
UNIVERSIDAD NACIONAL DE SALTA
FACULTAD DE CIENCIAS EXACTAS
DEPARTAMENTO DE MATEMÁTICA



MATEMÁTICA DISCRETA

APUNTES DE CÁTEDRA
AGOSTO DE 2008

THOMAS N. HIBBARD

Índice general

Capítulo 1. INTRODUCCIÓN	3
1. Algoritmos	3
Capítulo 2. LOS NÚMEROS NATURALES	9
1. Axiomas	9
2. División	12
3. Primos	14
4. Máximo común divisor	17
5. El Teorema Fundamental de la Aritmética	18
6. Aritmética Modular	21
7. Grupos	25
8. Cuerpos	28
9. Más sobre primalidad y factorización	32
10. Criptografía	34
Capítulo 3. TEORÍA DE GRAFOS	39
1. Grafos	39
2. Caminos y conectividad	40
3. Los puentes de Königsburg y ciclos de Euler	43
4. Estructuras de datos para grafos	46
5. Camino mínimo	50
6. Árboles	52
7. Redes de Transporte	54
Capítulo 4. AUTÓMATAS	63
1. Autómatas finitos y lenguajes regulares	64
2. Máquinas de Turing	69
Capítulo 5. GRAMÁTICAS	75
1. Lenguajes libres de contexto	76
2. Autómatas de Pila	80
Capítulo 6. FUNCIONES GENERADORAS Y ECUACIONES DE DIFERENCIAS	87
1. Introducción	87
2. Funciones generadoras racionales	89
3. Ecuaciones de diferencia	91
4. Una sola ecuación de diferencia	93
5. Sobre la aritmética sobre series infinitas	101
Capítulo 7. PROBABILIDAD DISCRETA	105
1. Introducción	105
2. Autómatas estocásticos	112
Capítulo 8. CADENAS DE MARKOV	119
Capítulo 9. LA INCOMPLETITUD DE LA ARITMÉTICA	133
1. Introducción	133

2.	Demostraciones	134
3.	El plan general	137
4.	Cálculo de predicados de primer orden	138
5.	El Teorema de Gödel	143

INTRODUCCIÓN

Si bien la matemática discreta no es nueva en el siglo 20 – la teoría de grafos fue fundada por Euler, y Pitágoras fue un pionero en la teoría de números – no fue muy respetada antes: se consideraba más bien una diversión, comparada con el trabajo serio del matemático en la matemática continua. Euclides tuvo que disfrazar su teoría de números como geometría.

Con el advenimiento de la computación electrónica, algunos problemas discretos empezaban a verse como urgentes. Los matemáticos puros seguían buscando sus generalizaciones: calcular algo concreto estaba por debajo de su dignidad, pero la gente cuyo trabajo sí era el de calcular algo concreto con las nuevas computadoras, en forma que salga bien confiable y en tiempo aceptable, se enfrentaba con nuevos problemas. Ahora los problemas de esta clase se conocen como problemas del diseño de algoritmos. Y siendo la computadora digital un aparato de naturaleza discreta, que analizado cuidadosamente se ve que trabaja sólo con números enteros, la matemática involucrada era discreta.

Es por eso que cada libro de matemática discreta tiene un capítulo dedicado a la noción de algoritmo.

No basta con algunas observaciones generales sobre esa noción, sino que, para que el alumno capte su verdadera significancia, es preciso que trabaje con algoritmos concretos, y además que los ponga en marcha en una computadora. Hay que seleccionar unos problemas que conducen a esto, y allí cada profesor va a tener su propio gusto. Y así también con respecto a los temas: casi siempre se elige la teoría de números y la teoría de grafos. Aquí hemos elegido también la teoría de lenguajes, incluyendo autómatas finitos y gramáticas, funciones generadoras, y, finalmente, la formalización de la noción de algoritmo, necesaria para el teorema de incompletitud de Gödel.

1. Algoritmos

No es que la matemática discreta no existía antes de la computadora, pero no existía el término. Euclides hacía matemática discreta, pero la tuvo que disfrazar como geometría para conseguir la atención de sus colegas. Leonardo Euler empezó la teoría de grafos en el siglo 18. John von Neumann empezó la teoría de los juegos al mismo tiempo que se estaban ideando las primeras computadoras electrónicas, cuando la división automática se hacía en segundos –en lugar de microsegundos– y estaba acompañada por una desconcertante serie de porrazos. Pero la matemática discreta no se consideraba parte de la corriente principal de la matemática, sino más bien una curiosa diversión.

