PPA

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

Manuel Panichelli

Deparatamento de Computación, FCEyN, UBA

Diciembre 2024

Introducción

Repaso de lógica

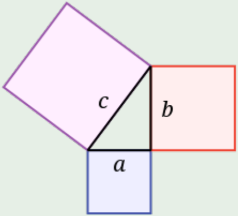
Definiciones (Conceptos centrales)

- **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 de teorema

Ejemplo (Teorema de Pitágoras)

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



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

Asistentes de demostración

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

[1] Terence Tao - Machine Assisted Proof

Asistentes de demostración

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

[1] Terence Tao - Machine Assisted Proof

Asistentes de demostración

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

[1] Terence Tao - Machine Assisted Proof

Asistentes de demostración

Constructivos

Coq
(Type theory)

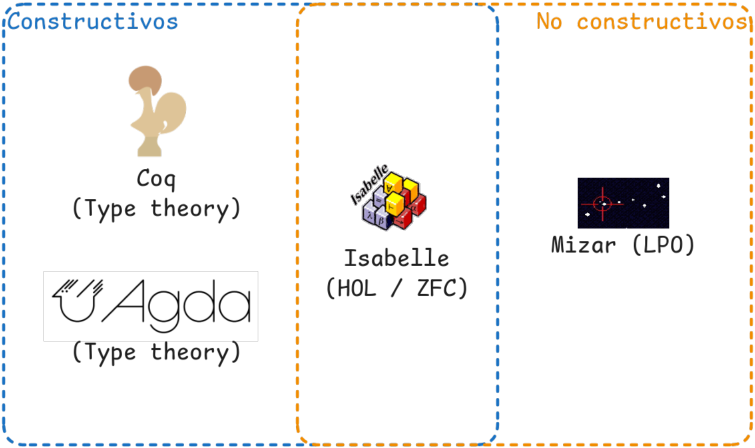
Agda
(Type theory)

Isabelle
(HOL / ZFC)

No constructivos

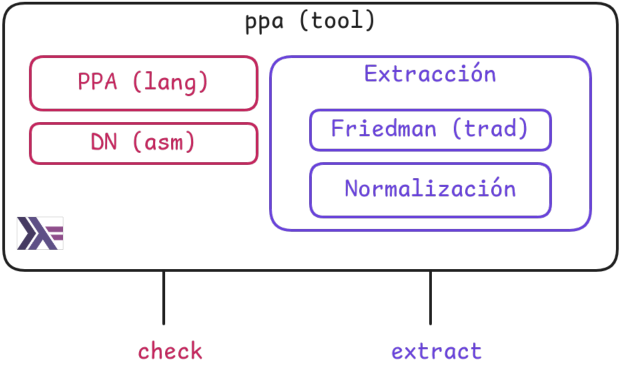
Mizar (LPO)

Asistentes de demostración



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

PPA



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

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

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.

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.

$\square$

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.

$\square$

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

Deducción natural (DN)

Lógica de primer orden

Definición (Términos)

Los términos están dados por la gramática:

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

Definición (Fórmulas)

Las fórmulas están dadas por la gramática:

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

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)

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

$$\frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \mathrm{E}{\to} \quad (\textit{modus ponens})$$

Definiciones

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

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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

Ejemplo (Demostración en DN)

$$\frac{\Gamma = (F \to X), (X \to R), F \vdash R}{(F \to X), (X \to R) \vdash F \to R} \ \mathrm{I}{\to}$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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

Ejemplo (Demostración en DN)

$$\dfrac{\dfrac{}{\Gamma = (F \to X), (X \to R), F \vdash R}\ \mathrm{E}{\to}}{(F \to X), (X \to R) \vdash F \to R}\ \mathrm{I}{\to}$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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

Ejemplo (Demostración en DN)

$$\dfrac{\dfrac{\Gamma \vdash X \to R \qquad\qquad \Gamma \vdash X}{\Gamma = (F \to X), (X \to R), F \vdash R}\ \mathrm{E}{\to}}{(F \to X), (X \to R) \vdash F \to R}\ \mathrm{I}{\to}$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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

