

PPA

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

Manuel Panichelli

Departamento de Computación, FCEyN, UBA

Diciembre 2024

Introducción

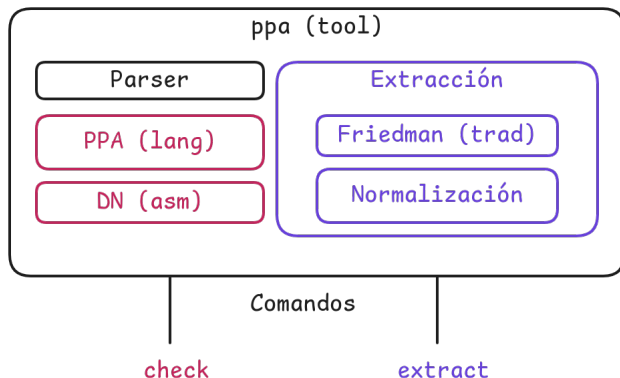
- Los **asistentes de demostración** son herramientas que facilitan la escritura y el chequeo de demostraciones por computadora.
- Usos usuales: formalización de teoremas matemáticos y verificación de programas.
- Ventajas:¹
 - Facilitan la colaboración a gran escala (mediante la confianza en el asistente).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

¹Terrence Tao - Machine Assisted Proof

Implementan distintas *teorías*. Ejemplos:

- Mizar (lógica de primer orden)
- Coq (teoría de tipos)
- Agda (teoría de tipos)
- Isabelle (lógica de orden superior / teoría de conjuntos ZF)

(TODO: No me gusta la palabra teoria, lo usamos para teorias de primer orden)



Diseñamos e implementamos en Haskell la herramienta `ppa` (*Pani's Proof Assistant*): un asistente de demostración para LPO **clásica**. Dos partes:

- El lenguaje **PPA** para escribir demostraciones.
- **Extracción de testigos**: De una demo de $\exists x.p(x)$, extrae t tq $p(t)$.

Implementación de extracción de testigos para lógica clásica de forma *indirecta*.

Representación de demostraciones

¿Cómo representamos las demostraciones? Ejemplo:

- Tenemos dos premisas
 - 1 Los alumnos que faltan a los exámenes, los reprueban.
 - 2 Si se reprueba un final, se recursa la materia.
- A partir de ellas, podríamos demostrar que si un alumno falta a un final, entonces recursa la materia.

Teorema

Si ((falta entonces reprueba) y (reprueba entonces recursa)) y falta, entonces recursa

Demostración.

- Asumo que falta. Quiero ver que recursa.
- Sabemos que si falta, entonces reprueba. Por lo tanto reprobó.
- Sabemos que si reprueba, entonces recursa. Por lo tanto recursó.



- **Problema:** Poco precisa. No se puede representar rigurosamente.
- Necesitamos **sistemas deductivos**: sistemas lógicos formales usados para escribir demostraciones de setencias.
- En particular usamos **deducción natural**. Compuesto por,
 - **Lenguaje formal**: lógica de primer orden.
 - **Reglas de inferencia**: lista de reglas que se usan para probar teoremas a partir de axiomas y otros teoremas. Por ejemplo, *modus ponens* (si es cierto $A \rightarrow B$ y A , se puede concluir B) o *modus tollens* (si es cierto $A \rightarrow B$ y $\neg B$, se puede concluir $\neg A$)
 - **Axiomas**: fórmulas de L que se asumen válidas. Todos los teoremas se derivan de axiomas. Se usan para modelar *teorías* de primer orden (por ej. teoría de estudiantes en la facultad).

Definición (Términos)

Los términos están dados por la gramática:

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

Definición (Fórmulas)

Las fórmulas están dadas por la gramática:

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

Deducción natural

Ejemplo

Vamos a demostrar el ejemplo informal en deducción natural. Lo modelamos para un alumno y materia particulares. Notamos:

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

Queremos probar entonces

$$\left((F \rightarrow X) \wedge (X \rightarrow R) \right) \rightarrow (F \rightarrow R)$$

Ejemplo

$$\begin{array}{c}
 \frac{}{\Gamma \vdash (F \rightarrow X) \wedge (X \rightarrow R)} \text{Ax} \\
 \hline
 \frac{}{\Gamma \vdash X \rightarrow R} \text{E}\wedge_1 \quad \frac{}{\Gamma \vdash X} \Pi \\
 \hline
 \frac{}{\Gamma = (F \rightarrow X) \wedge (X \rightarrow R), F \vdash R} \text{E}\rightarrow \\
 \hline
 \frac{}{(F \rightarrow X) \wedge (X \rightarrow R) \vdash F \rightarrow R} \text{I}\rightarrow \\
 \hline
 \vdash \left((F \rightarrow X) \wedge (X \rightarrow R) \right) \rightarrow (F \rightarrow R) \text{I}\rightarrow
 \end{array}$$

donde

$$\begin{array}{c}
 \frac{}{\Gamma \vdash (F \rightarrow X) \wedge (X \rightarrow R)} \text{Ax} \\
 \hline
 \frac{}{\Gamma \vdash F \rightarrow X} \text{E}\wedge_2 \quad \frac{}{\Gamma \vdash F} \text{Ax} \\
 \hline
 \Pi = \frac{}{\Gamma \vdash X} \text{E}\rightarrow
 \end{array}$$