Pero con la computadora, de golpe la inversión de una matriz de 50 por 50 no era trabajo de una semana sino de unos segundos, la simulación de una batalla se podía hacer en tiempo real, se podía modelar la economía de una nación por un año con bastante verosimilitud, y, en poco tiempo más, muchos problemas se abrieron a un estudio que antes parecía imposible.

Pero quizás la novedad principal que llegó con la computadora fue la realización concreta de algoritmos. Ya un algoritmo no era sólo una receta o una serie de instrucciones para hacer algo; residía dentro de la computadora en la forma de una serie de números, o si quiere, un solo número, porque la concatenación de una serie de números se puede leer como un solo número. Cambiando un solo dígito de ese número casi seguro iba a causar el fracaso del algoritmo. Se podía decir que los algoritmos habían pasado de la metamatemática a la matemática misma: habían llegado a ser objetos matemáticos.

Cabe mencionar que, en una de las muchas instancias en que la teoría se adelanta de la tecnología, en la década anterior, cuando Gödel necesitaba formalizar su requerimiento de la lógica que las demostraciones sean efectivamente computables, dió el primer intento de formalizar la noción de un algoritmo. Resultó después que no era completamente satisfactoria, aún que toda la lógica vista hasta este momento tienen demostraciones computables en ese sentido. Pero abrió por primera vez formalmente de qué era, formalmente, una función computable, contestado por Turing y Church con su famosa Tesis.

Es así que los algoritmos son fundamentales en la matemática discreta, y nuestra primera tarea es establecer una notación para describirlos.

Cada algoritmo está dirigido a calcular una función. Hay unos datos de entrada, los *argumentos*, y unos datos de salida (decimos “unos” pero pueden ser miles) que constituyen el valor de la función para los argumentos dados.

Aquí vamos a definir una función como una *expresión*, y vamos a ir definiendo el término *expresión* poco a poco.

Una expresión puede ser un número, y por ahora por *número* vamos a entender número natural. Por conveniencia incluimos 0 como número natural, dejando a un lado el debate de si 0 es natural en el sentido del lenguaje cotidiano. Cuando queremos enteros, ponemos un signo con un número natural, y cuando queremos un número real ponemos un par ordenado (a, b) de enteros para significar $a \times 10^b$ (o, a veces, $a \times B^b$ donde B , la base, es un número natural mayor que 1.)

Una expresión puede consistir en la invocación de una función ya conocida.

Ejemplo: $5 + 7$, 13×8 , $15 - 7$ (pero no $7 - 15$ mientras trabajamos sólo con naturales), 7^2 . Aquí las funciones son $+$, \times , $-$ y exponenciación respectivamente. Funciones fundamentales como éstas, a menudo se expresan con un sintaxis especial, *infixa* en el caso de $+$, \times y $-$.

Una expresión puede dar un valor lógico, “verdadero” o “falso”. Por ejemplo $5 < 7$ es verdadero y $7 < 5$ es falso.

Una noción clave es la de la *expresión condicional*. Lo vamos a escribir en la forma

si (expresión 1) expresión 2 **sino** expresión 3

donde expresión 1 es una expresión que da un valor lógico. El valor de tal expresión es el valor de la expresión 2 si el valor de la expresión 1 es verdadero, y es el de la expresión 3 si aquel es falso. Nótese que se evalúa la expresión 1 primero y después se evalúa la de las otras dos que corresponde. La otra opción, la de evaluar ambas de expresiones 2 y 3, y elegir el que corresponde al valor de la expresión 1, nos daría problema con una expresión como “si($y = 0$) 0 sino x/y ”, que daría “indefinido” siempre cuando $y=0$ por que $x/0$ es indefinido.