Ejemplo (Demostración en DN)

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash X \to R}\,\text{Ax} \qquad \Gamma \vdash X}{\Gamma = (F \to X), (X \to R), F \vdash R}\,\text{E}{\to}}{(F \to X), (X \to R) \vdash F \to R}\,\text{I}{\to}$$

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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

Ejemplo (Demostración en DN)

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash X \to R}\,\text{Ax} \qquad \dfrac{\dfrac{}{\Gamma \vdash F \to X}\,\text{Ax} \qquad \dfrac{}{\Gamma \vdash F}\,\text{Ax}}{\Gamma \vdash X}\,\text{E}{\to}}{\Gamma = (F \to X), (X \to R), F \vdash R}\,\text{E}{\to}}{(F \to X), (X \to R) \vdash F \to R}\,\text{I}{\to}$$

Reglas de inferencia

Otras reglas de inferencia

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

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$\bot$, I$\top$, LEM

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

- 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$\land$
- I$\lor_1$, I$\lor_2$, E$\lor$
- I$\forall$, E$\forall$, I$\exists$, E$\exists$
- E$\bot$, I$\top$, LEM

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

- 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 α-equivalentes (renombrando variables ligadas de forma apropiada, son iguales).

Sustitución

Eliminación de universal

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

Sustitución

Eliminación de universal

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

Definición (Sustitución)

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

Sustitución

Eliminación de universal

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

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 α-equivalente tq no ocurra)

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

PPA

Mizar $\rightsquigarrow$ Isar (Isabelle) $\rightsquigarrow$ *Mathematical Vernacular*[2]

Forma *natural* de representar demostraciones matemáticas. Ideas:

[2] De Freek Wiedijk

Mizar $\rightsquigarrow$ Isar (Isabelle) $\rightsquigarrow$ *Mathematical Vernacular*[2]

Forma *natural* de representar demostraciones matemáticas. Ideas:

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

[2] De Freek Wiedijk

Mathematical Vernacular

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

Forma *natural* de representar demostraciones matemáticas. Ideas:

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

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

  sin tener que demostrarlo a mano (automático).

[2] De Freek Wiedijk

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

Forma *natural* de representar demostraciones matemáticas. Ideas:

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

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

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

[2] De Freek Wiedijk

PPA

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

Ejemplo demostración

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

PPA

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

Ejemplo demostración

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

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

PPA

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

Ejemplo demostración

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

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

PPA

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

Ejemplo demostración

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

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

PPA

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

Ejemplo demostración

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

theorem "falta_entonces_recursa": forall A . forall M .
   falta(A, final(M)) -> recursa(A, M)
proof
   let A
   let M
   suppose "falta": falta(A, final(M))
   have "reprueba": reprueba(A, final(M)) by "ax1", "falta"
```

PPA

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

Ejemplo demostración

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

theorem "falta_entonces_recursa": forall A . forall M .
    falta(A, final(M)) -> recursa(A, M)
proof
    let A
    let M
    suppose "falta": falta(A, final(M))
    have "reprueba": reprueba(A, final(M)) by "ax1", "falta"
    thus recursa(A, M) by "ax2", "reprueba"
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`
I$\top$	`by`
E$\bot$	`by`

Comandos y reglas de inferencia

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

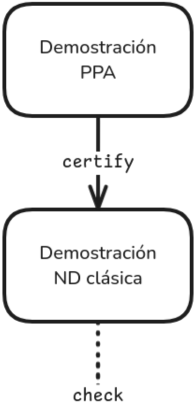
Regla	Comando
I$\wedge$	**by**
E$\wedge_1$	**by**
E$\wedge_2$	**by**
I$\rightarrow$	**suppose**
E$\rightarrow$	**by**
I$\neg$	**suppose**
E$\neg$	**by**
I$\top$	**by**
E$\bot$	**by**

Adicionales:

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

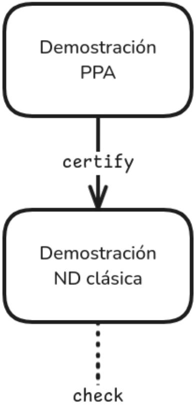
Certificados

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



Certificados

- Las demostraciones de PPA se **certifican** generando una demostración de deducción natural.
- Evita confiar en la implementación del asistente.
- Cumple con el **Criterio de de Bruijn** (sus demostraciones pueden ser chequeadas por un programa independiente)



Certificado de demostraciones

El procedimiento de certificado de una demostración es recursivo:

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

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

Figura: Ejemplo de certificado generado para un programa

by - El mecanismo principal de demostración

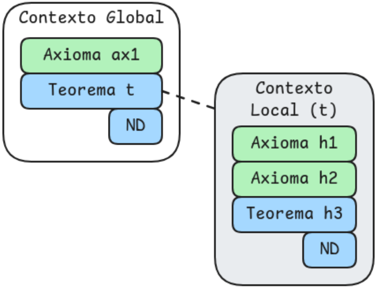
```
thus <form> by <h1>, ..., <hn>
have <name>: <form> by <h1>, ..., <hn>
```

- 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

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

Ejemplo sin cuantificadores (1/4)

By sin cuantificadores

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

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

$$\Gamma \vdash \big((a \to b) \land a\big) \to b$$

Ejemplo sin cuantificadores (2/4)

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

$$\Gamma, \neg\left[\big((a \to b) \land a\big) \to b\right] \vdash \bot$$

Ejemplo sin cuantificadores (2/4)

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

$$\Gamma, \neg[\big((a \to b) \wedge a\big) \to b] \vdash \bot$$

Definición (Eliminación de doble negación)

$$\dfrac{}{\neg\neg A \vdash A}\ \mathrm{E}\neg\neg \quad \equiv \quad \dfrac{}{\Gamma \vdash A \vee \neg A}\ \mathrm{LEM}$$

Definición (Introducción de negación)

$$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \mathrm{I}\neg \qquad\qquad \dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash \neg\neg A}\ \mathrm{I}\neg$$

Ejemplo sin cuantificadores (2/3)

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

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

Conversión a DNF - Reglas admisibles

Reglas admisibles para conversión a DNF

Pasos base

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

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

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

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

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

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

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

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

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

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

Pasos recursivos de congruencia

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

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

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

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

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

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

¡30 demostraciones!

Ejemplo sin cuantificadores (3/3)

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

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

$\square$

