PPA

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

Manuel Panichelli

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

Diciembre 2024

Introducción

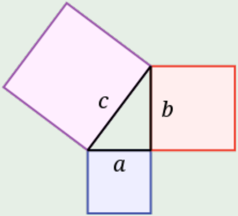
Definiciones (Conceptos centrales)

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

Ejemplo 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

- 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

- 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

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

[1] Terence Tao - Machine Assisted Proof

Asistentes de demostración

Constructivos

Coq
(Type theory)

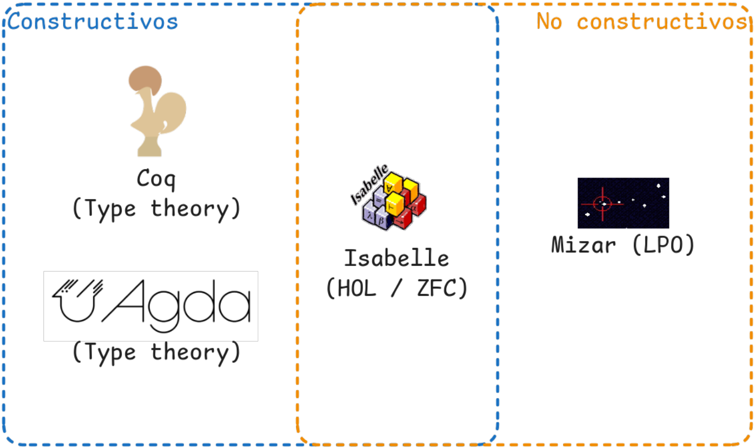
Agda
(Type theory)

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.

Asistentes de demostración

Constructivos

No constructivos

Coq
(Type theory)

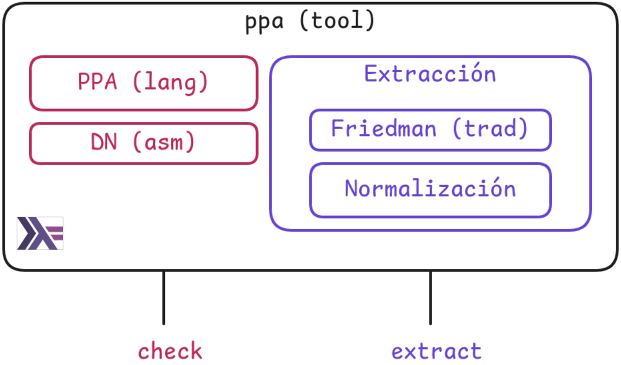
Agda
(Type theory)

Isabelle
(HOL / ZFC)

Mizar (LPO)

¿Dónde cae ppa?

PPA



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

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

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)

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:

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

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)

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \mathsf{I}{\to}$$

Modus ponens:

$$\frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \mathsf{E}{\to}$$

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

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:

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

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

Ejemplo (Demostración en DN)

$$(F \to X), (X \to R) \vdash F \to R$$

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)

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

Ejemplo de demostración en DN

Ejemplo (Teorema en DN)

Notamos:

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

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)

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

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)

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

Reglas de inferencia

Otras reglas de inferencia

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

Reglas de inferencia

Otras reglas de inferencia

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

Reglas admisibles

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

$$\big((A \to B) \wedge \neg B\big) \to \neg A$$

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

PPA

Mathematical Vernacular

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*.**

[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***.
- **Reglas de inferencia *declarativas***:

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

[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*.**
- **Reglas de inferencia *declarativas*:**

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

- **Sintaxis similar a un lenguaje de programación**

[2] De Freek Wiedijk

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

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*.

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*.

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*.

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 "h1": falta(A, final(M))
```

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

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 "h1": falta(A, final(M))
   have "h2": reprueba(A, final(M)) by "ax1", "h1"
```

PPA

Lenguaje PPA, inspirado en el *Mathematical Vernacular*.

