Introducción al Teorema de incompletitud de Gödel

30 de Mayo de 2012

Índice

Notación.

◦ De ahora en adelante las funciones con las que trabajaremos serán funciones de $\mathbb{N}^k$ en $\mathbb{N}$ para algún $k \in \mathbb{N}$ y las relaciones serán subconjuntos de $\mathbb{N}^k$ para algún $k \in \mathbb{N}$.

◦ Trabajaremos con el lenguaje de la aritmética $L_{Ar} = \{S, +, \cdot, 0, <\}$ y su teoría Ar. A las fórmulas del lenguaje L_{Ar} con n variables libres las denotaremos $For^{(n)}(L_{Ar})$. Usaremos la abreviatura $k_n = S\overset{n}{\cdots}S0$ para cualquier $n \in \mathbb{N}$.

◦ En ocasiones identificaremos un subconjunto R de $\mathbb{N}^m$ con una relación m-aria y viceversa. Por esa razón en vez de escribir $a \in R$ escribiremos $R(a)$. Así mismo, dados dos subconjuntos S y R de $\mathbb{N}^m$ escribiremos $R \wedge S$ en vez de $R \cap S$, $R \vee S$ en vez de $R \cup S$ y $\neg R$ en vez de $\mathbb{N}^m \backslash R$.

1. Teorema de incompletitud de Gödel

En este tema veremos el primer teorema de incompletitud de Gödel. Para ello, mostraremos que es posible codificar la aritmética. Es decir, vamos a probar que si L es un lenguaje finito que contiene al lenguaje de la aritmética L_{Ar} entonces existe una aplicación inyectiva

$$\ulcorner - \urcorner : For(L) \cup Ter(L) \to \mathbb{N}$$

con muy buenas propiedades. Como veremos más adelante, estas buenas propiedades giran alrededor del concepto de *recursividad*. No vamos a dar la definición de recursividad hasta la siguiente sección, pero sí podemos dar una idea intuitiva de lo que significa. Decimos que un conjunto $S \subset \mathbb{N}$ es recursivo si existe un algoritmo de forma que dado un número $a \in \mathbb{N}$ tenemos que si $a \in S$ entonces el algoritmo se detiene con un `TRUE` y si $a \notin S$ entonces se detiene con un `FALSE`. Diremos que una función $f : \mathbb{N}^n \to \mathbb{N}$ es recursiva si dados $a_1, \ldots, a_n \in \mathbb{N}$ podemos calcular $f(a_1, \ldots, a_n)$ mediante algún algoritmo. Con esta idea intuitiva de recursividad ya podemos enumerar las buenas propiedades a las que nos referíamos.

- El conjunto $Vble := \{\ulcorner v \urcorner : v \in V \text{ variable }\}$ es recursivo (Proposición 3.14).
- El conjunto $Term := \{\ulcorner t \urcorner : t \in Ter(L)\}$ es recursivo (Proposición 3.14).
- El conjunto $For := \{\ulcorner F \urcorner : F \in For(L)\}$ es recursivo (Proposición 3.14).
- Existe una cierta función $Sus : \mathbb{N}^3 \to \mathbb{N}$ recursiva tal que dados $s \in Ter(L)$, $F \in For(L)$ y una variable $x \in V$ entonces $Sus(\ulcorner F \urcorner, \ulcorner x \urcorner, \ulcorner s \urcorner) = \ulcorner F_x[s] \urcorner$ (Lema 3.16).

- Si nos dan una teoría T recursiva, es decir, tal que el conjunto $\{\ulcorner F \urcorner : F \in T\}$ es recursivo, entonces el conjunto $Tma_T := \{\ulcorner F \urcorner : T \vdash F\}$ es recursivamente numerable (Teorema 3.22). Esto quiere decir que existe un algoritmo de forma que para cualquier $F \in For(L)$ si $\ulcorner F \urcorner \in Tma_T$ entonces el algoritmo se detiene con un `TRUE` y si $\ulcorner F \urcorner \notin Tma_T$ entonces el algoritmo no se detiene.

Un resultado importante que nos relacionará los conjuntos recursivos con los conjuntos que podemos describir con la lógica de primer orden es el *Teorema de representabilidad*: cualquier subconjunto recursivo R de $\mathbb{N}$ es *representable*, es decir, existe una fórmula $G \in For^{(1)}(L_{Ar})$ tal que para cualquier $a \in \mathbb{N}$ tenemos que si $a \in R$ entonces $Ar \vdash G_x[k_a]$ y si $a \notin R$ entonces $Ar \vdash \neg G_x[k_a]$.

Ejercicio 1.1. Sean $R \subset \mathbb{N}$ y $G \in For^{(1)}(L_{Ar})$ de forma que G representa a R. Sea T una L-teoría coherente de forma que $L_{Ar} \subset L$ y $Ar \subset T$. Probar que $R = \{a \in \mathbb{N} : T \vdash G_x[k_a]\}$.

Ejercicio 1.2. Sean $R \subset \mathbb{N}$ y $G \in For^{(1)}(L_{Ar})$ tales que $R = \{a \in \mathbb{N} : Ar \vdash G_x[k_a]\}$. ¿Podemos deducir que la fórmula G representa a R?.

Llegados a este punto, recordemos unos de los problemas que atormentaban a los lógicos de principio de siglo: ¿podemos axiomatizar la Aritmética de forma que cualquier proposición verdadera pueda probarse? O ya en términos más técnicos:

Pregunta 1: ¿Existe una teoría T, coherente y recursiva, que extienda a la de la Aritmética y que sea completa?

Expliquemos (o recordemos) un poco los conceptos que aparecen en esta cuestión.

1. Decimos que T es coherente si $T \nvdash F \wedge \neg F$ para toda $F \in For(L)$. Alternativamente (por el Teorema de completitud) podemos decir que T es coherente si tiene algún modelo.

2. Decimos que una teoría T extiende la aritmética si es una teoría en un lenguaje L que contiene a L_{Ar} y para cualquier $F \in Ar$ tenemos que $T \vdash F$.

3. Decimos que T es completa si dada una fórmula $F \in FC(L)$ tenemos que $T \vdash F$ ó $T \vdash \neg F$

4. Decimos que una teoría T es recursiva si el conjunto $\{\ulcorner F \urcorner : F \in T\}$ es recursivo. Intuitivamente, pedir que la teoría sea recursiva quiere decir que tenemos cierto control sobre ella. En caso contrario, podríamos tomar directamente como teoría $Te(\mathbb{N}) := \{F \in FC(L_{Ar}) : \mathbb{N} \models F\}$. Sin embargo...¡qué sentido tiene considerar una teoría en el que todo lo que se puede probar es de hecho un axioma! Recordad que nuestro objetivo al fin y al cabo es fijar una serie de axiomas que de alguna forma nos resultan obvios y deducir de ellos otras propiedades que no son *a priori* tan evidentes. Por ejemplo, la conjetura de Goldbach dice (CG) *Todo número par mayor que 2 puede escribirse como suma de dos números primos.* No es evidente que esta afirmación sea cierta, de hecho, quizás no lo sea. Pero en cualquier caso, lo que uno desea es que si es cierta entonces pueda deducirse de una serie de axiomas básicos, por ejemplo, los de la Aritmética, y si es falsa que entonces sea su negación la que pueda deducirse de dichos axiomas. Si estamos considerando como axiomas toda la teoría $Te(\mathbb{N})$ entonces evidentemente o bien CG o bien $\neg$ CG puede probarse a partir de $Te(\mathbb{N})$ – ¡de hecho una de las dos es un axioma! – pero de qué nos sirve esto si no podemos decidir cuál de las dos es.

A raíz de esto último, uno quizá podría quedarse a medio camino y hacerse una pregunta menos ambiciosa. Decimos que una teoría T es decidible si el conjunto $Tma_T := \{\ulcorner F \urcorner : T \vdash F\}$ es recursivo.

Pregunta 2: ¿Existe una teoría coherente T que extienda a la Aritmética y que sea decidible?

Puede que la teoría no sea completa, pero si la respuesta es afirmativa al menos podremos saber si una proposición puede o no probarse.

Observación 1.3. Obsérvese que una respuesta afirmativa a la Pregunta 1 implica una respuesta afirmativa a la Pregunta 2. Ya dijimos que Tma_T es recursivamente numerable. Eso quería decir que existe un algoritmo de forma que para cualquier $F \in For(L)$ si $\ulcorner F \urcorner \in Tma_T$ entonces el algoritmo se detiene con un `TRUE` y si $\ulcorner F \urcorner \notin Tma_T$ entonces el algoritmo no da ningún resultado. Ahora bien, podemos crear un nuevo algoritmo: dada un fórmula $F \in For(L)$ aplicamos por un lado el algoritmo antiguo a F y por otro a $\neg F$. Como la teoría T es completa sabemos que $T \vdash F$ o $T \vdash \neg F$ y por tanto nuestro nuevo algoritmo seguro que se detiene. Si el algoritmo antiguo da como respuesta `TRUE` para F, entonces también diremos que el nuevo algoritmo da como respuesta `TRUE` para F. Si el algoritmo antiguo da como respuesta `TRUE` para $\neg F$, entonces diremos que el nuevo algoritmo da como respuesta `FALSE` para F.

En el siguiente teorema, enunciado en 1931, Kurt Gödel prueba que la respuesta a la segunda pregunta (y por tanto a la primera) es NO.