Definición (Reglas de inferencia)

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

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

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

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

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \text{E} \wedge_1$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \text{E} \wedge_2$$

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

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

Definición (Contexto de demostración)

Γ es un **contexto de demostración**, conjunto de fórmulas que se asumen válidas.

Notación: $\Gamma, \varphi = \Gamma \cup \{\varphi\}$

Definición (Relación de derivabilidad)

- \vdash es la **relación de derivabilidad** definida a partir de las *reglas de inferencia*. Permite escribir juicios $\Gamma \vdash \varphi$.
- Intuición: “ φ es una consecuencia de las suposiciones de Γ ”
- Decimos que φ es *derivable* a partir de Γ .
- El juicio es cierto si en una cantidad finita de pasos podemos concluir φ a partir de las fórmulas de Γ , los axiomas y las reglas de inferencia.

Definición (Reglas de inferencia)

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

$$\frac{\Gamma \vdash \perp}{\Gamma \vdash A} E_{\perp}$$

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

$$\frac{}{\Gamma \vdash \top} I_{\top}$$

$$\frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \perp} E_{\neg}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} I_{\vee_1}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} I_{\vee_2}$$

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

Definición (Sustitución)

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

Definición (Reglas de cuantificadores)

$$\frac{\Gamma \vdash A \quad x \notin fv(\Gamma)}{\Gamma \vdash \forall x.A} \text{I}\forall$$

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

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

$$\frac{\Gamma \vdash \exists x.A \quad \Gamma, A \vdash B \quad x \notin fv(\Gamma, B)}{\Gamma \vdash B} \text{E}\exists$$

Reglas admisibles

- Mencionamos *modus tollens* pero no aparece en las reglas de inferencia.
- Queremos un sistema lógico **minimal**: no agregamos las reglas **admisibles**, derivables a partir de las existentes.
- Se implementan como funciones o macros.

Lema (Modus tollens)

$$\frac{\frac{\frac{}{\Gamma \vdash (A \rightarrow B) \wedge \neg B} Ax}{\Gamma \vdash \neg B} E\wedge_2 \quad \frac{\frac{\frac{}{\Gamma \vdash (A \rightarrow B) \wedge \neg B} Ax}{\Gamma \vdash A \rightarrow B} E\wedge_1 \quad \frac{}{\Gamma \vdash A} Ax}{\Gamma \vdash B} E\rightarrow}{\Gamma = (A \rightarrow B) \wedge \neg B, A \vdash \perp} E\neg}{(A \rightarrow B) \wedge \neg B \vdash \neg A} I\neg}{\vdash (A \rightarrow B \wedge \neg B) \rightarrow \neg A} I\rightarrow$$

Sustitución sin capturas

Para la sustitución $A\{x := t\}$ queremos evitar la **captura de variables**, por ejemplo

$$(\forall y.p(x))\{x := y\} \stackrel{?}{=} \forall y.p(\textcolor{red}{y})$$

sustituyendo sin más, capturamos a la variable x que ahora está ligada. Lo evitamos **automáticamente**: cuando se encuentra con una captura, se renombra la variable ligada de forma que no ocurra

$$(\forall y.p(x))\{x := y\} = \forall \textcolor{red}{z}.p(y)$$

Alfa equivalencia

- Si tenemos una hipótesis $\exists x.p(x)$ queremos poder usarla para demostrar $\exists y.p(y)$.
- No son iguales, pero son **α -equivalentes**: si renombramos variables ligadas de forma apropiada, son iguales.
- Algoritmo naíf: cuadrático en la estructura de la fórmula, renombrando recursivamente.
- Algoritmo cuasilineal: manteniendo dos sustituciones, una por fórmula.

