1 INTRODUCCIÓN

Esta primera unidad tiene como objetivo presentar los conceptos básicos acerca de la resolución mecánica de problemas (sustentada sobre el concepto de algoritmo), presentar una breve historia de las máquinas de cálculo (capaces de ejecutar dichos algoritmos), mostrar el papel que puede desempeñar la naturaleza como fuente de inspiración computacional, y discutir la importancia de la modelización matemática, sentando las bases para la mejor comprensión de las siguientes unidades.

1 RESOLUCIÓN MECÁNICA DE PROBLEMAS

El hombre ha tenido siempre una sublime aspiración: la constante mejora de la calidad de vida. En ese menester, desde el comienzo de los tiempos se ha planteado cuestiones a las que ha tenido que hacer frente tratando de buscar respuestas. La formulación de estas cuestiones, el planteamiento de estos problemas, le ha llevado a intentar resolver los mismos, y en la medida de lo posible de una forma general, evitando tener que plantearse las mismas cuestiones al enfrentarse a las mismas situaciones una y otra vez. De este modo, el estudio de los problemas planteados le ha conducido, de manera natural, a la búsqueda de procedimientos sistemáticos que permitieran obtener soluciones a través de la realización de una serie de tareas elementales debidamente secuenciadas. Este proceso de mecanización de las soluciones permite sistematizar el modo de resolver los problemas cada vez que se repita la misma situación, facilita la transmisión del conocimiento y el aprendizaje científico, en general.

¿Qué tipo de problemas estamos generalmente interesados en resolver? Para responder a esta pregunta, hagamos una reflexión previa. Necesitamos, en primer lugar, establecer la distinción entre *problema abstracto*, como una clase o conjunto de problemas específicos, *concretos* o *instancias*. Por ejemplo, el problema "hallar la suma de dos números" es un problema abstracto, mientras que el problema "sumar 15 y 56" sería un problema concreto. Lógicamente, el problema de resolver la suma de 15 y 56 podrá tener interés para quien se plantea esa cuestión, pero lo más interesante será proporcionar un mecanismo para sumar

dos números, cualesquiera que sean, de modo que podamos aplicar la misma sistemática cada vez que nos encontremos ante la misma situación de tener que sumar dos números. Así, estaremos interesados en resolver problemas abstractos, a los que nos referiremos a veces de forma abreviada como problemas.

Por otra parte, ¿cómo podemos resolver un problema? ¿qué tipo de resolución debemos llevar a cabo? Analicemos esta cuestión...

¿Cuál es el método de resolución de un problema? Básicamente, podemos suponer que consta de tres etapas: estudio, diseño de la estrategia y proporcionar, si es posible, una solución. La diferencia fundamental estriba en la estrategia de resolución. Estaremos interesados en resolver los problemas de forma mecánica, dotando de un procedimiento mecánico (algoritmo) que proporcione una solución al problema, siempre que sea posible (lo cual no siempre estará garantizado).

Así, diremos que un problema es *resoluble algorítmicamente* si es posible elaborar un algoritmo tal que para cada instancia del problema (*dato de entrada*), nos proporciona una solución concreta. En caso contrario, diremos que es *irresoluble* (o *indecidible*).

Hablamos de un problema resoluble algorítmicamente en lugar de mecánicamente pero, ¿qué es un *algoritmo*?[1] Veamos algunas definiciones:

La definición ofrecida por la Real Academia,

> *Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema*,

nos da una primera idea intuitiva, muy cercana a lo que ya se comentaba anteriormente sobre *proporcionar un procedimiento sistemático compuesto por tareas elementales debidamente secuenciadas*. Hasta aquí estaríamos diciendo suficiente como definición general de *algoritmo*, si bien no es una definición plenamente satisfactible en el ámbito de la Computación. Para complementar lo anterior utilizaremos también la siguiente definición:

> *Es un método preciso susceptible de ser utilizado por un ordenador para la resolución de un problema.*

La principal diferencia entre las dos definiciones es la fuerte restricción que la condición *ser utilizado por un ordenador* impone a las operaciones permitidas. La resolución mecánica de problemas que da título a esta sección abre la posibilidad de usar máquinas capaces de ejecutar esas tareas y asistir al hombre en el proceso general de resolución de problemas, y para ello nos servimos de máquinas.

Vamos a precisar esta cuestión. Un algoritmo es un conjunto finito de reglas o instrucciones $I_1, \ldots, I_n$, tal que, dado un dato de entrada x, nos proporciona, al cabo de un número finito de pasos, un dato de salida, que es el resultado de aplicar el algoritmo al dato x. Para cada instrucción I_j del algoritmo se debe especificar:

[1] El origen de la palabra *algoritmo* se lo debemos al autor persa, Abu Ja'far Mohammed ibn Mussa al Khowarizmi (825 a.C.), que escribió un texto sobre Matemáticas donde recopiló algoritmos para el álgebra.

1. Cuál es la tarea a realizar.

2. Cuál es la siguiente instrucción a ejecutar. Si no hay siguiente instrucción, diremos que el algoritmo para o termina.

En cada paso de un algoritmo se pueden realizar una o más operaciones. Desde luego, para que un ordenador pueda ejecutar dichas operaciones es necesario exigir algunas condiciones:

- Cada operación debe de estar perfectamente definida, indicando clara e unívocamente qué se debe de hacer.

- Debe de ser efectiva, es decir, susceptible de ser realizada por una persona en tiempo finito.

Además, otra característica importante es que debe de parar después de un número finito de pasos. Esta condición no se exige para ciertos procedimientos, que suelen denominarse *procedimiento computacional*. Un ejemplo importante es el sistema operativo de un ordenador digital: está diseñado para controlar la ejecución de tareas, de manera que cuando no hay tareas por realizar continúa en estado de espera, hasta que una nueva es introducida.

