¿Todo o parte? El Teorema de los Parámetros

Hemos dado fin a dos de las etapas de que consta este curso: conocemos los rudimentos básicos del modelo clásico de la computabilidad y hemos asistido a la narración de los momentos críticos de su constitución. También hemos aprovechado para analizar algunas de las consecuencias filosóficas que todo este movimiento ha tenido y aún ha de tener.

He intentado que esta exposición pudiera ser entendida en todo momento como un análisis destinado a marcar la diferencia entre una interpretación informal e ingenua de la noción de tarea efectiva y otra formal y fundamentada desde un punto de vista científico. La primera consecuencia de todo ello es la disociación de dos características que la interpretación ingenua de esta noción mantiene unidas. Me refiero, como es evidente, al requisito según el cual cada paso en una rutina debe determinar por completo el siguiente y aquel otro según el cual para cada posible input debe existir un output. Presumir que este segundo requisito es condición necesaria para considerar una tarea como efectiva, o que es consecuencia de lo anterior es, como ya hemos visto, inadecuado. El tipo de tareas que en la actualidad ligamos a la existencia de máquinas de Turing encargadas de su ejecución incluyen propiamente todas aquellas que pueden quedar indeterminadas para algunos de sus inputs, siendo además imposible identificar de manera efectiva cuáles son éstas.

¿Hay otros resultados que puedan brindar más información acerca de la conducta y rasgos generales de la noción no ingenua de tarea efectiva? Los teoremas que nos ocupan a lo largo de los tres capítulos siguientes pueden ser vistos de este modo sin ejercer violencia alguna sobre su contenido. Se trata de analizar el Teorema de los parámetros, el Teorema de Rice y, finalmente, el Teorema de la recursión. En

todos los casos he huido deliberadamente de sus aplicaciones técnicas centrándome en lo que son sus consecuencias epistemológicas. Es posible que junto a estos resultados aún se puedan incluir otros con no menores consecuencias. Sin embargo, esta pequeña lista tiene la ventaja de figurar en todos y cada uno de los manuales básicos –o menos básicos- dedicados a la Teoría de la Computación.

¿Cómo es la computación por dentro? Hasta ahora casi todo lo que hemos venido discutiendo se refiere a su alcance, o a su relación con otras nociones afines. Sin embargo, no se puede decir que hayamos intentado responder muchas preguntas acerca de su forma concreta. El análisis de esta especie de conducta interna no se ofrece pensando tan sólo en el puro goce matemático. Es frecuente pensar en la Teoría de la Computación como una disciplina puramente instrumental, de hecho, es posible encontrar comentarios que parecen tomar las técnicas computacionales de modelización como poco más que un papel en blanco sobre el que presentar diseños más o menos sofisticados de aquello de lo que en cada caso se trata. Sabemos que el papel lo aguanta todo porque no es más que el soporte material sobre el que plasmamos nuestras ideas. ¿Se puede pensar de la Teoría de Computación en estos términos? Creo que a estas alturas ha quedado claro que el análisis de la noción de tarea efectiva tiene como primera víctima esa concepción puramente instrumental de la Teoría de la Computación. La interpretación formal de la noción de algoritmo tiene el efecto inmediato de imponer una serie de condiciones de posibilidad que proceden directamente de la estructura que es inherente a la capacidad humana para ejecutar tareas de manera efectiva. De dónde proceda esa especie de estructura abstracta que ahora me propongo investigar es, a buen seguro, una pregunta de muy otra índole.

A lo largo de este capítulo me voy a enfrentar al problema que se suscita en torno a la posibilidad de obtener rutinas complejas a partir de otras más simples, o a aquel otro que surge cuando nos planteamos descomponer rutinas complejas en sus partes constitutivas. ¿Qué reglas de composición se aplican a las tareas que resultan ser efectivas en los términos que la Teoría de la Computación establece?