Definición (Reglas de inferencia)

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\,\mathrm{E}\vee$$

$$\frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash \bot}\,\mathrm{E}\neg$$

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación

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

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación

$$\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)$$
$$\equiv \neg \Big[\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)\Big]$$

2. Convertimos a DNF ($\forall$ es *opaco*)

$$(\forall x.(p(x) \to q(x))) \land p(k) \land \neg q(k)$$

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación

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

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

2. Convertimos a DNF ($\forall$ es *opaco*)

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

3. No es refutable. E$\forall$ con $x := \mathrm{u}$ una **meta-variable** fresca.

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

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación

$$\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)$$
$$\equiv \neg \Big[\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)\Big]$$

2. Convertimos a DNF ($\forall$ es *opaco*)

$$(\forall x.(p(x) \to q(x))) \land p(k) \land \neg q(k)$$

3. No es refutable. E$\forall$ con $x := \mathrm{u}$ una **meta-variable** fresca.

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

4. Re-convertimos a DNF

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

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación
$$\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)$$
$$\equiv \neg \Big[\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)\Big]$$
2. Convertimos a DNF ($\forall$ es *opaco*)
$$(\forall x.(p(x) \to q(x))) \land p(k) \land \neg q(k)$$
3. No es refutable. E$\forall$ con $x := \mathrm{u}$ una **meta-variable** fresca.
$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$
4. Re-convertimos a DNF
$$(\neg p(\mathrm{u}) \land p(k) \land \neg q(k)) \lor (q(\mathrm{u}) \land p(k) \land \neg q(k))$$
5. Refutamos cada cláusula (**unificando**).

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación

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

$$\equiv \neg \Big[\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)\Big]$$

2. Convertimos a DNF ($\forall$ es *opaco*)

$$(\forall x.(p(x) \to q(x))) \land p(k) \land \neg q(k)$$

3. No es refutable. E$\forall$ con $x := \mathrm{u}$ una **meta-variable** fresca.

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

4. Re-convertimos a DNF

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

5. Refutamos cada cláusula (**unificando**).
   - $\neg p(\mathrm{u}) \land p(k) \land \neg q(k)$ tenemos $p(\mathrm{u}) \doteq p(k)$ con $\{\mathrm{u} := k\}$

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación
$$\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)$$
$$\equiv \neg \Big[\Big((\forall x.(p(x) \to q(x))) \land p(k)\Big) \to q(k)\Big]$$
2. Convertimos a DNF ($\forall$ es *opaco*)
$$(\forall x.(p(x) \to q(x))) \land p(k) \land \neg q(k)$$
3. No es refutable. E$\forall$ con $x := \mathrm{u}$ una **meta-variable** fresca.
$$(p(\mathrm{u}) \to q(\mathrm{u})) \land p(k) \land \neg q(k)$$
4. Re-convertimos a DNF
$$(\neg p(\mathrm{u}) \land p(k) \land \neg q(k)) \lor (q(\mathrm{u}) \land p(k) \land \neg q(k))$$
5. Refutamos cada cláusula (**unificando**).
   - $\neg p(\mathrm{u}) \land p(k) \land \neg q(k)$ tenemos $p(\mathrm{u}) \doteq p(k)$ con $\{\mathrm{u} := k\}$
   - $q(\mathrm{u}) \land p(k) \land \neg q(k)$ tenemos $q(\mathrm{u}) \doteq q(k)$ con $\{\mathrm{u} := k\}$

Certificador del by

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

1. Buscamos las hipótesis en el contexto
2. **Razonamos por el absurdo**: Asumiendo la negación buscamos una contradicción.
3. Convertimos la fórmula a **DNF**.
4. 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 $(\neg p(k,t), \forall x.\forall y.p(x,y))$

Certificador del by

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

1. Buscamos las hipótesis en el contexto
2. **Razonamos por el absurdo**: Asumiendo la negación buscamos una contradicción.
3. Convertimos la fórmula a **DNF**.
4. 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 indecidible por el *Teorema de Church*).

$$✗ \quad (\forall x.p(x) \to q(x)) \land (\forall x.p(x)) \to q(a)$$

Descarga de conjunciones

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

Descarga

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

Problema:

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

Descarga de conjunciones

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

Descarga

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

- Reordena la conjunción (tratando como conjunto).

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

Descarga de conjunciones

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

Descarga

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

- 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

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

- 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

Extracción simple

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

Extracción indirecta con instanciación