Teorema 1.4 (Primer teorema de incompletitud de Gödel). *Sea L un lenguaje finito que contiene al lenguaje L_{Ar} de la aritmética. Sea T una teoría en el lenguaje L coherente y que extiende a la teoría de la aritmética. Entonces T es indecidible. En particular, si T es recursiva, entonces no es completa.*

Demostración. Definimos el conjunto $R \subset \mathbb{N}^2$ de forma que

$$(a,b) \in R \text{ si y sólo si } Sus(b, \ulcorner x \urcorner, \ulcorner k_a \urcorner) \in Tma_T.$$

Observamos que si $b = \ulcorner F[x] \urcorner$ para alguna fórmula $F \in For^{(1)}(L)$ entonces

$$(a,b) \in R \Leftrightarrow Sus(b, \ulcorner x \urcorner, \ulcorner k_a \urcorner) = \ulcorner F_x[k_a] \urcorner \in Tma_T \Leftrightarrow T \vdash F_x[k_a].$$

Consideremos el conjunto $Q := \{a \in \mathbb{N} : (a,a) \notin R\}$ y veamos que no puede ser recursivo. Si Q lo fuese entonces por el teorema de representabilidad existiría una fórmula $G \in For^{(1)}(L_{Ar})$ que representa a Q y en particular por el Ejercicio 1.1 tenemos que $Q = \{a \in \mathbb{N} : T \vdash G_x[k_a]\}$. Si consideramos $b = \ulcorner G \urcorner$, entonces por la igualdad anterior $b \in Q$ si y sólo si $T \vdash G_x[k_b]$. Sin embargo, por definición de Q tenemos que $b \in Q$ si y sólo si $(b,b) \notin R$ si y sólo si $T \nvdash G_x[k_b]$, lo cual es una contradicción.

Ahora bien, si Tma_T fuese recursiva entonces Q sería claramente recursiva, lo cual es una contradicción con lo anterior. Por tanto Tma_T no es recursiva. □

Observación 1.5. ¿Pero cómo es la fórmula cuya existencia nos asegura el primer Teorema de Incompletitud de Gödel? Evidentemente no podemos saberlo puesto que no es una demostración constructiva. Es posible incluso que el contenido de esa fórmula no tenga un significado interesante para nosotros desde un punto de vista matemático. Sin embargo, estas esperanzas se desvanecen con el segundo resultado de incompletitud de Gödel: cualquier teoría recursiva que extiende a la Aritmética, si prueba su coherencia entonces es incoherente. Expliquemos esto un poco porque, ¿qué quiere decir eso de que "si prueba su coherencia"? Una teoría prueba fórmulas en un cierto lenguaje, no prueba frases. Gödel muestra que dada una teoría recursiva T en un lenguaje L que extiende a la Aritmética es posible encontrar una fórmula $Con(T) \in FC(L_{Ar})$ de forma que T es consistente si y sólo si $\mathbb{N} \models Con(T)$. Por supuesto esa fórmula expresa una cierta propiedad en el lenguaje de la aritmética, pero si la interpretamos en $\mathbb{N}$ e identificamos los números con las fórmulas en el lenguaje de T a través de la codificación entonces esa fórmula esta expresando la coherencia de T. A continuación, Gödel demuestra el *Segundo Teorema de Incompletitud*: si $T \vdash Con(T)$ entonces T es incoherente (véase la Sección 6.4.2 de [1]).

2. Recursividad

En esta sección vamos a definir rigurosamente la noción de recursividad y a estudiar sus propiedades.

2.1. Conjuntos y funciones recursivas

Definición 2.1. *Definimos el conjunto de funciones recursivas como el menor conjunto que satisface que:*

(R1) Contiene a las llamadas funciones recursivas básicas, las cuales son:

- *a) $I_i^n : \mathbb{N}^n \to \mathbb{N} : (a_1, \ldots, a_n) \mapsto a_i$ para cualquier $n \geqslant 1$ e $i \in \{1, \ldots, n\}$,*
- *b) las funciones usuales $+ : \mathbb{N}^2 \to \mathbb{N}$ y $\cdot : \mathbb{N}^2 \to \mathbb{N}$, y*
- *c) la función $K_< : \mathbb{N}^2 \to \mathbb{N}$ definida de forma que $K_<(a,b) = 0$ si $a < b$ y $K_<(a,b) = 1$ si $a \geqslant b$.*

(R2) Es cerrado por composición, es decir, si $g, h_1, \ldots, h_m$ son funciones recursivas la función f definida como $f(\bar{a}) := g(h_1(\bar{a}), \ldots, h_m(\bar{a}))$ es recursiva.

(R3) Es cerrado por el operador μ, es decir, si $g : \mathbb{N}^{k+1} \to \mathbb{N}$ es una función recursiva que satisface que para todo $\bar{a} \in \mathbb{N}^k$ existe $x \in \mathbb{N}$ tal que $g(\bar{a}, x) = 0$, entonces la función $f : \mathbb{N}^k \to \mathbb{N}$ definida como $f(\bar{a}) := \min\{x : g(\bar{a}, x) = 0\}$ es recursiva. De ahora en adelante escribiremos $f(\bar{a}) = \mu x g(\bar{a}, x)$.

Definición 2.2. *Un conjunto $R \subset \mathbb{N}^m$ es recursivo si la función $K_R : \mathbb{N}^m \to \mathbb{N}$ definida como $K_R(\bar{a}) = 0$ si $\bar{a} \in R$ y $K_R(\bar{a}) = 1$ si $\bar{a} \notin R$, es una función recursiva.*

Observación 2.3. ¿Pero qué tiene que ver esta definición con la que dimos en la Sección 1? En realidad, la de la Sección 1 no es una buena definición, ya que el concepto de "algoritmo" es muy vago. En cualquier caso, es fácil creerse que un conjunto o función recursiva en el sentido de la Definición 2.1 es calculable por medio de un algoritmo (por ejemplo, por un programa en C++). El recíproco es lo que se conoce como *Tesis de Church*. Obsérvese que no es algo que se pueda probar, ya que como hemos dicho ni siquiera tenemos un concepto claro de algoritmo. Debemos pensar en él más bien como en un principio, corroborado por ejemplo por el hecho de que cualquier conjunto o función calculable por medio de un lenguaje de programación conocido es recursivo. Es más, alrededor de los años 30, Alan Turing ideó lo que hoy se conocen como *Máquinas de Turing*. No vamos a entrar en detalle, pero digamos que estas Máquinas de Turing son como una versión abstracta de un ordenador moderno. Con esta noción, Turing definió un conjunto o una función *computable* de la misma forma que hicimos nosotros en la Sección 1, pero en vez de utilizar el (vago) concepto de "algoritmo" utilizó el concepto de "calculable por una máquina de Turing". Sin embargo, y apoyando la Tesis de Church, es posible probar que computable implica recursivo, razón por la cual también se la conoce como *Tesis de Church-Turing*.

Ejercicio 2.4. Mostrar que la función $\mathbb{N} \to \mathbb{N} : y \mapsto y + y$ es recursiva.

Ejercicio 2.5. Mostrar que la función $OP : \mathbb{N}^2 \to \mathbb{N} : (a, b) \mapsto (a + b)(a + b) + a + 1$ es recursiva e inyectiva.

2.2. Teorema de representabilidad

Definición 2.6. *Sea $n > 0$ y sea $f : \mathbb{N}^n \to \mathbb{N}$ una función. Sea $F \in For^{(n+1)}(L_{Ar})$. Decimos que F representa a la función f si para cualquier $\bar{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$ tenemos que $Ar \vdash F[k_{a_1}, \ldots, k_{a_n}, y] \leftrightarrow y = k_{f(\bar{a})}$.*

Ejemplo 2.7. La fórmula $F[x_1, x_2, y] : y = x_1 + x_2$ representa la función $+ : \mathbb{N}^2 \to \mathbb{N}$. Tenemos que mostrar que $Ar \vdash y = k_m + k_n \leftrightarrow y = k_{m+n}$. Es evidente (ejercicio) que por el Teorema de igualdad basta probar que $Ar \vdash k_m + k_n = k_{m+n}$. Lo demostramos por inducción sobre n. Para $n = 0$ tenemos

$F1 : \forall x\ x + 0 = x$	Axioma $Ar1$
$F2 : \forall x\ x + 0 = x \to k_m + 0 = k_m$	Axioma $C2$
$F2 : k_m + 0 = k_m$	MP

Supongamos que es cierto para n y probémoslo para $n + 1$.

$F1 : \forall x \forall y\ x + Sy = S(x + y)$	Axioma $Ar4$
$F2 : k_m + Sk_n = S(k_m + k_n)$	Axioma $C2$ + MP
$F3 : k_m + k_n = k_{n+m}$	Hipótesis de inducción
$F4 : S(k_m + k_n) = Sk_{n+m}$	Igualdad
$F5 : k_m + Sk_n = Sk_{n+m}$	Igualdad

Así pues hemos probado $Ar \vdash k_m + k_n = k_{m+n}$ como queríamos.

Notación 2.8. Recuérdese que estamos relacionando conjuntos con relaciones. Por ejemplo, asociamos la relación $=$ con el conjunto $\{(a, b) \in \mathbb{N}^2 : a = b\}$. En particular dados $R \subset \mathbb{N}^k$ y $\bar{a} \in \mathbb{N}^k$ estamos denotando $\bar{a} \in R$ con $R(\bar{a})$ y $\bar{a} \notin R$ con $\neg R(\bar{a})$.

