Introducción a la correspondencia entre pruebas y programas:

Relaciones entre HA y PA

Alexandre Miquel

abril de 2021

Plan

1. Recursión primitiva en HA y PA
2. Propiedades de HA y PA "con lenguaje amplio"
3. Traducción negativa de Gödel-Gentzen
4. *R*-traducción de Friedman

Plan

Introducción

- En las clases anteriores, presentamos PA (Aritmética clásica) y HA (Aritmética intuicionista) con un lenguaje mínimo de términos:

Términos $t, u \quad ::= \quad x \mid 0 \mid s(t) \mid t + u \mid t \times u$

- Sin embargo, el lenguaje de fórmulas de HA/PA permite representar mucho más funciones, como por ejemplo la función $(n, m) \mapsto n^m$, y más generalmente: todas las funciones recursivas primitivas

- **Objetivo de esta parte:** Definir las funciones recursivas primitivas y ver cómo se pueden integrar en el lenguaje (y la teoría) de HA/PA

Funciones iniciales

Se llaman funciones iniciales a las siguientes funciones:

- La **función nula** $z : \mathbb{N} \to \mathbb{N}$, definida por

$$z(n) := 0$$

(para todo $n \in \mathbb{N}$)

- La **función sucesor** $s : \mathbb{N} \to \mathbb{N}$, definida por

$$s(n) := n + 1$$

(para todo $n \in \mathbb{N}$)

- Las **proyecciones** $\pi_i^k : \mathbb{N}^k \to \mathbb{N}$ $(k \geq i \geq 1)$, definidas por

$$\pi_i^k(n_1, \ldots, n_k) := n_i$$

(para todo $(n_1, \ldots, n_k) \in \mathbb{N}^k$)

Esquemas de composición y de recursión primitiva

- **Esquema de composición** A partir de $f_1, \ldots, f_p : \mathbb{N}^k \to \mathbb{N}$ y $g : \mathbb{N}^p \to \mathbb{N}$, definir la función $h : \mathbb{N}^k \to \mathbb{N}$ por

$$h(n_1, \ldots, n_k) := g(f_1(n_1, \ldots, n_k), \ldots, f_p(n_1, \ldots, n_k))$$

(para todo $(n_1, \ldots, n_k) \in \mathbb{N}^k$)

Se nota $h = g \circ (f_1, \ldots, f_p)$

- **Esquema de recursión primitiva** A partir de $f : \mathbb{N}^k \to \mathbb{N}$ y $g : \mathbb{N}^{k+2} \to \mathbb{N}$, definir la función $h : \mathbb{N}^{k+1} \to \mathbb{N}$ por:

$$\begin{aligned} h(0, n_1, \ldots, n_k) &:= f(n_1, \ldots, n_k) \\ h(n+1, n_1, \ldots, n_k) &:= g(n, h(n, n_1, \ldots, n_k), n_1, \ldots, n_k) \end{aligned}$$

(para todos $n \in \mathbb{N}$ y $(n_1, \ldots, n_k) \in \mathbb{N}^k$)

Se nota $h = \mathbf{rec}(f, g)$

Funciones recursivas primitivas (1/3)

Definición (Funciones recursivas primitivas)

El conjunto de las funciones recursivas primitivas es el mínimo conjunto $\subseteq \bigcup_{k\geq 1}(\mathbb{N}^k \to \mathbb{N})$ que contiene todas las funciones iniciales

$$\boldsymbol{z}, \qquad \boldsymbol{s}, \qquad \boldsymbol{\pi}_i^k \quad (k \geq i \geq 1)$$

y está cerrado por los esquemas de composición y de recursión primitiva:

$$(f_1, \ldots, f_p, g) \mapsto g \circ (f_1, \ldots, f_p), \qquad (f, g) \mapsto \mathbf{rec}(f, g)$$

Notación: $\mathbf{PR}_k$ = conjunto de las funciones recursivas primitivas de aridad k

También se puede definir inductivamente la familia $(\mathbf{PR}_k)_{k\geq 1}$ por las reglas:

$$\frac{}{\boldsymbol{z} \in \mathbf{PR}_1} \qquad \frac{}{\boldsymbol{s} \in \mathbf{PR}_1} \qquad \frac{}{\boldsymbol{\pi}_i^k \in \mathbf{PR}_k}\ (k\geq i\geq 1)$$

$$\frac{f_1 \in \mathbf{PR}_k \quad \cdots \quad f_p \in \mathbf{PR}_k \quad g \in \mathbf{PR}_p}{g \circ (f_1, \ldots, f_p) \in \mathbf{PR}_k} \qquad \frac{f \in \mathbf{PR}_k \quad g \in \mathbf{PR}_{k+2}}{\mathbf{rec}(f, g) \in \mathbf{PR}_{k+1}}$$

Funciones recursivas primitivas (2/3)

Observaciones:

- Las funciones aritméticas básicas son recursivas primitivas:

$$n \mapsto n+1, \qquad (n,m) \mapsto n+m, \qquad (n,m) \mapsto nm,$$
$$(n,m) \mapsto n^m, \qquad (n,m) \mapsto \min(n,m), \qquad (n,m) \mapsto \max(n,m),$$
$$(n,m) \mapsto n \dot{-} m \quad \text{(resta truncada)}, \quad (n,m) \mapsto n \div m \quad \text{(div. euclidiana)},$$
$$(n,m) \mapsto n \,\%\, m \quad \text{(resto euclidiano)}, \quad n \mapsto n!, \quad n \mapsto \mathrm{fib}(n), \quad \text{etc.}$$

- Si $f : \mathbb{N}^k \to \mathbb{N}$ es recursiva primitiva, entonces las funciones $g, h : \mathbb{N}^k \to \mathbb{N}$ definidas por:

$$g(n, n_2, \ldots, n_k) := \sum_{m \leq n} f(m, n_2, \ldots, n_k)$$

$$h(n, n_2, \ldots, n_k) := \prod_{m \leq n} f(m, n_2, \ldots, n_k)$$

son recursivas primitivas

Funciones recursivas primitivas (3/3)

Observaciones (continuación):

- Todas las funciones recursivas primitivas son totales y computables[1]
- En particular, el conjunto $\mathbf{PR}_k$ es numerable (para todo $k \geq 1$)
- Sin embargo, existen funciones totales y computables que no son recursivas primitivas, como por ejemplo la función de Ackermann ack : $\mathbb{N}^2 \to \mathbb{N}$ definida por:

$$\begin{aligned} \text{ack}(0, n) &:= n + 1 \\ \text{ack}(m + 1, 0) &:= \text{ack}(m, 1) \\ \text{ack}(m + 1, n + 1) &:= \text{ack}(m, \text{ack}(m + 1, n)) \end{aligned}$$