Con esto podemos definir funciones muy útiles, como por ejemplo la siguiente:

$$\text{máx}(x, y) = \text{si } (x > y) \ x \text{ sino } y$$

Note aquí que los nombres de las funciones que definimos pueden tener varias letras, en este caso “max”. Ahora teniendo este “max” como conocido, podemos definir

$$\text{máx}(x, y, z) = \text{máx}(x, \text{máx}(y, z)),$$

lo cual ilustra otra cosa: que podemos dar el mismo nombre a dos funciones distintas mientras tienen distinto número de argumentos. Y entonces podemos definir

$$\text{máx}(x, y, z, w) = \text{máx}(x, \text{máx}(y, z, w)),$$

o si preferimos,

$$\text{máx}(x, y, z, w) = \text{máx}(\text{máx}(x, y), \text{máx}(z, w)).$$

Cuando escribimos así la definición de una función, digamos f , es permitido invocar a f en la misma expresión que usamos para definir f . Por ejemplo, aunque vamos a suponer que todo el mundo sabe qué quiere decir x^y cuando x e y son naturales no ambos 0, podemos eliminar toda duda escribiendo

$$x^n = \text{si } (n = 0) \ 1 \text{ sino } x \times x^{n-1}.$$

(Esto definiría $0^0 = 1$ pero incluimos en la definición que no vale cuando los dos argumentos son 0.) Pero este es mal algoritmo. Para 2^{5000} por ejemplo hace 4999 multiplicaciones, pero se puede hacer como $((2^{10})^{10})^5$ con solo 22.

Ejercicio 1. (1) Realice x^n en Maple como procedimiento $\text{pot}(x,n)$, definiendo $\text{pot} := \text{proc}(x,n)$ if $n = 0$ then 1 else x^* . . . fi end;

Ejercicio 2. (1) Explique cómo calcular 2^{5000} con menos de 50 multiplicaciones. Sugerencia: $2^{5000} = (2^{1000})^5$, que multiplica 1005 veces en lugar de 5. Siga aplicando esta idea. Realice este cálculo como una expresión en Maple, empleando el procedimiento pot del ejercicio 1.

Ejercicio 3. (1) Explique por qué el mínimo número de multiplicaciones para engendrar x^n es igual al mínimo número de adiciones para engendrar n , empezando con 1.

Ejercicio 4. (3) Explique cómo calcular x^n con no más que $2(\lceil \log_2 n \rceil - 1)$ multiplicaciones. (No pedimos que lo explique en la notación algorítmica compacta porque hay que engendrar una sucesión y no hemos hablado todavía de ese tipo de datos – explique en buen castellano no más.)

Ejercicio 5. (3) Demuestre, examinando valores chicos de n , que el algoritmo del ejercicio 4 es óptimo para algunos n pero no lo es para todos. (Alternativamente, si su algoritmo es óptimo para todo n , mándelo a publicar inmediatamente).

Cuando la expresión que define a f invoca a f se dice que es una definición *recursiva*. Las definiciones recursivas son fundamentales; sin ellas no se puede definir nada de interés. Debe notarse que no hablamos de una sucesión de instrucciones, donde instrucción 10 puede ser “vuelva a la instrucción 5”, pero si estuviéramos hablando así, el volver a la instrucción 5 es una invocación recursiva de la función que se está evaluando cuando estamos en la instrucción 5.

También puede ocurrir que en la definición de f invocamos una g que invoca a f . También es una función recursiva. O puede ser que f invoque a g que invoca a h que invoca a f , etcétera. Claro que no hay garantía de que tales definiciones siempre vayan a tener sentido, por ejemplo $f(x) = g(x) + 1$ y $g(x) = f(x) - 1$. ¿Qué idiota trataría de evaluar eso? ¡Ese idiota es la computadora!

Examinemos esta función interesante:

$$\text{fibonacci}(n) = \text{si}(n < 2) \text{ } n \text{ sino } f(n-1) + f(n-2)$$

Los primeros términos de esta *sucesión de Fibonacci* son 0,1,1,2,3,5,8,13,21,34. Ud. puede continuar la sucesión, obteniendo el próximo término simplemente sumando los dos últimos que tiene.

Ejercicio 6. Realice la función *fibonacci* como procedimiento en Maple, escribiendo

$$\text{fibonacci} := \text{proc}(n) \text{ if } n < 2 \text{ then } n \text{ else fibonacci}(n-1) + \text{fibonacci}(n-2) \text{ ;}$$

Pruébalo invocando varios $\text{fibonacci}(n)$ con $n < 20$. Ahora haga $\text{trace}(\text{fibonacci})$ e invocar $\text{fibonacci}(8)$ (no más que 8 porque hay mucha salida). La palabra de inglés *trace* quiere decir rastrear. Cause el rastreo de toda la actividad de la función *fibonacci* cada vez que se invoca.