Definición 2.9. *Sea $n > 0$ y sea $R \subset \mathbb{N}^n$. Decimos que la fórmula $G \in For^{(n)}(L_{Ar})$ representa al conjunto R si para cualquier $\bar{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$ tenemos que si $R(\bar{a})$ entonces $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}]$ y si $\neg R(\bar{a})$ entonces $Ar \vdash \neg G[k_{a_1}, \ldots, k_{a_n}]$.*

Observación 2.10. Por el Ejercicio 1.2 las siguientes condiciones no son equivalentes:

(i) $R(a)$ entonces $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}]$ y si $\neg R(a)$ entonces $Ar \vdash \neg G[k_{a_1}, \ldots, k_{a_n}]$

(ii) $R(a)$ si y solo si $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}]$.

Ejemplo 2.11. La relación $=$ es representable por la fórmula $F[x,y] : x = y$. Efectivamente, si $m = n$ entonces $Ar \vdash k_m = k_n$ por el axioma I1, el axioma C2 y MP. Veamos por inducción en n que si $m > n$ entonces $Ar \vdash \neg k_m = k_n$. Efectivamente, si $n = 0$ tenemos la demostración:

$$\begin{array}{ll} F1 : \forall x \neg Sx = 0 & \text{Axioma } Ar1 \\ F2 : \neg Sk_{m-1} = 0 & \text{Axioma } C2 \text{ + MP} \end{array}$$

Supongamos que es cierto para n y probémoslo para $n+1$. Dado $m > n+1$ tenemos que:

$$\begin{array}{ll} F1 : \forall x \forall y (Sx = Sy \to x = y) & \text{Axioma } Ar2 \\ F2 : Sk_{m-1} = Sk_n \to k_{m-1} = k_n & \text{Axioma } C2\text{+MP} \\ F3 : (Sk_{m-1} = Sk_n \to k_{m-1} = k_n) \to (\neg k_{m-1} = k_n \to \neg Sk_{m-1} = Sk_n) & \text{Tautología} \\ F4 : \neg k_{m-1} = k_n \to \neg Sk_{m-1} = Sk_n & \text{MP} \\ F5 : \neg k_{m-1} = k_n & \text{Hipótesis inducción} \\ F6 : \neg Sk_{m-1} = Sk_n & \text{MP.} \end{array}$$

Al leer la definición de un conjunto representable uno/una podría pensar: ¿no podríamos haber dicho, al igual que en la definición de recursividad, que un conjunto es representable si la función $K_R : \mathbb{N}^n \to \mathbb{N}$ definida como $K_R(\bar{a}) = 0$ si $\bar{a} \in R$ y $K_R(\bar{a}) = 1$ si $\bar{a} \notin R$, es representable? La respuesta es sí.

Proposición 2.12. *Sea $R \subset \mathbb{N}^n$. El conjunto R es representable si y solo si la función K_R es representable.*

Demostración. [2, Pag.128, Lemma 1] Probamos solo la dirección de izquierda a derecha. Sea $G[x_1, \ldots, x_n]$ la fórmula que representa al conjunto R. Veamos que la fórmula $F[x_1, \ldots, x_n, y] : (G[x_1, \ldots, x_n] \wedge y = k_0) \vee (\neg G[x_1, \ldots, x_n] \wedge y = k_1)$ representa la función K_R. Es decir, dados $a_1, \ldots, a_n \in \mathbb{N}$ debemos mostrar que $Ar \vdash F[k_{a_1}, \ldots, k_{a_n}, y] \leftrightarrow y = k_{K_R(a_1,\ldots,a_n)}$. Si $K_R(a_1, \ldots, a_n) = 0$ entonces la fórmula $G[k_{a_1}, \ldots, k_{a_n}] \to (F[k_{a_1}, \ldots, k_{a_n}, y] \leftrightarrow y = k_0)$ es una tautología y por tanto, puesto que en este caso $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}]$, deducimos que $Ar \vdash F[k_{a_1}, \ldots, k_{a_n}, y] \leftrightarrow y = k_{K_R(a_1,\ldots,a_n)}$. Si $K_R(a_1, \ldots, a_n) = 1$ entonces la fórmula $\neg G[k_{a_1}, \ldots, k_{a_n}] \to (F[k_{a_1}, \ldots, k_{a_n}, y] \leftrightarrow y = k_1)$ es una tautología y por tanto, puesto que en este caso $Ar \vdash \neg G[k_{a_1}, \ldots, k_{a_n}]$, deducimos que $Ar \vdash F[k_{a_1}, \ldots, k_{a_n}, y] \leftrightarrow y = k_{K_R(a_1,\ldots,a_n)}$.

Para probar la otra dirección basta mostrar que si $F[x_1, \ldots, x_n, y]$ representa la función K_R entonces la fórmula $G[x_1, \ldots, x_n] = F[x_1, \ldots, x_n, k_0]$ representa el conjunto R (ejercicio). □

El objetivo principal de esta sección es el siguiente teorema.

Teorema 2.13 (de representabilidad). *Los conjuntos y funciones recursivas son representables.*

No vamos a ver la demostración con detalle, haremos referencia a las páginas concretas en [2] donde se pueden encontrar las pruebas. En cualquier caso, observamos que la estrategia para probar el resultado no puede ser otra que la siguiente: dado que por la Proposición 2.12 basta probar que toda función recursiva es representable, lo que hay que hacer es demostrar que las funciones recursivas primitivas son representables y probar que el conjunto de funciones representables es cerrada por composición y por el operador μ.

Lema 2.14. *Las funciones recursivas básicas son representables.*

Demostración. [2, Pag. 127-128] En el Ejemplo 2.7 ya hemos visto que $+$ es representable. Dejamos como ejercicio probar que la formula $F[x_1, x_2, y] : y = x_1 \cdot x_2$ representa a $\cdot$ y que la fórmula $F[x_1, x_2] : x_1 < x_2$ representa a la relación $<$. En particular, por la Proposición 2.12 la función $K_<$ será representable. Por último, nótese que dados $n \in \mathbb{N}$ e $1 \leqslant i \leqslant n$ la función I_i^n es representable por la fórmula $F[x_1, \ldots, x_n, y] : y = x_i$. □

Lema 2.15. *El conjunto de funciones representables es cerrado por composición.*

Demostración. Sean $g_1, \ldots, g_m : \mathbb{N}^n \to \mathbb{N}$ y $h : \mathbb{N}^m \to \mathbb{N}$ funciones representables. Veamos que la función $f(\bar{a}) = h(g_1(\bar{a}), \ldots, g_m(\bar{a}))$ es representable. Para cada $i \in \{1, \ldots, m\}$ sea $G_i[x_1, \ldots, x_n, y_i]$ una formula que representa a g_i y sea $H[y_1, \ldots, y_m, z]$ una fórmula que representa a h y con las variables $x_1, \ldots, x_n, y_1, \ldots, y_m, z$ distintas. Veamos que f es representable por la fórmula

$$F[x_1, \ldots, x_n, z] : \exists y_1 \cdots \exists y_m (\bigwedge_{i=1}^{m} G_i[\bar{x}, y_i] \wedge H[\bar{y}, z]).$$

Sean $a_1 \ldots, a_n \in \mathbb{N}$ y denotemos $c := f(\bar{a})$ y $b_i := g_i(\bar{a})$. Obsérvese que por definición $h(b_1 \ldots, b_m) = c$, $Ar \vdash G[k_{b_1}, \ldots, k_{b_m}, z] \leftrightarrow z = k_c$ y $Ar \vdash H_i[k_{a_1}, \ldots, k_{a_n}, y_i] \leftrightarrow y_i = k_{b_i}$ para cada $i \in \{1, \ldots, m\}$. Por el ejercicio 6 de la Hoja 3, para cada $i \in \{1, \ldots, m\}$ tenemos

$$Ar \vdash \exists y_1 \cdots \exists y_m (\bigwedge_{i=1}^{m} H_i[k_{a_1}, \ldots, k_{a_n}, y_i] \wedge G[y_1, \ldots, y_m, z]) \leftrightarrow \exists y_1 \cdots \exists y_m (\bigwedge_{i=1}^{m} y_i = k_{b_i} \wedge G[y_1, \ldots, y_m, z])$$

Aplicando reiteradamente el Corolario 4.8 (de los apuntes) tenemos que

$$Ar \vdash \exists y_1 \cdots \exists y_m (\bigwedge_{i=1}^{m} y_i = k_{b_i} \wedge G[y_1, \ldots, y_m, z]) \leftrightarrow G[k_{b_1}, \ldots, k_{b_m}, z]$$

y por el ejercicio 6 de la Hoja 3

$$Ar \vdash \exists y_1 \cdots \exists y_m (\bigwedge_{i=1}^{m} H_i[k_{a_1}, \ldots, k_{a_n}, y_i] \wedge G[y_1, \ldots, y_m, z]) \leftrightarrow G[k_{b_1}, \ldots, k_{b_m}, z]$$

Finalmente de nuevo por el ejercicio 6 de la Hoja 3

$$Ar \vdash \exists y_1 \cdots \exists y_m (\bigwedge_{i=1}^{m} H_i[k_{a_1}, \ldots, k_{a_n}, y_i] \wedge G[y_1, \ldots, y_m, z]) \leftrightarrow z = k_c.$$

□

Lema 2.16. *Las funciones representables son cerradas por medio del operador μ.*

Demostración. Vamos a usar la siguiente afirmación cuya demostración se puede encontrar en [2, Pag. 129, Lemma 3]:

(*) Sea $F \in For(L_{Ar})$ y sea y una variable que no aparece en F. Si $Ar \vdash \neg F_x[k_i]$ para todo $0 \leqslant i < n$ y $Ar \vdash F_x[k_n]$ entonces $Ar \vdash (F \wedge \forall y(y < x \to \neg F_x[y])) \leftrightarrow x = k_n$.