Volvamos por un momento a las operaciones permitidas. Una forma de cumplir las exigencias antes reseñadas es escribir tales operaciones en un lenguaje de programación (bien diseñado, para que cada sentencia escrita en dicho lenguaje posea un único significado). Un *programa* es la expresión de un algoritmo en un lenguaje de programación. Para el estudio de los algoritmos no es necesario (y desde cierto punto de vista, puede que, incluso, no sea aconsejable) manejar su traducción a un lenguaje concreto. Sin embargo, sí debemos describirlos con cierta exactitud. Para ello, se utiliza un lenguaje (*pseudo–código*) para leerlos con facilidad. Las instrucciones del lenguaje que adoptaremos en este capítulo son traducibles de forma prácticamente automática a cualquier lenguaje de programación:

1. $variable \leftarrow expresión$

2. **si** $condición$ **entonces** $instrucción$

3. **si** $condición$ **entonces** $instrucción$ **sino** $instrucción$

4. **mientras** $condición$ **hacer** $instrucciones$

5. **repetir** $instrucciones$ **hasta que** $condición$

6. **para** $variable \leftarrow valor\ inicial$ **hasta** $valor\ final$ **hacer** $instrucciones$ (En algunos casos puede especificarse el salto.)

7. etiqueta: *instrucción*

8. **ir a** *etiqueta*

9. **Procedimiento** *nombre* (*lista de parámetros*)
10. **Función** *nombre* (*lista de parámetros*)
11. *devolver*

Esta formalización es útil para el estudio de ciertos aspectos de los algoritmos, tales como la corrección y el coste. Sin embargo, no es buena para tratar otro tipo de problemas.

Las cuestiones que se abordan en el estudio de los algoritmos son:

1. **Diseño:** De todas las cuestiones, ésta es la más difícil de sistematizar. En cierto modo, podemos equiparar el acto de crear o diseñar algoritmos con el descubrimiento y *demostración* de nuevos resultados[2]. No obstante, existen ciertas pautas o técnicas de diseño que son útiles para este fin. Así, la metodología que se sigue para *enseñar a diseñar* consiste en exponer algoritmos clásicos que ilustran los distintos esquemas. Véanse, por ejemplo, [1] y [5]. Otro acercamiento consiste en exponer métodos de construcción de algoritmos a partir de *especificaciones* (condiciones formalizadas que caracterizan el problema). Este acercamiento es limitado, pero se puede considerar como el establecimiento de una buena fundamentación de la programación (véase, por ejemplo, [2]).

2. **Corrección:** El diseño es un acto de creación, que podemos suponer igual de difícil que demostrar. Como en una demostración, puede ocurrir que se cometan errores. Una rigurosa aproximación a la verificación era, quizás, una de las cuestiones sobre algoritmos que se encontraba más olvidada. El estudio de los algoritmos no se puede limitar al diseño. En el otro extremo podemos colocar ciertos métodos efectivos implícitos en algunas pruebas (por ejemplo, en Matemáticas, la demostración del teorema del rango para matrices) donde la pregunta es ¿la prueba nos asegura que el algoritmo es correcto?

   Para responder a este tipo de cuestiones debemos responder a dos preguntas:

   a) ¿Está el algoritmo bien definido? Es decir, ¿el algoritmo para sobre todo dato de entrada y devuelve un resultado?

   b) ¿Calcula lo que realmente pretendemos? (¿devuelve una solución?).

3. **Análisis:** la ejecución de un algoritmo obliga al procesador a realizar operaciones y a usar espacio de memoria para almacenar datos. El análisis de un algoritmo consiste en determinar la cantidad de recursos (tiempo y espacio, por ejemplo) que se necesita para su ejecución, utilizando herramientas matemáticas. La definición de complejidad que utilicemos debe de ser capaz de precisar, dados dos algoritmos que resuelven el mismo problema,

[2] Afirmación debida a C.A.R. Hoare, que justifica adecuadamente en su ya clásico trabajo [4], donde expone una *teoría matemática de la programación*.

cual de los dos es *mejor* (tiene menor coste) *independientemente de la máquina que lo ejecuta*.

4. **Computabilidad:** Un algoritmo A (respectivamente una instancia suya, un programa P), define una aplicación $f_A : D \to R$ (respectivamente f_P) del conjunto de ejemplares o datos de entrada en el conjunto de resultados. Un problema de computabilidad es el siguiente: Dada una función $f : D \to R$, ¿Existe un algoritmo A tal que $f = f_A$ (es decir, que resuelve el problema)?

2 HISTORIA DE LAS MÁQUINAS DE CÁLCULO. POTENCIA Y LIMITACIONES

Este capítulo está centrado en dos conceptos fundamentales: computación y máquina. Para ilustrar de manera informal estos términos, imaginemos por un instante que disponemos de un terreno en las afueras de la ciudad en la que vivimos y tenemos mucha ilusión por construir allí una casa que sirva de esparcimiento de la familia. Entonces, en primer lugar, hablaríamos con un equipo de arquitectos al que detallaríamos nuestros gustos y preferencias. Al cabo de un tiempo, el equipo nos mostraría una serie de papeles con letras, símbolos, gráficos, etc., tras los cuales existe una gran cantidad de cálculos, algunos de ellos muy sofisticados. Allí aparecería el esquema/bosquejo de un edificio así como las distintas partes del mismo, especificando la composición de materiales, su proporción, orden de ejecución, etc. Son los *planos* del edificio.

A continuación, hablaríamos con un constructor para que elaborara un presupuesto y, una vez aceptado, se encargara de implementar las ideas que subyacen en esos planos, asesorado por los arquitectos, y utilizando los materiales y la tecnología adecuada. Podría suceder que el diseño de los planos fuera muy vanguardista, hasta el punto de que la tecnología actual no permitiera la edificación correspondiente a esos planos.

En este símil, los planos constituyen un *modelo de computación* y el edificio construido viene a ser una *máquina* de ese modelo. Se ha utilizado el término *modelo* como calificativo de computación.

De forma más precisa, se puede afirmar que un *modelo de computación* consiste, básicamente, en definir de manera rigurosa qué es un *procedimiento mecánico* o *algoritmo*. Para ello, se detalla cómo se puede identificar o reconocer un tal procedimiento (*sintaxis* del modelo) y cómo se ejecuta para materializar cálculos (*semántica* del modelo). Así pues, la sintaxis del modelo determina la apariencia física de los procedimientos y la semántica indica la forma en que éstos se comportan. Pueden existir distintos modelos de computación, según el criterio considerado para precisar los procedimientos específicos de los mismos. Ahora bien, ¿qué existe en la trastienda de un modelo de computación? Tres cosas, a saber, Matemáticas, Matemáticas y más Matemáticas.