```
axiom padre_es_padre: forall A. es_padre(A, padre(A))
theorem todos_tienen_padre: forall Q. exists P. es_padre(Q, P)
proof
    let Q
    take P := padre(Q)
    thus es_padre(Q, padre(Q)) by "padre_es_padre"
end
```

Extracción indirecta con instanciación

```
axiom padre_es_padre: forall A. es_padre(A, padre(A))
theorem todos_tienen_padre: forall Q. exists P. es_padre(Q, P)
proof
   let Q
   take P := padre(Q)
   thus es_padre(Q, padre(Q)) by "padre_es_padre"
end

axiom def_abuelo: forall P. forall Q. forall R.
   (es_padre(P, Q) & es_padre(Q, R)) <-> es_abuelo(P, R)
theorem todos_tienen_abuelo: forall A. exists B. es_abuelo(A, B)
```

Extracción indirecta con instanciación

```
axiom padre_es_padre: forall A. es_padre(A, padre(A))
theorem todos_tienen_padre: forall Q. exists P. es_padre(Q, P)
proof
   let Q
   take P := padre(Q)
   thus es_padre(Q, padre(Q)) by "padre_es_padre"
end

axiom def_abuelo: forall P. forall Q. forall R.
   (es_padre(P, Q) & es_padre(Q, R)) <-> es_abuelo(P, R)
theorem todos_tienen_abuelo: forall A. exists B. es_abuelo(A, B)
proof
   let A
   consider X st "h1": es_padre(A, X) by "todos_tienen_padre"
   consider Y st "h2": es_padre(X, Y) by "todos_tienen_padre"
   take B := Y
   thus es_abuelo(A, Y) by "h1", "h2", "def_abuelo"
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 pepe
Running program... OK! (size = 146)
Translating... OK! (raw size = 136, reduced size = 9)
Checking translated... OK!
Extracted witness: padre(padre(pepe))
       of formula: es_abuelo(pepe, padre(padre(pepe)))
```

Extracción por el absurdo

```
axiom juanEsBajo: bajo(juan)

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

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

Extracción por el absurdo

Extracción por el absurdo

```
axiom juanEsBajo: bajo(juan)

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

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

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

Extracción por el absurdo

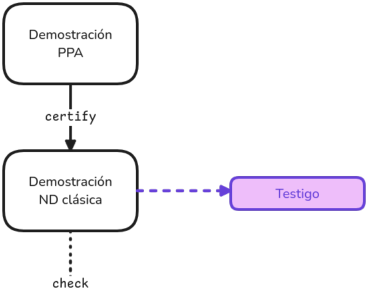
Extracción por el absurdo

```
axiom juanEsBajo: bajo(juan)

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

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

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

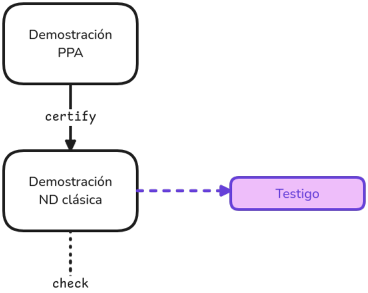
Lógica clásica

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



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

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



Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

Sea C algo indecidible (tipo $\mathtt{HALT}$), queremos ver que vale

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

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

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

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 $y = 0$. $\square$

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

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

podemos demostrarlo razonando por casos con LEM de $C \lor \neg C$

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

¡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$\neg\neg$ $\equiv$ LEM.
- Existen fórmulas que admiten *solo demostraciones no constructivas* (i.e. clásicas) Ejemplo: $\neg(A \wedge B) \to \neg A \vee B$ solo es válido en lógica clásica.

Lógica intuicionista

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

Características:

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

Lógica intuicionista

lógica intuicionista = lógica clásica − LEM

Características:

- No tiene LEM[3], entonces siempre es constructiva.

[3] Ni principios de razonamiento equivalentes, como E¬¬

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

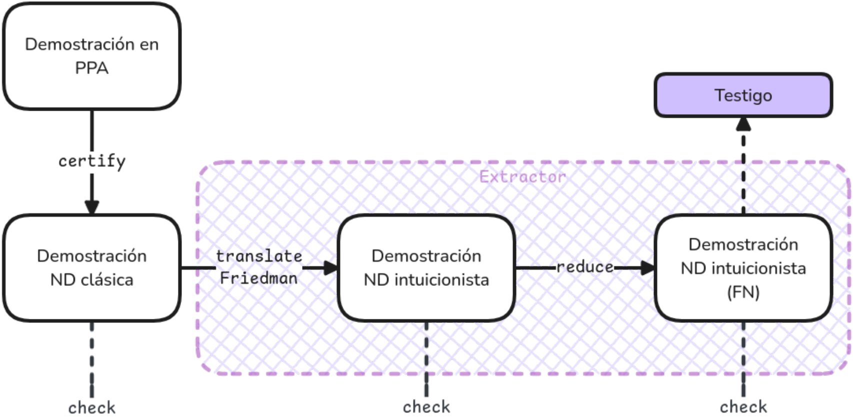
Características:

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

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

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

Estrategia de extracción indirecta



Normalización

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

$$\dfrac{\dfrac{}{\Gamma \vdash A}\,\mathsf{Ax}_{h_1} \qquad \dfrac{}{\Gamma \vdash B}\,\mathsf{Ax}_{h_2}}{\dfrac{\Gamma = h_1 : A, h_2 : B \vdash A \wedge B}{h_1 : A, h_2 : B \vdash A}\,\mathsf{E}\wedge_1}\,\mathsf{I}\wedge$$

Normalización o reducción

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Definición (Reducción de conjunción)

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

Normalización o reducción

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Definición (Reducción de conjunción)

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

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

Normalización de implicación

Definición (Normalización de implicación)

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

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

Normalización de implicación

Definición (Normalización de implicación)

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

- Primer idea: $\cancel{\Pi_B \rhd \Gamma \vdash B}$
- Π_B requiere $h : A$, agregada por $\mathrm{I}{\to}_h$
- Correcto: usar Π_B, pero *sustituyendo* todas las ocurrencias de la hipótesis h por la demostración Π_A (sin capturas).

Normalización de implicación

Definición (Normalización de implicación)

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

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

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,

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

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,

$$\begin{array}{c} \Pi_A \qquad \Pi_B \\ \vdots \\ \Pi \end{array} \quad \overset{\rightsquigarrow}{\begin{pmatrix} \Pi_A \overset{*}{\rightsquigarrow} \Pi_A^* \\ \Pi_B \overset{*}{\rightsquigarrow} \Pi_B^* \end{pmatrix}} \quad \begin{array}{c} \color{red}{\Pi_A^*} \qquad \color{red}{\Pi_B^*} \\ \vdots \\ \Pi \end{array}$$

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$\neg\neg$ $\equiv$ 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 ($\mathsf{E}\neg\neg \equiv \mathsf{LEM}$) pero en intuicionista es más débil.

Teorema

$$\begin{array}{c} \Pi \\ \Gamma \vdash_{\mathbf{c}} A \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \Pi^N \\ \Gamma^N \vdash_{\mathbf{I}} A^N \end{array}$$

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 ($E\neg\neg \equiv$ LEM) pero en intuicionista es más débil.

Teorema

$$\begin{array}{c} \Pi \\ \Gamma \vdash_{\mathbf{C}} A \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \Pi^N \\ \Gamma^N \vdash_{\mathbf{I}} \color{red}{A^N} \end{array}$$

Problema: Necesitamos la misma fórmula

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

El truco de Friedman

Teorema (Traducción de Friedman)

*Sea φ una fórmula **conjuntiva** y todas las fórmulas de Γ sean **F-fórmulas**. Si tenemos*

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

Podemos generar una nueva demostración Σ tal que

$$\Sigma \rhd \Gamma \vdash_I \forall y_1 \ldots \forall y_n . \exists x . \varphi(x, y_1, \ldots, y_n).$$

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

Traducción de doble negación relativizada

Definición (Negación relativizada)

Podemos ver a $\neg A \equiv A \to \bot$. Definimos $\neg_R A \equiv A \to R$

Definición (Traducción de doble negación relativizada)

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

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

$$\Pi \rhd \Gamma \vdash_C \psi$$

Queremos demostrar la misma fórmula en intuicionista. Pasos:

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

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

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: 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

$$\Pi \rhd \Gamma \vdash_C \psi$$

Queremos demostrar la misma fórmula en intuicionista. Pasos:

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

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

2. Usarla para demostrar la fórmula original.
Restricción: ψ debe ser Π_2 con φ **conjuntiva**.

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

3. Mantener el contexto (reemplazando Ax por $A \vdash_I A^{\neg\neg}$)
Restricción: Axiomas (Γ) deben ser **F-fórmulas**.

$$\Sigma \rhd \Gamma \vdash_I \forall y_1 \ldots \forall y_n . \exists x . \varphi(x, y_1, \ldots, y_n). \quad \square$$

Tipos de fórmulas

Definición (Gramática de fórmulas)