- Se puede demostrar que toda función total y computable cuya complejidad está acotada (superiormente) por una función recursiva primitiva también es recursiva primitiva

[1] Definiremos la noción de función computable en el curso sobre el cálculo lambda

Representación de las funciones recursivas primitivas

En lo siguiente, se escribe $\overline{n} :\equiv s^n(0)$ al entero de Peano n

Teorema (Representación de las funciones recursivas primitivas)

Para toda función recursiva primitiva $f : \mathbb{N}^k \to \mathbb{N}$:

(1) Existe una fórmula aritmética $A_f(x_1, \ldots, x_k, y)$ tal que

$$\mathrm{HA} \vdash \forall y \left(A(\overline{n_1}, \ldots, \overline{n_k}, y) \Leftrightarrow y = \overline{f(n_1, \ldots, n_k)} \right)$$

para todo $(n_1, \ldots, n_k) \in \mathbb{N}^k$

(2) $\mathrm{HA} \vdash \forall x_1 \cdots \forall x_k \, \exists! y \, A_f(x_1, \ldots, x_k, y)$

Observaciones:

- (1) expresa que la función f es representable en HA por la fórmula $A_f(x_1, \ldots, x_k, y)$. Se puede demostrar más generalmente que todas las funciones computables (parciales o totales) son representables en HA
- (2) expresa que la totalidad de la función f se puede demostrar en HA. Esto no se cumple para todas las funciones computables, ni siquiera para todas las funciones computables totales

Integración de las funciones recursivas primitivas

Como todas las funciones recursivas primitivas son representables en HA, es natural integrarlas en el lenguaje.

Dos opciones posibles:

1. Añadir un nuevo símbolo de función para cada definición recursiva primitiva, usando el mecanismo de extensión definicional
2. Cambiar el lenguaje de términos para integrar de modo primitivo un mecanismo de recursión primitiva

En ambos casos, la extensión construida (de HA/PA) es conservativa

Extensiones definicionales (recordatorio)

Definición (Extensión definicional)

Sea $\mathscr{T}$ una teoría clásica (resp. intuicionista) con un teorema de la forma

$$\mathscr{T} \vdash \forall x_1 \cdots \forall x_k \, \exists! y \, A(x_1, \ldots, x_k, y)$$

Se llama extensión definicional de $\mathscr{T}$ (con respecto al teorema anterior) a la teoría $\mathscr{T}'$ clásica (resp. intuicionista) obtenida añadiendo a $\mathscr{T}$ un nuevo símbolo de función f de aridad k con el nuevo axioma:

$$\forall x_1 \cdots \forall x_k \, A(x_1, \ldots, x_k, f(x_1, \ldots, x_k)) \qquad (\in \mathrm{Ax}(\mathscr{T}'))$$

Teorema (Conservatividad de las extensiones definicionales)

Toda extensión definicional $\mathscr{T}' \supseteq \mathscr{T}$ es conservativa, en lógica clásica como en lógica intuicionista

Obs.: La demostración (constructiva) se basa en la definición de una traducción $B \mapsto B^*$ del lenguaje de $\mathscr{T}'$ en el lenguaje de $\mathscr{T}$ que elimina el nuevo símbolo de función f, remplazando cada ocurrencia por su «definición» $A(x_1, \ldots, x_k, y)$

Opción 1: añadiendo nuevos símbolos de función (1/2)

Se define el nuevo lenguaje de HA/PA del siguiente modo:

- Se considera la familia $(\mathscr{F})_{k\geq 1}$ de conjuntos de símbolos de función (uno para cada aridad $k \geq 1$) definida inductivamente por las reglas:

$$\frac{}{\boldsymbol{z} \in \mathscr{F}_1} \qquad \frac{}{\boldsymbol{s} \in \mathscr{F}_1} \qquad \frac{}{\boldsymbol{\pi}_i^k \in \mathscr{F}_k}\ (k\geq i\geq 1)$$

$$\frac{f_1 \in \mathscr{F}_k \quad \cdots \quad f_p \in \mathscr{F}_k \quad g \in \mathscr{F}_p}{g \circ (f_1, \ldots, f_p) \in \mathscr{F}_k} \qquad \frac{f \in \mathscr{F}_k \quad g \in \mathscr{F}_{k+2}}{\mathbf{rec}(f, g) \in \mathscr{F}_{k+1}}$$

Obs.: Ahora, las notaciones $\boldsymbol{z}$, $\boldsymbol{s}$, $\boldsymbol{\pi}_i^k$, etc. designan símbolos

- Se definen los términos del nuevo lenguaje por:

Términos $\quad t, u \quad ::= \quad x \quad | \quad 0 \quad | \quad f(t_1, \ldots, t_k) \qquad (f \in \mathscr{F}_k)$

- No cambia la gramática de las fórmulas

Opción 1: añadiendo nuevos símbolos de función (2/2)

En este nuevo lenguaje, los axiomas de HA/PA son los siguientes:

(1) $\forall x\,(\boldsymbol{z}(x) = 0)$

(2) $\forall x\,(\boldsymbol{s}(x) \neq 0)$

(3) $\forall x_1 \cdots \forall x_k\,(\boldsymbol{\pi}_i^k(x_1, \ldots, x_k) = x_i)$ $\qquad (k \geq i \geq 1)$

(4) $\forall x_1 \cdots \forall x_k\,\big(h(x_1, \ldots, x_k) = g(f_1(x_1, \ldots, x_k), \ldots, f_p(x_1, \ldots, x_k))\big)$
para cada símbolo $h \in \mathscr{F}_k$ de la forma $h \equiv g \circ (f_1, \ldots, f_p)$

(5) $\begin{cases} \forall x_1 \cdots \forall x_k\,\big(h(0, x_1, \ldots, x_k) = f(x_1, \ldots, x_k)\big) \\ \forall x\,\forall x_1 \cdots \forall x_k\,\big(h(\boldsymbol{s}(x), x_1, \ldots, x_k) = g(x, h(x, x_1, \ldots, x_k), x_1, \ldots, x_k)\big) \end{cases}$
para cada símbolo $h \in \mathscr{F}_{k+1}$ de la forma $h \equiv \mathbf{rec}(f, g)$

(6) Esquema de inducción (formulación usual)

Obs.: Como la función predecesor (recursiva primitiva) está en el lenguaje, ya se puede derivar que el sucesor es inyectivo sin axioma específico

Opción 2: la recursión primitiva en los términos (1/2)

En lugar de introducir nuevos símbolos de funciones, se añade un mecanismo de recursión primitiva en el lenguaje de términos:

Términos $t, u \quad ::= \quad x \mid 0 \mid s(t) \mid \mathtt{rec}(t, u, (x, y)u')$

- En la construcción $\mathtt{rec}(t, u, (x, y)u')$:
  - t es el término sobre el cual se hace la recursión
  - u es el término inicial
  - $(x, y)u'$ es el término de iteración, expresado en función de las variables x (entero anterior) e y (llamada recursiva)

  Formalmente, tenemos que:

$$FV(\mathtt{rec}(t, u, (x, y)u')) = FV(t) \cup FV(u) \cup (FV(u') \setminus \{x, y\})$$

- Como en la opción anterior, las fórmulas no cambian

¡Cuidado! Debido a la presencia de la construcción $\mathtt{rec}(t, u, (x, y)u')$ (símbolo ligador) en los términos, este lenguaje ya no es un lenguaje de primer orden

Opción 2: la recursión primitiva en los términos (2/2)

En este nuevo lenguaje, los axiomas de HA/PA son los siguientes:

(1) $\forall x\,(s(x) \neq 0)$

(2) $\begin{cases} \forall \vec{z}\,(\mathtt{rec}(0,\ u,\ (x,y)u') = u) \\ \forall \vec{z}\,(\mathtt{rec}(s(t),\ u,\ (x,y)u') = u'[x := t][y := \mathtt{rec}(t,\ u,\ (x,y)u')]) \end{cases}$

para todos t, u, u' tales que $FV(t,u) \subseteq \{\vec{z}\}$, $FV(u') \subseteq \{x, y, \vec{z}\}$

(3) Esquema de inducción (formulación usual)

Ejemplos: Se definen $t + u$, $t \times u$, $\mathrm{pred}(t)$ y $t!$ por:

$$\begin{aligned} t + u &:\equiv \mathtt{rec}(u,\ t,\ (x,y)s(y)) \\ t \times u &:\equiv \mathtt{rec}(u,\ 0,\ (x,y)(y + t)) \\ \mathrm{pred}(t) &:\equiv \mathtt{rec}(t,\ 0,\ (x,y)x) \\ t! &:\equiv \mathtt{rec}(t,\ 1,\ (x,y)(x \times y)) \end{aligned}$$

Ejercicio: Verificar que estas definiciones cumplen las identidades deseadas

Observaciones

- Muy frecuentemente en la literatura, se supone que el lenguaje de HA/PA provee un símbolo de función para cada (definición de) función recursiva primitiva (Opción 1)

  $\rightsquigarrow$ Esta presentación "con lenguaje amplio" es conservativa con respecto a la presentación "con lenguaje mínimo"

- Trabajar "con lenguaje amplio" tiene muchas ventajas...
  - En la práctica: comodidad, expresividad, etc.
  - En la teoría: definición de la jerarquía aritmética (véase más adelante)

- También se puede definir una aritmética computacional $\mathrm{HA}^{\cong}$ "con lenguaje amplio": basta con integrar las ecuaciones definientes de todas las funciones recursivas primitivas en la reducción $t \succ t'$
  - La reducción (extendida) sobre los términos y las fórmulas sigue siendo confluente y fuertemente normalizante
  - El teorema de eliminación de cortes se extiende naturalmente a la aritmética computacional "con lenguaje amplio"

Plan

Fórmulas recursivas primitivas (1/4)

A partir de ahora, se trabaja sólo en HA/PA "con lenguaje amplio"

Definición (Fórmula sin cuantificadores)

Una fórmula es sin cuantificadores cuando no contiene ningún $\forall$ o $\exists$

Más generalmente, se definen las cuantificaciones acotadas por:

$$\begin{aligned}(\forall x \leq t)A(x) &:\equiv \forall x\,(x \leq t \Rightarrow A(x))\\ (\exists x \leq t)A(x) &:\equiv \exists x\,(x \leq t \wedge A(x))\end{aligned} \qquad (x \notin FV(t))$$

Definición (Fórmula con cuantificaciones acotadas)

Una fórmula es con cuantificaciones acotadas (c.c.a.) cuando todas las cuantificaciones que ocurren en dicha fórmula están acotadas, es decir:

$$\begin{aligned}\textbf{Fórmulas c.c.a.}\quad A, B \quad ::= \quad & t = u \mid \top \mid \bot\\ \mid\ & A \Rightarrow B \mid A \wedge B \mid A \vee B\\ \mid\ & (\forall x \leq t)A \mid (\exists x \leq t)A \qquad (x \notin FV(t))\end{aligned}$$

Intución: Fórmula c.c.a. = fórmula "con información finita"

Fórmulas recursivas primitivas (2/4)

Definición (Fórmulas recursivas primitivas)

Una fórmula $A(x_1, \ldots, x_k)$ con variables libres en $\{x_1, \ldots, x_k\}$ es recursiva primitiva cuando existe $f \in \mathbf{PR}_k$ tal que

$$\mathrm{HA} \vdash \forall x_1 \cdots \forall x_k \left(A(x_1, \ldots, x_k) \Leftrightarrow f(x_1, \ldots, x_k) = 0 \right)$$

Obs.: Como la igualdad es decidible en HA, cada fórmula $A(x_1, \ldots, x_k)$ recursiva primitiva también es decidible en HA:

$$\mathrm{HA} \vdash \forall x_1 \cdots \forall x_k \left(A(x_1, \ldots, x_k) \vee \neg A(x_1, \ldots, x_k) \right)$$

(¡Cuidado! Hay fórmulas decidibles que no son recursivas primitivas)

Teorema (Reducción de complejidad)

En HA (con lenguaje amplio), todas las fórmulas con cuantificaciones acotadas son recursivas primitivas (y luego decidibles)

Dicho de otro modo: Se puede remplazar cualquier fórmula $A(\vec{x})$ con cuantificaciones acotadas por una fórmula atómica de la forma $f(\vec{x}) = 0$

Fórmulas recursivas primitivas (3/4)

Demostración. Por inducción sobre la fórmula c.c.a. $A(x_1, \ldots, x_k)$ se construye $f_A \in \mathbf{PR}_k$ tal que $\mathsf{HA} \vdash \forall x_1 \cdots \forall x_k \, (A(x_1, \ldots, x_k) \Leftrightarrow f_A(x_1, \ldots, x_k) = 0)$, distinguiendo los siguientes casos:

- $A(x_1, \ldots, x_k) \equiv t(x_1, \ldots, x_k) = u(x_1, \ldots, x_k)$. Basta con tomar

$$f_A(x_1, \ldots, x_k) := (t(x_1, \ldots, x_k) \dot{-} u(x_1, \ldots, x_k)) + (u(x_1, \ldots, x_k) \dot{-} t(x_1, \ldots, x_k))$$

- $A(x_1, \ldots, x_k) \equiv \top$. Basta con tomar $f_A(x_1, \ldots, x_k) := 0$
- $A(x_1, \ldots, x_k) \equiv \bot$. Basta con tomar $f_A(x_1, \ldots, x_k) := 1$
- $A(x_1, \ldots, x_k) \equiv B(x_1, \ldots, x_k) \wedge C(x_1, \ldots, x_k)$. Basta con tomar

$$f_A(x_1, \ldots, x_k) := f_B(x_1, \ldots, x_k) + f_C(x_1, \ldots, x_k)$$

- $A(x_1, \ldots, x_k) \equiv B(x_1, \ldots, x_k) \vee C(x_1, \ldots, x_k)$. Basta con tomar

$$f_A(x_1, \ldots, x_k) := f_B(x_1, \ldots, x_k) \times f_C(x_1, \ldots, x_k)$$

(...)

