más que punto de partida de investigaciones pertinentes: es resultado de un análisis que el 'rigor informal' debe hacer lo más riguroso posible.

Por su parte, la reducción del cálculo infinitesimal a análisis *no-standard* permite a Robinson la aplicación de la Teoría de Modelos al examen de la acción recíproca que ha existido entre las ideas filosóficas y las ideas técnicas en temas y momentos que son relevantes en el desarrollo del cálculo infinitesimal; desde Newton y Leibniz hasta Cauchy y Weierstrass.

Más estrictamente filosófica puede considerarse la segunda contribución de Kreisel. Rechaza en ella la tesis de Gödel de que si las Matemáticas fueran construcciones mentales serían recursivas, al tiempo que defiende el carácter no-mecanicista del razonamiento matemático. Interpretado éste como fenómeno biológico conduciría a la naturaleza no-mecanicista de la Biología.

También relacionado con el tema de la recursividad —la Teoría de las Funciones Recursivas es uno de los logros más significados de los últimos tiempos— es el artículo de Harley Rogers. La repercusión del desarrollo de la teoría es notoria: Lógica y Fundamentos, insolubilidad, invarianza, estructuas y análisis recursivos, etc. Los trabajos de Kleene, Mostowski y Post permiten a Rogers fijar el año 1943 como límite entre dos épocas de la historia de la teoría. La solución de las dificultades básicas en la primera de ellas posibilita el espectacular desarrollo posterior.

Finalmente, Tarski —recurriendo a los resultados de la Lógica, la Teoría de Conjuntos y la Teoría de Modelos— hace de la Geometría euclidiana —un problema ya clásico de la Filosofía de las Matemáticas— 'a topic of mathematical investigation' al establecer con rigurosidad el lenguaje de la misma y las correspondientes reglas de demostración.

Sin duda, los trabajos recopilables serían muchos más. Incluso —como señala Hintikka— se nota la ausencia de áreas tan significativas como la Teoría de Conjuntos. No obstante, *The Philosophy of Mathematics* resulta ser un excelente libro cuya utilidad no se limita a las personas interesadas en la Filosofía de las Matemáticas.

R. Beneyto

BARRON, D. W.: *Recursive techniques in programming.* London. Macdonald & Co., 1968. 64 págs.

El libro de D. W. Barron pertenece a una serie de trabajos monográficos sobre computación que dirige el profesor Stanley Gill, del Imperial College (London). En cuatro capítulos y un apéndice suministra una detallada información sobre el uso de la Recursión en computadores, insertando al final de cada capítulo la bibliografía más importante sobre el tema.

El uso de técnicas recursivas en programación —en lugar de las iterativas, más comunes— es objeto de polémica entre los programadores; algunos las consideran un lujo innecesario. El libro de D. W. Barron constituye un alegato en favor de las técnicas recursivas, poniendo de manifiesto diversos casos en los que una huida de la programación recursiva daría como resultado programas más esotéricos y artificiales, y señalando como necesaria una estructuración igualmente recursiva para el caso de programas que hubieran de ser mucho mayores que los actuales, debido a que su montaje tendría que realizarse a base de fragmentos ordenados de una manera jerárquica.

El concepto de recursión, de uso frecuente en matemáticas y lógica, se refiere tanto a procesos —el de integración doble, por ejemplo— como a un modo de definir funciones —funciones recursivas, definiciones recursivas— (un proceso es recursivo si se contiene él mismo como subproceso; una función está definida recursivamente si lo está en términos de sí misma). En programación se trata corrientemente con procesos inherentemente recursivos —evaluación de una integral doble, evaluación de una función de Ackerman, etc.— y también con datos que de suyo poseen una estructura recursiva —expresiones algebraicas, listas (*lists*)—. Es obvio que en estos casos resulta óptima la utilización de sistemas que permiten la recursión —IPL, ALGOL, lenguajes funcionales: LISP—. La polémica a que nos hemos referido anteriormente sólo encuentra su lugar si lo que se trata de programar son datos que no poseen necesariamente estructura recursiva, o procesos que no son inherentemente recursivos (por ejemplo, evaluación de una función factorial, máximo común divisor de dos números naturales, etc.) ante este tipo de casos, el programador debe elegir entre un programa (recursivo), muy cercano al lenguaje matemático convencional de las definiciones recursivas, y otro (iterativo) más artificial pero de ejecución (*running*) más rápida y de más sencillo *debugging* (correción de errores). Cada caso concreto debe implicar una decisión por una de las dos técnicas o por la combinación de ambas, pero es improcedente el rechazo sin más de las técnicas recursivas.

