UN MARCO GENERAL PARA RESULTADOS SOBRE LIMITACIONES DE LOS SISTEMAS FORMAES

Rafael Grimson
Departamento de Matemática Universidad de Buenos Aires
Buenos Aires, Argentina

Resumen

En este artículo se estudian diferentes limitaciones de los sistemas formales y se les da un marco común. Principalmente trabajaremos con limitaciones relacionadas con las nociones de expresabilidad en la aritmética de Peano y de computabilidad. Daremos un principio general que engloba una amplia gama de resultados sobre limitaciones de los formalismos.

Abstract

In this article different formal systems limitations are studied and a common abstract setting is given to them. We will mainly work with limitation related to the notions of expressibility in Peano arithmetic and computability. A general principle will be given and some results on formal systems limitations will be deduced from it.

1. Introducción

Desde Gödel, Turing y Tarski se han mostrado diversas limitaciones de los sistemas formales. Algunas de éstas tienen una forma común, una esencia común. El objetivo del presente trabajo es dar un marco abstracto general en el que una amplia gama de estos resultados pueden ser vistos como consecuencias de un Principio de Indefinibilidad.

Este escrito es un resumen de mi tesis de licenciatura en Ciencias Matemáticas (UBA) que fue dirigida por el Dr. Guillermo Martínez a quien quiero agradecer por su gran apoyo. Dada la longitud del presente trabajo no nos es posible introducir toda la notación que usaremos para enunciar y demostrar los teoremas expuestos, de todas formas trabajaremos con el simbolismo más usual en estos casos. Usaremos el lenguaje de programación *S* dado por Davis, Sigal y Weyuker en [1994] para las demostraciones que involucren nociones de computabilidad y el simbolismo dado por Mendelson en [1997] para la aritmética de Peano.

2. Algunos Resultados Clásicos

2.1 El problema de la detención de Turing

En 1936 Alan Turing publicaba su trabajo "On computable numbers with an application to the Entscheidungsproblem" [Turing 1936-37]. En él demostró que el predicado *HALT(x,y)* que es verdadero si y sólo si el programa con código *y* de detiene para valor de entrada *x* no es un predicado computable. Una demostración moderna es la siguiente:

Teorema 1: El predicado *HALT(x,y)* no es computable.

Demostración: La demostración es por reducción al absurdo. Supongamos que el predicado es computable y consideremos el siguiente programa al que llamaremos *P*:

[A] IF HALT(X, X) GOTO A

Es claro que *P* computa la función que está indefinida si *HALT(x, x)* y está definida y su resultado es cero en caso de ser *ÿHALT(x, x)*. Para llegar a un absurdo consideremos y_0=#*P* (y_0 es el número de programa de *P*) entonces tenemos que para todo *x* vale

$HALT\ (x, y_0 \leftrightarrow \neg HALT(x, x)$

En particular vale para $x=y_0$ de donde obtenemos una contradicción. Esta provino de suponer que el predicado *HALT(x, y)* es computable.

2.2 El problema de la verdad de Tarski

Así como nos preguntamos si un programa puede determinar si otro programa se detiene o no también podemos preguntarnos, en el marco de la Aritmética de Peano (AP), si puede escribirse una fórmula con una variable libre, *x*, que exprese al predicado *TRUE(x)*: "*x* es el código de una fórmula verdadera". Este problema es conocido como el problema de la verdad de Tarski. Por analogía con el problema de la detención vamos a reformular la pregunta de la siguiente manera. Sea *T(x,y)* el predicado que expresa: "*x* es el código de una fórmula con una variable libre. Al ser reemplazada la variable libre por el valor de *y* la fórmula en cuestión resulta una sentencia verdadera". ¿Es *T(x,y)* expresable en AP?

Teorema 2: El predicado *T(x,y)* no es expresable en AP.

Demostración: Nuevamente razonaremos por absurdo. Supongamos que *t(x, y)* es una fórmula que expresa a *T(x, y)*. Entonces $\neg\tau(y, y)$ expresa a $\neg T(y, y)$. Sea *c* el código de $\neg\tau(y,y)$. Consideremos *T(c, y)*.

T(c, y) ´ $\neg\tau(y, y)$ es verdadera $\leftrightarrow$ $\neg T(y, y)$.

Ya que esta equivalencia vale para todo *y* vale en particular para *y=c* con lo que llegamos a un absurdo. Concluimos que *T(x, y)* no es expresable en AP. ♦

Es interesante observar la similitud entre esta demostración y la del problema de la detención. Fue ésta la que dio el impulso inicial al Principio de indefinibilidad. El principio se basa en una abstracción de estas demostraciones.