La aparición de los primeros métodos formales/precisos de razonamiento (posiblemente en Mesopotamia, y mejorados sustancialmente en la civilización griega) proporcionó una herramienta interesante para poder expresar de forma algo más precisa el concepto intuitivo de *procedimiento mecánico* o *algoritmo*.

Seguidamente se ilustra el concepto de *algoritmo* y otras nociones directamente relacionadas con él, mediante un ejemplo. Imaginemos por un instante que se desea preparar un plato de cocina de acuerdo con cierta receta que aparece en una revista. Entonces se va al mercado a fin de comprar todos los ingredientes necesarios para su elaboración, se seleccionan los utensilios adecuados (microondas, sartenes, cuchillos, platos, horno, etc.) que se van a utilizar y nos ponemos manos a la obra, siguiendo escrupulosamente, paso a paso, cada una de las instrucciones que están secuenciadas en la receta. Al cabo de un cierto tiempo (según la destreza del cocinero/usuario) tendremos preparado el plato deseado.

En este ejemplo, la *máquina* (*hardware*) que realiza la ejecución estaría formada por el cocinero (que *interpreta* la receta, haciendo de *unidad de control*, y la lleva a efecto, haciendo de *unidad aritmética*, de cálculo) y los utensilios usados. La propia cocina nos permite almacenar (a modo de una *unidad de memoria*) algunas tareas parciales que se van realizando para conseguir materializar la receta (el sofrito, las patatas peladas, etc.). El conjunto de todos los ingredientes, en las cantidades señaladas en la receta, constituirían el *dato de entrada*; la receta es el *algoritmo* (o *software*), y el plato elaborado es el *dato de salida*. Estos son los elementos básicos que intervienen en la ejecución de un procedimiento mecánico por parte de una máquina: datos de entrada y de salida (E/S), software y hardware.

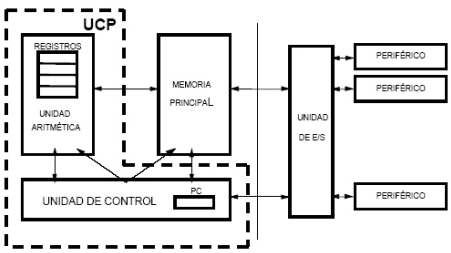

FIGURA 1: Arquitectura de von Neumann.

Una *máquina* de un modelo de computación es un dispositivo (virtual o físico) susceptible de ejecutar los procedimientos del modelo (los *cálculos*). Las máquinas son las encargadas de asistir al hombre en la dura tarea de materializar los cálculos y, por tanto, le ayudan a resolver problemas de la vida real. Una vez que se dispone de un modelo de computación, se ha de encargar a los *técnicos especialistas* que lo implementen en una máquina usando, por ejemplo, la tecnología electrónica. Serían esos técnicos, guiados por los diseñadores del modelo, los que llevarían a cabo la tarea de construir una máquina del mismo. Podría suceder que algún modelo de computación diseñado no pudiera ser implementado en una máquina con la tecnología disponible en la actualidad.

A lo largo de la historia, el hombre ha diseñado máquinas para que le asistan en la tarea de realizar ciertos cálculos tediosos. Desde el ábaco (aproximadamente, año 1000 a. C.) hasta la máquina tabuladora de H. Hollerit (1886), pasando por las tablas de J. Neper (1617), la regla de cálculo de E. Gunter (1620) y W. Oughtred (1623), la máquina de B. Pascal (1642), la de G. Leibniz (1670-1694), el telar de J. M. Jacquard (1801-1805) o la máquina de diferencias de Ch. Babbage (1821). Estas máquinas de *propósito específico* sólo permitían la resolución de un conjunto restringido de problemas.

Se dice que una máquina es de *propósito general* o *universal* si es capaz de resolver todos los problemas que son resolubles de manera mecánica, de acuerdo con la idea intuitiva de este término (tesis de Church-Turing).

FIGURA 2: Alan Turing.

FIGURA 3: Alonzo Church.

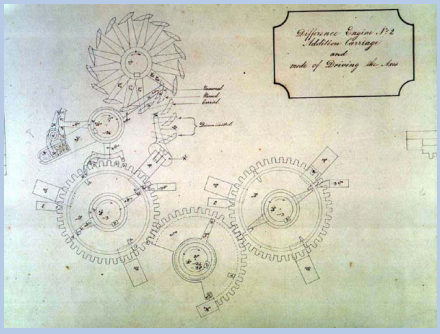

FIGURA 4: La máquina analítica.

Entre 1847 y 1849, Ch. Babbage diseñó los planos de la primera máquina de *propósito general* que el propio autor denominaría *máquina analítica*. Ese dispositivo no pudo ser construido ya que no se disponía de la tecnología adecuada. El *arquitecto* había sido demasiado vanguardista e, incluso, rupturista.

No obstante, estas ideas inspiraron a J. von Neumann y a sus discípulos, J. P. Eckert y J. W. Mauchly para diseñar unos nuevos planos (*arquitectura de von Neumann*) que darían lugar a la construcción en 1946 del primer ordenador de propósito general, el ENIAC (Electronic Numerical Integrator And Computer).

¿Qué hacemos con una máquina? Resolver problemas. Analicemos un poco esto...

Vamos a comenzar con un problema de tipo doméstico: en nuestro hogar se ha deteriorado un interruptor de la luz, ¿qué se necesita para repararlo? En

primer lugar, conocimiento; en segundo lugar, capacidad para implementar ese conocimiento; y, finalmente, disponer del material necesario. En este contexto podríamos distinguir entre *saber resolver* el problema y *ser capaz* de resolverlo. Una persona podría arreglar el interruptor sin saber cómo hacerlo, bastaría que fuese capaz de ejecutar las instrucciones que le indicara un experto en electricidad, disponiendo del material necesario. Podría suceder que una persona supiera qué se ha de hacer para arreglar un interruptor pero, en cambio, no fuera capaz de hacerlo debido a cierta incapacidad física o a que no dispusiera del material adecuado.