Las definiciones recursivas, con vistas a la programación, pueden expresarse muy adecuadamente mediante un tipo de notación (las "expresiones condicionales" del LISP) debida a Mc.Carthy; [1] el autor la utiliza con alguna modificación para definir funciones que así se dejan traducir de modo inmediato a un programa recursivo ALGOL capaz de evaluarlas, contrastando por su sencillez con otra programación no-recursiva del problema en el mismo lenguaje.

En el segundo capítulo, muy ilustrativo, el autor ejemplifica distintos casos del cálculo numérico —Resolución de ecuaciones, Integración aproximada, Particiones, etc.— para los que resulta posible,

---

[1] McCarthy, J.: 'Recursive functions of symbolic expressions and their computation by machine.' *Commun. Assoc. Computing Machinery*, Vol. 3, No. 4, p. 184, 1960.

cuando no más conveniente —Particiones— una programación recursiva; expone también diversos tópicos de la computación —compilación, clasificación (*sorting*), manipulación de expresiones algebraicas, inteligencia artificial— en los que el uso de la recursión constituye una ventaja o una necesidad. El proceso de expresiones algebraicas, por ejemplo, resulta simplificado si se elige una representación dentro de la máquina —implementando un lenguaje recursivo— que refleje la estructura recursiva de éstas; tal representación tendría estructura de lista (*list structure*). Cualquier expresión algebraica consta esencialmente de distintos operadores y operandos adecuadamente combinados; todos ellos pueden ordenarse en una lista (*list*) que conste de tres partes, de este modo: operador / (operando, / operando); si los operandos fuesen a su vez expresiones constituirían sub-listas de la misma estructura. La expresión se procesaría por medio de un programa con dos rutinas recursivas (OPERADOR y OPERANDO) que irían siendo llamadas durante la ejecución.

La ejecución de programas recursivos (capítulo 3.º) precisa de ciertos mecanismos *software* —implementados por medio de macrotécnicas— o de máquinas que los contengan como parte de su *hardware* (ejemplos de éstas son los computadores: English Electric KDF9 7 y Burroughs B5000). Estos mecanismos, aunque pueden alcanzar un alto grado de complicación, pueden reducirse en esencia a un *stack* o *push-down store* (memoria o parte de memoria que comprende cierto número de registros en columna (*hardware stack*) o funciona como si los comprendiese (*software stack*) y de la que entran o salen datos (verdaderamente, *words*) de uno en uno, de modo que siempre es el último que ha entrado el primero en salir) y dos subrutinas que además de realizar cualquier operación programada se encargan de restaurar a su primitivo estado, respectivamente, los registros del *stack*.

Estos elementos, básicamente necesarios para la ejecución de cualquier programa recursivo, hacen mucho más lento el proceso si son empleados en el caso de programas iterativos; por lo cual han tenido que crearse sistemas —PL/1, CPL— con cuya utilización el programador puede indicar a la máquina qué fragmentos de un programa debe tratar como recursivos y cuáles no. Irons-Feuerzig y Hawkins-Huxtable, en la misma línea, han propuesto métodos mediante cuya aplicación el computador realizaría la distinción automáticamente.

El último capítulo del libro lo dedica el autor a dejar constancia de los intentos realizados —Mc.Carthy— por establecer una teoría matemática de la computación, mediante la cual pudiera probarse la equivalencia de programas que realizan —recursiva e iterativamente— el mismo trabajo. Este tan interesante campo de investigación podría posibilitar la transformación automática de relaciones recursivas a definiciones iterativas.

E. Casabán