Programación y matemática

Los objetos matemáticos se describen usando un lenguaje al que llamamos lenguaje matemático. Como este lenguaje tiene pautas claras que indican cuáles descripciones tienen sentido y cuales no, podemos llamarlo lenguaje formal o formalismo: identificamos una expresión con sentido por su forma. Es así que podemos reconocer un objeto matemático en la expresión "(2 + 1) * 5"; y un sinsentido en la expresión "/* 111 5 -+ 34".

Las expresiones del lenguaje matemático se evalúan utilizando reglas de transformación que describen cómo al ser aplicadas a expresiones (una o más de una) se obtiene otra expresión “más simple” que la original. “Más simple” significa que el número de pasos de reducción para llegar a obtener una expresión para la cual no hay regla a aplicar (o sea, no puede ser reducida), es menor. La expresión que no puede ser reducida se denomina *forma canónica* de la expresión (en realidad, de todas las obtenidas en los pasos de reducción). El término “reducir” proviene del hecho de que el número de pasos hasta la forma canónica es menor y el término “evaluar” proviene de que la forma canónica se denomina también el “valor” de la expresión.

Ejemplo: (2 + 1) * 5 puede reducirse a la expresión 3 * 5 y ésta a 15. Decimos que 15 *está en* forma canónica o *es* la forma canónica de la expresión (2 + 1) * 5 o que (2 + 1) * 5 *tiene* forma canónica 15.
Las reglas de transformación que se aplican son la funciones + y *.
3 * 5 es “más simple” que (2+1)*5 porque el número de pasos de reducción hasta la forma canónica es menor (desde (2+1)*5 a 15 hay dos pasos, mientras que de 3 *5 hay un solo paso).

Las reglas pueden ser aplicadas siguiendo distintas *estrategias*, pero por cualquiera de ellas *se llega a una misma forma canónica* (si la misma existe). Por ejemplo, la expresión (2 + 1) * 5 puede reducirse a la expresión 3 * 5 y ésta a 15, o a la expresión 10 + 5 y luego a 15.
Algunas expresiones que no tienen forma canónica, por ejemplo 2/0, porque la regla involucrada (/) no está definida para denominador 0.

Tipos de expresiones
Los elementos de un conjunto pueden ser las letras del abecedario. Podemos definir “palabras” como los subconjuntos de ese conjunto de letras.
Y podemos definir relaciones y funciones sobre conjuntos de letras o de

palabras. Por ejemplo, podemos definir una operación que dadas dos palabras, nos devuelva otra formada por las letras de la primera seguidas por las de la segunda.
Cuando buscamos una palabra en un diccionario (de papel), utilizamos una relación de orden entre las palabras. Sabemos por ejemplo, que “gato” está después de “casa” y antes que “sol” y esa relación determina el algoritmo de búsqueda que empleamos para buscarla.

¿Cree Ud. que puede definirse en matemática ese algoritmo de búsqueda?
Si, se puede y este problema es uno de los tipos de problemas que se tratan en matemática discreta. También puede *programarse* es decir, representarse en un lenguaje de programación.

En matemática discreta y en programación, a las letras (y símbolos en general, por ejemplo los del teclado) se les llama *caracteres* y a las palabras se les llama *cadenas de caracteres* o *strings* y suelen denotarse entre comillas, por ejemplo “Sylvia” es la cadena formada por los caracteres *“S”*, *“y”*, *“l”*, *“v”*, *“i”*, *“a”*.

Expresiones booleanas
Uno de los conjuntos con los que trabajamos es el conjunto llamado **Bool**, formado por los valores verdadero y falso, o *true* y *false* en inglés que es lo que usan los lenguajes de programación (y nosotros por comodidad).
O sea que Bool = {true, false}.
Las expresiones que evalúan o reducen a true o a false se denominan expresiones booleanas de la misma forma que las expresiones que evalúan a un valor entero, se denominan expresiones enteras.
Todo lo que hemos dicho vale para expresiones booleanas, por ejemplo, 2*(3+1) > 4 reduce a 2*4 > 4 reduce a 8 > 4 reduce a *true*. El valor *true* es la forma canónica de las expresiones y es una expresión canónica (o sea un valor). La regla que se aplica en este caso es la denotada por >.
También está definida la la relación de orden lexicográfico entre palabras (la que usa el diccionario). Por ejemplo, “abc” < “abd” evalúa a true.