Fórmulas recursivas primitivas (4/4)

Demostración (continuación y fin).

- $A(x_1,\ldots,x_k) \equiv B(x_1,\ldots,x_k) \Rightarrow C(x_1,\ldots,x_k)$. Basta con tomar

$$f_A(x_1,\ldots,x_k) := (1 \dot{-} f_B(x_1,\ldots,x_k)) \times f_C(x_1,\ldots,x_k)$$

- $A(x_1,\ldots,x_k) \equiv (\forall y \leq t(x_1,\ldots,x_k))B(y,x_1,\ldots,x_k)$. Basta con tomar

$$f_A(x_1,\ldots,x_k) := \sum_{y \leq t(x_1,\ldots,x_k)} f_B(y,x_1,\ldots,x_k)$$

- $A(x_1,\ldots,x_k) \equiv (\exists y \leq t(x_1,\ldots,x_k))B(y,x_1,\ldots,x_k)$. Basta con tomar

$$f_A(x_1,\ldots,x_k) := \prod_{y \leq t(x_1,\ldots,x_k)} f_B(y,x_1,\ldots,x_k)$$

□

Obs.: La demostración anterior muestra cómo calcular de modo efectivo el valor de verdad de cualquier fórmula con cuantificaciones acotadas (en el modelo estándar) mediante una función recursiva primitiva adecuada (con la convención $\mathcal{V} = 0$ y $\mathcal{F} \neq 0$)

Formas prenexas (LK)

Sea $\mathscr{L}$ un lenguaje de primer orden cualquiera

Definición (Forma prenexa)

Una formula $A \in \mathscr{L}$ está en forma prenexa cuando es de la forma

$$A \equiv Qx_1 \cdots Qx_n\, A_0$$

con $Q_1, \ldots, Q_n \in \{\forall, \exists\}$ y donde $A_0 \in \mathscr{L}$ es sin cuantificadores

Proposición (Existencia de la forma prenexa)

Toda fórmula $A \in \mathscr{L}$ es clásicamente equivalente a una fórmula en forma prenexa:

$$\vdash_{\mathsf{NK}} A \Leftrightarrow Qx_1 \cdots Qx_n\, A_0$$

para algunos $n \in \mathbb{N}$, $Q_1, \ldots, Q_n \in \{\forall, \exists\}$ y $A_0 \in \mathscr{L}$ sin cuantificadores

Demostración. Por inducción sobre A. □

¡Cuidado! Este resultado sólo se cumple en lógica clásica

La jerarquía aritmética (LK) (1/3)

Para toda fórmula aritmética A (abierta o cerrada), se observa que:

- o bien $A \Leftrightarrow \exists \vec{x}_1 \, \forall \vec{x}_2 \, \exists \vec{x}_3 \, \cdots \, \overline{Q} \vec{x}_n \, A_0$ (clase Σ^0_n)
- o bien $A \Leftrightarrow \forall \vec{x}_1 \, \exists \vec{x}_2 \, \forall \vec{x}_3 \, \cdots \, Q \vec{x}_n \, A_0$ (clase Π^0_n)

donde A_0 es una fórmula sin cuantificadores (o con cuantificaciones acotadas)

Definición (Clases de complejidad Σ^0_n, Π^0_n, Δ^0_n)

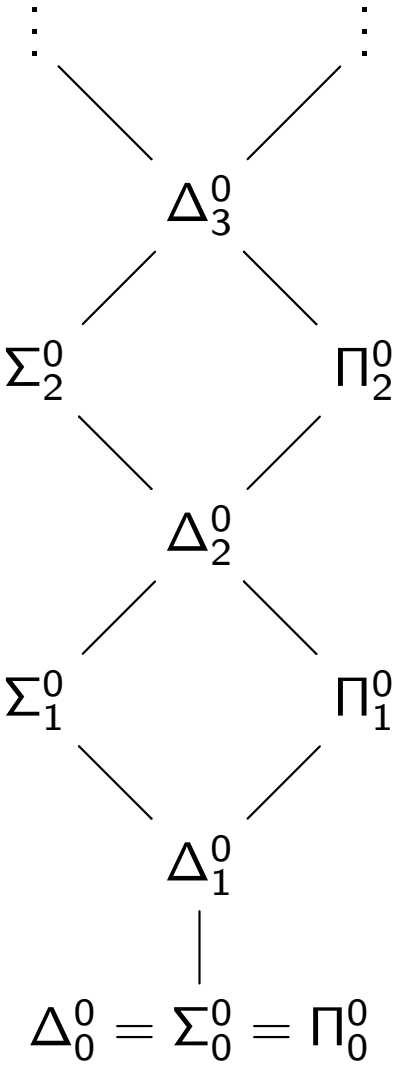
Para todo $n \in \mathbb{N}$, se definen los conjuntos de fórmulas Σ^0_n y Π^0_n por:

$$
\begin{array}{rcll}
A \in \Sigma^0_0 = \Pi^0_0 & \text{sii} & \mathrm{PA} \vdash A \Leftrightarrow A_0 & \text{para alguna fórmula } A_0 \text{ c.c.a.} \\
A \in \Sigma^0_{n+1} & \text{sii} & \mathrm{PA} \vdash A \Leftrightarrow \exists \vec{x} \, B & \text{para alguna fórmula } B \in \Pi^0_n \\
A \in \Pi^0_{n+1} & \text{sii} & \mathrm{PA} \vdash A \Leftrightarrow \forall \vec{x} \, B & \text{para alguna fórmula } B \in \Sigma^0_n
\end{array}
$$