Recordemos que el problema de la verdad implica demostrar que el predicado *TRUE(x)* no es expresable. Para esto necesitamos usar la función *E(x,y)* definida a continuación. Supongamos que *y* es un número natural y *x* es el código de una fórmula con una variable libre, digamos *x* es el código de $\phi_x(v)$. Entonces *E(x, y)* da como resultado el código de la fórmula que resulta de sustituir todas las apariciones libres de *v* en la fórmula con código *x* por el valor de *y*. En símbolos $E(x, y) = Cod(\phi_x(v)[y])$. La función *E(x y)* es una función expresable en AP. La demostración de este hecho es quizá demasiado técnica para el presente trabajo y puede ser encontrada en la literatura.

Teorema 3: El predicado *TRUE(x)* no es expresable en AP.

Demostración: La demostración de este teorema es ahora muy sencilla. Si *TRUE(x)* fuera expresable en AP también lo sería el predicado *T(x,y)* (ya que *T(x,y)=TRUE(E(x,y))* y la expresabilidad de predicados es una clase cerrada por composición con funciones expresables). Sabemos por el Teorema 2 que *T(x,y)* no es expresable y entonces podemos afirmar la no expresabilidad de *TRUE(x)* en AP. ♦

3. El Principio de Indefinibilidad

3.1 Marco Abstracto

Sean *A* y *U* conjuntos y *0* un elemento distinguido de *A*. Sea $f:U \rightarrow A$ una función. Diremos que *f* expresa al predicado P_f definido por

$P_f(u) \leftrightarrow f(u) \neq 0$

Es decir, P_f es un predicado con dominio U y $P_f(u)$ es verdadero sii $f(u)$ no es igual al elemento $0 \in A$.

Dada una clase C de funciones con dominio incluido en U^k e imagen en A (con $0 \in A$), si $f \in C$ diremos que el predicado P_f pertenece a la clase C. Llamaremos a f una *función característica* de P_f. Observar que toda clase de funciones con estas propiedades tiene asociados dos conjuntos U y A.

Definición: Una clase C de funciones con dominio en U^k ($k \geq 0$) e imagen en A se dirá *cerrada por casos* si cada vez que $f(x_1,...,x_n,y)$ y $g(x_1,...,x_n,y)$ sean funciones de la clase C y $Q(y)$ sea un predicado perteneciente a la clase C, la función $r(x_1,...,x_n,y)$ definida por

$$r(x_1,..., x_n y) = \begin{cases} f(x1,...x_n) si Q(y) \\ g(x1,...x_n) si \neg Q(y) \end{cases}$$

también pertenezca a la clase C.

Ejemplos:

(1) Sea $U=A=IN$ y sea CFE la clase de funciones expresables en AP. Veremos que CFE es cerrada por casos: Si $\pi_1(x,z)$ expresa a $f(x)$, $\pi_2(x,z)$ expresa a $g(x)$ y $\_(y,w)$ expresa la característica de $Q(y)$ entonces $(\ \ w)(\_(y,w)^{r_{c_r}}$ $((\pi_1(x,z) \ \ w\pi 0) \ (\pi_2(x,z) \ \ w=0))$ expresa a la función $r(x,y)$.

(2) Sea ahora CFC la clase de funciones parcialmente computables. Veremos que CFC es cerrada por casos: sean $f(x)$ y $g(x)$ funciones parcialmente computables y sea $Q(y)$ un predicado parcialmente computable entonces el siguiente programa computa a la función $r(X_1,X_2)$

IF $Q(X_2)$ *GOTO A*
IF $\neg Q(X_2)$ *GOTO B*
[A] $Y \leftarrow f(X_1)$
GOTO E
[B] $Y \leftarrow g(X1)$

(3) Consideremos la clase de funciones booleanas $B=\{f:IN^k \rightarrow 2, k \geq 0\}$. En este caso $U=IN$ y $A=2=\{0,1\}$. Dadas dos funciones, $f(x)$ y $g(x)$, y un predi-

cado *Q(y)*, todos pertenecientes a la clase *B*, considerando *h(y)* la función característica de *Q(y)* se ve claramente que *p(x,y) = (h(y) f(x)) (ÿh(y) g(x))* también pertenece a la clase *B*. Como la función *p(x,y)* toma los mismos valores que la función *r(x,y)* de la definición anterior podemos afirmar que *B* es cerrada por casos.