Ejemplo

$$\begin{array}{ll} (\exists x.f(x)) \stackrel{\alpha}{=} (\exists y.f(y)) & \{\}, \{\} \\ \iff f(x) \stackrel{\alpha}{=} f(y) & \{x \mapsto z\}, \{y \mapsto z\} \\ \iff x \stackrel{\alpha}{=} y & \{x \mapsto z\}, \{y \mapsto z\} \\ \iff z = z. & \end{array}$$

PPA

Hay una forma canónica de representar demostraciones matemáticas².
Descubierta e implementada independientemente en Mizar, Isar (Isabelle),
etc. Combinación de ideas:

- **Deducción natural en estilo de *Fitch***. Notación equivalente, demostraciones como listas de fórmulas en lugar de árboles.
- **Reglas de inferencia *declarativas***: una forma de afirmar que $A_1, \dots, A_n \vdash A$ es válida, sin tener que demostrarlo a mano.
- **Sintaxis similar a un lenguaje de programación** en lugar al lenguaje natural.

² *Mathematical Vernacular* de Freek Wiedijk

Diseñamos e implementamos el *lenguaje* PPA, inspirado en el *mathematical vernacular*. Veamos la **interfaz** completa y luego la implementación.

Ejemplo demostración

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

Un **programa** de PPA consiste en una lista de **declaraciones**, que pueden ser

- **Axiomas**: fórmulas que se asumen válidas

```
axiom <name> : <form>
```

- **Teoremas**: fórmulas junto con sus demostraciones.

```
theorem <name> : <form>
```

```
proof
```

```
  <steps>
```

```
end
```

- **Variables** (<var>)

$(_ | [A-Z]) [a-zA-Z0-9_ -] * (\') *$

- **Identificadores** (<id>)

$[a-zA-Z0-9_ - \backslash ? ! \# \$ \% * \backslash + \backslash < \backslash > \backslash = \backslash ? \backslash @ \backslash ^] + (\') *$

- **Nombres** (<name>)

Pueden ser identificadores o strings arbitrarios encerrados por comillas dobles.

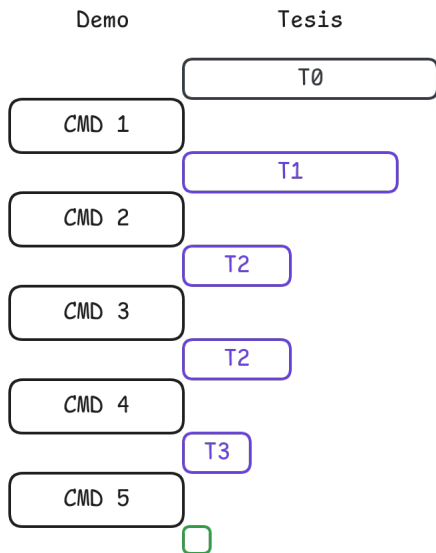
$\text{<id> } | \text{ \" } [^\"] * \text{ \" }$

Términos:

- Variables: `<var>`
- Funciones: `<id>(<term>, ..., <term>)`

Funciones:

- Predicados: `<id>(<term>, ..., <term>)`
- `<form>` & `<form>`
- `<form>` | `<form>`
- `<form>` `->` `<form>`
- `<form>` `<->` `<form>`
- `~ <form>`
- **exists** `<var>` . `<form>`
- **forall** `<var>` . `<form>`
- **true, false**
- `(<form>)`



- Lista de **comandos** que reducen sucesivamente la *tesis* (fórmula a demostrar) hasta agotarla por completo.
- Corresponden aproximadamente a reglas de inferencia de deducción natural.
- Tienen disponible un **contexto** con todas las hipótesis asumidas (axiomas) o demostradas (teoremas y comandos que demuestran hipótesis auxiliares).

by - El mecanismo principal de demostración

- `<form> by <h1>, ..., <hn>` afirma que la fórmula es una consecuencia lógica de las fórmulas que corresponden a las hipótesis provistas.
- Los nombres de las hipótesis son del tipo `<name>` (o bien identificadores o *strings* arbitrarios).
- Por debajo usa un *solver* completo para lógica proposicional pero heurístico para primer orden.
- Se usa para eliminar implicaciones y universales.
- Usado por dos comandos principales: **thus** y **have**.

Thus

thus <form> **by** <h1>, ..., <hn>

Si <form> es *parte* de la tesis, y el *solver* puede demostrar la implicación, lo demuestra automáticamente y lo descarga de la tesis.

Eliminación de implicación

```
1 axiom ax1: a -> b
2 axiom ax2: b -> c
3
4 theorem t1: a -> c
5 proof
6   suppose a: a
7
8   // La tesis ahora es c
9   thus c by a, ax1, ax2
10 end
```

Eliminación de universal

```
1 axiom ax: forall X . f(X)
2
3 theorem t: f(n)
4 proof
5   thus f(n) by ax
6 end
```

have <name>: <form> **by** <h1>, ..., <hn>

Análogo a **thus**, pero introduce una afirmación *auxiliar* sin reducir la tesis, agregándola al contexto.

