Capítulo 15: Aritmetización de la Sintaxis
Computability and Logic, Quinta edición

Números de Gödel

- El objetivo central del capítulo es codificar expresiones por medio de números. Los números que codifican expresiones reciben el nombre de 'números de Gödel' o 'códigos de Gödel'.
- Podemos codificar no sólo símbolos y expresiones, sino secuencias de expresiones, secuencias de secuencias de expresiones, etc.
- Decimos que un conjunto de expresiones es *recursivo* si y sólo si el conjunto de números de Gödel de esas expresiones es recursivo.
- Un lenguaje es recursivo si y sólo si el conjunto de números de Gödel de los símbolos del lenguaje es recursivo.

Un esquema de codificación

Símbolo	código	Símbolo	código	Símbolo	código	
(	1	)	19	,	199	
$\sim$	2	$\vee$	29			
$\exists$	3					
$=$	4					
x_0	5	x_1	59	x_2	599	
A^0_0	6	A^1_0	68	A^2_0	688	.....
A^1_0	68	A^1_1	689	A^2_1	6889	
A^2_0	688	A^1_2	6899	A^2_2	68899	
...	...	...	...	...	...	
f^0_0	7	f^1_0	78	f^2_0	788	
f^0_1	79	f^1_1	789	f^2_1	7889	
f^0_2	799	f^1_2	7899	f^2_2	78899	
...	...	...	...	...	...	

Secuencias de símbolos

- En el lenguaje de la aritmética, $<$ es A_0^2, $\mathbf{0}$ es f_0^0, $'$ es f_0^1, $+$ es f_0^2 y $\cdot$ es f_1^2. Por ende, $<$ tiene código 688, $\mathbf{0}$ tiene código 7, $'$ tiene código 78, $+$ tiene código 788 y $\cdot$ tiene código 7889.
- Los números de Gödel no solamente codifican expresiones individuales sino también secuencias de expresiones. Para eso, definimos una función de concatenación $*$ tal que $s * t$ es el número de Gödel de la secuencia que consiste en la secuencia codificada por s seguida por la secuencia codificada por t.
- Es decir, si la expresión E tiene número de Gödel e y la expresión D tiene número de Gödel d, la expresión ED que se obtiene concatenándolas, tiene el número de Gödel que se obtiene poniendo a e junto con d.
- Consideremos la oración $(\mathbf{0} = \mathbf{0} \vee \sim \mathbf{0} = \mathbf{0})$. Dicha oración tiene número de Gödel 174729274719.
- De forma más general, $e * d = e \cdot 10^{lg(d,10)+1} + d$. Recuerde que lg es una de las funciones logaritmo del capítulo 7. Por tanto, sabemos que $*$ es primitiva recursiva.
- $lg(d, 10) + 1$ nos da el número de dígidos en el número d. Luego $e \cdot 10^{lg(d,10)+1}$ nos da el número e seguido por tantos ceros como dígitos hay en d. Luego, $e \cdot 10^{lg(d,10)+1} + d$ nos da el número e seguido del número d.
- $e * d = e \cdot \underbrace{10^{lg(d,10)+1}}_{100...0} + d$ (donde tenemos tantos 0s como dígitos en d).
- Símbolos, expresiones y secuencias de expresiones. Fórmulas y pruebas.

Proposición 15.1 Las operaciones lógicas son recursivas

Prueba Sea n el número de Gödel de $\sim$. Sea $neg(x) = n * x$. Luego, si x es el código de una fórmula, $neg(x)$ será el código de su negación. Sean l, d y r los números de Gödel para el paréntesis izquierdo, la disyunción y el paréntesis derecho, respectivamente. Si x e y son números de Gödel para fórmulas, $disj(x, y) = l * x * d * y * r$ es el número de Gödel para la disyunción de esas fórmulas. Si e es el número de Gödel de $\exists$, $exquant(v, x) = e * v * x$ es el número de Gödel para la cuantificación existencial con la variable con número de Gödel v de la fórmula con número de Gödel x. El resto de las operaciones lógicas no presentan complicaciones. ■

Otro esquema de codificación

Símbolo	código
(	1
)	3
,	5
$\sim$	7
$\vee$	9
$\exists$	11
$=$	13
x_i	$2 \cdot 5^i$
A_i^n	$2^2 \cdot 3^n \cdot 5^i$
f_i^n	$2^3 \cdot 3^n \cdot 5^i$

- Ahora, en el lenguaje de la aritmética, $<$ tiene código $2^2 \cdot 3^2 \cdot 5^0$, $\mathbf{0}$ tiene código $2^3 \cdot 3^0 \cdot 5^0$, $'$ tiene código $2^3 \cdot 3^1 \cdot 5^0$, $+$ tiene código $2^3 \cdot 3^2 \cdot 5^0$ y $\cdot$ tiene código $2^3 \cdot 3^2 \cdot 5^1$.