Además, se escribe: $\Delta^0_n := \Sigma^0_n \cap \Pi^0_n$

$$
\Sigma/\Pi/\Delta^0_n \quad \left\{ \begin{array}{l} \textbf{Superíndice } 0 = \text{nivel de la aritmética} \\ \textbf{Subíndice } n = \text{número de ``bloques'' de cuantificaciones} \\ \boldsymbol{\Sigma} = \text{empieza con } \exists \; / \; \boldsymbol{\Pi} = \text{empieza con } \forall \; / \; \boldsymbol{\Delta} = \boldsymbol{\Sigma} \cap \boldsymbol{\Pi} \end{array} \right.
$$

La jerarquía aritmética (LK) (2/3)



- Inclusiones: $\Sigma^0_n, \Pi^0_n \subset \Delta^0_{n+1} \subset \Sigma^0_{n+1}, \Pi^0_{n+1}$
- Las clases Σ^0_n, Π^0_n, Δ^0_n están cerradas por $\wedge$, $\vee$ y por las cuantificaciones acotadas
- Además Δ^0_n está cerrada por $\neg$
- La negación $\neg$ intercambia Σ^0_n y Π^0_n
- Mediante una biyección primitiva recursiva entre $\mathbb{N}^k$ y $\mathbb{N}$, se pueden agrupar cuantificaciones de misma naturaleza:

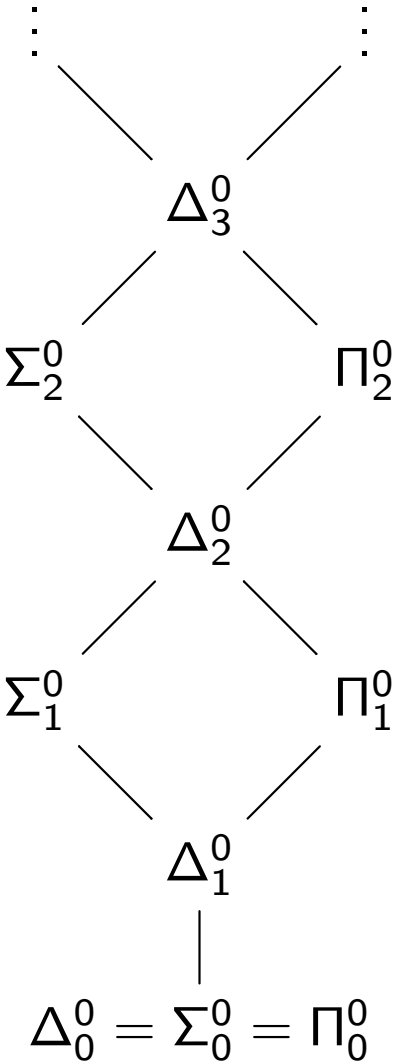
$$Qx_1 \cdots Qx_k \; A(x_1, \ldots, x_k) \Leftrightarrow Qx \, A(\pi^k_1(x), \ldots, \pi^k_k(x))$$

- Por lo tanto:

$A(\vec{z}) \in \Sigma^0_n$ sii
$A(\vec{z}) \Leftrightarrow \exists x_1 \, \forall x_2 \, \exists x_3 \cdots Qx_n \, f(x_1, \ldots, x_n, \vec{z}) = 0$

$A(\vec{z}) \in \Pi^0_n$ sii
$A(\vec{z}) \Leftrightarrow \forall x_1 \, \exists x_2 \, \forall x_3 \cdots \overline{Q}x_n \, f(x_1, \ldots, x_n, \vec{z}) = 0$

La jerarquía aritmética (LK) (3/3)



A través del modelo estándar, se puede ver cada fórmula aritmética $A(\vec{x})$ como un subconjunto

$$\{\vec{n} \in \mathbb{N}^k : \mathbb{N} \models A(\vec{n})\} \subseteq \mathbb{N}^k$$

Por lo tanto, también se puede ver cada clase de complejidad (Σ^0_n, Π^0_n, Δ^0_n) de la jerarquía aritmética como una clase de complejidad de conjuntos $\subseteq \mathbb{N}^k$

- Π^0_1 = conjuntos co-recursivamente enumerables
- Σ^0_1 = conjuntos recursivamente enumerables
- Δ^0_1 = conjuntos recursivos
- $\Delta^0_0 = \Sigma^0_0 = \Delta^0_0$ = conjuntos rec. primitivos

Plan

La traducción negativa de Gödel-Gentzen

Sea $\mathscr{L}$ un lenguaje de primer orden cualquiera

Definición (Traducción $A \mapsto A^{\mathbf{G}}$ de Gödel-Gentzen)

A cada fórmula $A \in \mathscr{L}$ se asocia otra fórmula $A^{\mathbf{G}} \in \mathscr{L}$ definida por:

$$
\begin{array}{rcll}
(p(t_1,\ldots,t_k))^{\mathbf{G}} & :\equiv & \neg\neg p(t_1,\ldots,t_k) & \\
\top^{\mathbf{G}} & :\equiv & \top & \\
\bot^{\mathbf{G}} & :\equiv & \bot & \\
(\neg A)^{\mathbf{G}} & :\equiv & \neg A^{\mathbf{G}} & \\
(A \Rightarrow B)^{\mathbf{G}} & :\equiv & A^{\mathbf{G}} \Rightarrow B^{\mathbf{G}} & \\
(A \wedge B)^{\mathbf{G}} & :\equiv & A^{\mathbf{G}} \wedge B^{\mathbf{G}} & \\
(A \vee B)^{\mathbf{G}} & :\equiv & \neg(\neg A^{\mathbf{G}} \wedge \neg B^{\mathbf{G}}) & \left(\underset{\mathrm{LJ}}{\Leftrightarrow} \quad \neg\neg(A^{\mathbf{G}} \vee B^{\mathbf{G}})\right) \\
(\forall x\, A)^{\mathbf{G}} & :\equiv & \forall x\, A^{\mathbf{G}} & \\
(\exists x\, A)^{\mathbf{G}} & :\equiv & \neg\forall x\, \neg A^{\mathbf{G}} & \left(\underset{\mathrm{LJ}}{\Leftrightarrow} \quad \neg\neg\exists x\, A^{\mathbf{G}}\right)
\end{array}
$$

Intuición: $\vee$ clásico $=$ doble negación del $\vee$ intuicionista
$\exists$ clásico $=$ doble negación del $\exists$ intuicionista

Propiedades de la traducción $A \mapsto A^{\mathbf{G}}$

Obs.: $FV(A^{\mathbf{G}}) = FV(A)$

Proposición (Propiedades de la traducción $A \mapsto A^{\mathbf{G}}$)

Fijado un lenguaje de primer orden $\mathscr{L}$:

(1) $\vdash_{\mathsf{NK}} A^{\mathbf{G}} \Leftrightarrow A$ (para todo $A \in \mathscr{L}$)

(2) $\vdash_{\mathsf{NJ}} A^{\mathbf{G}} \Leftrightarrow \neg\neg A^{\mathbf{G}}$ (para todo $A \in \mathscr{L}$)

(3) Si $\Gamma \vdash_{\mathsf{NK}} A$, entonces $\Gamma^{\mathbf{G}} \vdash_{\mathsf{NJ}} A^{\mathbf{G}}$

(4) En particular: $\vdash_{\mathsf{NK}} A$ implica $\vdash_{\mathsf{NJ}} A^{\mathbf{G}}$

Demostración.