Eliminación de implicación en dos pasos

```
1 axiom ax1: a -> b
2 axiom ax2: b -> c
3
4 theorem t1: a -> c
5 proof
6   suppose a: a
7   have b: b by a, ax1
8   thus c by b, ax2
9 end
```

Hipótesis anterior

Ambas pueden referirse a la hipótesis anterior con guión medio (-), y pueden hacerlo implícitamente usando **hence** y **have**.

Comando	Alternativo	¿Reduce la tesis?
thus	hence	Sí
have	then	No

Eliminación en dos pasos

```
1 axiom ax1: a -> b
2 axiom ax2: b -> c
3
4 theorem t1: a -> c
5 proof
6   suppose a: a
7   have b: b by a, ax1
8   thus c by b, ax2
9 end
```

Alternativas equivalentes

```
proof
  suppose a: a
  have b: b by -, ax1
  thus c by -, ax2
end

proof
  suppose -: a
  then -: b by ax1
  hence c by ax2
end
```

- El **by** es opcional
- Si se omite, la fórmula debe ser demostrable por el *solver* sin partir de ninguna hipótesis
- Vale para todas las tautologías proposicionales.

Tautología proposicional

```
1  theorem "distributiva de negación sobre disyunción":  
2       $\sim(a \mid b) \leftrightarrow \sim a \ \& \ \sim b$   
3  proof  
4      thus  $\sim(a \mid b) \leftrightarrow \sim a \ \& \ \sim b$   
5  end
```

Comandos y reglas de inferencia

Regla	Comando
LEM	cases
Ax	by
$I\exists$	take
$E\exists$	consider
$I\forall$	let
$E\forall$	by
$I\forall_1$	by
$I\forall_2$	by
$E\forall$	cases

Regla	Comando
$I\wedge$	by
$E\wedge_1$	by
$E\wedge_2$	by
$I\rightarrow$	suppose
$E\rightarrow$	by
$I\neg$	suppose
$E\neg$	by
IT	by
$E\perp$	by

Suppose ($\text{I}\rightarrow / \text{I}\neg$)

suppose <name>: <form> ($\text{I}\rightarrow / \text{I}\neg$)

- Si la tesis es una implicación $A \rightarrow B$, agrega el antecedente A como hipótesis con el nombre dado y reduce la tesis al consecuente B
- Viendo la negación como una implicación $\neg A \equiv A \rightarrow \perp$, permite introducir negaciones, tomando $B = \perp$.

Introducción de implicación

```
1 theorem "suppose":  
2   a -> (a -> b) -> b  
3 proof  
4   suppose h1: a  
5   suppose h2: a -> b  
6   thus b by h1, h2  
7 end
```

Introducción de negación

```
1 theorem "not intro":  
2   ~b & (a -> b) -> ~a  
3 proof  
4   suppose h: ~b & (a -> b)  
5   suppose a: a  
6   hence false by h, a  
7 end
```

cases by <h1>, ..., <hn> (\vdash / \vdash)

- Permite razonar por casos a partir de una disyunción. Para cada uno, se debe demostrar la tesis en su totalidad.
- Si los casos son <f1> a <fn>, tiene que valer
<f1> | ... | <fn> **by**
<h1>, ..., <hn>.
- Se puede omitir el **by** para razonar mediante LEM (casos φ y $\neg\varphi$).

Cases

```
1  theorem "cases":  
2    (a & b) | (c & a) -> a  
3  proof  
4    suppose h: (a & b) | (c & a)  
5    cases by h  
6      case a & b  
7        hence a  
8      case right: a & c  
9        thus a by right  
10   end  
11  end
```

Take (\exists)

take <var> := <term> (\exists)

- Introduce un existencial instanciando su variable y reemplazándola por un término.
- Si la tesis es **exists** $X . p(X)$, luego de **take** $X := a$, se reduce a $p(a)$.

Consider ($E\exists$)

consider <var> **st** <name>: <form> **by** <h1>, ..., <hn> ($E\exists$)

- Si se puede justificar **exists** X . $p(X)$, permite razonar sobre tal X .
- Agrega <form> como hipótesis al contexto, con nombre <name>. No reduce la tesis.
- Debe valer **exists** <var> . <form> **by** <h1>, ..., <hn>
- Permite α -equivalencias: Si podemos justificar **exists** X . $p(X)$, podemos usarlo como **consider** Y **st** h : $p(Y)$ **by**

let <var> (\forall)

- Permite demostrar un cuantificador universal.
- Si la tesis es **forall** $X . p(X)$, luego de **let** X , la tesis se reduce a $p(X)$.
- Permite renombrar la variable, por ejemplo luego de **let** Y la tesis se reduce a $p(Y)$.