- Extendemos el código a todas las secuencias finitas de símbolos usando la descomposición en primos, como en el capítulo 7.

- Por ejemplo, si tomamos la expresión $\mathbf{0} = \mathbf{0}$ que, en la notación oficial del capítulo 9 es $= (\mathbf{0}, \mathbf{0})$, le asociamos la siguiente secuencia de números: $(13, 1, 8, 5, 8, 3)$. Luego, el código de la oración es $2^6 \cdot 3^{13} \cdot 5^1 \cdot 7^8 \cdot 11^5 \cdot 13^8 \cdot 17^3$.

- En el caso de las expresiones que consisten de un sólo símbolo, diferenciamos el código de la expresión *en tanto símbolo* del código de la expresión *en tanto expresión*. Así, por ejemplo, el código de $=$ en tanto símbolo es 13, pero su código en tanto expresión es $2 \cdot 3^{13}$.

- También podemos extender el código a secuencias finitas de expresiones. El código de una secuencia finita de expresiones es simplemente el código para una secuencia finita de números naturales, donde las entradas de la secuencia son a su vez códigos de expresiones.

Fórmulas, pruebas y recursividad

- *Ser una variable*: $Var(x)$ si y sólo si $\exists i < x(x = 2 \cdot 5^i)$.
- *Ser un predicado*: $Pred(x)$ si y sólo si $\exists n < x \exists i < x(x = 2^2 \cdot 3^n \cdot 5^i)$.
- *Ser un predicado n-ádico*: $Pred(n, x)$ si y sólo si $\exists i < x(x = 2^2 \cdot 3^n \cdot 5^i)$.
- De la misma forma, es posible codificar otras nociones, como la de ser un símbolo de función $Func(x)$, ser un símbolo de función n-ádica $F(n, x)$, ser una constante de individuo $Const(x)$, ser un término $Term(x)$, etc.
- Un ejemplo un poco más complicado es el de *ser una fórmula atómica*. Para ello, recordemos las siguientes funciones criptográficas del capítulo 7: $lh(x)$, $ent(x, i)$, $last(x)$. Luego, tenemos $FormAt(x)$ si y sólo si

  $$\begin{aligned}
  &\exists n < x : \\
  &lh(x) = 2n + 2\ \& \\
  &Pred(n, ent(x, 0))\ \& \\
  &ent(x, 1) = 1\ \& \\
  &last(x) = 3\ \& \\
  &\forall i(1 < i \leq lh(x) - 1) : \\
  &\exists y(i = 2y + 1) \rightarrow ent(x, i) = 5\ \& \\
  &\exists y(i = 2y) \rightarrow Term(ent(x, i))
  \end{aligned}$$

Proposición 15.2 El conjunto de fórmulas es recursivo. También lo es el de oraciones.

Prueba Vea la seccion 15.2. ■

Proposición 15.3 Si Γ es un conjunto recursivo de oraciones, la relación 'Σ es una deducción de la oración D a partir de Γ' es una relación recursiva.

Prueba Vea la seccion 15.3. ■

- Estos dos resultados son cruciales para la prueba del primer teorema de incompletitud de Gödel.
- En el capítulo 16 veremos que todos los conjuntos y relaciones recursivas son *representables* (en un sentido técnico a precisar) en cualquier teoría aritmética adecuada (también en un sentido técnico a precisar).
- Por la proposición 15.2, esto significa que hay un predicado $Form(x)$ tal que para todo número n, la aritmética prueba $Form(\mathbf{n})$ si y sólo si n es el número de Gödel de una fórmula del lenguaje. (Lo mismo vale para las oraciones).
- Más sorprendente aún, por la proposición 15.3, hay un predicado $Prf(x, y)$ tal que para todos los números n y m, la aritmética prueba $Prf(\mathbf{m}, \mathbf{n})$ si y sólo si m es el número de Gödel de una prueba de la oración con número de Gödel n.

- Desde un punto de vista conceptual, podríamos decir que la aritmética es capaz de 'hablar' (via la codificación) sobre sus propias fórmulas y sus propias pruebas, entre otras cosas.

Corolario 15.4 El conjunto de oraciones deducible de un conjunto recursivo de oraciones Γ es semirecursivo.

Prueba Sea Rxy la siguiente relacion:

$Rsd \leftrightarrow d$ es el número de Gödel de una oración y s es el número de Gödel de una deducción de esta oración a partir de Γ.