A continuación, consideremos el siguiente problema de tipo numérico:

PARIDAD

Dado un número natural, determinar su paridad

Para su resolución sería suficiente reconocer números y saber una regla de caracterización de los números pares; por ejemplo, son aquellos cuya última cifra es 0,2,4,6 u 8. También sería suficiente, saber dividir por 2 y conocer la regla de que un número es par si su división por 2 es exacta, y sólo en ese caso.

¿Qué capacidades debería tener una máquina para poder solucionar cualquier problema resoluble mecánicamente?

Una máquina de cálculo de propósito general es una entidad capaz de realizar de manera autónoma las siguientes tareas básicas: (a) saber sumar 1; (b) saber distinguir si un número es o no igual a 0; (c) restar 1 a un número mayor que cero; y d) ¡obedecer!

En esta definición nos estamos limitando al caso de máquinas que sólo manejan números naturales. No obstante, gracias al uso de codificaciones adecuadas, ésta no es una restricción excesiva.

Para que una máquina resuelva de hecho un cierto problema, en primer lugar se ha de disponer de un procedimiento mecánico que lo resuelva y, a continuación, se ha de usar un determinado lenguaje a fin de que la máquina *entienda* ese procedimiento y pueda ejecutar las tareas indicadas en el mismo y en el orden correcto. Así por ejemplo, en el problema antes indicado, un procedimiento mecánico de resolución descrito en un lenguaje informal sería el siguiente:

Entrada: un número natural x.
- 1.- Si x vale 0, devuelve PAR.
- 2.- Si no,
  - 3.- Al número x réstale 1 y sigue llamándolo x.
  - 4.- Si x vale 0, devuelve IMPAR.
  - 5.- Si no,
    - 6.- Al número x réstale 1 y sigue llamándolo x.
    - 7.- Vuelve a empezar (vuelve a la instrucción número 1).

Una máquina de cálculo de propósito general (no humana) tiene una serie de capacidades pero, en cambio, no tiene el conocimiento necesario para resolver un problema. Por ello, si un experto humano dispone de un procedimiento mecánico que resuelve un problema, entonces éste puede ser resuelto con la ayuda de la máquina con tal de comunicarse ella a través de un lenguaje especial que la máquina *entienda*: un *lenguaje de programación*. La expresión de un procedimiento mecánico en un lenguaje de programación se denomina programa. Así por ejemplo, el anterior procedimiento mecánico se puede escribir de la siguiente manera en un lenguaje de programación llamado `GOTO`:

$$
\begin{aligned}
&[\,B\,]\ IF\ X \neq 0\ GOTO\ A\\
&\qquad Y \leftarrow Y + 1\\
&\qquad IF\ Y \neq 0\ GOTO\ E\\
&[\,A\,]\ X \leftarrow X - 1\\
&\qquad IF\ X \neq 0\ GOTO\ B
\end{aligned}
$$

La expresión anterior describe un programa `GOTO` que será comprensible por toda máquina que posea un traductor (*compilador*) de ese lenguaje. Además, la comunicación tendría que ser implementada a través de algún dispositivo físico; por ejemplo, un teclado en el caso de ordenadores convencionales. En ese programa, la respuesta está codificada en la variable Y a través de su contenido: el valor $Y = 1$ se identifica con *número par* y el valor $Y = 0$ con *número impar*.

En relación con la resolución del problema anterior ¿qué sucedería si una de las tareas básicas que realiza la máquina fuera la división por 2? En este caso, un procedimiento mecánico que resuelve el problema podría ser descrito de la siguiente manera:

Entrada: un número natural x.
- 1.- Divide x por 2.
- 2.- Si el resto vale 0, devuelve PAR.
- 3.- Si no, devuelve IMPAR.

Un programa GOTO extendido que lo resuelve sería el siguiente:

$$Z \leftarrow RM(DIV(X,2))$$
$$IF\ Z = 0\ GOTO\ E$$
$$Y \leftarrow Y + 1$$

En este caso, el lenguaje utilizado sería más simple al haber considerado expresiones como, por ejemplo, $Z \leftarrow RM(DIV(X,2))$ que no son instrucciones elementales en el lenguaje inicial. En esa expresión, la variable auxiliar Z tiene como valor el resto de la división del número X por 2.

¿Son las máquinas omnipotentes? ¿carecen de limitaciones? Respondamos brevemente a esta cuestión.

Con la irrupción de los ordenadores, los matemáticos trabajaron activamente en formalizar la noción de algoritmo y se crearon varios modelos formales de computación, para dotar así de un *significado* formal al concepto de *ejecución en una máquina*: máquinas de Turing, funciones recursivas... Es muy importante reseñar que es con esta formalización con la que se pueden plantear –y resolver– algunas cuestiones fundamentales, como, por ejemplo, mostrar un problema irresoluble algorítmicamente. Estudiar las limitaciones de estos modelos formales es el objetivo de la Teoría de la Computabilidad. Nótese que hay una cierta asimetría en este tipo de cuestiones. Para afirmar que un cierto problema es *resoluble algorítmicamente* basta encontrar un algoritmo que lo resuelva. En cambio, para afirmar que es indecidible es imprescindible disponer de una definición formal de algoritmo, para poder afirmar que *ningún* algoritmo, según esta definición, lo resuelve[3]. El desarrollo de la teoría de la recursión es la pieza fundamental en la resolución de este tipo de cuestiones. No vamos a extendernos en las mismas, estudiadas como hemos comentado por la Teoría de la Computabilidad, pero podemos concluir indicando que las máquinas tienen sus limitaciones, hay problemas no resolubles algorítmicamente, indicidibles.

Además, como analiza la Teoría de la Complejidad Computacional, existen problemas resolubles algorítmicamente pero que presentan limitaciones en cuanto al tiempo que requeriría, en el mejor de los casos, la mejor solución al problema.

Afortunadamente contamos con máquinas reales cada vez más rápidas para solucionar estos problemas. ¿Es suficiente?