(1) Por inducción sobre la fórmula A (ejercicio)

(2) Por inducción sobre la fórmula A (ejercicio)

(3) Por inducción sobre la derivación de $\Gamma \vdash A$ en el sistema NK, usando (2) para interpretar la regla del absurdo (ejercicio)

(4) Sigue inmediatamente de (3) □

El caso de la Aritmética

- Se observa que $\mathrm{HA} \vdash A^{\mathbf{G}}$ para todo axioma $A \in \mathrm{Ax}(\mathrm{PA})$

  (1) $\mathrm{HA} \vdash \forall x\, \neg\neg(x + 0 = x)$

  (2) $\mathrm{HA} \vdash \forall x\, \forall y\, \neg\neg(x + s(y) = s(x + y))$

  (3) $\mathrm{HA} \vdash \forall x\, \neg\neg(x \times 0 = 0)$

  (4) $\mathrm{HA} \vdash \forall x\, \forall y\, \neg\neg(x \times s(y) = x \times y + x)$

  (5) $\mathrm{HA} \vdash \forall x\, \forall y\, (\neg\neg(s(x) = s(y)) \Rightarrow \neg\neg(x = y))$

  (6) $\mathrm{HA} \vdash \forall x\, \neg\neg\neg(s(x) = 0)$

  (7) $\mathrm{HA} \vdash \forall \vec{z}\, [A^{\mathbf{G}}(\vec{z}, 0) \wedge \forall x\, (A^{\mathbf{G}}(\vec{z}, x) \Rightarrow A^{\mathbf{G}}(\vec{z}, s(x))) \Rightarrow \forall x\, A^{\mathbf{G}}(\vec{z}, x)]$
  para cada fórmula $A(\vec{z}, x)$ con variables libres $\{\vec{z}, x\}$

Teorema

$\mathrm{PA} \vdash A$ si y sólo si $\mathrm{HA} \vdash A^{\mathbf{G}}$

Corolario: PA y HA son equiconsistentes: $\mathrm{PA} \approx \mathrm{HA}$

Traducción de otras teorías

El mismo método permite demostrar más generalmente que:

- PA2 $\approx$ HA2 (Aritmética de segundo orden)
- PAn $\approx$ HAn (Aritmética de n-ésimo orden)
- PAω $\approx$ HAω (Aritmética de alto orden)

Obs.: Este método no se aplica de modo directo a la teoría de conjuntos, pues la traducción del axioma de extensionalidad no es derivable:

$$\mathrm{IZF} \not\vdash \underbrace{\forall x\,\forall y\,[\forall z\,(\neg\neg(z\in x)\Leftrightarrow\neg\neg(z\in y))\ \Rightarrow\ \neg\neg(x=y)]}_{\left(\forall x\,\forall y\,[\forall z\,(z\in x\Leftrightarrow z\in y)\ \Rightarrow\ x=y]\right)^{\mathbf{G}}}$$

Solución: Trabajar en teorías de conjuntos no extensionales (i.e. sin $=$) equiconsistentes a (I)Z, (I)ZF. Esto permite demostrar que:

- IZ $\approx$ Z
- IZF$_C$ $\approx$ ZF [Friedman 1973]

(IZF$_C$ = ZF intuicionista con esquema de Colección en lugar de Remplazo)

Variante: la traducción $A \mapsto A^{\mathbf{G}'}$ (1/2)

Observación: En HA, tenemos que: $x = y \;\Leftrightarrow\; \neg\neg(x = y)$

Esto motiva la siguiente modificación en la traducción de Gödel-Gentzen:

Definición (Traducción $A \mapsto A^{\mathbf{G}'}$)

A cada fórmula $A \in \mathscr{L}_{\mathsf{PA}}$ se asocia la fórmula $A^{\mathbf{G}'} \in \mathscr{L}_{\mathsf{HA}}$ definida por:

$$
\begin{array}{rcll}
(t = u)^{\mathbf{G}'} & :\equiv & t = u & \text{(en lugar de } \neg\neg(t = u)) \\
\top^{\mathbf{G}'} & :\equiv & \top & \\
\bot^{\mathbf{G}'} & :\equiv & \bot & \\
(\neg A)^{\mathbf{G}'} & :\equiv & \neg A^{\mathbf{G}'} & \\
(A \Rightarrow B)^{\mathbf{G}'} & :\equiv & A^{\mathbf{G}'} \Rightarrow B^{\mathbf{G}'} & \\
(A \wedge B)^{\mathbf{G}'} & :\equiv & A^{\mathbf{G}'} \wedge B^{\mathbf{G}'} & \\
(A \vee B)^{\mathbf{G}'} & :\equiv & \neg(\neg A^{\mathbf{G}'} \wedge \neg B^{\mathbf{G}'}) & \left(\underset{\mathsf{HA}}{\Leftrightarrow} \quad \neg\neg(A^{\mathbf{G}'} \vee B^{\mathbf{G}'})\right) \\
(\forall x\, A)^{\mathbf{G}'} & :\equiv & \forall x\, A^{\mathbf{G}'} & \\
(\exists x\, A)^{\mathbf{G}'} & :\equiv & \neg\forall x\, \neg A^{\mathbf{G}'} & \left(\underset{\mathsf{HA}}{\Leftrightarrow} \quad \neg\neg\exists x\, A^{\mathbf{G}'}\right)
\end{array}
$$

Variante: la traducción $A \mapsto A^{\mathbf{G}'}$ (2/2)

Obs.: La única diferencia entre las traducciones $A^{\mathbf{G}}$ y $A^{\mathbf{G}'}$ es:

$$(t = u)^{\mathbf{G}} \;:\equiv\; \neg\neg(t = u) \qquad \text{mientras} \qquad (t = u)^{\mathbf{G}'} \;:\equiv\; t = u$$

Por lo tanto:

Lema: $\mathrm{HA} \vdash A^{\mathbf{G}'} \Leftrightarrow A^{\mathbf{G}}$ (para toda fórmula A)

Proposición (Propiedades de la traducción $A \mapsto A^{\mathbf{G}'}$)