Definición: Diremos que la clase *C* es *cerrada por composición con constantes* si dada una función *(n+1)*-aria $f(x_1,...x_n,y)$ *C* y *u U* la función *n*-aria $f(x_1,...x_n,u)$ también pertenece a la clase *C*.

Definición: Si *A=U* diremos que la clase *C* es *cerrada por composición* si dadas $f_1(x_1,...x_n)$, ..., $f_m(x_1,...x_n)$ y $g(x_1,...x_m)$ en *C*, la función $g(f_1(x_1,...x_n), ..., f_m(x_1,...x_n))$ pertenece también a la clase *C*.

Ejemplos:
La clase *CFE* es cerrada por composición y cerrada por composición con constantes. Lo mismo ocurre con la clase *CFC*. En el caso de la clase *B* podemos afirmar que es cerrada por composición con constantes pero no tiene sentido preguntarse si es cerrada por composición pues *U=IN≠A=2*.

Antes de introducir al conjunto *L* de expresiones y a la función *I(e,n)* de interpretación de expresiones como funciones vamos a motivar su definición. En el caso del problema de la detención tenemos por un lado los programas en el lenguaje *S*, por otro lado las funciones parcialmente computables y entre ambos hay una relación que llamamos *computa*. En el caso de la Aritmética de Peano tenemos a las fórmulas de AP (que hacen un papel equivalente a los programas), las funciones expresables (jugando el rol de las funciones computables) y la relación *expresa* entre fórmulas y funciones. En el marco abstracto que vamos a dar supondremos que nos es dado un conjunto *L* de expresiones (programas, fórmulas u otras) y una función $I: L \times IN \rightarrow C$ que a cada par *(expresión, número)* le asigna una función *n*-aria de la clase *C*, *I(e,n)*. Si la función *I* es sobreyectiva entonces decimos que $< L,C,I >$ es un *lengua - je de funciones*.

Así, llamando *Ps* al conjunto de programas en *S* y *FAP* a las fórmulas de *AP*, tenemos que $<FAP, CFE, I>$ y $<Ps, CFC, J>$ son lenguajes de funciones. Donde *I* es tal que dada $\_(x_1,...x_n,z)$ *FAP* si expresa a una función $f(x_1,...x_n)$ entonces $I(\_, n) = f$ y si la fórmula _ no expresa una función *n*-aria entonces *I(_, n)* corresponde con la función *n*-aria constantemente nula. *J(P, n)* es la función *n*-aria computada por el programa *P*.

Trabajaremos también con un tercer lenguaje de funciones, que utilizaremos para reobtener el resultado de Tarski, compuesto por <*FAP, B, K*> donde $K(\_(x_1,...x_n), n)(a_1,...a_n)$ es *0* o *1* dependiendo si $\_[a_1,...a_n]$ es verdadera o falsa y $K(\_(x_1,...x_n), m)$ es constantemente *0* si $m \neq n$. Es decir, si _ tiene exactamente *n* variables libres entonces *K(_, n)* es la función característica de la relación "$a_1,...a_n$ satisfacen _"; si no entonces *K(_, n)* es la función *n*-aria constantemente nula.

Por último, si contamos con una función $c: L \rightarrow U$ inyectiva llamaremos a <*L, C, I, c*> un *lenguaje de funciones con codificación*. Así, a cada programa le corresponde un número natural, cada fórmula tiene su número de gödel. En general notaremos con e_u a la expresión con código *u* y c_e al código de la expresión *e*.

Teorema 4 (Principio de Indefinibilidad): Sea <*L, C, I, c*> un lenguaje de funciones con codificación tal que la clase *C* es cerrada por casos y por composición con constantes. Si *Q* es una propiedad sobre *C* y existen dos funciones *n*-arias, *f* y *g*, tales que *Q(f)* y ¬*Q(g)* entonces el predicado $\overline{Q}(y)=Q(I(e_y,n+1)_y)$ no pertenece a la clase *C*.