Sea $g : \mathbb{N}^{n+1} \to \mathbb{N}$ una función representable por la fórmula $G[x_1, \ldots, x_n, y, z]$ y tal que para cualesquiera $a_1, \ldots, a_n \in \mathbb{N}$ existe un $b \in \mathbb{N}$ con $g(\bar{a}, b) = 0$. Veamos que la función $f(\bar{a}) = \mu x g(\bar{a}, x)$ es representable por la fórmula $F[\bar{x}, y] : G_z[k_0] \wedge \forall w(w < y \to \neg G_{y,z}[w, 0])$, donde w es una nueva variable que no tiene ninguna aparición en G. Sean $a_1, \ldots, a_n \in \mathbb{N}$ y denotemos $b := f(a_1, \ldots, a_n)$ y $c_i = g(a_1, \ldots, a_n, i)$ para cualquier $i \in \mathbb{N}$. Por definición $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}, k_i, z] \leftrightarrow z = k_{c_i}$. En particular, usando Generalización, el Axioma C2 y MP obtenemos que $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}, k_i, k_0] \leftrightarrow k_0 = k_{c_i}$. Si $i < b$ entonces $c_i \neq 0$, es decir, $Ar \vdash \neg k_0 = k_{c_i}$ y por lo anterior $Ar \vdash \neg G[k_{a_1}, \ldots, k_{a_n}, k_i, k_0]$. Por otro lado, $Ar \vdash k_0 = k_{c_b}$ ya que $c_b = 0$. Deducimos que $Ar \vdash G[k_{a_1}, \ldots, k_{a_n}, k_b, k_0]$. Finalmente aplicamos (*). □

Ejercicio 2.17 (Difícil). Demostrar la implicación de derecha a izquierda de la afirmación (*) del lema anterior. Podéis usar el siguiente hecho: para cualquier fórmula $F \in For(L_{Ar})$ y $n \in \mathbb{N}$ tenemos que $Ar \vdash F_x[k_0] \to \cdots \to F_x[k_{n-1}] \to x < k_n \to F$. Lo probamos por inducción en n. Si $n = 0$ entonces es trivial ya que en este caso lo que hay que mostrar es $Ar \vdash x < 0 \to F$ lo cual se sigue de $Ar7$ (obsérvese que la fórmula $\neg x < 0 \to x < 0 \to F$ es una tautología). Supongamos que es cierto hasta n y probémoslo para $n + 1$. Por un lado, por hipótesis de inducción tenemos que $Ar \vdash F_x[k_0] \to \cdots \to F_x[k_{n-1}] \to x < k_n \to F$. Por otro lado, por el Teorema de Igualdad tenemos que $Ar \vdash x = k_n \to (F \leftrightarrow F_x[k_n])$. Denotemos $G_1 : F_x[k_0] \to \cdots \to F_x[k_{n-1}] \to F_x[k_n] \to x < k_n \to F$ y $G_2 : x = k_n \to (F \leftrightarrow F_x[k_n])$. La fórmula

$$G_1 \to G_2 \to (F_x[k_0] \to \cdots \to F_x[k_{n-1}] \to F_x[k_n] \to (x < k_n \vee x = k_n) \to F)$$

es una tautología, y por tanto $Ar \vdash F_x[k_0] \to \cdots \to F_x[k_{n-1}] \to F_x[k_n] \to (x < k_n \vee x = k_n) \to F$. Finalmente, por el axioma Ar8 tenemos que $Ar \vdash x < k_{n+1} \leftrightarrow (x < k_n \vee x = k_n)$ y usando el ejercicio 6 de la Hoja 3 llegamos al resultado.

2.3. Propiedades de los conjuntos y funciones recursivas

A continuación vamos enumerar una serie de propiedades de los conjuntos recursivos.

Proposición 2.18. *Sea Q un conjunto recursivo y sean $h_1, \ldots, h_m : \mathbb{N}^k \to \mathbb{N}$ funciones recursivas. Sea P el subconjunto de $\mathbb{N}^k$ definido por $P(\bar{a})$ si y solo si $Q(h_1(\bar{a}), \ldots, h_m(\bar{a}))$. Entonces P es un conjunto recursivo.*

Demostración. [2, Pag. 110, R4] □

Proposición 2.19. *Sea $P \subset \mathbb{N}^{k+1}$ un conjunto recursivo tal que para cualquier $\bar{a} \in \mathbb{N}^k$ existe $b \in \mathbb{N}$ con $P(\bar{a}, b)$. Entonces la función $f(\bar{a}) = \mu x P(\bar{a}, x) := \min\{x \in \mathbb{N} : P(\bar{a}, x)\}$ es recursiva.*

Demostración. [2, Pag. 110, R5] □

Proposición 2.20. *Para cualquier $n > 0$ y $a \in \mathbb{N}$ la función constante $\mathbb{N}^n \to \mathbb{N} : (x_1, \ldots, x_n) \mapsto a$ es recursiva.*

Demostración. [2, Pag. 111, R6] □

Proposición 2.21. *Sean $P, Q \subset \mathbb{N}^n$ conjuntos recursivos. Entonces $\neg P$ y $P \vee Q$ son recursivos.*

Demostración. [2, Pag. 111, R7] □

Observación 2.22. Con la notación de la proposición anterior, los conjuntos $P \wedge Q$, $P \to Q$ y $P \leftrightarrow Q$ también son recursivos.

Proposición 2.23. *Las relaciones $<$, $\leqslant$, $\geqslant$, $=$ son recursivas. O dicho de otra forma, los subconjuntos de $\mathbb{N}^2$ descritos por estas relaciones son recursivos.*

Definición 2.24. *Dado un conjunto $P \subset \mathbb{N}^{k+1}$, $\bar{a} \in \mathbb{N}^k$ y $a \in \mathbb{N}$, definimos el operador μ acotado como*

$$\mu x < a P(\bar{a}, x) := \mu x (P(\bar{a}, x) \vee x = a).$$

Obsérvese que $\mu x < a P(\bar{a}, x)$ es el menor natural más pequeño que a tal que $P(\bar{a}, a)$ en caso de que exista; si no existe, entonces $\mu x < a P(\bar{a}, x) = a$.

Proposición 2.25. *[Operador μ acotado] Si $P \subset \mathbb{N}^{k+1}$ es un conjunto recursivo entonces la función $f : \mathbb{N}^k \to \mathbb{N}$ dada por $f(a, \bar{a}) := \mu x < a P(\bar{a}, x)$ es recursiva.*

Demostración. [2, Pag. 112, R9] □

Corolario 2.26. *Sea $P \subset \mathbb{N}^{k+1}$ un conjunto recursivo y sea $f : \mathbb{N}^m \to \mathbb{N}$ una función recursiva. Entonces la función $g : \mathbb{N}^{m+k} \to \mathbb{N}$ dada por*

$$g(\bar{b}, \bar{a}) := \mu x < f(\bar{b}) P(\bar{a}, x)$$

es recursiva.

Demostración. Ejercicio (usando la Proposición 2.25). □

Proposición 2.27 (Cuantificadores acotados). *Si $R \subset \mathbb{N}^{k+1}$ es un conjunto recursivo entonces los conjuntos P y Q de $\mathbb{N}^{k+1}$ dados por $P(a, \bar{a})$ si solo si $\exists x < a R(\bar{a}, x)$ y $Q(a, \bar{a})$ si solo si $\forall x < a R(\bar{a}, x)$, son recursivos.*

Demostración. Ejercicio (usando la Proposición 2.25). □

Corolario 2.28. *Sea $R \subset \mathbb{N}^k$ un conjunto recursivo y sea $f : \mathbb{N}^m \to \mathbb{N}$ una función recursiva. Entonces los conjuntos P y Q de $\mathbb{N}^{m+k}$ dados por $P(\bar{b}, \bar{a})$ si solo si $\exists x < f(\bar{b}) R(\bar{a}, x)$ y $Q(\bar{b}, \bar{a})$ si solo si $\forall x < f(\bar{b}) R(\bar{a}, x)$ son recursivos.*

Demostración. Ejercicio (usando la proposición anterior). □

Ejercicio 2.29. Demuestra que el conjunto $Div := \{(a_1, a_2) \in \mathbb{N}^2 : a_2$ divide a $a_1\}$ es recursivo.

Ejercicio 2.30. Usa las propiedades anteriores para mostrar que el conjunto el conjunto de números primos es recursivo.

Proposición 2.31 (Definición de funciones por casos). *Sean $g_1, \ldots, g_s : \mathbb{N}^n \to \mathbb{N}$ funciones recursivas y sean $R_1 \ldots, R_s \subset \mathbb{N}^n$ relaciones recursivas tales que para cualquier $\bar{a}$ existe un único $j = 1, \ldots, s$ con $R_j(\bar{a})$. Entonces la función*

$$f(\bar{a}) = \begin{cases} g_1(\bar{a}) & \text{si } R_1(\bar{a}), \\ \vdots & \vdots \\ g_s(\bar{a}) & \text{si } R_s(\bar{a}). \end{cases}$$

es recursiva.