Lenguajes de programación
Un lenguaje de programación es un formalismo en el cual se representan objetos por medio de expresiones del lenguaje, que se evalúan utilizando reglas de transformación sobre las expresiones. Las reglas ***también*** se representan en el lenguaje, de modo que puedan ser aplicadas por un computador y la evaluación de las expresiones se hace entonces

automáticamente. Las reglas son ***algoritmos y que también pueden ser representados en el lenguaje matemático*.**

Existen lenguajes de programación en los cuales pueden representarse las pruebas de propiedades y ejecutarse (semi) automáticamente (el "semi" es porque hay partes de las pruebas que son indecidibles). Estos lenguajes de programación se llaman "asistentes de pruebas".

Citamos dos opiniones que definen la relación entre matemática y computación bajo una perspectiva que compartimos.

La primera está extraída de la traducción castellana del articulo "La revolución en matemáticas" de Marshall Stone, aparecido en 1978 en "La enseñanza de las matemáticas modernas" de Jean Piaget y otros autores. Refiriéndose a la introducción de computadores, a los que llama "potentes auxiliares" en el quehacer matemático, dice así:

> Después de la introducción de estos potentes auxiliares los matemáticos ya no tienen que preocuparse de reducir sus soluciones a una forma sencilla de calcular a mano o con calculadoras corrientes; su misión ha pasado a ser la de traducir estos cálculos a programas de cuya ejecución se encargarán computadoras electrónicas de gran velocidad, lo que ha dado lugar a la aparición de una nueva rama de las matemáticas dedicada a la teoría y práctica de la confección de programas.

Unos años mas tarde, en "On the cruelty of really teaching computer science", Dijkstra define ciencia de la computación con estas palabras:

> It is -and will always be- concerned with the interplay between mechanized and human symbol manipulation, usually referred to as "computing" and "programming" respectively.[1]

Y agrega:

> In the map of academic disciplines, computer science has to be located in the direction of formal mathematics and applied logic.[2]

[1] Concierne a la relación entre la manipulación simbólica mecanizada y la humana, referidas como computar y programar respectivamente.
[2] En el mapa de las disciplinas académicas, la ciencia de la computación debe ser ubicada entre la matemática formal y la lógica aplicada.

El lenguaje matemático es manejado por humanos, mientras que un lenguaje de programación es manejado por un autómata, un computador. Esto plantea algunas consideraciones, la primera de las cuales es que ambos lenguajes han sido diseñados por humanos y por lo tanto pueden contener inexactitudes o ambigüedades. Ahora bien, el lenguaje matemático *es usado* por humanos, quienes pueden advertir las inexactitudes y/o ambigüedades, de una manera similar a lo que sucede con el lenguaje natural. Sin embargo, un lenguaje de programación *es usado* por un computador, que ejecutará o realizará las instrucciones sin detenerse a reflexionar ni en la pertinencia ni en el significado de las mismas, actuando siempre como si estuvieran completas y fueran exactas. Por ello, el programar exige rigurosidad en el uso del lenguaje, es más, cumplir con este requisito, es una de las mayores dificultades en la construcción de software y una de las fuentes de errores del mismo. Hay lenguajes de programación que ayudan a superar estas inexactitudes mejor que otros. Por otro lado, es importante educar a los estudiantes en el uso riguroso del lenguaje desde cursos iniciales. Las prácticas que no se adquieren en el sistema educativo son más difíciles de adquirir fuera del mismo (o no se adquieren nunca). Por otro lado la rigurosidad exigida en programación es beneficiosa para todos los estudiantes, no sólo para aquellos que seguirán estudios en computación, como lo señalan autores como Dowek y J. Wing.