Demostración: Sean *f* y *g* funciones *n*-aria tales que *Q(f)* y ¬*Q(g)*. Si $\overline{Q}$ está en la clase *C* definimos la función *r(x, y)* de la siguiente forma (escribimos *x* en lugar de $x_1,...x_n$):

$$r(x, y) = \begin{cases} f(x), si \neg \overline{Q}(y) \\ g(x), si\ \overline{Q}(y) \end{cases}$$

Dado que la clase *C* es cerrada por casos, la función *r(x, y)* pertenece a dicha clase y, por lo tanto, existe una expresión *e*Œ *L* tal que *I(e, n+1)=r*. Sea c_r el código de *e*. r_{c_r} (es decir *r(x,* c_r*)* como función de *x*) es igual a *f* o igual a *g* dependiendo del valor de $\overline{Q}(c_r)$. Por otra parte $\overline{Q}(c_r)$ es igual (por definición) al valor de $Q(r_{c_r})$. Teniendo en cuenta que *Q(f)* y *ÿQ(g)* podemos afirmar que $Q(r_{c_r}) \leftrightarrow r_{c_r} = f \leftrightarrow \neg Q(c_r) \leftrightarrow \neg Q(r_{c_r})$. Absurdo, que provino de suponer que $\overline{Q}$ pertenece a la clase *C*. ♦

3.2 Primeras Consecuencias del Principio de Indefinibilidad

Veremos ahora aplicaciones del Principio:

Corolario: El predicado *HALT(x,y)* no es computable.

Demostración: Sea *Q* la propiedad definida sobre *CFC* de la siguiente forma: si *f* es

una función *n*-aria entonces *Q(f)* es verdadera si y sólo si *f* nunca está indefinida (i.e. si *n>0* entonces *dom(f)=IN*^{n} y si *n=0* entonces *f* es una función constante). Consideremos los siguientes programas de una línea:

P1: [A] IF Y = 0 GOTO A
y

P2: Y¨Y+1

Estos programas computan, respectivamente, a la función que está siempre indefinida y a la función constantemente uno. Resulta entonces que $Q(\psi_{P_1})$ y $\neg Q(\psi_{P_2})$. Por lo tanto el predicado $\overline{Q}$ *(y) = Q(J(e$_y$,1)$_y$)* no pertenece a la clase *CFC*. Por las definiciones de *J* y de *Q*, ^{Q} *(y)* es verdadero si y sólo si el programa número *y* termina para valor de entrada *y*. Ya que el predicado *HALT(y, y)* tiene la misma definición obtenemos como consecuencia del Principio de Indefinibilidad que *HALT(y, y)* no pertenece a la clase *CFC*. Concluimos que *HALT(x, y)* no es computable.♦

Corolario: El predicado *T(x,y)* no es expresable.

Demostración: Sea *Q* la propiedad definida sobre *B* de la siguiente forma: *Q(f)* es verdadera si y sólo si *f* es constantemente uno (o sea, *f* esta asociada a una tautología). Es claro que la propiedad *Q* cumple las hipótesis del teorema. Entonces el predicado $\overline{Q}$ *(y)=Q(K(*$\_$*$_y$,1)$_y$)* no pertenece a la clase *B*. Observar que $\overline{Q}$ *(y)* es verdadero si y sólo "$_$$_y$ tiene una sola variable libre y $\_$$_y$*(v)[y]* es verdadera" ($_$$_y$ representa aquí a la fórmula con número de gödel *y*). Concluimos que *T(y, y)* no es expresable y por lo tanto *T(x, y)* tampoco lo es. ♦

4. Generalización del Principio

Para demostrar el teorema de la verdad en (2.2) usamos la inexpresabilidad de *T(x, y)* y la función *E(x, y)*. En este apartado vamos a abstraer este procedimiento.

4.1 Principio de Indefinibilidad Generalizado

Teorema 5: Bajo las hipótesis del teorema anterior si además *U = A*, la clase *C* es cerrada por composición y existe una función *E(c,z)* en la clase *C* tal que dado el código de una expresión, *c*, y un elemento del universo, *z*, da como resultado el código de una expresión *e* tal que *I(e, n)=I(e$_c$, n+1)$_z$* entonces $\overline{Q}'(y) = Q(I(e_y, n))$ no pertenece a la clase *C*.

Demostración: Razonaremos por absurdo. Sea *h(y)* una función característica del predicado $\overline{Q}$'(y) y sea *q(c) = h(E(c,c))*. Si *h* perteneciera a la clase *C* entonces *q(c)* también pertenecería a dicha clase por ser esta cerrada por composición; pero *q(c)* es una función característica del predicado $\overline{Q}$ *(c)* que sabemos que no pertenece a la clase *C* por el teorema 4. Concluimos que $\overline{Q}$ *'(y)* no es un predicado perteneciente a la clase *C*.♦

Corolario: El predicado *TRUE(x)* no es expresable.