Demostración. [2, Pag. 113, R12] □

Proposición 2.32 (Diferencia truncada). *La función*

$$a \dot{-} b = \begin{cases} a - b & \text{si } b \leqslant a, \\ 0 & \text{si } a < b, \end{cases}$$

es recursiva.

Demostración. Ejercicio usando la Proposición 2.31. □

Proposición 2.33 (Definición de relaciones por casos). *Sean $Q_1, \ldots, Q_s \subset \mathbb{N}^n$ y $R_1, \ldots, R_s \subset \mathbb{N}^n$ relaciones recursivas. Entonces la relación*

$$P(\bar{a}) \text{ si y solo si } \begin{cases} Q_1(\bar{a}) & \text{si } R_1(\bar{a}), \\ \vdots & \vdots \\ Q_s(\bar{a}) & \text{si } R_s(\bar{a}). \end{cases}$$

es recursiva.

Demostración. [2, Pag. 114, R13] □

3. Codificación de la sintaxis

3.1. La función beta de Gödel

Lema 3.1 (La función beta de Gödel). *Existe una función $\beta : \mathbb{N}^2 \to \mathbb{N}$ recursiva que satisface*

1. *$\beta(a, i) \leqslant a \dot{-} 1$ para cualesquiera $a, i \in \mathbb{N}$, y*
2. *para cualesquiera $a_0, \ldots, a_{n-1} \in \mathbb{N}$ existe $a \in \mathbb{N}$ tal que $\beta(a, i) = a_i$ para todo $i < n$.*

Demostración. Recuérdese el conjunto Div y la función OP de los ejercicios 2.5 y 2.29. Definimos la función β como

$$\beta(a, i) = \mu x < (a \dot{-} 1)\exists y < a \exists z < a[a = OP(y, z) \wedge Div(y, 1 + (OP(x, i) + 1)z)].$$

Puesto que Div y OP son recursivos y por las propiedades de la sección anterior, deducimos que β es recursiva. Por la propia definición del operador acotado $\mu x < (a \dot{-} 1)$ tenemos que para todo $(a, i) \in \mathbb{N}^2$ se satisface $\beta(a, i) \leqslant a \dot{-} 1$. Sean $a_0, \ldots, a_{n-1} \in \mathbb{N}$ y veamos que existe $a \in \mathbb{N}$ tal que $\beta(a, i) = a_i$ para todo $i < n$. Sea $c = \text{máx}\{OP(a_i, i) + 1 : i < n\}$ y

$$\tilde{z} := c! \, .$$

Evidentemente, para todo $0 < m \leqslant c$ tenemos que m divide a $\tilde{z}$. Es fácil comprobar también que los elementos del conjunto $A = \{1 + j\tilde{z} : 0 < j \leqslant c\}$ son coprimos dos a dos (Ejercicio). Sea

$$\tilde{y} := \prod_{i=0}^{n-1} (1 + (OP(a_i, i) + 1)\bar{z}).$$

Tomemos $a := OP(\tilde{y}, \tilde{z}) > 0$ y comprobemos que $\beta(a, i) = a_i$ para cualquier $0 \leqslant i \leqslant n - 1$. Fijado $0 \leqslant i \leqslant n - 1$, observamos que:

- $a_i < OP(a_i, i) < c < \tilde{z} < a$ y en particular $a_i < a - 1$ (ya que tenemos más de una acotación estricta),
- $\tilde{z}, \tilde{y} < OP(\tilde{y}, \tilde{z}) = a$, y
- dado que $0 \leqslant i \leqslant n - 1$, por la propia definición de $\tilde{y}$ tenemos que $Div(\tilde{y}, 1 + (OP(a_i, i) + 1)\tilde{z})$.

Así pues, lo único que resta por comprobar es que a_i es el mínimo que satisface estas condiciones. Sea $b < a_i$ para el cual existen $y < a$ y $z < a$ con $a = OP(y, z)$ y $Div(y, 1 + (OP(b, i) + 1)z)$. Como la función OP es inyectiva, deducimos que $y = \tilde{y}$ y $z = \tilde{z}$. Por otro lado, puesto que $b < a_i$, deducimos que $OP(b, i) + 1 < OP(a_i, i) + 1 \leqslant c$. Puesto que los elementos de A son coprimos dos a dos, el hecho de que $1 + (OP(b, i) + 1)\tilde{z}$ divide a $\tilde{y}$ implica que $OP(b, i) + 1 = OP(a_k, k) + 1$ para algún $k = 0, \ldots, n - 1$. Finalmente, puesto que OP es inyectiva, $(b, i) = (a_k, k)$, es decir, $b = a_i$ lo cual es una contradicción. □

Observación 3.2. Usando (1) deducimos que $\beta(0, i) = 0$ para todo $i \in \mathbb{N}$ y que si $a \neq 0$ entonces $\beta(a, i) < a$ para todo $i \in \mathbb{N}$.

3.2. Números secuencia

Definición 3.3. *Para cada $n \in \mathbb{N}$ definimos la función recursiva*

$$\begin{array}{rcl} \langle - \rangle_n : \mathbb{N}^n & \to & \mathbb{N} \\ (a_0, \ldots, a_{n-1}) & \mapsto & \langle a_0, \ldots, a_{n-1} \rangle_n \end{array}$$

dada por

$$\langle a_0, \ldots, a_{n-1} \rangle_n = \mu x (\beta(x, 0) = n \wedge \beta(x, 1) = a_0 \wedge \cdots \wedge \beta(x, n) = a_{n-1}).$$

Si $n = 0$ entonces $\langle - \rangle_0 = \mu x \beta(x, 0) = 0$. Escribiremos $\langle - \rangle$ en lugar de $\langle - \rangle_n$ si se deduce del contexto.

Definición 3.4. *La unión de las imágenes de la funciones $\langle - \rangle_n$ son los números secuencia. Es decir, $a \in \mathbb{N}$ es un número secuencia si y sólo si existen $n \in \mathbb{N}$ y $a_0, \ldots, a_{n-1} \in \mathbb{N}$ tales que $a = \langle a_0, \ldots, a_{n-1} \rangle_n$. Al conjunto de números secuencia lo denotaremos con $\mathcal{S}ec$.*

Proposición - Definición 3.5. *Las siguientes funciones son recursivas:*

1. $\lg : \mathbb{N} \to \mathbb{N} : a \mapsto \lg(a) := \beta(a, 0)$,
2. $(-)_- : \mathbb{N}^2 \to \mathbb{N} : (a, i) \mapsto (a)_i := \beta(a, i + 1)$.

Observación 3.6. Sea $a \in \mathbb{N}$ un número secuencia. Entonces existen $n \in \mathbb{N}$ y $a_0, \ldots, a_{n-1} \in \mathbb{N}$ tales que $a = \langle a_0, \ldots, a_{n-1} \rangle_n$. En particular, por definición tenemos que $\lg(a) = \beta(a, 0) = n$. De la misma forma, para cualquier $i < n$ tenemos que $(a)_i = \beta(a, i + 1) = a_i$. Es más, por la Observación 3.2 si $a \neq 0$ entonces $n = \lg(a) = \beta(a, 0) < a$ y $a_i = (a)_i = \beta(a, i + 1) < a$.

Ejercicio 3.7. Demuestra que para todo $n \in \mathbb{N}$ la función $\langle - \rangle_n$ es inyectiva.

Ejercicio 3.8. Demuestra que para cualesquiera $n, m \in \mathbb{N}$ distintos se tiene que las imágenes de $\langle - \rangle_n$ y $\langle - \rangle_m$ son disjuntas.

Proposición 3.9. *El conjunto de los números secuencia $\mathcal{S}ec$ es un conjunto recursivo.*

Demostración. Basta observar que $\mathcal{S}ec(a)$ si y solo si $a = \langle (a)_0, \ldots, (a)_{\lg(a)-1} \rangle$, es decir, si a es el menor x tal que $\lg(x) = \lg(a) \wedge (x)_0 = (a)_0 \wedge \cdots \wedge (x)_{\lg(a)-1} = (a)_{\lg(a)-1}$, lo cual podemos expresar de forma recursiva como

$$\forall x < a (\lg(x) \neq \lg(a) \vee \exists i < \lg(a)((x)_i \neq (a)_i)).$$

□

El siguiente resultado será esencial para poder definir conjuntos y funciones por recurrencia.

Proposición 3.10 (Definición recurrente). *(a) Dada una función $f : \mathbb{N}^m \to \mathbb{N}$, $m > 0$, sea la función*

$$\hat{f}(a, \bar{b}) = \langle f(0, \bar{b}), \ldots, f(a-1, \bar{b}) \rangle$$

si $a > 0$ y $\hat{f}(0, \bar{b}) = 0$. Entonces f es recursiva si y sólo si $\hat{f}$ es recursiva.
(b) Sea $g : \mathbb{N}^{m+1} \to \mathbb{N}$ una función recursiva y sea $f : \mathbb{N}^m \to \mathbb{N}$ una función tal que $f(a, \bar{b}) = g(\hat{f}(a, \bar{b}), a, \bar{b})$. Entonces f es recursiva.