La irrupción de los ordenadores electrónicos a mediados del siglo pasado significó un avance cualitativo en la resolución de problemas concretos que, hasta ese momento, habían sido inabordables. En el año 1983, la euforia producida por tan importante logro quedó frenada en seco cuando R. Churchouse demostró la existencia de un límite superior en la velocidad de cálculo de los procesadores construidos a partir de chips electrónicos. Desgraciadamente, el panorama se

[3] En 1936 Alan Turing ofrece el primer ejemplo de problema indecidible, el *problema de la parada*, que describiremos más adelante.

oscureció notablemente al constatarse que ese límite es claramente insuficiente para atacar la resolución de problemas muy importantes de la vida real. Más concretamente, se demostró que tales problemas jamás podrían ser resueltos por máquinas cuyo soporte físico fuese el electrónico, a menos que sucediera un *milagro*, expresado en forma de refutación de una famosa conjetura ($\mathbf{P} \neq \mathbf{NP}$) que es, hoy día, uno de los problemas abiertos más importante de la ciencia (para más detalles, ver [3] y [6]).

Ante esa tesitura, el hombre se plantea la posibilidad de diseñar nuevas máquinas que respondan a otra forma de realizar cálculos y cuyo soporte físico sea distinto, con el fin de que puedan llegar a superar la barrera inherente a los dispositivos electrónicos que, a partir de ahora, denominaremos *convencionales*. En ese escenario aparece la Naturaleza viva como fuente de inspiración, dando origen a la disciplina de la *Computación Natural*, cuya finalidad es el análisis de modelos y técnicas computacionales inspiradas por la Naturaleza, tratando de comprender el mundo que nos rodea en términos de procesamiento de la información.

3 LA NATURALEZA VIVA COMO FUENTE DE INSPIRACIÓN COMPUTACIONAL

Las máquinas electrónicas son dispositivos físicos y, como tales, tienen unas limitaciones tanto en lo que respecta a la velocidad de cálculo como a la miniaturización de las componentes físicas que lo integran.

> A finales de la década de los cincuenta del pasado siglo, el premio Nobel R.P. Feynman describió los ordenadores *sub–microscópicos* e introdujo el concepto teórico de *computación a nivel molecular*, postulándolo como una innovación revolucionaria en la carrera por la miniaturización. Las ideas de R.P. Feynman adquirieron especial relevancia a partir de 1983, cuando R. Churchhouse demostró la existencia de una limitación en la velocidad de cálculo de las máquinas electrónicas, así como en el tamaño de los microprocesadores. R.P. Feymann llegó a afirmar que *algún día sería posible almacenar la Enciclopedia Británica en la punta de una uña*.

En este contexto, la búsqueda de modelos alternativos de computación se hace necesaria y está encaminada al diseño de nuevas máquinas que puedan superar, algún día, la barrera que se deduce del resultado de R. Churchhouse. En las últimas décadas, esta búsqueda ha propiciado la introducción de nuevos modelos de computación sustancialmente distintos de los clásicos o convencionales (máquinas de Turing, funciones recursivas, λ-cálculo, etc.) que proporcionan una mejora importante en la cuantificación de los recursos computacionales, en el marco de una posible implementación práctica.

La *Computación Natural* es una disciplina cuyo objetivo es el estudio e implementación de los procesos dinámicos que se dan en la Naturaleza viva y que son

susceptibles de ser interpretados como procedimientos de cálculo. Es decir, trata de capturar el modo en que la Naturaleza actúa/opera sobre la materia y la forma en que lleva *calculando* desde hace miles de millones de años. La Naturaleza viva ha sido fuente de inspiración computacional desde mediados de los cuarenta del pasado siglo.

> En 1943, W.S. McCulloch y W. Pitts propusieron el primer modelo de redes neuronales artificiales (precursores de los autómatas finitos), inspirándose en la organización y funcionamiento del cerebro, sirviendo de ejemplo para modelos posteriores de M. Minsky y F. Rosenblatt, entre otros. En 1975, J. Holland introdujo los algoritmos genéticos, modelo computacional inspirado en procesos de evolución y selección natural de los seres vivos. Estos algoritmos establecen una analogía entre el conjunto de soluciones de un problema y la colección de individuos de una población natural, codificando la información de cada solución en una cadena a modo de cromosoma. En palabras del propio Holland: *Se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de evolución simulada*. Tanto las redes neuronales artificiales como los algoritmos genéticos han sido utilizados para mejorar el rendimiento de las máquinas convencionales e incluso su propia arquitectura.

La Computación Natural nace como disciplina a finales de 1994, a raíz del experimento de L. Adleman que resolvió una instancia pequeña de un problema *presuntamente intratable*, desde el punto de vista computacional (cualquier solución mecánica conocida necesita una cantidad de recursos que es exponencial en el tamaño de la instancia), usando técnicas de biología molecular para la manipulación de moléculas de ADN (para más detalles, ver [7] y [9]).

4 LA IMPORTANCIA DE LA MODELIZACIÓN MATEMÁTICA

En cierto sentido se puede afirmar que el intento de entender ciertas partes del mundo ha sido un sueño constante de la humanidad. Los esfuerzos encaminados a capturar exactamente esa parte concreta del mundo que es realmente significativa, en ciertos contextos, con el fin de destilar su esencia, ha conducido a avances importantes en la ciencia.

Informalmente, un *modelo* de un cierto sistema complejo es una representación concreta, abstracta, conceptual, gráfica o formal, que permite estudiar, analizar, describir, explicar y razonar acerca del mismo. Así por ejemplo, un modelo de un objeto físico trata de imitar la apariencia o fisonomía de ese objeto como, por ejemplo, sería la maqueta de un barco o un dibujo de una cierta persona. Lo que importa de un modelo es, básicamente, la información que es capaz de proporcionar acerca del sistema que representa. Con un modelo se obtiene una imagen simplificada de la realidad que, por tanto, nunca podrá contener todos y cada uno de los detalles del sistema real que describe y representa. Es muy importante

saber distinguir entre componentes y factores que son esenciales, de otros que no lo son tanto y que, en consecuencia, no deberían ser incorporados al modelo. En resumen, los modelos son abstracciones o representaciones simplificadas de ciertos aspectos o sistemas del mundo real que tienen su interés y utilidad en determinados contextos.