Descarga de conjunciones

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

Descarga simple

```
1 theorem "and discharge":  
2   a -> b -> (a & b)  
3 proof  
4   suppose "a" : a  
5   suppose "b" : b  
6   // La tesis es a & b  
7   hence b by "b"  
8  
9   // La tesis es a  
10  thus a by "a"  
11 end
```

Descarga compleja

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

equivalently <form>

- Permite reducir la tesis a una fórmula equivalente
- Se puede usar por ejemplo para descarga de conjunciones, o para razonar por el absurdo mediante la eliminación de la doble negación.

Descarga de conjunción

```
1 axiom a1: ~a
2 axiom a2: ~b
3
4 theorem "ejemplo" : ~(a | b)
5 proof
6   equivalently ~a & ~b
7   thus ~a by a1
8   thus ~b by a2
9 end
```

Razonamiento por el absurdo

```
theorem t: <form>
proof
  equivalently ~~<form>
  suppose <name>: ~<form>
  // Demostración de <form>
  // por el absurdo,
  // asumiendo ~<form>
  // y llegando a una
  // contradicción (false).
end
```

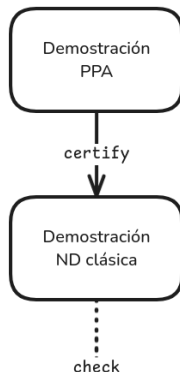
```
claim <name>: <form>
```

Permite demostrar una afirmación auxiliar. Útil para ordenar las demostraciones sin tener que definir otro teorema.

```
theorem t: <form1>
proof
  claim <name>: <form2>
  proof
    // Demostración de <form2>.
  end
  // Demostración de <form1> refiriéndose a <name>.
end
```


Certificador

- Las demostraciones de PPA se *certifican* generando una demostración de deducción natural.
- No deberían generarse demostraciones erróneas, pero son chequeadas independientemente como mecanismo de *fallback*.



Criterio de de Bruijn

Un asistente de demostración cumple con el criterio de de Bruijn si satisface que sus demostraciones puedan ser chequeadas por un programa independiente, pequeño y confiable.

Contexto global

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

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

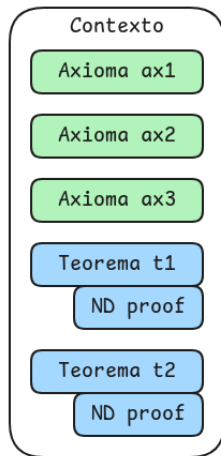


Figura: Contexto resultante de certificar un programa

El certificado de una demostración es recursivo: se certifica cada comando, generando una demostración en deducción natural cuyas premisas son el certificado del resto de la demostración en PPA.

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

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

Figura: Ejemplo de certificado generado para un programa

Contexto local

Cada demostración tiene un contexto local a ella con las hipótesis agregadas por ciertos comandos (**suppose**, **consider**, **have**, **claim**, etc.). Necesaria para obtener las fórmulas asociadas a las hipótesis en el **by**.

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

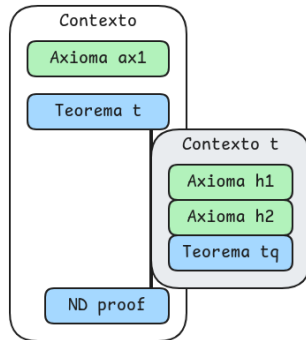


Figura: Ejemplo de contexto local

Certificado del by

Teniendo $\Gamma = \{h_1 : B_1, \dots, h_n : B_n\}$, para certificar

thus A **by** h_1, \dots, h_n :

- 1 Buscamos las hipótesis en el contexto. Queremos demostrar

$$B_1 \wedge \dots \wedge B_n \rightarrow A$$

- 2 **Razonamos por el absurdo**: Asumiendo la negación buscamos una contradicción

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

- 3 Convertimos la negación a forma normal disyuntiva (**DNF**)

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

- 4 Buscamos una **contradicción** refutando cada cláusula individualmente. Será refutable si

- Contiene \perp o dos fórmulas opuestas $(a, \neg a)$,
- Eliminando existenciales consecutivos y reiniciando el proceso, se consigue una refutación $(\neg p(k), \forall x.p(x))$

Ejemplo sin cuantificadores (1/2)

Tenemos el siguiente programa

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

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

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

- ❷ Negamos la fórmula

$$\neg[((a \rightarrow b) \wedge a) \rightarrow b]$$

Ejemplo sin cuantificadores (2/2)

3 La convertimos a DNF

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

4 Refutamos cada cláusula

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

Ejemplo con cuantificadores (1/3)

Tenemos el siguiente programa

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

- ❶ Para certificar **thus** q(a) **by** ax1, ax2 hay que generar una demostración para la implicación

$$\left(\left(\forall x. (p(x) \rightarrow q(x)) \right) \wedge p(a) \right) \rightarrow q(a)$$

- ❷ Negamos la fórmula

$$\neg \left[\left(\left(\forall x. (p(x) \rightarrow q(x)) \right) \wedge p(a) \right) \rightarrow q(a) \right]$$

Ejemplo con cuantificadores (2/3)

- 3 La convertimos a DNF

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

como a los ojos de DNF un \forall es opaco, a pesar de que dentro tenga una implicación, la fórmula ya está en forma normal.