La Tesis de Church-Turing

Imaginemos una rutina definida en términos de una serie de instrucciones recogidas en algún manual diseñado a tal efecto. En sus páginas se nos indica cómo actuar una vez se dan las condiciones iniciales que se suponen procedentes del entorno. Estas condiciones pueden ser identificadas a todos los efectos con los inputs de esta rutina. Es perfectamente posible imaginar una circunstancia en la que sólo se suministran, al menos de momento, ciertos inputs de aquellos cuya presencia da lugar al inicio de esa tarea. ¿Podemos proceder a anticipar tarea esperando que se complete la serie de inputs no suministrada inicialmente? ¿Puede una tarea efectiva ser descompuesta en subrutinas igualmente efectivas? Hay que reconocer que cuando se plantea esta cuestión en términos puramente intuitivos, cualquier respuesta parece realmente posible. Es un hecho que la descripción informal de rutinas parece estar presidida por la existencia de un objetivo que se realiza en la totalidad de esa rutina sin que pueda ser localizado en ninguna de sus partes. Es más, a veces parece difícil atribuir significado alguno a una subrutina distinto al de formar parte de la rutina en la que realiza su función. ¿Qué tiene que decir la Teoría de la Computación acerca de estas consideraciones que, por informales, son en esta ocasión ciertamente obscuras?

Dos rutinas que operan sobre sus respectivos inputs podrían llegar a encadenarse de tal modo que los outputs de una sean los inputs de la otra dando lugar de este modo a una única rutina de mayor complejidad. En esta ocasión parece bastante claro, desde un punto de vista plenamente intuitivo, incluso, que el resultado es también una tarea efectiva. Lo que nos importa ahora es saber si la composición de subrutinas se preserva vía códigos. Es decir, ¿podemos establecer el código de la rutina resultante a partir de los códigos de las rutinas utilizadas en su formación? Caso de no ser así habría que aceptar un peculiar desajuste entre los nombres que corresponden a rutinas consideradas efectivas y esas mismas rutinas.

El tercer asunto que tratamos aquí tiene que ver con la posibilidad de analizar la complejidad de un problema a través del algoritmo asociado a su resolución. Estudiaremos, en realidad, la manera de reducir unos problemas a otros sirviéndonos de la mención a un problema tomado como testigo. Esta mención se efectúa a través

de su código y se muestra la forma en que es posible transformar un problema en otro mediante operaciones sobre sus códigos.

Si analizamos estas conjeturas o experimentos mentales pronto apreciamos que lo que anda esta vez en juego es el equilibrio existente entre las partes de una tarea efectiva y los códigos empleados para representar esas mismas rutinas. El rendimiento filosófico de esta cuestión me parece evidente, y también me parece obvia la falta de sensibilidad generalmente manifestada ante sus posibilidades. No obstante, pronto tenderemos oportunidad de entender hasta qué punto estamos ante un rasgo realmente definitorio de la computabilidad formal.

El teorema que permite responder a todas estas cuestiones se debe, una vez más, al ingenio de S. Kleene. En la actualidad es frecuente referirse al él bajo el rótulo de *teorema de los Parámetros*, pero también se puede encontrar en los manuales bajo la descripción de *teorema s-m-n.* Esto sólo hace referencia a cierta elección terminológica, completamente convencional, por la que Kleene opta en las primeras presentaciones de su teorema.

[1] Teorema de los parámetros: Sea $\varphi^e(x_1,...x_m,y_1,...y_n)$ una función computable. Entonces para cualesquiera enteros positivos m, n existe una función computable totalmente definida $s(e,y_1,...y_n)$ la cual satisface lo siguiente:

$$-\varphi^{s(e,y_1,...y_n)}(x_1,...x_m)= \varphi^e(x_1,...x_m,y_1,...y_n).$$

Esquema de demostración: Tómese una función computable cualquiera del tipo $\varphi^e(x_1,...x_m,y_1,...y_n)$. A continuación acordamos fijar los argumentos $<y_1,...y_n>$ asignándoles un valor constante que representamos mediante una secuencia $<k_1,...k_n>$. La función resultante se comporta, tal y como hemos dicho ya, como una nueva función $\psi(x_1,...x_m)$ cuya máquina