Por la proposición 15.3, si Γ es un conjunto recursivo de oraciones, Rxy es una relación recursiva. Luego, podemos definir el conjunto Sx de números de Gödel de oraciones deducibles a partir de Γ como sigue:

$$Sd \leftrightarrow \exists s Rsd.$$

Como S se obtiene cuantificando existencialmente sobre una relación recursiva, es un conjunto semirecursivo. ■

Corolario 15.5 [*El teorema de Completitud de Gödel, forma abstracta*] El conjunto de oraciones válidas es semirecursivo.

Prueba El conjunto de oraciones válidas es el conjunto de oraciones deducibles a partir de $\Gamma = \varnothing$. Por lo tanto, este corolario se sigue del colorario previo. ■

Algunas nociones nuevas

- Decimos que un conjunto de oraciones Γ *prueba* una oración D si hay una deduccion de D a partir de Γ. Las oraciones que pueden probarse a partir de Γ son los *teoremas* de Γ.
- Una *teoría* T es un conjunto de oraciones que contiene a todas las oraciones del lenguaje que pueden probarse a partir de ese conjunto. Luego, los teoremas de una teoría T son simplemente las oraciones de la teoría.
- Una teoría T es *(finitamente) axiomatizable* si existe un conjunto recursivo (finito) Γ de oraciones tal que T consiste en las oraciones que pueden probarse a partir de Γ.
- Un conjunto de oraciones Γ es *completo* si para toda oración B del lenguaje, B es una consecuencia de Γ o $\sim B$ es una consecuencia de Γ. Una teoría T es *completa* si para toda oración B del lenguaje, $B \in T$ o $\sim B \in T$.
- Un conjunto de oraciones Γ es *consistente* si no toda oración es una consecuencia de Γ. Una teoría T es *consistente* si no toda oración pertenece a T.

- Un conjunto de oraciones Γ es *decidible* si el conjunto de oraciones de su language que son consecuencias de Γ es recursivo. Una teoría T es *decidible* si y sólo si T es recursiva.

Corolario 15.7 Sea T una teoría axiomatizable. Si T es completa, entonces T es decidible.

Prueba Sea T^* el conjunto de números de Gödel de los teoremas de T. Por el Corolario 15.4, $T*$ es semirecursivo. Ahora, o bien T es consistente o bien no lo es. Si no lo es, T^* es el conjunto de todos los números que codifican oraciones. Por la Proposición 15.2, T es recursivo. Si T es consistente, al ser también completo, sabemos que para cada oración D, D es un teorema de T o $\sim D$ es un teorema de T. Luego, el complemento de T es la unión de los conjuntos X e Y donde $X =$ el conjunto de los números que codifican oraciones que no son teoremas de T, e $Y =$ el conjunto de los números que no codifican oraciones. Y es recursivo por la Proposición 15.2 y X es semirecursivo porque se obtiene sustituyendo la función recursiva $neg(x)$ en el conjunto semirecursivo T^*. Luego, $X \cup Y$ es semirecursivo. Pero $X \cup Y$ es el complemento de T^*, y T^* también era semirecursivo. Luego, por el teorema de Kleene, T^* es recursivo. Y por ende T es decidible. ■

Ejercicios

1. Teniendo en cuenta el primer esquema de codificación ofrecido, muestre que el conjunto $Var(x)$ de números de codifican alguna variable es recursivo.
2. Sea Γ un conjunto de oraciones, y sea T el conjunto de oraciones del lenguaje de Γ que son deducibles a partir de Γ. Muestre que T es una teoría. (Sugerencia: asuma que T no es una teoría e intente llegar a una contradicción).
3. Sean $A_1, A_2, A_3, ...$ oraciones tales que ninguna A_n es probable a partir de la conjunción de las A_m para $m < n$. Sea T una teoría que consiste en todas las oraciones que son probables a partir de las A_i. Muestre que T no es finitamente axiomatizable o, en otras palabras, que no hay un conjunto finito distinto de oraciones $B_1, B_2, ..., B_m$ tales que T es el conjunto de las oraciones que son probables a partir de las B_i. (Sugerencia: asuma que T es finitamente axiomatizable para llegar a una contradicción).
4. Para un lenguaje que contiene únicamente dos predicados diádicos, considere interpretaciones donde el dominio consiste de los enteros positivos de 1 a k. ¿Cuántas interpretaciones de este tipo hay? ¿Y si el lenguaje tiene n predicados m-ádicos?
5. Una oración D es *finitamente válida* si toda interpretación finita es un modelo de D. Esboce un argumento, *asumiendo la tesis de Church*, para concluir que el conjunto de las oraciones que *no* son finitamente válidas es semirecursivo. (Sugerencia: Utilice la proposición 12.4 y una versión más general del ejercicio anterior).