Las leyes de Newton pueden ser consideradas como modelos en tanto que proporcionan información cualitativa acerca de la dinámica aproximada de ciertos sistemas. Como suele suceder con todos los modelos, dichas leyes contienen simplificaciones; en efecto, una velocidad constante presupone en muchos casos que no existe fricción, la caída de una manzana de un árbol admite que no hay resistencia del aire y que la masa es constante, etc. A pesar de estas simplificaciones, no hay duda que las leyes de Newton son muy útiles en numerosos cálculos mecánicos ya que las simplificaciones sólo introducen errores menores que, además, pueden ser evaluados, por ejemplo, en estos casos por medio de otras leyes físicas sobre fricción o resistencia del aire. En función de la precisión que se quiera obtener en nuestros cálculos, habría que analizar si las simplificaciones efectuadas son adecuadas o necesitan ser revisadas con otras consideraciones adicionales.

Existe un problema de "equilibrio" a la hora de diseñar un modelo ya que, por una parte, éste debe incorporar únicamente las características del sistema que sean esenciales o relevantes en el contexto del problema científico que trata de ser resuelto; es decir, deben ser lo suficientemente simples con el fin de entenderlos y manejarlos de la mejor y más eficiente manera posible. Pero, por otra parte, deben ser lo suficientemente ricos a fin de proporcionar comportamientos que puedan ser sorprendentes, interesantes, útiles y significativos. Así por ejemplo, los mapas geográficos son modelos de ciertas realidades y el diseño de los mismos estará en función de su utilidad para propósitos que pueden ser muy diversos como, por ejemplo, mapas para aeronaves, coches, trenes, barcos, lugares arqueológicos, genoma de organismos vivos, etc.

El uso de modelos es intrínseco a cualquier actividad científica y es básico a la hora de abordar ciertos problemas de la vida real en donde, además, como es bien conocido existen muchas interacciones entre otros factores que operan de manera simultánea. La construcción de modelos nos puede ayudar a describir esas interacciones, a relacionar las experiencias u observaciones previas con otras actuales y a analizar qué podría suceder en el futuro bajo determinadas circunstancias o escenarios iniciales.

Como ya hemos comentado, los científicos usan regularmente abstracciones o simplificaciones de la realidad tales como diagramas, grafos, leyes, relaciones, etc., con el fin de tratar de entender mejor ciertos aspectos concretos que estudian, analizan y examinan. El desarrollo de modelos suele requerir una aproximación multidisciplinar ya que en el propio problema que se estudia pueden incidir relaciones muy variadas entre diferentes disciplinas. Los médicos, químicos, biológos, ecólogos y economistas, entre otros, han estado siempre familiarizados con unos mecanismos de modelización, generalmente informales, estrechamente relacionados con los experimentos que realizaban en el laboratorio o sobre el

terreno. En este contexto, las Matemáticas y la Informática han sido utilizadas por dichos colectivos simplemente como herramientas auxiliares para el mejor desarrollo cuantitativo de esos experimentos. Sin embargo, durante las últimas décadas se ha producido un extraordinario avance en las técnicas usadas en la experimentación así como en la extracción y procesamiento de la información, que ha propiciado la recopilación de una masiva cantidad de datos sobre los sistemas dinámicos analizados; por ejemplo, en la búsqueda de las moléculas involucradas en ciertas funciones y en el análisis de su estructura como fue el caso de la insulina, en la secuenciación del genoma de distintas especies, incluida la humana, y la búsqueda de genes, en el análisis de factores en ecosistemas que inciden en la evolución de especies en peligro de extinción o de especies exóticas invasoras, etc. Así se llegó a la conclusión que el avance en las técnicas de laboratorio o de campo, unido a los distintos avances tecnológicos, únicamente habían proporcionado algo así como la *partitura* que describía la complejidad de los sistemas dinámicos que se analizaban, sin darnos información directa acerca de cómo interpretar la música o de cómo componer nuestras propias melodías. Entonces surge el problema de cómo manejar esa ingente cantidad de datos a fin de extraer, por una parte, información cualitativa que puede ser deducida a partir de los datos y, por otra, realizar predicciones acerca de los comportamientos posibles de los sistemas a lo largo del tiempo ante difererentes escenarios. Los modelos proporcionan, en general, mucha más información acerca del problema objeto de estudio que un simple análisis estadístico de observaciones ya que tienen la ventaja de poder incorporar gran parte del conocimiento teórico sobre el tema.

Los logros alcanzados durante finales del siglo pasado, tanto en Biología celular y molecular como en Ecología y dinámica de poblaciones, en general, en Economía así como, por supuesto, en Ciencias de la Computación tanto en su vertiente teórica como práctica, han propiciado la convergencia de dichas disciplinas a través del uso de modelos formales, con el objetivo de conseguir progresos científicos significativos en la ciencia.

4. 1 Modelos formales

Un *modelo formal* es una abstracción de un aspecto concreto del mundo real dentro de un marco matemático que trata de resaltar algunos aspectos relevantes del sistema objeto de estudio a través del uso de teorías formales. Así pues, un modelo formal es una especie de traducción simplificada de una parte de la realidad a un nuevo sistema expresado en términos específicamente matemáticos; es decir, en el contexto de una teoría formal.

El proceso de modelización formal consta de un conjunto de etapas o fases semiformales que nos guían en la tarea de diseño, en la descripción en un lenguaje formal, así como en su implementación para la evaluación y el análisis. Dicho proceso se puede estructurar como sigue:

- *Identificación y delimitación del fenómeno objeto de estudio*

  El desarrollo de modelos formales es un proceso arduo en donde, con frecuencia, habrá que reconsiderar los supuestos, las simplificaciones, etc. realizadas en su diseño inicial. El primer paso consiste en establecer la parte específica del sistema que se quiere modelizar detallando las componentes más relevantes, los objetivos concretos que se pretenden alcanzar, las cuestiones particulares que se trata de responder, así como los procesos de validación y análisis que se van a considerar.

- *Definición y formulación formal*

  Seguidamente hay que especificar o traducir en el contexto de una teoría formal, la descripción informal de las componentes del sistema que se estudia, los objetivos y las cuestiones a tratar.