asociada pasamos a describir. En primer lugar consideramos una subrutina M^k la cual finalizará en una configuración cuya cinta presenta como contenido la secuencia $<x_1,...x_m, k_1,...k_n>$ siempre que inicia su rutina sobre una cinta de cálculo que contiene la secuencia $<x_1,...x_m>$. La tabla de M^k constará, por tanto, de una parte inicial destinada a recorrer el contenido de la secuencia de una configuración inicial para proceder entonces a rellenar las celdas que en número y disposición indiquen las restantes instrucciones de su tabla. Una vez finalizada esa rutina, la secuencia final resultante es adoptada como secuencia inicial de la subrutina que ejecuta la máquina cuyo índice es ***e***, esto es, M^e. Si representamos mediante $M^k\&M^e$ la máquina que se obtiene de concatenar ambas rutinas, podremos afirmar que la máquina $M^k\&M^e$ computa precisamente la función $\psi(x_1,...x_m)$ indicada anteriormente, o en otras palabras, concluimos que dicha función es computable. Parece evidente que el índice de $M^k\&M^e$ depende en exclusiva de los índices ***k*** y ***e***, sucediendo además que ***k*** depende tan sólo de los valores de la secuencia $<k_1,...k_n>$. Puesto que las funciones de codificación y descifrado de máquinas son, como ya se demostró en su momento, recursivas primitivas, podemos, dada una secuencia $<k_1,...k_n>$ y un índice ***e*** hallar la tabla de la máquina $M^k\&M^e$ para entonces calcular su índice. Pero eso supone tanto como afirmar que el índice de $M^k\&M^e$ puede ser efectivamente calculado a partir de los valores de $<k_1,...k_n>$ y ***e***. Puesto que las funciones que permiten obtener esa conclusión son funciones de cifrado y descifrado podemos decir entonces que son totales, de dónde concluimos que para cualesquiera valores de $<y_1,...y_n>$ y ***e*** existe una función recursiva primitiva $s(e,<y_1,...y_n>)$ que arroja el índice ***e'*** de la función $\psi(x_1,...x_m)$. En resumen, la función $s(e,<y_1,...y_n>)$ satisface la identidad propuesta en el teorema y está totalmente definida.

La demostración del teorema indica la posibilidad de reforzar su enunciado sosteniendo no sólo la computabilidad de la función totalmente definida $s(e,<y_1,...y_n>)$, sino su pertenencia a la clase de las funciones recursivas primitivas. No es que esto importe mucho desde un punto de vista general, pero aporta nuevos datos acerca de la complejidad, del tipo de relaciones de composición, que tienen lugar entre rutinas e índices de rutinas. Si resulta más frecuente la formulación que aquí hemos dado ello se debe, tal vez, al mayor impacto que provoca en nuestras intuiciones la capacidad de calcular de manera efectiva $s(e,<y_1,...y_n>)$ para cualesquiera valores de sus argumentos. Ese sólo dato basta para garantizar la inexistencia de porciones de la rutina a que da lugar una función computable que no puedan ser estudiadas separadamente en términos de otras funciones computables. Obsérvese que si $s(e,<y_1,...y_n>)$ no estuviese definida -aun siendo computable- para alguna elección de valores de $<y_1,...y_n>$ entonces la función computable $\varphi^e(x_1,...x_m,y_1,...y_n)$ contendría fragmentos de su rutina virtualmente no computables o, computables tan sólo cuando son completados con datos ajenos a dicha rutina.

Hemos demostrado en definitiva, que las funciones computables pueden ser tomadas a partir de ahora como un agregado de subrutinas cuyos índices podemos identificar y combinar mediante el sólo uso de funciones recursivas primitivas.

El teorema de los parámetros tiene una aplicación inmediata sobre dos colecciones de problemas de evidente peso e interés: la reducción de la computabilidad de una función dada a la de otra ya conocida, y la localización de funciones complejas a través de operaciones definidas sobre los índices de las funciones que la componen.

Como ejemplo del primer tipo de aplicación escogemos un problema de considerable importancia. Pero antes una definición.

[2] *Conjunto diagonal*. Por K, o *conjunto diagonal*, designaremos aquella colección de enteros positivos definida como:

- $K=\{x / \varphi^x(x)$ está definida$\}$.