Demostración: La demostración es inmediata considerando la clase *C* como la clase de funciones booleanas (*B*), la función *I* como la función *K* definida en (3.1) y *Q(f)* verdadera si y sólo si *f* es constantemente uno. Al interpretar el predicado $\overline{Q}$' *(y)* observamos que es equivalente a *TRUE(y)* con lo que obtenemos el resultado buscado.♦

4.2 Teorema de Rice Clásico

A partir del Principio de Indefinibilidad Generalizado podemos demostrar el Teorema de Rice. El teorema afirma que dado un subconjunto propio y no vacío, *G*, de *CFC* el conjunto de códigos de programas que computan funciones en *G* no es un computable. Analizándolo para el caso en que *G* contiene una sola función vemos que el conjunto de todos los programas que computan una función dada no es un posible capturarlo, a través de la codificación, de manera recursiva. La limitación no es intrínseca de las funciones recursivas, veremos que es posible extenderlo al caso de la aritmética de Peano. En general el resultado muestra una nueva limitación de los sistemas formales: no es posible expresar desde la teoría al conjunto de todas las expresiones con un mismo significado.

Teorema 6 (Rice): Si *G* es un subconjunto propio y no vacío del conjunto de funciones unarias parcialmente computables entonces $R_\Gamma = \{n \quad IN /$ el programa con código *n* computa una función perteneciente a $\Gamma\}$ no es computable.

Demostración: Sea *Q(f)* la propiedad definida sobre *CFC*: *Q(f)* $\leftrightarrow$ *fŒG*. Sabemos que *CFC* es cerrada por casos y cerrada por composición con constantes. Dado que el conjunto *G* es propio y no vacío, el predicado *Q* cumple con las hipótesis del Principio de Indefinibilidad Generalizado. Entonces el predicado $\overline{Q}$ *(y) = Q(J(*e_y*,1)*$_y$*)* no pertenece a la clase *CFC*, o sea, no es un predicado computable. Por la definición de *J* y de *Q* resulta que $\overline{Q}$ *'(y)* es verdadero si y sólo si la función unaria computada por el programa con código *y* es una función perteneciente al conjunto Γ, es decir, si *y* R_Γ. Concluimos que R_Γ no es computable.♦

4.3 Teorema de Rice para la Aritmética de Peano

Al ver la demostración del teorema de Rice a partir del Principio de Indefinibilidad podemos pensar en usar el mismo procedimiento para aplicarlo a la Aritmética de Peano. Esta idea no presenta dificultades y da como resultado el siguiente teorema:

Teorema 7: Si Γ es un subconjunto propio y no vacío del conjunto de funciones unarias expresables en AP y R_Γ el subconjunto de *IN* que contiene a todos los códigos de fórmulas que expresan funciones de Γ, entonces R_Γ no es expresable en AP.

Demostración: La demostración es completamente análoga a la del caso anterior.♦

4.4 Programas Elegantes

Definición: Sea *P* un programa en el lenguaje *S*. Diremos que *P* es *elegante* si ningún programa de longitud menor computa la misma función.

Observar que cada función computable tiene al menos una versión elegante (podría ocurrir que existan dos programas de la misma longitud que computen la misma función). Dado el código de un programa puede ocurrir que sea el código de un programa elegante o que no lo sea. Sea *E(n)* el predicado que es verdadero si y sólo si *n* es el código de un programa elegante. ¿Es computable *E(n)*? La respuesta en negativa y es posible demostrarlo usando los resultados de esta sección. Al aplicar el Principio puede utilizarse el predicado $Q_k(f)$: "*f* posee una versión elegante de longitud menor que *k*". Sugerimos al lector que intente realizar una demostración formal del resultado.

Bibliografía

Chaitin, Gregory (1997). *The Limits of Mathematics*, Berlín-Heidelberg-N.York, Springer-Verlag.

Davis, Martin, Ron Sigal y Elaine J. Weyuker. (1994). *Computability, Complexity and Languages*, Los Altos (Cal.), Morgan Kaufmann.

Ladrière, Jean (1969). *Limitaciones Internas de los Formalismos*. Trad. cast. de José Blasco, Madrid, Tecnos 1969.

Mendelson, Elliot (1997). *Introduction to Mathematical Logic*. 4ta. ed. Londres, Chapman and Hall 1997.

Rice, H. G. (1952). "Clases of recursively enumerable sets and their decision problems". *Transactions of the AMS*, 74, pp.358-366.

Smullyan, Raymond (1992). *Gödel's Incompleteness Theorems*. Oxford *et al*., Oxford University Press.

Turing, Alan M. (1936-37). "On computable numbers with an application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society.*