- 4 Buscamos una contradicción refutando cada cláusula. No hay forma encontrando literales opuestos o \perp , por ej. la cláusula $p(a)$ no es refutable.

Ejemplo con cuantificadores (3/3)

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

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

- 6 Convertimos a DNF

$$\begin{aligned} & (p(u) \rightarrow q(u)) \wedge p(a) \wedge \neg q(a) \\ & \equiv (\neg p(u) \vee q(u)) \wedge p(a) \wedge \neg q(a) \\ & \equiv ((\neg p(u) \wedge p(a)) \vee (q(u) \wedge p(a))) \wedge \neg q(a) \\ & \equiv (\neg p(u) \wedge p(a) \wedge \neg q(a)) \vee \\ & \quad (q(u) \wedge p(a) \wedge \neg q(a)) \end{aligned}$$

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

Desafío

¡Hay que generar una demostración en deducción natural!

Pasos

- **Razonamiento por el absurdo:** mediante las *reglas admisibles* cut y eliminación de la doble negación ($E_{\neg\neg}$).
- **Conversión a DNF:** mediante la implementación de un *sistema de reescritura*.
- **Contradicciones:** mediante la *regla admisible* $E_{\wedge\varphi} + E_{\vee} + I_{\neg}$.
- **Eliminación de cuantificadores universales:** mediante unificación y E_{\forall} .

Razonamiento por el absurdo

$$\vdash B_1 \wedge \dots \wedge B_n \rightarrow A \overset{?}{\rightsquigarrow} \neg(B_1 \wedge \dots \wedge B_n \rightarrow A) \vdash \perp$$

Teorema (DNeg Elim)

$$\frac{\frac{}{\neg\neg A \vdash A}}{E_{\neg\neg}}$$

Teorema (cut)

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

Lema (Razonamiento por el absurdo)

$$\frac{\frac{\vdots}{\Gamma, \neg A \vdash \perp}}{\Gamma \vdash \neg\neg A} \text{ } I_{\neg} \quad \frac{\frac{\Gamma \vdash \neg\neg A}}{\Gamma \vdash A} \text{ } cut, E_{\neg\neg}$$

Conversión a DNF

Implementamos una traducción mediante el siguiente sistema de reescritura. **Algoritmo:** reescribir de a un paso hasta que no cambie (clausura de Kleene)

$$\neg\neg a \rightsquigarrow a$$

eliminación de $\neg\neg$

$$\neg\perp \rightsquigarrow \top$$

$$\neg\top \rightsquigarrow \perp$$

$$a \rightarrow b \rightsquigarrow \neg a \vee b$$

definición de implicación

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

distributiva de \neg sobre \wedge

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

distributiva de \neg sobre \vee

$$(a \vee b) \wedge c \rightsquigarrow (a \wedge c) \vee (b \wedge c)$$

distributiva de \wedge sobre \vee (der)

$$c \wedge (a \vee b) \rightsquigarrow (c \wedge a) \vee (c \wedge b)$$

distributiva de \wedge sobre \vee (izq)

$$a \vee (b \vee c) \rightsquigarrow (a \vee b) \vee c$$

asociatividad de \vee

$$a \wedge (b \wedge c) \rightsquigarrow (a \wedge b) \wedge c$$

asociatividad de \wedge

Conversión a DNF - Congruencias

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

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

Congruencias

$$A \vdash A' \Rightarrow A \wedge B \vdash A' \wedge B$$

$$A \vdash A' \Rightarrow A \vee B \vdash A' \vee B$$

$$A' \vdash A \Rightarrow \neg A \vdash \neg A'$$

\neg es contravariante

Para demostrar $\neg A \vdash \neg A'$ no necesitamos una demostración de $A \vdash A'$, sino de $A' \vdash A$.

\Rightarrow para todas las reescrituras, incluso las congruencias, tenemos que demostrarlas en ambos sentidos.