K representa, como parece evidente, el conjunto de índices de funciones –máquinas- asociado al problema de parada. Al comentar este resultado ya vimos que no había forma de saber cuándo una de estas funciones estaba o no definida, lo cual supone ahora reconocer que no disponemos de un algoritmo que responda *sí* o *no* a cada pregunta acerca de la pertenencia de un entero positivo a K.

La relación entre conjuntos numéricos y funciones computables lleva a introducir una serie de conceptos fundamentales sobre los que me parece adecuado detenernos un momento. Nuestra experiencia matemática enseña que un conjunto cuyos elementos son exactamente aquellos que satisfacen una determinada propiedad no siempre nos permite juzgar y dirimir cuestiones del tipo: ¿es *i* un elemento de S? Por suerte, el grado de dificultad que este tipo de problemas puede ser analizado a través de una serie de nociones que sacan provecho de todo lo visto acerca de la computabilidad de funciones numéricas.

[3] *Conjuntos decidibles (computables)*. Diremos que un conjunto S es *decidible* syss

i. la afirmación $i \in S$ puede ser determinada de forma efectiva y

ii. $i \notin S$ puede ser establecida del mismo modo.

Si S solamente satisface [3.i], esto es, si dispone de un método efectivo para dirimir si $i \in S$, pero no para determinar si $i \notin S$, entonces S recibe el nombre de conjunto *semi-decidible* (semi-computable).

A primera vista la distinción entre conjuntos decidibles y semi-decidibles puede resultar truculenta. Si un conjunto semi-decidible es aquel que dispone de un procedimiento efectivo para determinar si $i \in S$, entonces parece evidente que si ese algoritmo concluye sin arrojar como resultado que $i \in S$ ello será por la sencilla razón de que $i \notin S$. El error en este juicio se encuentra en que el proceso podría comportarse de tal manera que resultase imposible saber si finalmente se detiene arrojando alguna respuesta. Si se nos informa, además de que no se tiene certeza de que el procedimiento finalice para todos los valores acerca de los que ha de juzgar, entonces resulta evidente que un cómputo significativamente largo que aún no haya finalizado no puede ser identificado con ningún juicio acerca de la pertenencia de un natural *i* a un conjunto S. Podría suceder que el algoritmo concluya algunos pasos más adelante informando de que $i \in S$ o, por el contrario, podría suceder que el algoritmo continuase indefinidamente sin arrojar valor alguno.

Para conectar esta discusión con el análisis previo acerca de la computabilidad se precisa involucrar a las funciones numéricas de alguna forma. Dado que el problema que se analiza es el de la pertenencia de un elemento a un conjunto y que las respuestas sólo pueden ser “sí” o “no”, podemos escoger dos naturales como representantes de tales extremos, sean estos “1”, “0” respectivamente, e interpretar ahora que una función $f^n: ù^n \rightarrow \{0,1\}$ define un conjunto de tuplas de números naturales de acuerdo al esquema $\{<k_1,...k_n> / f^n(<k_1,...k_n>)=1\}$. Admitiremos que tales funciones, denominadas *funciones características*, definen siempre un conjunto con independencia de la definición concreta de f^n o de su carácter, efectivo o no efectivo. De este modo, aceptamos como medida de la complejidad de un conjunto numérico la propia de las funciones que lo definen.

Esta forma de incorporar el tópico de la decidibilidad en el campo de la computabilidad de funciones numéricas permite elaborar interesantes alternativas a las nociones que se han introducido líneas atrás. Así,

[4] *Conjuntos recursivos*: Diremos que un subconjunto S de $ù^n$ es *recursivo* syss su función característica f_S definida del modo

$f_S(<x_1,...x_n>)=$ 1 si $<x_1,...x_n> \in S$
0 si $<x_1,...x_n> \notin S$,

es una función recursiva (computable).

[5] *Conjuntos semi-recursivos*: Diremos que un subconjunto S de $ù^n$ es *semi-recursivo* syss su función característica f_S definida del modo

$f_S(<x_1,...x_n>)=$ 1 si $<x_1,...x_n> \in S$
indefinida en otro caso,

es una función recursiva (computable).