(1) $\mathrm{PA} \vdash A^{\mathbf{G}'} \Leftrightarrow A$ (para todo $A \in \mathscr{L}_{\mathrm{PA}}$)

(2) $\mathrm{PA} \vdash A$ si y sólo si $\mathrm{HA} \vdash A^{\mathbf{G}'}$

Cuando A no contiene ni $\vee$ ni $\exists$, tenemos que $A^{\mathbf{G}'} \equiv A$ y por lo tanto:

Teorema

Para toda fórmula aritmética A sin $\vee$ ni $\exists$, tenemos que:

$$\mathrm{PA} \vdash A \qquad \text{si y sólo si} \qquad \mathrm{HA} \vdash A$$

Interpretación computacional de las pruebas de $A^{\mathbf{G}}$

Idea: Combinar la traducción $A \mapsto A^{\mathbf{G}}$ (de PA a HA) con el teorema de eliminación de cortes (en HA) para analizar las pruebas clásicas

- **¿Propiedad de la existencia?** Una derivación cerrada y sin cortes de $(\exists x\, A(x))^{\mathbf{G}} \equiv \neg\forall x\, \neg A^{\mathbf{G}}(x)$ es de la forma:

$$\dfrac{\begin{array}{c}\vdots\ ?\\ \forall x\, \neg A^{\mathbf{G}}(x) \ \vdash_{\mathsf{HA}^{\cong}}\ \bot\end{array}}{\vdash_{\mathsf{HA}}\ \neg\forall x\, \neg A^{\mathbf{G}}(x)}\ (\neg\text{-in})$$

  $\rightsquigarrow$ ¡No se puede decir nada más!

- **¿Propiedad de la disyunción?** Problema análogo (verificarlo)

Fracaso: Se puede demostrar (por técnicas de realizabilidad) que la fórmula $A^{\mathbf{G}}$ no tiene ningún contenido computacional interesante

Necesidad de modificar la traducción $\rightsquigarrow$ traducción de Friedman

Plan

La traducción de Friedman [Friedman 1978]

Sea $\mathscr{L}$ un lenguaje de primer orden cualquiera

Definición (Traducción $A \mapsto A^{\mathbf{F}}$ de Friedman)

Fijada una fórmula cualquiera $R \in \mathscr{L}$ (posiblemente abierta), se define la traducción de Friedman $A \mapsto A^{\mathbf{F}}$ inducida por R por las ecuaciones:

$$
\begin{array}{rcll}
(p(t_1,\ldots,t_k))^{\mathbf{F}} & :\equiv & p(t_1,\ldots,t_k) \vee R & \\
\top^{\mathbf{F}} & :\equiv & \top & \\
\bot^{\mathbf{F}} & :\equiv & R & \\
(\neg A)^{\mathbf{F}} & :\equiv & A^{\mathbf{F}} \Rightarrow R & \\
(A \Rightarrow B)^{\mathbf{F}} & :\equiv & A^{\mathbf{F}} \Rightarrow B^{\mathbf{F}} & \\
(A \wedge B)^{\mathbf{F}} & :\equiv & A^{\mathbf{F}} \wedge B^{\mathbf{F}} & \\
(A \vee B)^{\mathbf{F}} & :\equiv & A^{\mathbf{F}} \vee B^{\mathbf{F}} & \\
(\forall x\, A)^{\mathbf{F}} & :\equiv & \forall x\, A^{\mathbf{F}} & (\text{si } x \notin FV(R)) \\
(\exists x\, A)^{\mathbf{F}} & :\equiv & \exists x\, A^{\mathbf{F}} & (\text{si } x \notin FV(R))
\end{array}
$$

Obs.: Antes de calcular $A^{\mathbf{F}}$, se necesita cambiar los nombres de las variables ligadas en la fórmula A de tal modo que $BV(A) \cap FV(R) = \varnothing$

Propiedades de la traducción $A \mapsto A^{\mathbf{F}}$

Obs.: $FV(A^{\mathbf{F}}) \subseteq FV(A) \cup FV(R)$

Proposición (Propiedades de la traducción $A \mapsto A^G$)

Fijados un lenguaje de primer orden $\mathscr{L}$ y una fórmula $R \in \mathscr{L}$:

(1) $\vdash_{\mathsf{NK}} A^{\mathbf{F}} \Leftrightarrow A \vee R$ (para todo $A \in \mathscr{L}$)

(2) $\vdash_{\mathsf{NJ}} R \Rightarrow A^{\mathbf{F}}$ (para todo $A \in \mathscr{L}$)

(3) Si $\Gamma \vdash_{\mathsf{NJ}} A$, entonces $\Gamma^{\mathbf{F}} \vdash_{\mathsf{NJ}} A^{\mathbf{F}}$

(4) En particular: $\vdash_{\mathsf{NJ}} A$ implica $\vdash_{\mathsf{NJ}} A^{\mathbf{F}}$

Obs.: $\nvdash_{\mathsf{NJ}} A \Rightarrow A^{\mathbf{F}}$ (en general)

Demostración.

(1) Por inducción sobre la fórmula A (ejercicio)

(2) Por inducción sobre la fórmula A (ejercicio)

(3) Por inducción sobre la derivación de $\Gamma \vdash A$ en el sistema NJ, usando (2) para interpretar la regla ($\bot$-elim) (*ex falso quod libet*)

(4) Sigue inmediatamente de (3) $\square$

El caso de la Aritmética

- Se observa que $\mathrm{HA} \vdash A^{\mathbf{F}}$ para todo axioma $A \in \mathrm{Ax}(\mathrm{HA})$

  (1) $\mathrm{HA} \vdash \forall x\,(x + 0 = x \ \vee \ R)$

  (2) $\mathrm{HA} \vdash \forall x \forall y\,(x + s(y) = s(x + y) \ \vee \ R)$

  (3) $\mathrm{HA} \vdash \forall x\,(x \times 0 = 0 \ \vee \ R)$

  (4) $\mathrm{HA} \vdash \forall x \forall y\,(x \times s(y) = x \times y + x \ \vee \ R)$

  (5) $\mathrm{HA} \vdash \forall x \forall y\,(s(x) = s(y) \vee R \ \Rightarrow \ x = y \vee R)$

  (6) $\mathrm{HA} \vdash \forall x\,(s(x) = 0 \Rightarrow R)$

  (7) $\mathrm{HA} \vdash \forall \vec{z}\,[A^{\mathbf{F}}(\vec{z}, 0) \wedge \forall x\,(A^{\mathbf{F}}(\vec{z}, x) \Rightarrow A^{\mathbf{F}}(\vec{z}, s(x))) \ \Rightarrow \ \forall x\, A^{\mathbf{F}}(\vec{z}, x)]$
  para cada fórmula $A(\vec{z}, x)$ con variables libres $\{\vec{z}, x\}$