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

Lema

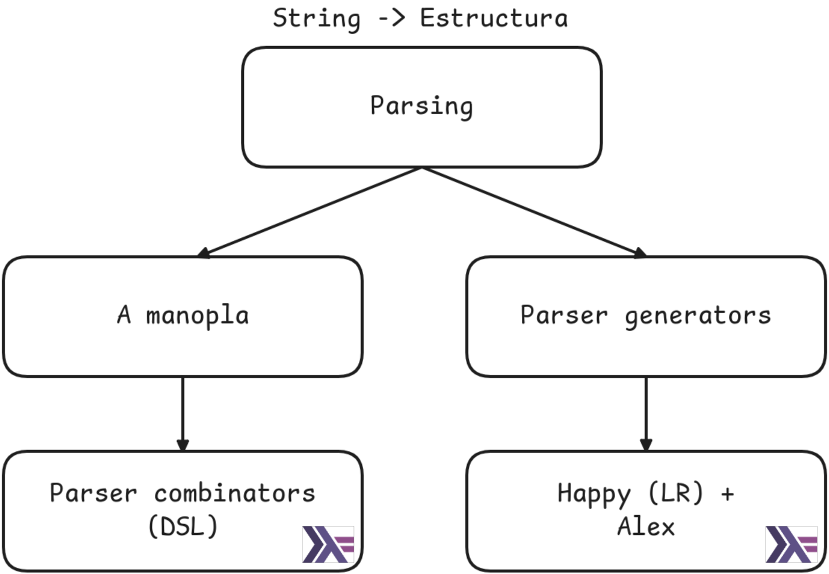
Sea F una F-fórmula. Vale $F \vdash_I F^{\neg\neg}$.

Lema

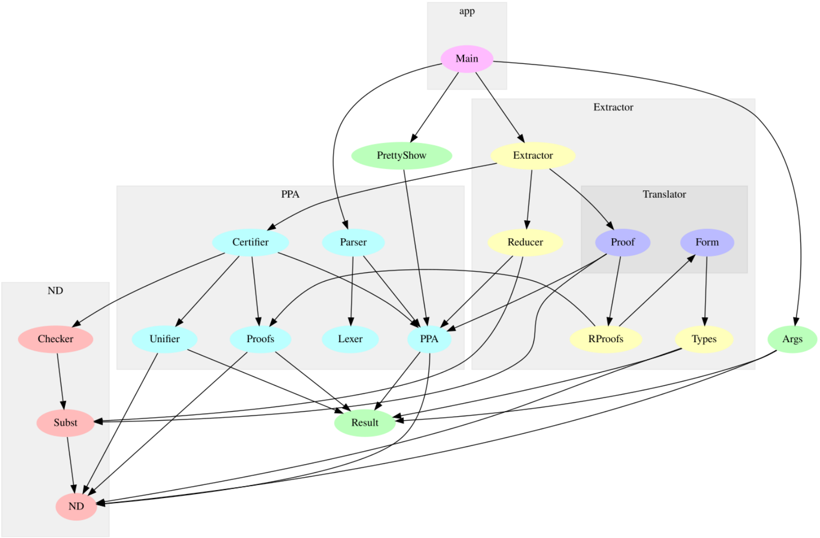
Sea C una fórmula conjuntiva. Vale $\neg_R C \vdash_I \neg_R C^{\neg\neg}$.

Detalles de implementación

Parser y lexer



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.

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.
- Desafíos

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.
- Desafíos
  - Haskell.

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.
- Desafíos
  - Haskell.
  - Demostraciones generadas automáticamente difíciles de *debuggear*.

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.
- Desafíos
  - Haskell.
  - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras

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.
- Desafíos
  - Haskell.
  - Demostraciones generadas automáticamente difíciles de *debuggear*.
- Otras mejoras
  - Permitir importar archivos, implementar biblioteca estándar.

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.
- 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).

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.
- 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.

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.
- 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

- **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** (E$\vee$).
  - *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** ($E\vee$).
  - *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