Una vez se analiza el problema de la decidibilidad de un conjunto numérico con el concurso de sus funciones características se aprecia que éstas son, en realidad, un artificio algo complejo y tal vez innecesario. La conexión entre la complejidad de subconjuntos de $ù^n$ y las funciones numéricas es mucho más íntima de lo que podría parecer a la luz de las definiciones anteriores.

Imaginemos un listado de los números naturales en su orden estándar -listado, por tanto, infinito y puramente virtual- que presenta la característica de marcar con un símbolo auxiliar "*" algunos de los números que allí figuran. Los naturales así marcados

constituyen evidéntemente un subconjunto $S \subseteq \grave{u}$ cuyos elementos pueden ser extraídos siguiendo un cierto orden. En general, cualquier subconjunto S en $\grave{u}^n$ para el cual exista un procedimiento efectivo capaz de listar sus elementos en un cierto orden recibirá el nombre de *conjunto efectivamente enumerable*, resultando el procedimiento aludido una *enumeración* del mismo. Puede ser un interesante ejercicio demostrar que la noción de conjunto semi-decidible, semi-recursivo y efectivamente enumerable coinciden para subconjuntos que, como S pueden ser ordenados de manera efectiva. Sucede entonces que marcar elementos en un listado de los naturales en su orden estandar es un proceso que puede ser puesto en relación con una función numérica la cual arroja un cierto resultado convenido cuando el argumento evaluado pertenece al conjunto en cuestión: ese resultado constituye una marca. Imaginemos que el resultado convenido o marca consiste, simplemente, en emparejar el argumento analizado con un número natural, sea éste el que fuere y no necesariamente el mismo en todos los casos. Podemos interpretar esa lista como una función numérica parcial en la que los naturales a los cuales se les adjunta un valor módulo f_i^n son exactamente el dominio de dicha función. Si la función parcial así descrita es recursiva, entonces los argumentos para los que ésta arroja un valor pueden ir siendo enumerados efectivamente según se obtienen. Esta apreciación se traduce en:

[6] *Conjuntos recursivamente enumeranbles (r.e.)*. Diremos que un subconjunto $S \subseteq \grave{u}^n$ es *recursivamente enumerable*, r.e. en símbolos, syss constituye el dominio $dom(f_i^n)$ de una función recursiva (computable) parcial f_i^n.

Conjuntos semi-decidibles, semi-recursivos, efectivamente enumerables y, finalmente recursivamente enumerables coinciden en analizar el mismo concepto matemático. No obstante, la última de las nociones indicadas ofrece la mejor conexión intuitiva entre conjuntos y funciones numéricas. El siguiente resultado es buena prueba de ello:

[7] Teorema: Un conjunto $S \subseteq \omega^n$ es recursivo syss tanto él como su complementario son recursivamente enumerables.

Esquema de la demostración: Basta observar que un conjunto recursivo es aquel para el cual existe un procedimiento efectivo para establecer tanto si $i \in S$ como si $i \notin S$. Puesto que S es r.e., entonces si $i \in S$, existe una función recursiva que determina ese extremo de manera efectiva. Si $i \notin S$, entonces pertenece a su complementario módulo ω^n, y puesto que éste es también r.e., obtenemos la misma conclusión que en el caso anterior.

Volvamos ahora a examinar la circunstancia en la que se encuentra el conjunto K definido en [2]. El conjunto K es un conjunto r.e., es fácil percatarse de ello, para el cual existe un argumento directo que establece que no puede ser recursivo –su complementario no es, por tanto, r.e.-. ¿Cuántos tipos distintos de conjuntos aquejados de este problema hay? ¿Se puede reducir el género de indecidibilidad que presentan otros conjuntos, otros problemas, al que afecta a K?

En lo que sigue me interesa hacer referencia no sólo al carácter r.e. de un subconjunto formado por enteros positivos, sino también a la función asociada a dicho subconjunto. Así, si Γ=dom(f) para una función computable f identificable como φ^e, el conjunto Γ aparecerá siempre como Γ^e.

A continuación vamos a considerar un conjunto r.e. Γ^e asociado a la función computable $\varphi^e(x)$ y definimos a partir de esta última una nueva función del siguiente modo:

[8] $\psi(e,x,y) = \varphi^e(x)$.