Conversión a DNF - Reglas admisibles

Reglas admisibles para conversión a DNF

Pasos base

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Ejemplo

$$\begin{array}{c}
 \text{Ax} \frac{}{(\neg a \wedge a \wedge \neg b)} \quad \Pi_L \quad \frac{\Gamma_1 \vdash b \wedge a \wedge \perp}{\Gamma, b \wedge a \wedge \perp \vdash \perp} \text{Ax} \\
 \frac{\Gamma \vdash \vee (b \wedge a \wedge \perp) \quad \Gamma, \neg a \wedge a \wedge \neg b \vdash \perp}{\Gamma = (\neg a \wedge a \wedge \neg b) \vee (b \wedge a \wedge \perp) \vdash \perp} \text{E}\vee \quad \text{E}\wedge_{\perp}
 \end{array}$$

donde

$$\Pi_L = \frac{\frac{\Gamma_1 \vdash \neg a \wedge a \wedge \neg b}{\Gamma_1 \vdash \neg a} \text{Ax} \quad \frac{\Gamma_1 \vdash \neg a \wedge a \wedge \neg b}{\Gamma_1 \vdash a} \text{Ax}}{\Gamma_1 = \Gamma, b \wedge a \wedge \perp \vdash \perp} \text{E}\neg$$

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

$$\frac{\Gamma \vdash \varphi_1 \wedge \dots \wedge \varphi_i \wedge \dots \wedge \varphi_n \quad n \in \mathbb{N}}{\Gamma \vdash \varphi_i} E\wedge_{\varphi_i}$$

- **Completo** para lógica proposicional y **heurístico** para primer orden.
- Esto es aceptable, la validez de LPO es indecidible (Teorema de Church).
- Elimina los \forall consecutivos de a lo sumo una hipótesis. Pero le faltan más cosas.

Ejemplo de falla en eliminación

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

(TODO: Agregar esto)

Extracción de testigos

Extracción simple

```
1 axiom ax: p(v)
2 theorem t: exists X . p(X)
3 proof
4   take X := v
5   thus p(v) by ax
6 end
```

Extraemos un testigo de $\exists X.p(X)$: v , tal que valga $p(v)$.

Extracción con instanciación

```
1 axiom ax: forall Y . p(Y, v)
2 theorem t:
3   forall X. exists V . p(X, V)
4 proof
5   let X
6   take V := v
7   thus p(X, v) by ax
8 end
```

Extraemos un testigo v y podemos instanciar X en lo que sea, por ej. la siguiente fórmula es demostrable: $p(f(n, m), v)$

Extracción indirecta

```
1  axiom ax1: q(m)
2  axiom ax2: forall X. q(X) -> p(X)
3
4  theorem t1: exists X. q(X)
5  proof
6      take X := m
7      thus q(m) by ax1
8  end
9
10 theorem t2: exists X. p(X)
11 proof
12     consider Y st h: q(Y) by t1
13     take X := Y
14     hence p(Y) by ax2
15 end
```

Extracción **indirecta** de **theorem** t2 nos da el testigo m.

Extracción por el absurdo (1/2)

Extracción por el absurdo

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


Extracción por el absurdo (2/2)

```
1 theorem hayAlguienBajo: exists X. bajo(X)
2 proof
3   equivalently ~~exists X. bajo(X)
4   suppose h1: ~exists X. bajo(X)
5
6   claim h2: forall X. ~bajo(X)
7   proof
8     let X
9     suppose h3: bajo(X)
10    claim h4: exists Y. bajo(Y)
11    proof
12      take Y := X
13      hence bajo(X) by h3
14    end
15    hence false by h1
16  end
17  hence false by noTodoElMundoEsAlto
18 end
```

Sin **take** (\exists) explícito,
igual podemos extraer
el testigo a partir del
theorem
hayAlguienBajo: juan.

Buscamos un mecanismo general tal que,

Mecanismo de extracción de testigos

A partir de una demostración en PPA para una fórmula de la forma

$$\forall x_0 \dots \forall x_n \exists y. \varphi(x_0, \dots, x_n, y),$$

- 1 la certifique generando una demostración en deducción natural **clásica**
- 2 a partir de ella extraiga u tal que, para t_0, \dots, t_n cuales quiera, valga

$$\varphi(t_0, \dots, t_n, u)$$

La lógica clásica **no es constructiva** por LEM

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

Demostración no constructiva

Ejemplo (Fórmula sin demostración constructiva)

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

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

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

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

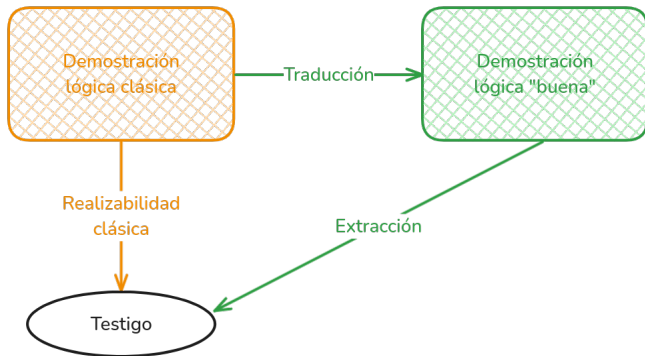
¡No nos dice cual es cierto! No es *constructiva*. No tenemos forma de saber si es cierto C o $\neg C$ (indecidible).