En lo anterior, se supone que $FV(R) \cap \{\vec{z}, x, y\} = \varnothing$

Teorema

Si $\mathrm{HA} \vdash A$, entonces $\mathrm{HA} \vdash A^{\mathbf{F}}$ (¡Cuidado! $\not\Leftarrow$)

Π^0_2-conservatividad (1/3)

El interés de la traducción de Friedman aparece en la demostración de:

Teorema [Friedman 1978]

PA es una extensión Π^0_2-conservativa de HA, es decir:

$$\text{PA} \vdash \forall \vec{x}\, \exists \vec{y}\, f(\vec{x}, \vec{y}) = 0 \qquad \text{sii} \qquad \text{HA} \vdash \forall \vec{x}\, \exists \vec{y}\, f(\vec{x}, \vec{y}) = 0$$

para toda función recursiva primitiva $f(\vec{x}, \vec{y})$

Esto implica más generalmente que

$$\text{PA} \vdash \forall \vec{x}\, \exists \vec{y}\, A(\vec{x}, \vec{y}) \qquad \text{sii} \qquad \text{HA} \vdash \forall \vec{x}\, \exists \vec{y}\, A(\vec{x}, \vec{y})$$

para toda fórmula $A(\vec{x}, \vec{y})$ con cuantificaciones acotadas

Π^0_2-conservatividad (2/3)

Demostración. Sin perdida de generalidad, se puede restringir al caso de las fórmulas de la forma $\forall x\, \exists y\, f(x,y) = 0$.

Supongamos:	$\mathsf{PA} \vdash \forall x\, \exists y\, f(x,y) = 0$	
Entonces:	$\mathsf{HA} \vdash (\forall x\, \exists y\, f(x,y) = 0)^{\mathbf{G}'}$	(corrección de $A \mapsto A^{\mathbf{G}'}$)
Es decir:	$\mathsf{HA} \vdash \forall x\, \neg\neg \exists y\, f(x,y) = 0$	(def. de $A \mapsto A^{\mathbf{G}'}$)
Entonces:	$\mathsf{HA} \vdash \neg\neg \exists y\, f(x_0,y) = 0$	($\forall$-elim con $x := x_0$)
Sea:	$R :\equiv \exists y\, f(x_0,y) = 0$	(truco de Friedman)
Tenemos que:	$\mathsf{HA} \vdash (\neg\neg \exists y\, f(x_0,y) = 0)^{\mathbf{F}}$	(corrección de $A \mapsto A^{\mathbf{F}}$)
Es decir:	$\mathsf{HA} \vdash (\exists y\, (f(x_0,y) = 0 \vee R) \Rightarrow R) \Rightarrow R$	(def. de $A \mapsto A^{\mathbf{F}}$)
Es decir:	$\mathsf{HA} \vdash ((\exists y\, f(x_0,y) = 0) \vee R \Rightarrow R) \Rightarrow R$	(pues $y \notin FV(R)$)
Es decir:	$\mathsf{HA} \vdash (R \vee R \Rightarrow R) \Rightarrow R$	(def. de R)
Por otro lado:	$\mathsf{HA} \vdash R \vee R \Rightarrow R$	(obvio)
Y por lo tanto:	$\mathsf{HA} \vdash R$	(modus ponens)
Es decir:	$\mathsf{HA} \vdash \exists y\, f(x_0,y) = 0$	(def. de R)
Luego:	$\mathsf{HA} \vdash \forall x\, \exists y\, f(x,y) = 0$	($\forall$-intro) $\square$

Π^0_2-conservatividad (3/3)

Adaptando la misma técnica a otros formalismos, Friedman (1978) demostró que:

- PA2 es una extensión Π^0_2-conservativa de HA2
- PAn es una extensión Π^0_2-conservativa de HAn (para todo $n \geq 2$)
- PAω es una extensión Π^0_2-conservativa de HAω
- Z es una extensión Π^0_2-conservativa de IZ
- ZF es una extensión Π^0_2-conservativa de IZF$_C$

Observación: En teoría de conjuntos, las fórmulas Π^0_2 son de la forma

$$(\forall \vec{x} \in \omega)(\exists \vec{y} \in \omega)\, f(\vec{x}, \vec{y}) = 0$$

donde $f(\vec{x}, \vec{y})$ es cualquier función recursiva primitiva

Regla de Markov

El principio de Markov es el principio de doble eliminación de la negación restringido a las fórmulas Σ^0_1:

MP $\quad \neg\neg\exists x\, A(x) \;\Rightarrow\; \exists x\, A(x)$

para toda fórmula $A(x)$ sin cuantificadores (o con cuantificaciones acotadas)

- Tenemos que $\mathrm{PA} \vdash \mathbf{MP}$, pero $\mathrm{HA} \nvdash \mathbf{MP}$ (en general)
- Sin embargo:

Teorema (Regla de Markov)

La regla de Markov $\dfrac{\mathrm{HA} \vdash \neg\neg\exists x\, A(x)}{\mathrm{HA} \vdash \exists x\, A(x)}$ es admisible en HA

(en el contexto vacío) donde $A(x)$ es cualquier fórmula con cuantificaciones acotadas

Demostración. Si $\mathrm{HA} \vdash \neg\neg\exists x\, A(x)$, entonces $\mathrm{PA} \vdash \exists x\, A(x)$, y por lo tanto $\mathrm{HA} \vdash \exists x\, A(x)$ por Π^0_2-conservatividad. □

Extracción de programas en lógica clásica

- Sea una derivación

$$d \;:\; \mathrm{PA} \;\vdash\; \forall x_1 \cdots \forall x_k \, \exists y \; A(\vec{x}, y)$$

donde $A(\vec{x}, y)$ es una fórmula con cuantificaciones acotadas

- Combinando las traducciones de Gentzen-Gödel y de Friedman con el truco de Friedman, se deduce otra derivación:

$$d' \;:\; \mathrm{HA} \;\vdash\; \forall x_1 \cdots \forall x_k \, \exists y \; A(\vec{x}, y)$$

- Usando el teorema de eliminación de cortes en HA, se construye una función computable $f : \mathbb{N}^k \to \mathbb{N}$ (total) tal que

$$\mathrm{HA} \;\vdash\; A(n_1, \ldots, n_k, f(n_1, \ldots, n_k))$$

para todos $n_1, \ldots, n_k \in \mathbb{N}$

- **Obs.:** Se puede extraer f directamente a partir de la derivación d (clásica), usando técnicas de realizabilidad clásica [Krivine 2006, Miquel 2009]