Puesto que $\varphi^e(x)$ es computable, $\psi(e,x,y)$ lo será a su vez. Si aplicamos ahora el teorema de los parámetros -en una variante algo liberal- sobre $\psi(e,x,y)$ obtenemos una nueva función:

[9] $\psi(e,x,y)= \varphi^{s(e,x)}(y)$

cuya conducta merece la pena analizar. Obsérvese que por su definición en [9] $\varphi^{s(e,x)}(y)$ satisface lo siguiente:

[10] 1. $x\in\Gamma^e \Rightarrow \forall y\ \varphi^{s(e,x)}(y)$ está definida
2. $x\notin\Gamma^e \Rightarrow \forall y\ \varphi^{s(e,x)}(y)$ queda indefinida.

En otras palabras, vemos que es una condición suficiente que $x\in\Gamma^e$ para que $\varphi^{s(e,x)}(y)$ sea total y a su vez, que basta que $x\notin\Gamma^e$ para que $\varphi^{s(e,x)}(y)$ sea la función vacía. Tomemos la primera condición del problema, ¿cuándo podemos garantizar que, fijados los valores *x* y *x*, la función $\varphi^{s(e,x)}(y)$ en la variable *y* está definida?. Es evidente que obtenemos este resultado cuando hacemos depender el argumento de esa función de los parámetros *x* y *e* una vez fijada la función $\varphi^{s(e,x)}$, o en otras palabras, cuando evaluamos si $\varphi^{s(e,x)}(s(e,x))$ está definida. Es evidente que el análisis del caso en el que $x\notin\Gamma^e$ depara idéntica conclusión siguiéndose entonces de la definición de K que,

[11] $x\in\Gamma^e$ syss $s(e,x)\in K$, para la función computable totalmente definida $s(e,x)$.

Siempre es posible considerar que *e* es un entero que se puede incorporar en la definición de una nueva función obtenida a partir de s(e,x) de modo que $\sigma(x)=s(e,x)$. Esto permite transformar [11] en lo siguiente

[12] $x \in \Gamma$ syss $\sigma(x) \in K$, para una función computable totalmente definida $\sigma(x)$.

Desde un punto de vista puramente intuitivo, parece que dos subconjuntos cualesquiera A,B de ù exhiben una conexión explícita por lo que al *problema de la decisión* se refiere si sucede que cada afirmación del tipo i0A puede analizarse equivalentemente y de manera uniforme como una afirmación del tipo j0B para algún *j*. Dicho de otro modo, la complejidad de A es *reductible* a la de B, A#B en símbolos, si existe una función f computable totalmente definida tal que para cada x0A sucede que x0A syss f(x)0B. Desde este punto de vista, el teorema de los parámetros ha permitido demostrar que:

[13] Teorema: Todo subconjunto r.e. Γ de ù valida que Γ K.

Esquema de demostración: La función $\sigma(x)$ invocada en [12] es del tipo requerido.

El uso de la expresión *A es reductible a B* debe interpretarse como la confirmación de que cualquier enunciado del tipo $i \in A$ puede analizarse, sin aumento de complejidad, como una cuestión relativa a la pertenencia de f(i) a B. Eventualmente puede resultar más fácil este extremo que el primero. Es evidente que si para cada par $i,j \in A$ sucede que si f(i)=f(j) entonces i=j, en otras palabras, f(x) es 1-1, entonces la pertenencia de f(i) a B no tiene más consecuencias sobre A que la pertenencia de *i* a A. En tal caso, decimos no sólo que A es reductible a B, sino que es *1-reductible*, $A \ {}_1B$ en símbolos. Es evidente a partir de lo dicho que si $A \ {}_1B$ y B es recursivo (r.e.), entonces A es recursivo (r.e.).

Finalizaré ofreciendo una breve discusión del uso del teorema de los parámetros para representar funciones complejas obtenidas a partir de operaciones sobre otras funciones previamente dadas. Primero analizamos un caso bastante elemental y frecuente en la literatura referido a la operación de composición de funciones. Se trata de localizar esquemáticamente la función que permite determinar la función que resulta al componer otras dos previamente conocidas.