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 "h1": falta(A, final(M))
   have "h2": reprueba(A, final(M)) by "ax1", "h1"
   thus recursa(A, M) by "ax2", "h2"
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**

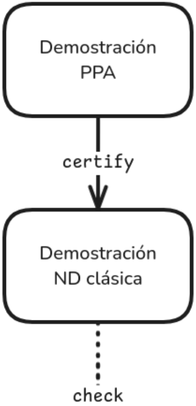
Regla	Comando
I$\wedge$	**by**
E$\wedge_1$	**by**
E$\wedge_2$	**by**
I$\rightarrow$	**suppose**
E$\rightarrow$	**by**
I$\neg$	**suppose**
E$\neg$	**by**
I$\top$	**by**
E$\bot$	**by**

Adicionales:

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

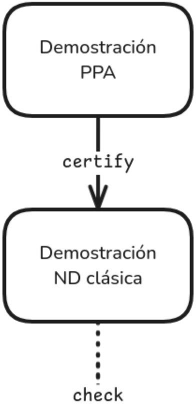
Certificados

- 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**



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
```

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

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
```

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

```
PImpI
    { hypAntecedent = "h"
    , proofConsequent =
        PExistsI
            { inst = TFun "v" []
            , proofFormWithInst = PAx "h"
            }
    }
```

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

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

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

Ejemplo sin cuantificadores (2/4)

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

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

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)

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

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

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

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

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

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

□

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

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

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*)

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

Ejemplo con cuantificadores

Definición (Eliminación de universal)

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

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

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

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.

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

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.

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

4. Re-convertimos a DNF

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

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

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

5. Refutamos cada cláusula (**unificando**).

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.

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

4. Re-convertimos a DNF

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

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

Ejemplo con cuantificadores

1. Supongamos que tenemos que resolver siguiente implicación

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

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

Para certificar un **by**:

1. Buscamos las hipótesis en el contexto y armamos la fórmula
2. **Razonamos por el absurdo**
3. Convertimos la fórmula a **DNF**.
4. Demostramos una **contradicción** refutando cada cláusula
   - Contiene $\perp$ o dos fórmulas opuestas $(a, \neg a)$,
   - Eliminando universales **consecutivos** $(\neg p(k, t), \forall x.\forall y.p(x, y))$

Certificador del by

Para certificar un **by**:

1. Buscamos las hipótesis en el contexto y armamos la fórmula
2. **Razonamos por el absurdo**
3. Convertimos la fórmula a **DNF**.
4. Demostramos una **contradicción** refutando cada cláusula
    - Contiene $\perp$ o dos fórmulas opuestas $(a, \neg a)$,
    - Eliminando universales **consecutivos** $(\neg p(k,t), \forall x.\forall y.p(x,y))$

Alcance

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

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

Descarga de conjunciones

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

Problema:

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

Descarga

```
axiom "ax1": a & e
axiom "ax2": b & c
axiom "ax3": d
theorem "and discharge":
    (a & b) & ((c & d) & e)
proof
    thus a & e by "ax1"
    thus d by "a3"
    thus b & c by "ax2"
end
```

Descarga de conjunciones

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

Descarga

```
axiom "ax1": a & e
axiom "ax2": b & c
axiom "ax3": d
theorem "and discharge":
    (a & b) & ((c & d) & e)
proof
    thus a & e by "ax1"
    thus d by "a3"
    thus b & c by "ax2"
end
```

- Reordena la conjunción (tratando como conjunto).

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

Descarga de conjunciones

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

Descarga

```
axiom "ax1": a & e
axiom "ax2": b & c
axiom "ax3": d
theorem "and discharge":
    (a & b) & ((c & d) & e)
proof
    thus a & e by "ax1"
    thus d by "a3"
    thus b & c by "ax2"
end
```

- Reordena la conjunción (tratando como conjunto).

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

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

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

Descarga de conjunciones

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

Descarga

```
axiom "ax1": a & e
axiom "ax2": b & c
axiom "ax3": d
theorem "and discharge":
    (a & b) & ((c & d) & e)
proof
    thus a & e by "ax1"
    thus d by "a3"
    thus b & c by "ax2"
end
```

- Reordena la conjunción (tratando como conjunto).