Teorema

Existen dos números irracionales a y b tales que a^b es racional.

Existen fórmulas que admiten demostraciones constructivas y no constructivas, y otras solo no constructivas.

Clases de estrategias de extracción



Clases de estrategias de extracción de demostraciones en lógica clásica:

- **Directas:** Extraer directamente de demostraciones clásicas. Técnicas de *realizabilidad clásica* (Semánticas de λ -cálculos clásicos).
- **Indirectas:** Convertir la demostración a una lógica que se porte mejor y extraer de ahí.

lógica intuicionista = lógica clásica – LEM

Características:

- No tiene LEM³, entonces siempre es constructiva.
- Noción de *forma normal* buena: una demostración de un \exists debería comenzar con $\text{I}\exists$:

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

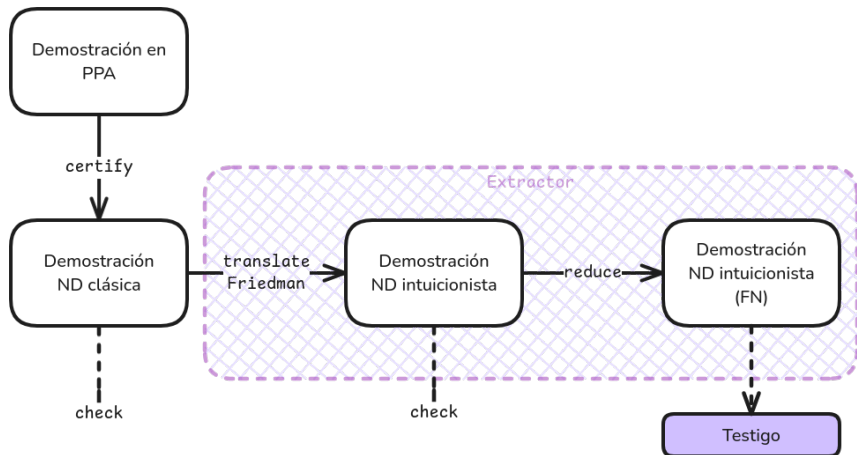
- Proceso de normalización análogo a reducción de λ -cálculo (su semántica operacional), visto desde el isomorfismo Curry-Howard.

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

- La **traducción de Friedman** permite embeber la lógica clásica en la intuicionista, para *algunas* fórmulas.
- Traduce demostraciones de fórmulas Π_2 : de la forma

$$\forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n)$$

Estrategia de extracción indirecta - Aporte principal



Traducción de doble negación relativizada

Definición (Traducción de doble negación relativizada)

Sea $\neg_R A \equiv A \rightarrow R$, se define la traducción de doble negación relativizada:

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

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

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

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

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

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

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

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

Teorema

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

Traducción de doble negación relativizada

Definición (Traducción de doble negación relativizada)

Sea $\neg_R A \equiv A \rightarrow R$, se define la traducción de doble negación relativizada:

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

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

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

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

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

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

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

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

Teorema

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

Teorema (Traducción de Friedman)

Sea Π una demostración clásica de $\forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n)$, y φ una fórmula conjuntiva. Si tenemos

$$\Gamma \vdash_C \forall y_1 \dots \forall y_n. \exists x. \varphi(x, y_1, \dots, y_n),$$

podemos generar una demostración intuicionista de la misma fórmula

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

Definición (Fórmulas conjuntivas)

Generadas por

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

El truco de Friedman

Teorema

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

Inducción estructural sobre la demostración.
mostrar: introducción del and, LEM, lemas

- evitar desvíos superfluos (mostrar ejemplo). Simplificamos hasta que esté en forma normal - isomorfismo curry howard

- regla de $E \rightarrow$, motivar sustituciones sin captura

- estrategia original, lenta - estrategia gross knuth

- lenta (reinicia focos de evaluación) - incompleta (reglas permutativas)

Manteniendo el contexto

Animación de la reducción

Detalles de implementación

- modulos (para dar una idea de la cantidad y complejidad del código) -
- haskell - comandos check y extract

diagrama para los dos caminos para hacer un parser. - happy y alex

Conclusiones y trabajo futuro

Conclusiones