Demostración. (a) Es obvio que $\hat{f}(a, \bar{b}) = \mu x(\lg(x) = a \wedge \forall i < a (x)_i = f(i, \bar{b}))$. Por tanto si f es recursiva entonces $\hat{f}$ es recursiva. Por el contrario, tenemos que $f(a, \bar{b}) = (\hat{f}(a+1, \bar{b}))_a$ y por tanto si $\hat{f}$ es recursiva entonces f también es recursiva.
(b) Sea la función

$$h(a, \bar{b}) = \mu x(\mathcal{S}ec(x) \wedge \lg(x) = a \wedge \forall i < a (x)_i = g(In(x, i), i, \bar{b})),$$

donde $In(x, i) := \mu y(\lg(y) = i \wedge \forall j < i (y)_j = (x)_j)$. Obsérvese que $In(x, i) = \langle (x)_0, \ldots, (x)_{i-1} \rangle$. Para que la función h esté bien definida debemos probar que al menos existe un x tal que $\mathcal{S}ec(x) \wedge \lg(x) = a \wedge \forall i < a (x)_i = g(In(x, i), i, \bar{b})$. De hecho, vamos a probar que el único que lo satisface es $\hat{f}(a, \bar{b})$. Sea x satisfaciendo la condición anterior. Entonces $x = \langle (x)_0, \ldots, (x)_{a-1} \rangle$. Basta ver por inducción que para todo $i < a$ tenemos que $(x)_i = f(i, \bar{b})$. Para $i = 0$, $(x)_0 = g(In(x, 0), 0, \bar{b}) = g(0, 0, \bar{b})$. Por hipótesis, $f(0, \bar{b}) = g(\hat{f}(0, \bar{b}), 0, \bar{b}) = g(0, 0, \bar{b})$. Supongamos que es cierto para i y probémoslo para $i + 1$. Por inducción $In(x, i+1) = \langle (x)_0, \ldots, (x)_i \rangle = \langle f(0, \bar{b}), \ldots, f(i, \bar{b}) \rangle = \hat{f}(i+1, \bar{b})$. Por tanto $(x)_{i+1} = g(In(x, i+1), i+1, \bar{b}) = g(\hat{f}(i+1, \bar{b}), i+1, \bar{b}) = f(i+1, \bar{b})$.

Finalmente, como g es recursiva, $h = \hat{f}$ es recursiva y por el apartado (a) deducimos que f es recursiva. □

Observación 3.11. Intuitivamente, la función $\hat{f}(a, \bar{b})$ contiene toda la información a cerca de $f(0, \bar{b}), \ldots, f(a-1, \bar{b})$. Por esta razón, en la igualdad $f(a, \bar{b}) = g(\hat{f}(a, \bar{b}), a, \bar{b})$ lo que estamos expresando en realidad es que f se calcula de forma recurrente, es decir, sabiendo los valores $a, \bar{b}, f(0, \bar{b}), \ldots, f(a-1, \bar{b})$ podemos calcular el valor de $f(a, \bar{b})$. En particular, puesto que lo hemos probado para funciones, también es cierto para relaciones, es decir, que una relación definida de forma recurrente es recursiva.

3.3. Codificación de la sintaxis

En esta sección vamos a mostrar cómo codificar un lenguaje de primer orden L finito que contiene el lenguaje L_{Ar}. Obsérvese sin embargo que el procedimiento se puede reproducir para cualquier lenguaje L de primer orden que sea finito (o incluso recursivo). Fijemos nuestro conjunto de variables $V = \{v_0, v_1, \ldots\}$. A cada símbolo del lenguaje L le vamos a asignar un símbolo de la siguiente forma. A la variable v_i le asignamos el número $SN(v_i) := 2i$. Para el resto símbolos lógicos hacemos la asignación $SN(\forall) := 1$, $SN(=) := 3$, $SN(\to) := 5$ y $SN(\neg) := 7$. Para los símbolos de L_{Ar} hacemos las asignaciones $SN(S) := 9$, $SN(+) := 11$, $SN(\cdot) := 13$, $SN(0) := 15$, $SN(<) := 17$. Al resto de símbolos de L le asignamos un número impar de forma que dos símbolos distintos no tengan asignado el mismo número. Cualquier otra asignación sería perfectamente válida, lo único importante es que acabemos con una función $SN : L \to \mathbb{N}$ inyectiva y tal que $SN(v_i) = 2i$. A continuación vamos a definir una aplicación

$$\ulcorner - \urcorner : For(L) \cup Ter(L) \to \mathbb{N}$$

por inducción en la complejidad:

- Si $s \in Ter(L)$ es una variable o una constante entonces $\ulcorner s \urcorner = \langle SN(s) \rangle$.
- Si $s \in Ter(L)$ es de la forma $ft_1 \cdots t_k$ donde f es un símbolo de función k-aria entonces definimos $\ulcorner s \urcorner = \langle SN(f), \ulcorner t_1 \urcorner, \ldots, \ulcorner t_k \urcorner \rangle$.
- Si $F \in For(L)$ es de la forma $Rt_1 \cdots t_k$ donde R es un símbolo de función k-aria entonces definimos $\ulcorner F \urcorner = \langle SN(R), \ulcorner t_1 \urcorner, \ldots, \ulcorner t_k \urcorner \rangle$.
- Si $F \in For(L)$ es de la forma $\to HG$ entonces $\ulcorner F \urcorner = \langle SN(\to), \ulcorner H \urcorner, \ulcorner G \urcorner \rangle$.
- Si $F \in For(L)$ es de la forma $\neg G$ entonces $\ulcorner F \urcorner = \langle SN(\neg), \ulcorner G \urcorner \rangle$.
- Si $F \in For(L)$ es de la forma $\forall v_i G$ entonces $\ulcorner F \urcorner = \langle SN(\forall), \ulcorner SN(v_i) \urcorner, \ulcorner SN(G) \urcorner \rangle$.

Por ejemplo, $\ulcorner S0 + 0 \urcorner = \ulcorner +S00 \urcorner = \langle SN(+), \ulcorner S0 \urcorner, \ulcorner 0 \urcorner \rangle = \langle SN(+), \langle SN(S), SN(0) \rangle, \langle SN(0) \rangle \rangle$.

Ejercicio 3.12. Sea $F \in For(L)$ y sean G una subfórmula de F y $t \in Ter(L)$ un término que aparece en F. Probar que $\ulcorner G \urcorner, \ulcorner t \urcorner \leqslant \ulcorner F \urcorner$ (véase la Observación 3.6).

Observación 3.13. En la siguiente proposición vamos a probar de forma rigurosa que el conjunto de códigos de fórmulas y términos es recursivo. Pero antes, veamos de forma intuitiva por qué es esto cierto (usando la Tesis de Church). Resulta evidente que si nosotros conocemos la función SN entonces dada una fórmula o término podemos calcular su código. Por el contrario, obsérvese que dado un número $a \in \mathbb{N}$ entonces podemos saber si corresponde al código de un término o de una fórmula. Efectivamente, dado $a \in \mathbb{N}$ lo primero que hacemos es ver si $a \in \mathcal{S}ec$. Si no es es así, entonces no puede ser el código de ningún término o fórmula. Si $a \in \mathcal{S}ec$ entonces $a = \langle a_0, \ldots, a_n \rangle$ y de hecho podemos calcular $a_0, \ldots, a_n$ usando las funciones lg y $(-)_-$ de la Proposición 3.5. Si $n = 0$ y a_0 es el valor que toma SN en una constante o variable entonces tendremos que a es el código de dicho símbolo. Si $n > 0$ y a_0 es el valor al aplicar SN a un símbolo de función n-aria entonces debemos seguir analizando si los números $a_1, \ldots, a_n$ corresponden a códigos de términos. Obsérvese que el proceso termina ya que $a_1, \ldots, a_n < a$ (véase la Observación 3.6). Si a_0 es el símbolo de una función k-aria con $k \neq n$ entonces a no es el código de una fórmula ni de un término. Si $n > 0$ y a_0 es el valor que toma SN en $\forall$, $\rightarrow$, $\neg$, $=$ o un símbolo de relación entonces la forma de proceder es similar. Observamos que como corolario de esta discusión tenemos que la función $\ulcorner - \urcorner$ es inyectiva y su imagen, la cual está contenida en $\mathcal{S}ec$, es un conjunto recursivo.

Proposición 3.14. *Los conjuntos $Vble$, $Term$, $AFor$, For y FB de códigos de variables, términos, fórmulas atómicas, fórmulas y fórmulas básicas en el lenguaje L respectivamente son recursivos.*

Demostración. Veamos con detalle que $Vble$ es recursivo. Observamos que

$$Vble(a) \text{ si y solo si } a = \langle (a)_0 \rangle \wedge \exists y{<}a[(a)_0 = y + y]. \tag{1}$$

La relación $=$ es recursiva, así cómo la función identidad y las funciones $(-)_0$ y $\langle - \rangle$. Tenemos por tanto que por 2.18 el conjunto $\{a \in \mathbb{N} : a = \langle (a)_0 \rangle\}$ es recursivo. Por otro lado, la función $+$ es recursiva y por tanto también lo es el conjunto $\{(a, y) \in \mathbb{N} : (a)_0 = y + y\}$. En particular por 2.27 tenemos que $\{a \in \mathbb{N} : \exists y < a[(a)_0 = y + y]\}$ es recursivo. Por último aplicamos 2.22 y deducimos que $Vble$ es recursivo. Para probar que $Term$ es recursivo, imaginemos por un momento que $L = L_{Ar}$. En este caso tenemos que

$$Term(a) \text{ si y solo si } \begin{cases} 0 = 0 & \text{si } a = \langle SN(0) \rangle, \\ Term((a)_1) & \text{si } a = \langle SN(S), (a)_1 \rangle, \\ Term((a)_1) \wedge Ter((a)_2) & \text{si } a = \langle SN(+), (a)_1, (a)_2 \rangle \vee a = \langle SN(\cdot), (a)_1, (a)_2 \rangle, \\ Vble(a) & \text{en otro caso.} \end{cases}$$

el cual es recursivo por 3.11. Si el lenguaje L fuese distinto de L_{Ar} y tuviésemos más símbolos de funciones entonces por cada símbolo f de función k-aria tendríamos que añadir la siguiente línea a la expresión anterior: $Term((a)_1) \wedge \cdots Term((a)_k)$ si $a = \langle SN(f), (a)_1, \ldots, (a)_k \rangle$.