- *Implementación o simulación computacional*.

  A continuación hay que detallar la forma de implementar o, en su caso, aproximar numéricamente o simular el modelo usando ordenadores electrónicos, a fin de poder manejarlo y extraer información fiable. En este punto es fundamental la corrección de los algoritmos usados para capturar la semántica del modelo considerado debido, básicamente, a la gran cantidad de parámetros, variables y estructuras involucradas en los elementos fundamentales (por ejemplo, funciones celulares, redes de genes, rutas señalizadoras de proteínas, interacciones entre individuos de poblaciones, etc.) de los sistemas dinámicos que se estudian. Esta tarea entra dentro de la Ingeniería del software.

- *Validación y calibración*:

  En el estudio y análisis de sistemas complejos es posible que determinados parámetros o bien no se hayan podido calcular experimentalmente o bien se han obtenido dentro de un rango de posibles valores. Por ello, antes de realizar simulaciones habrá que proceder a *calibrar* nuestro modelo; es decir, habrá que usar un método de estimación de parámetros y realizar un análisis de sensitividad para determinar qué parámetros son realmente cruciales en el modelo. El proceso de calibración debe ser seguido de un proceso de validación contrastando los resultados de la simulación del modelo con datos fiables obtenidos experimentalmente. Esta etapa proporciona, obviamente, un proceso iterativo de *refinamiento*. Conviene advertir que el método de validación depende, en cierto sentido, de los objetivos del modelo y, por ello, habría que traducir, en cierta manera, esos objetivos en criterios de validación. La validación debe proporcionar el grado de fiabilidad o de incertidumbre del modelo.

- *Análisis y chequeo*.

  Una vez que se ha encontrado un conjunto de parámetros fiables del modelo y éste ha sido validado, podemos centrarnos en algunas cuestiones que se

formularon como objetivos cuando se inició el proceso de modelización. Existen diferentes métodos de análisis según los modelos formales diseñados que van desde una simple generación de simulaciones en un ordenador electrónico hasta complicados análisis estadísticos, pasando por el uso de técnicas de *model checking*.

En un modelo formal de un sistema complejo se pueden distinguir una serie de componentes o elementos básicos: *variables de estado*, *funciones de control* o *variables externas*, *expresiones matemáticas*, *parámetros* y *constantes universales*, si las hubiere.

1. Las variables de estado permiten describir completamente el sistema objeto de estudio en un instante determinado. La elección de estas variables es esencial en el diseño del modelo.

2. Las funciones de control o variables son expresiones "externas" que influyen sobre el estado del sistema. El modelo tratará de analizar cómo los cambios en los valores de esas variables o funciones pueden afectar al estado del sistema.

3. Las expresiones matemáticas (ecuaciones, reglas de reescritura, etc.) describen las relaciones entre las funciones de control y las variables de estado. Se usan para representar procesos físicos, químicos, biológicos y de comportamiento de poblaciones, en general.

4. Los parámetros son coeficientes que intervienen en la representación matemática de los procesos a fin de especificar algunos hechos relevantes de los elementos que intervienen en el modelo.

5. Las constantes universales, en su caso, que intervienen en el sistema analizado y que no dependen, propiamente, del mismo.

Parece claro que para evitar que un modelo sea innecesariamente complejo habría que restringir "tanto como se pueda" el número de componentes básicas.

Es bien conocido que en muchos sistemas complejos reales no es posible realizar experimentos con todo el sistema completo a fin de obtener información. No obstante, recurriendo a modelos de dichos sistemas será posible realizar modificaciones sobre las funciones de control y variables a fin de estudiar los cambios producidos en los mismos a través de una evaluación de los cambios en las variables de estado.

4. 2 Modelización computacional

Recordemos que un modelo de computación consiste, básicamente, en la definición formal del concepto de procedimiento mecánico. En este marco, la resolución de un problema abstracto tiene la ventaja de poder ser implementada en una

máquina y, por tanto, de ser ejecutada para valores que especifican problemas concretos.

Está ampliamente aceptado ([10]) que un buen modelo formal debe satisfacer, al menos, las cuatro siguientes propiedades: *relevancia*, *comprensibilidad*, *extensibilidad* y *tratablidad matemático–computacional*.

- Un modelo debe ser *relevante*, en el sentido de capturar las propiedades básicas o esenciales del fenómeno investigado de una manera unificada, tanto en su estructura como en su comportamiento a lo largo del tiempo.

- Un modelo debe facilitar una mejor *comprensibilidad* del sistema que se estudia; es decir, los formalismos abstractos usados en el modelo deberían corresponderse bien con los conceptos informales e ideas del sistema, de tal manera que pueda ser fácilmente interpretado por los expertos en el sistema objeto de estudio.

- Un modelo debe ser fácilmente *extensible* y *escalable* a otros niveles de organización y fácilmente modificable con el fin de incluir nuevo conocimiento o eliminar hipótesis falsas. En general, nuestro conocimiento del sistema es dinámico y casi constantemente se está generando nueva información sobre el mismo; por ello, el modelo debería ser fácilmente extensible, en el sentido de poder incorporar esa nueva información.

- Un modelo debe ser *tratable computacionalmente* en tanto en cuanto debe permitir su implementación en dispositivos electrónicos a fin de poder ejecutar simulaciones que permitan manejar el modelo y estudiar la dinámica del sistema en diferentes escenarios, a través de la manipulación de las condiciones experimentales en el modelo, sin necesidad de realizar experimentos complejos y costosos en el laboratorio o sobre el terreno, posibilitando, de esta manera, su análisis matemático y computacional.

El desarrollo de modelos en un marco computacional ofrece la ventaja de poder manejarlos usando máquinas o simuladores del propio marco. La habilidad de los modelos computacionales para manejar la complejidad hacen de ellos potentes instrumentos para explorar la naturaleza. Estos modelos son esenciales para entender más y mejor sistemas complejos en donde los cálculos directos podrían tardar años o, incluso, siglos. Son aplicables tanto a nivel micro (por ejemplo, a nivel de reacciones químicas moleculares, rutas señalizadoras de proteínas) como a nivel macro (por ejemplo, a nivel de ecosistemas, galaxias, etc.) y están posibilitando la aparición de importantes y útiles herramientas científicas. De la misma forma que la interacción entre teoría y experimentación es la fuerza que dinamiza la ciencia, se puede afirmar que, actualmente, la modelización computacional y la simulación están en el propio corazón del método científico que podríamos calificar como moderno. En este caso, como en tantos otros, la teoría y los experimentos prácticos van de la mano.