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

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

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

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

Extracción de testigos

Extracción simple

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

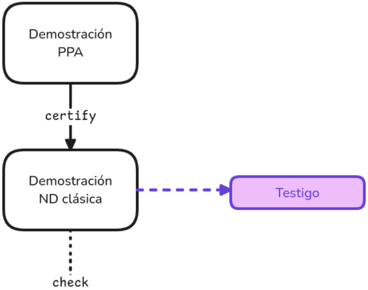
Extracción simple

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
```

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

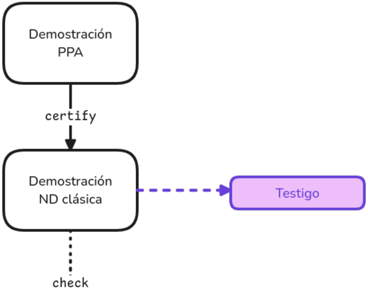
- Buscamos un mecanismo general para extraer testigos de los certificados.



Lógica clásica

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

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



Ejemplo (Fórmula sin demostración constructiva)

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

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

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

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

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

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

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 $y = 0$. $\square$

¡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

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

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*.

¿Entonces por qué lógica clásica?

- Permite razonar por el absurdo, con $\mathrm{E}\neg\neg \equiv \mathrm{LEM}$.
- Existen fórmulas que admiten *solo demostraciones clásicas.* Ejemplo:

$$\neg(A \land B) \to \neg A \lor B.$$

Lógica intuicionista

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

Características:

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

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

Características:

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

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

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

Características:

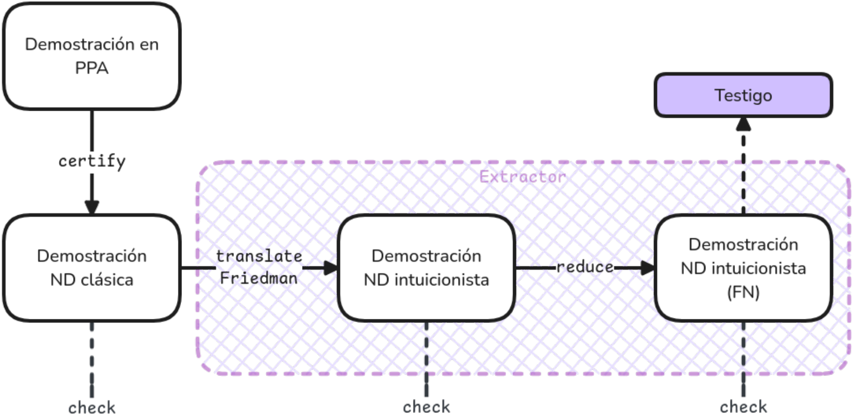
- No tiene LEM[3], entonces siempre es constructiva.
- Proceso de normalización con *forma normal* buena, una demostración de un $\exists$ comienza con I$\exists$

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

- Siempre permite extracción de testigos.

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

Estrategia de extracción indirecta



Traducción de Friedman

Traducción de doble negación

- Queremos *embeber* lógica clásica a intuicionista (no son equivalentes)
- Traducción de **doble negación**: método general.
- **Intuición**: "agregar una doble negación a todo".
- En clásica son equivalentes ($\mathrm{E}\neg\neg \equiv \mathrm{LEM}$) 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 ($E\neg\neg \equiv$ 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

$$\dfrac{\Pi}{\Gamma \vdash_{\boldsymbol{C}} A} \quad \rightsquigarrow \quad \dfrac{\Pi^N}{\Gamma^N \vdash_{\boldsymbol{I}} \color{red}{A^N}}$$

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)

Si tenemos

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

Podemos generar una nueva demostración Σ *tal que*

$$\Sigma \rhd \Gamma \vdash_I \psi$$

Si ψ *y* Γ *cumplen con ciertas restricciones.*

Traducción de doble negación relativizada

Definición (Negación relativizada)

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

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

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

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

Funcionamiento de traducción de Friedman

Partiendo de

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

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

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