Vamos a finalizar dando una descripción de $AFor$, For y FB parecida a la que hemos dado para $Vble$ y $Term$. La forma de deducir de esta expresión que son recursivos es similar usando los resultados de la Sección 2.3 y la Proposición 3.11. Tan sólo escribimos el caso $L = L_{Ar}$, el caso general se resolvería añadiendo líneas de forma parecida a como hemos hecho anteriormente para los términos.

$$AFor(a) \text{ si y solo si } a = \langle (a)_0, (a)_1, (a)_2 \rangle \wedge ([(a)_0 = SN(=)] \vee [(a)_0 = SN(<)]) \wedge Term((a)_1) \wedge Term((a)_2)$$

$$For(a) \text{ si y solo si } \begin{cases} For((a)_1) & \text{si } a = \langle SN(\neg), (a)_1 \rangle, \\ For((a)_1) \wedge For((a)_2) & \text{si } a = \langle SN(\rightarrow), (a)_1, (a)_2 \rangle, \\ Vble((a)_1) \wedge For((a)_2) & \text{si } a = \langle SN(\forall), (a)_1, (a)_2 \rangle, \\ AFor(a) & \text{en otro caso} \end{cases}$$

$$FB(a) \text{ si y sólo si } For(a) \wedge (AFor(a) \vee (a)_0 = SN(\forall)).$$

□

El siguiente resultado es esencial para poder ”reconocer” una demostración.

Proposición 3.15. *El conjunto $AxLog_{_L} = \{\ulcorner F\urcorner : F \text{ axioma lógico de } L\}$ es recursivo.*

Para ello necesitaremos primero probar lo siguiente.

Lema 3.16. *(a) Existe una función recursiva $Sus : \mathbb{N}^3 \to \mathbb{N}$ tal que para cualquier fórmula F, término t y variable x tenemos que $Sus(\ulcorner t\urcorner, \ulcorner x\urcorner, \ulcorner s\urcorner) = \ulcorner t_x[s]\urcorner$ y $Sus(\ulcorner F\urcorner, \ulcorner x\urcorner, \ulcorner s\urcorner) = \ulcorner F_x[s]\urcorner$.*
(b) Existe un conjunto recursivo $lib \subset \mathbb{N}^2$ tal que para cualquier fórmula F, término t y variable x tenemos que $lib(\ulcorner F\urcorner, \ulcorner x\urcorner)$ si y sólo si x aparece libre en F y $lib(\ulcorner t\urcorner, \ulcorner x\urcorner)$ si y sólo si x aparece en t.

Demostración. Basta definir

$$Sus(a,b,c) \text{ si y solo si } \begin{cases} c & \text{si } Vble(a) \wedge a = b, \\ \langle (a)_0, Sus((a)_1, b, c)\rangle & \text{si } a = \langle (a)_0, (a)_1\rangle, \\ \langle (a)_0, Sus((a)_1, b, c), Sus((a)_2, b, c)\rangle & \text{si } a = \langle (a)_0, (a)_1, (a)_2\rangle \wedge (a)_0 \neq SN(\forall), \\ \langle (a)_0, (a)_1, Sus((a)_1, b, c)\rangle & \text{si } a = \langle SN(\forall), (a)_1, (a)_2\rangle \wedge (a)_1 \neq b, \\ a & \text{en otro caso.} \end{cases}$$

$$lib(a,b) \text{ si y solo si } \begin{cases} a = b & \text{si } Vble(a), \\ lib((a)_1, b) & \text{si } a = \langle (a)_0, (a)_1\rangle, \\ lib((a)_1, b) \wedge lib((a)_2, b) & \text{si } a = \langle (a)_0, (a)_1, (a)_2\rangle \wedge (a)_0 \neq SN(\forall), \\ lib((a)_2, b) \wedge (a)_1 \neq b & \text{en otro caso.} \end{cases}$$

□

Demostración de la Proposición 3.15. No vamos a dar la demostración de este resultado, pero sí vamos a dar indicaciones. Los axiomas lógicos son la unión de las tautologías, los axiomas de cuantificadores (C1 y C2) e igualdad (I1, I2 e I3). Así que basta ver que cada uno de estos conjuntos es recursivos ya que entonces su unión también lo será (Proposición 2.21). En realidad, los axiomas de cuantificadores y los de igualdad no son problemáticos. Por ejemplo, si llamamos también I2 al conjunto de códigos de axiomas de tipo I2, tenemos que I2(a) si y sólo si

$$For(a) \wedge \exists x_1 < a \exists x_2 < a \exists y_1 < a \exists y_2 < a [Vble(x_1) \wedge Vble(x_2) \wedge Vble(y_1) \wedge Vble(y_2) \wedge$$
$$[a = \langle SN(\to), \langle SN(=), x_1, y_1\rangle, \langle SN(\to), \langle SN(=), x_2, y_2\rangle, \langle SN(=), \langle SN(+), x_1, y_1\rangle, \langle SN(+), x_2, y_2\rangle\rangle\rangle\rangle \vee$$
$$a = \langle SN(\to), \langle SN(=), x_1, y_1\rangle, \langle SN(\to), \langle SN(=), x_2, y_2\rangle, \langle SN(=), \langle SN(\cdot), x_1, y_1\rangle, \langle SN(\cdot), x_2, y_2\rangle\rangle\rangle\rangle].$$

Los axiomas de cuantificadores también tienen una estructura muy característica y no resulta difícil convencerse de que el conjunto de sus códigos es recursivo. Por ejemplo, para demostrar la recursividad de C1 basta observar que

$$\text{C1}(a) \text{ si y sólo si } For(a) \wedge \exists x < a \exists y < a \exists z < a (Vble(x) \wedge For(y) \wedge For(z) \wedge \neg lib(y, x) \wedge$$
$$a = \langle SN(\to), \langle SN(\forall), x, \langle SN(\to), y, z\rangle\rangle, \langle SN(\to), y, \langle SN(\forall), x, z\rangle\rangle\rangle).$$

Quizás no esté tan claro el porqué de que el conjunto de códigos de las tautologías sea recursivo. Partimos del siguiente hecho que no vamos a probar: existe una función recursiva $Bas : \mathbb{N} \to \mathbb{N}$ de forma que si $a = \ulcorner F\urcorner$ entonces $Bas(a) = \langle \ulcorner G_1\urcorner, \ldots, \ulcorner G_k\urcorner\rangle$, donde $G_1, \ldots, G_k$ son las subfórmulas básicas de F ordenadas de forma que $\ulcorner G_1\urcorner < \cdots < \ulcorner G_k\urcorner$. Definimos el conjunto recursivo $v \subset \mathbb{N}^2$ tal que $v(a,m)$ si y sólo si

$$\mathcal{S}ec(m) \wedge \lg(m) = \lg(Bas(a)) \wedge \forall i < \lg(m) \exists < 2 (m)_i = \langle (Bas(a))_i, e\rangle.$$

Obsérvese que si $a = \ulcorner F\urcorner$ y $Bas(a) = \langle \ulcorner G_1\urcorner, \ldots, \ulcorner G_k\urcorner\rangle$ entonces $v(a,m)$ si y solo si $m = \langle\langle \ulcorner G_1\urcorner, e_1\rangle, \ldots, \langle \ulcorner G_k\urcorner, e_k\rangle\rangle$ para algunos $e_i \in \{0,1\}$. Es decir, m codifica una valoración de verdad de las fórmulas básicas de F. Obsérvese también que en este caso $m < \langle\langle a, 1\rangle, \overset{\lg(Bas(a))}{\ldots}, \langle a, 1\rangle\rangle$ (Ejercicio 3.12). A continuación, definimos el conjunto recursivo

$$Val(a,m) \text{ si y solo si } \begin{cases} ((m)_0)_1 = 1 & \text{si } FB(a), \\ \neg Val((a)_1, m) & \text{si } (a)_0 = SN(\neg), \\ \neg Val((a)_1, m) \vee Val((a)_2, m) & \text{si } (a)_0 = SN(\to), \\ 0 = 0 & \text{en otro caso.} \end{cases}$$

Obsérvese que si $a = \ulcorner F\urcorner$ y $v(a,m)$ entonces $Val(a,m)$ si y sólo F toma el valor 1 para la valoración asociada a m. Finalmente, $Tau(a)$ si y solo si $For(a) \wedge \forall m < \langle\langle a, 1\rangle, \overset{\lg(Bas(a))}{\ldots}, \langle a, 1\rangle\rangle Val(a,m)$. □

Definición 3.17. *Sea T una L-teoría. Decimos que T es recursiva si el conjunto $Ax_T := \{\ulcorner F \urcorner : F \in T\}$ es recursivo.*

Proposición 3.18. *Sea T una L-teoría recursiva. Entonces el conjunto*

$$Dem_T(a) \text{ si y solo si } a = \langle \ulcorner F_1 \urcorner, \dots, \ulcorner F_n \urcorner \rangle \text{ y } (F_1, \dots, F_n) \text{ es una demostración en } T,$$

es recursiva.