Uno de los objetivos fundamentales de cualquier modelo formal es su capacidad de *predicción*; es decir, la posibilidad de realizar conjeturas o hipótesis plausibles acerca del posible comportamiento del sistema modelizado que es objeto de investigación, ante diferentes escenarios que puedan ser considerados de interés por los expertos.

Existen diversas formas de enfocar el proceso de modelización formal de sistemas complejos. A continuación, enumeramos algunas aproximaciones.

1. En relación con el escalado del espacio, se puede distinguir entre enfoque *macroscópico*, *microscópico* y *mesoscópico*. En el primero de ellos, el sistema se observa como un todo, sus componentes se representan con poco detalle y no se proporcionan mecanismos de interacción entre ellas. En el enfoque microscópico, cada parte del sistema se representa con bastante detalle; por ejemplo, en el caso de fenómenos moleculares, se tiene en cuenta cada molécula y se especifica su posición y momento. Esta aproximación proporciona mucha más información que la anterior pero es intratable desde el punto de vista computacional en la mayoría de casos. Por último, el enfoque mesoscópico se centra en el número de componentes individuales que integran el sistema, sólo tiene en cuenta las partes del sistema que se consideran más relevantes, despreciando parámetros como la posición y el momento, en el caso de las moléculas. Esta aproximación es más tratable que la microscópica y, a la vez, es capaz de manejar y conservar información más relevante que la macroscópica.

2. De acuerdo con el tipo de análisis realizado, un modelo formal puede ser *cuantitativo* o *cualitativo*. El primero proporciona información y datos cuantitativos acerca del sistema que se estudia, mientras que el segundo proporciona información cualitativa acerca del sistema y de su dinámica. A su vez, según el tipo de datos cuantitativos generados y el carácter de la especificación del sistema, un modelo puede ser *discreto* o *continuo*. En el primero de ellos, las componentes del sistema modelizado son representadas mediante individuos o entidades discretas y los datos generados son, también, discretos. En un modelo *continuo*, las componentes del sistema son representadas a través de variables continuas y los datos generados son continuos.

3. Respecto de su dinámica o evolución, los modelos se clasifican en *determinista* y *no determinista*. En el primero de ellos, existe una única posible evolución del modelo a partir de unos parámetros o situación inicial, mientras que en el segundo existen muchas posibles evoluciones del modelo a partir de unos parámetros iniciales. En esa dinámica, el paso de una configuración a una configuración siguiente se realiza, por ejemplo, con una cierta aleatoriedad; por tanto, a la hora de obtener resultados mínimamente fiables a partir de un determinado escenario, es necesario realizar un número elevado de simulaciones y trabajar con medias, evaluando la desviación típica y el intervalo de confianza.

4. Finalmente, en relación con el origen de la información utilizada para su diseño, los modelos formales pueden ser *empíricos* o *heurísticos*. El primero de ellos es construido a partir de observaciones directas o resultados experimentales, mientras que los modelos heurísticos son diseñados a través de los mecanismos conocidos del sistema que se estudia.

Los modelos computacionales y sus eventuales predicciones pueden provocar ciertos recelos en muchos entornos de expertos. Desconfianza como la que encontró Galileo tras realizar las primeras observaciones astronómicas con un telescopio y publicar sus descubrimientos en los que se afirmaba la existencia de montañas lunares, las lunas de Júpiter, los anillos de Saturno, etc. Esas afirmaciones violaban las creencias aristotélicas acerca del cosmos, llegando los críticos a afirmar que esos descubrimientos eran, propiamente, artefactos derivados del modelo usado por Galileo, ilusiones ópticas creadas por el propio telescopio a imagen y semejanza de un caleidoscopio.

En ese contexto se podría afirmar que un modelo de computación viene a ser como un telescopio virtual que, en lugar de analizar la inmensidad del universo, se centra en alguna parte específica del mismo con el objetivo de conocerla hasta el más mínimo detalle, a fin de comprenderla íntegramente e, incluso, predecir su comportamiento.

BIBLIOGRAFÍA

[1] A. Aho, J. Hopcroft, J. Ullman. The design and analysis of computer algorithms. Addison Wesley (1974).

[2] J.L. Balcázar. Programación metódica. McGraw Hill (1993).

[3] S. Cook. The P versus NP Problem. Manuscript prepared for the Clay Mathematics Institute for the Millennium Prize Problems, November 2000. Versión online: www.claymath.org/millennium/

[4] C.A.R Hoare. *A axiomatic basis for computer programming* Comm. of ACM 12 (1969) 576–583.

[5] U. Manber. Introduction to algorithms. A creative approach. Addison Wesley (1989).

[6] J. Pavlus. ¿Qué significa "P vs NP" para el resto de nosotros? Versión online: www.technologyreview.es/read_article.aspx?id=36497

[7] M.J. Pérez Jiménez. Modelos de Computación Natural. En A. Nepomuceno, F.J. Salguero y F. Soler (eds.) *Bases biológicas, lingüísticas, lógicas y computacionales para la conceptualización de la mente*. Mergablum, Edición y Comunicación, Sevilla, 2004, pp. 199–245.

[8] M.J. Pérez Jiménez, F. Sancho Caparrini. *Computación celular con membranas: Un modelo no convencional.* Editorial Kronos, Sevilla, 2002.

[9] M.J. Pérez Jiménez, F. Sancho Caparrini. *Máquinas moleculares basadas en ADN.* Secretariado de Publicaciones, Universidad de Sevilla. Colección de divulgación científica, número 2, 2003.

[10] A. Regev, E. Shapiro. The π-calculus as an abstraction for biomolecular systems. *Modelling in Molecular Biology*,2004, pp. 219–266.