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

Funcionamiento de traducción de Friedman

Partiendo de

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

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

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

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

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

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

Funcionamiento de traducción de Friedman

Partiendo de

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

Queremos demostrar la misma fórmula en ND intuicionista. Pasos:

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

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

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)

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

Lema

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

Lema

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

Normalización

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Luego

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

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Luego

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

Normalización o reducción

Motivación: evitar *"desvíos superfluos"*.

Ejemplo

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

Luego

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

Definición (Reducción de conjunción)

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

Algoritmo de reducción

Definición (Otras reglas)

Además, hay reglas para simplificar

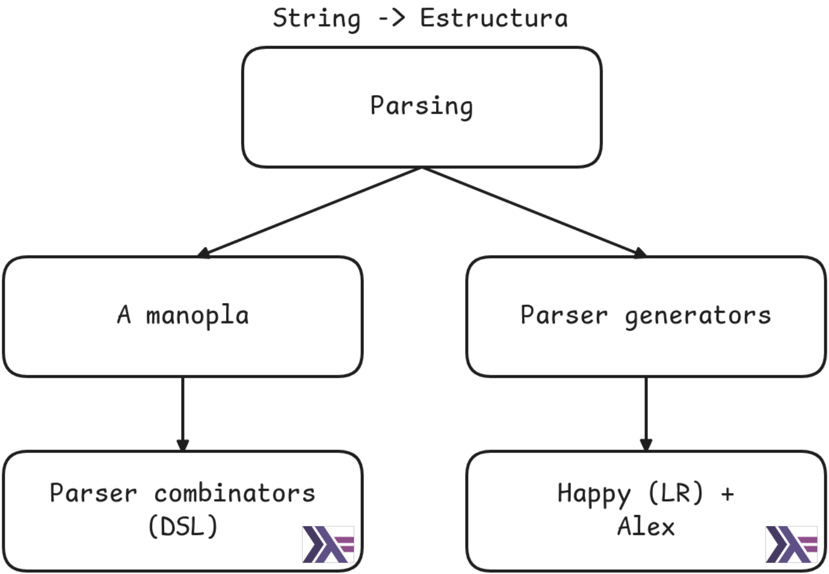
- E$\exists$ con I$\exists$, E$\forall$ con I$\forall$.
- E$\neg$ con I$\neg$, E$\vee$ con I$\vee$.
- E$\rightarrow$ con I$\rightarrow$.

Idea del algoritmo

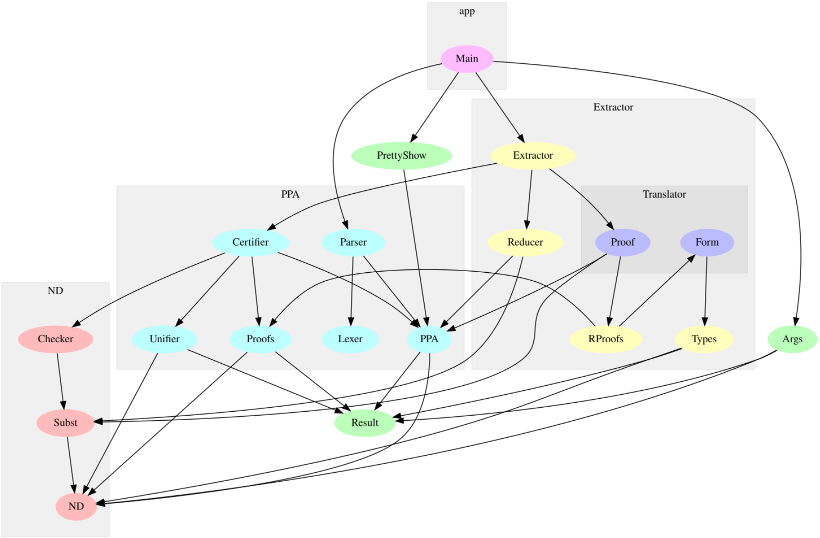
Aplicar las reglas de reducción sucesivamente hasta que no se pueda y la demostración esté en *forma normal*.

Detalles de implementación

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 de Friedman

Conclusiones - Extracción de testigos

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

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

Conclusiones - Extracción de testigos

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

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

Conclusiones - Extracción de testigos

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

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

Conclusiones - Extracción de testigos

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

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

Conclusiones - Extracción de testigos

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

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

Conclusiones - Extracción de testigos

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

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

Conclusiones - Extracción de testigos

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

- **Desafío**: Problemas aparecieron en la integración de las 3 partes.
- Traducción de Friedman
  - **Extensión**: A más de un $\exists$.
  - **Desafío**: Demostraciones de F-fórmulas y conjuntivas.
  - **Limitación**: Refinar la definición de fórmulas conjuntivas y explorar aparente vínculo con *fórmulas de Harrop*.
- Reducción
  - Solo contempla introducciones y eliminaciones del mismo conectivo.
  - **Incompleta**: no contempla *reducciones permutativas*. *Mejora*: Implementarlas. Ejemplo que queda afuera: **cases** ($E\vee$).
  - **Ineficiente**: en cada paso reinicia la búsqueda de todos los focos de evaluación. *Mejora*: Usar una *máquina abstracta*.

¡Gracias!

github.com/mnPanic/tesis