Demostración. Es fácil comprobar que los siguientes conjuntos

- $MP(a,b,c)$ si y solo si existen fórmulas F y G tales que $a = \ulcorner F \urcorner$, $b = \ulcorner F \to G \urcorner$ y $c = \ulcorner G \urcorner$,
- $Gen(a,b)$ si y solo si existe una fórmula F y una variable v_i tal que $a = \ulcorner F \urcorner$, $b = \ulcorner \forall v_i F \urcorner$,

son recursivos. Efectivamente, $MP(a,b,c)$ si y solo si $For(a) \wedge For(b) \wedge b = \langle SN(\to), a, c \rangle$ y $Gen(a,b)$ si y solo si $For(a) \wedge \exists c < b(Vble(c) \wedge b = \langle SN(\forall), c, a \rangle)$.

Por último basta observar que $Dem_T(a)$ si y sólo si

$$\mathcal{S}ec(a) \wedge \lg(a) \neq 0 \wedge \forall i < \lg(a)(Ax_T((a)_i) \vee AxLog_L((a)_i) \vee \exists j < i \exists k < i[MP((a)_j, (a)_k, (a)_i) \vee Gen((a)_j, (a)_i)]).$$

□

3.4. Teorías decidibles

Sea L un lenguaje de primer orden. Como comentamos al principio de la Sección 3.3, podemos imitar el procedimiento de dicha sección y obtener una codificación de la sintaxis de L.

Definición 3.19. *Decimos que una L-teoría T es decidible si el conjunto $Tma_T = \{\ulcorner F \urcorner : T \vdash F\}$ es recursivo.*

Utilizando las secciones anteriores podemos probar que dada una teoría recursiva, el conjunto de teoremas es recursivamente numerable.

Definición 3.20. *Decimos que un conjunto $R \subset \mathbb{N}^n$ es recursivamente numerable si existe $Q \subset \mathbb{N}^{m+1}$ recursivo tal que $R(\bar{a})$ si y sólo si $\exists x Q(\bar{a}, x)$.*

Observación 3.21. ¿Cómo podemos ligar esta definición con la que dimos en la Sección 1? Imaginemos que tenemos un conjunto recursivamente numerable $R \subset \mathbb{N}^m$ en el sentido de la definición anterior. Eso quiere decir que existe un conjunto $Q \subset \mathbb{N}^{m+1}$ recursivo tal que $R(\bar{a})$ si y sólo si $\exists x Q(\bar{a}, x)$. Como Q es recursivo, existe un algoritmo que nos permite decidir si un elemento está o no en Q. Vamos a crear un nuevo algoritmo para R: dado $\bar{a} \in \mathbb{N}^m$, vamos cogiendo cada número natural $x \in \mathbb{N}$ y aplicamos el algoritmo de Q a la tupla $(\bar{a}, x)$. Si para algún $x \in \mathbb{N}$ resulta que el algoritmo de Q da como respuesta `TRUE` para la tupla $(\bar{a}, x)$, entonces nuestro nuevo algoritmo se para y da como respuesta también `TRUE`. Resulta evidente que este algoritmo nuevo que hemos construido para R satisface lo que pedimos en su momento en la Sección 1: si $R(\bar{a})$ entonces el algoritmo parará con un `TRUE` y si $\neg R(\bar{a})$ entonces el algoritmo no parará. Por el contrario, supongamos que tenemos un conjunto $R \subset \mathbb{N}^m$ recursivamente numerable en el sentido de la Sección 1. Definimos un nuevo conjunto $Q \subset \mathbb{N}^{m+1}$ de forma que $Q(\bar{a}, x)$ si y sólo si tras x pasos del algoritmo de R obtenemos que $R(\bar{a})$. Evidentemente Q es recursivo desde el punto de vista de los algoritmos y, por la Tesis de Church, recursivo desde el punto de vista formal.

Teorema 3.22. *Si una L-teoría T es recursiva entonces el conjunto $Tma_T = \{\ulcorner F \urcorner : T \vdash F\}$ es recursivamente numerable.*

Demostración. Basta observar que $Tma_T(a)$ si y solo si $\exists x(Dem_T(x) \wedge (x)_{\lg(x)-1} = a)$. □

Proposición 3.23. *Un conjunto $R \subset \mathbb{N}^n$ es recursivo si y sólo si R y $\neg R$ son recursivamente numerables.*

Demostración. Sea $R \subset \mathbb{N}^n$ un conjunto recursivo. Consideremos el conjunto $Q \subset \mathbb{N}^{n+1}$ tal que $Q(\bar{a}, x)$ si y sólo si $R(\bar{a})$, el cual es claramente recursivo. Para deducir que tanto R como $\neg R$ son recursivamente numerables basta observar que $R(\bar{a})$ si y sólo si $\exists x Q(\bar{a}, x)$ y $\neg R(\bar{a})$ si y sólo si $\exists x \neg Q(\bar{a}, x)$.

Supongamos ahora que R y $\neg R$ son recursivamente numerables, es decir, existen $Q_1, Q_2 \subset \mathbb{N}^n$ recursivos tales que $R(\bar{a})$ si y sólo si $\exists x Q_1(\bar{a}, x)$ y $\neg R(\bar{a})$ si y sólo si $\exists x Q_2(\bar{a}, x)$. Observamos que dado $\bar{a} \in \mathbb{N}^n$ siempre existe x tal que $Q_1(\bar{a}, x) \vee Q_2(\bar{a}, x)$ y por tanto podemos considerar la función recursiva $f(\bar{a}) := \mu x(Q_1(\bar{a}, x) \vee Q_2(\bar{a}, x))$. Basta probar entonces que $R(\bar{a})$ si y solo si $Q_1(\bar{a}, f(\bar{a}))$. Efectivamente, si $R(\bar{a})$ entonces no existe $x \in \mathbb{N}$ tal que $Q_2(\bar{a}, x)$ y por tanto $f(\bar{a})$ es el mínimo $x \in \mathbb{N}$ tal que $Q_1(\bar{a}, x)$, y en particular $Q_1(\bar{a}, f(\bar{a}))$. Si por el contrario $Q_1(\bar{a}, f(\bar{a}))$ entonces $\exists x Q_1(\bar{a}, x)$ y por tanto $R(\bar{a})$. □

La idea de la proposición anterior es la que usamos en su momento en la Observación 1.3 para probar intuitivamente el siguiente teorema.

Teorema 3.24. *Si una L-teoría T es recursiva y completa entonces T es decidible.*

Demostración. Si T no es coherente entonces $Tma_T = For(L)$ es recursivo. Podemos suponer entonces que T es coherente. Por el Teorema 3.22 tenemos que Tma_T es recursivo numerable. Así pues, por la proposición anterior basta ver que $\neg Tma_T$ es recursivo numerable. Dada una fórmula $F \in For(L)$ y $a = \ulcorner F \urcorner$ veamos que el número de variables que aparecen en F es como mucho a. Efectivamente, si v_i aparece en F entonces $i \leqslant 2i \leqslant SN(v_i) \leqslant \ulcorner v_i \urcorner \leqslant \ulcorner F \urcorner = a$ (véase el Ejercicio 3.12) . En particular, la fórmula $\forall v_0 \cdots \forall v_a F$ es una clausura universal de F. A continuación vamos a definir una función recursiva $cl : \mathbb{N} \to \mathbb{N}$ de forma que si $a = \ulcorner F \urcorner$ para alguna fórmula $F \in For(L)$ entonces $cl(a) = \ulcorner \forall v_0 \cdots \forall v_a F \urcorner$. Efectivamente, basta considerar primero la función $f : \mathbb{N}^2 \to \mathbb{N}$ tal que $f(0, a) = a$ y $f(n + 1, a) = \langle SN(\forall), SN(v_n), f(n, a)\rangle$. La función f es recursiva y por tanto también lo es la función $cl(a) := f(a + 1, a)$.

Dada $F[v_0, \ldots, v_a] \in For(L)$, por la regla de generalización y el axioma $C2$ tenemos que $T \vdash F$ si y sólo si $T \vdash \forall v_0 \cdots \forall v_a F$. En particular, puesto que T es coherente y completa, $T \nvdash F$ si y sólo si $T \vdash \neg \forall v_0 \cdots \forall v_a F$. Así pues $\neg Tma_T(a)$ si y solo si $\neg For(a) \vee Tma_T(\langle SN(\neg), cl(a)\rangle)$, y dado que Tma_T es recursivamente numerable deducimos que $\neg Tma_T$ también lo es. □

Ejemplo 3.25. Las teorías CAC_0, CAC_p, CRC, CORC, GADST, OLDSE son decidibles.

Referencias

[1] R. Cori, D. Lascar, Mathematical Logic, Part 2, Oxford University Press 2001.

[2] Joseph R. Shoenfield, *Mathematical logic*, Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont. 1967.