 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*.

- 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
  - Cantidad de demostraciones a escribir.

- 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
  - Cantidad de demostraciones a escribir.
  - 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
  - Cantidad de demostraciones a escribir.
  - 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
  - Cantidad de demostraciones a escribir.
  - 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
  - Cantidad de demostraciones a escribir.
  - 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
  - Cantidad de demostraciones a escribir.
  - 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
  - Cantidad de demostraciones a escribir.
  - 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

- **Extensión:** A más de un  $\exists$ .
- **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

- **Extensión:** A más de un  $\exists$ .
- **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

- **Extensión:** A más de un  $\exists$ .
- **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* (mezclando introducciones y eliminaciones de conectivos distintos).
  - Hay algunas demostraciones que no se van a poder reducir a una forma normal útil. Ej: **cases** (EV).
  - *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

- **Extensión:** A más de un  $\exists$ .
- **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* (mezclando introducciones y eliminaciones de conectivos distintos).
  - Hay algunas demostraciones que no se van a poder reducir a una forma normal útil. Ej: **cases** (EV).
  - *Mejora:* Implementarlas.
- **Ineficiente:** en cada paso reinicia la búsqueda de todos los focos de evaluación. *Mejora:* Usar una *máquina abstracta*.

¡Gracias!



[github.com/mnPanic/tesis](https://github.com/mnPanic/tesis)