[14] Sean $\varphi^i(x)$, $\varphi^j(x)$ sendas funciones computables entre las cuales se define por composición la función $\varphi^i(\varphi^j(x))$. Nada impide que consideremos *i*, *j* como valores constantes de una función $\psi(i,j,x)$, la cual por el T. de normalización existe y es tal que:

- $\varphi^i(\varphi^j(x))=\psi(i,j,x)$.

Puesto que $\psi(i,j,x)$ es computable, tendrá un índice *e* de manera que $\varphi^i(\varphi^j(x))=\varphi^e(i,j,x)$. Tomamos esta última función y aplicamos el T. de los parámetros:

- $\varphi^e(i,j,x)=\varphi^{s(e,i,j)}(x)$.

Escogemos ahora una función computable totalmente definida de modo que $\sigma(i,j)=s(e,i,j)$, de donde resulta finalmente que:

- $\varphi^i(\varphi^j(x))=\varphi^{\sigma(i,j)}(x)$.

Este resultado no tiene, ciertamente, nada de impresionante, simplemente muestra cómo usar el Teorema de los parámetros para hallar el índice de una función compleja mediante el uso de una función definida únicamente sobre los índices de las

funciones dadas. Sin embargo, sugiere un uso algo más audaz del teorema el cual pasamos a ilustrara continuación.

En [14] la operación de composición se establece sobre funciones fijadas previamente mediante sus índices, detalle que obliga a considerar una función $\psi(i,j,x)$ con valores prefijados. Tómese ahora la función suma x+y definida sobre los naturales y considérense que la variables involucradas toman valores entre los índices de las funciones computables. Eso hace que la operación aludida consista, en realidad, en la función $\varphi^x(z)+\varphi^y(z)$ definida ahora para cualesquiera x,y,z en ù. Si utilizamos el T. de lo parámetros en toda su potencia obtenemos que:

[15] *Ejemplo*: $\varphi^x(z)+\varphi^y(z)=\psi(x,y,z)$ para alguna función computable $\psi(x,y,z)$.

Obviamente entonces,

- $\psi(x,y,z)=\varphi^e(x,y,z)$ para algún entero positivo *e*.

Aplicando el T. de los parámetros,

- $\varphi^e(x,y,z)=\varphi^{s(e,x,y)}(z)$, para alguna función computable totalmente definida s(e,x,y).

Escogemos $\sigma(x,y)=s(e,x,y)$, de donde resulta finalmente que
- $\varphi^x(z)+\varphi^y(z)=\varphi^{\sigma(x,y)}(z)$, para cualesquiera x,y,z.

La posibilidad de definir la operación suma sobre funciones de modo que seamos siempre capaces de obtener para cualesquiera *x*,*y* la función resultante se expresa afirmando que la operación suma es *efectiva sobre índices*. Puesto que la técnica utilizada en [15] no es sino una generalización de la utilizada en [14], podemos concluir

que la composición de funciones y en general cualquier función computable, es igualmente efectiva sobre índices.

En definitiva, hemos podido observar que la manipulación de rutinas -funciones computables- a través de sus índices muestra unos principios de composición que se acomodan perfectamente bien a aquellos que se pueden establecer directamente para los propios objetos a que hacen mención. Quizá sea esto lo que de un modo u otro esperábamos encontrar, pero estoy convencido de que muchas de las preguntas que hemos podido responder aquí ni siquiera hubiéramos podido formularlas adecuadamente en términos puramente informales.

Orientación bibliográfica.

Este teorema se puede encontrar en un estado bastante puro en **[Kleene, 1952]** secc. 65. No obstante, esta referencia no es la más aconsejable para una correcta comprensión del problema. **[Salomaa, 1985]** secc. 4.2 presenta una demostración que apela a la Tesis de Church. Versiones más elementales se pueden encontrar en **[Bridges, 1994]** cap. 3 o en **[Rogers, 1967]**.

Para la cuestión relativa a los conjuntos r.e. y los grados de reducibilidad véase **[Boolos, 1974]** secc. 1 y 15. También en **[Bridges, 1994]** loc. cit. Un tratamiento más extenso de la reducibilidad se puede encontrar en **[Enderton, 1977]** , secc. 8.