Pero note que esa idiota de computadora va a seguir esta definición a la letra, como hemos visto en el ejercicio 6, en lugar de simplemente agregar otro término de la sucesión sumando los dos últimos. Por ejemplo, para $f(4)$ va a calcular primero $f(3) = f(2) + f(1) = f(1) + f(0) + f(1)$, y luego $f(2) = f(1) + f(0)$ otra vez, y así hace $f(1) + f(0) + f(1) + f(1) + f(0)$, cinco invocaciones de f . Sea $t(n)$ el tiempo que demora la computadora en calcular $\text{fibonacci}(n)$, y supongamos que $t(1) = t(2) = 1$ microsegundo. Entonces para $n > 1$, $t(n) \geq t(n-1) + t(n-2)$ y resulta que $t(n) \geq \text{fibonacci}(n)$. Entonces para calcular $\text{fibonacci}(50)$ la computadora demoraría por lo menos 3 horas y media. Pero podríamos hacerlo a mano, aún demorando un promedio de 30 segundos en cada suma, en 25 minutos, y yo acabo de calcular con Maple $\text{fibonacci}(50) = 12586269025$ en una pequeña fracción de segundo.

Lo que pasa es que la definición, aunque funcionalmente correcta, es ineficiente. Es un mal algoritmo. Mejor es éste:

$$\begin{aligned} \text{fibonacci}(n) &= \text{si}(n < 2) \text{ } n \text{ sino } \text{fibonacci}(0, 1, n - 2) \\ &\quad \text{donde} \\ \text{fibonacci}(x, y, n) &= \text{si}(n = 0) \text{ } x + y \text{ sino } \text{fibonacci}(y, x + y, n - 1) \end{aligned}$$

Ejercicio 7. (4) Demuestre que el último algoritmo es correcto demostrando lo siguiente:

Si $f(n) = \text{si}(n = 0) \text{ } a \text{ sino } \text{si}(n = 1) \text{ } b \text{ sino } f(n - 1) + f(n - 2)$
y si $g(x, y, n) = \text{si}(n = 0) \text{ } x + y \text{ sino } g(y, x + y, n - 1)$
entonces $g(a, b, n) = f(n + 2)$.

Sugerencia: Demuestre por inducción en k que $g(a, b, n) = g(f(k), f(k + 1), n - k)$ para todo $k \leq n$.

Ejercicio 8. Realice este nuevo algoritmo para fibonacci en Maple. Haga `trace(fibonacci)` nuevamente (al redefinir “fibonacci” se cancela su `trace`) e invoque `fibonacci(8)`. Haga `untrace(fibonacci)`; luego invoque `fibonacci(50)`.

El próximo ejemplo será el de calcular una raíz cuadrada. Algoritmo:

$$\begin{aligned} \text{raíz}(a) &= \text{raíz}(a, 3) \\ &\quad \text{donde} \\ \text{raíz}(a, x) &= \text{si}(x = a/x) \text{ } x \text{ sino } \text{raíz}(a, \frac{x + \frac{a}{x}}{2}). \end{aligned}$$

Medio malo este algoritmo porque nunca para, excepto en casos especiales como $a = 9$. Pero pruébelo viendo los sucesivos valores de x . Por ejemplo, para $a = 1000000$ son 166668.1667, 83337.08332, 41674.54139, 20849.26843, 10448.61588, 5272.161168, 2730.918352, 1548.547770, 1097.157047, 1004.301796, 1000.009213. Claramente x tiende al límite 1000, la raíz cuadrada de 1000000. Bueno, cuando se trata de la raíz cuadrada no podemos pretender más que llegar cerca. Entonces para tener un algoritmo que termina, terminamos cuando x y a/x están suficientemente cerca; es decir, elegimos un ϵ y terminamos cuando $|x - a/x| < \epsilon$:

$$\begin{aligned} \text{raíz}(a, \epsilon) &= \text{raíz}(a, 3, \epsilon) \\ &\quad \text{donde} \\ \text{raíz}(a, x, \epsilon) &= \text{si}(|x - \frac{a}{x}| < \epsilon) \text{ } x \text{ sino } \text{raíz}(a, \frac{x + \frac{a}{x}}{2}, \epsilon) \end{aligned}$$

Ahora cuando hacemos esto a mano no vamos a calcular a/x dos veces, sino vamos a guardarlo para copiar la segunda vez que lo usamos. Pero si damos esto a algún idiota (por ejemplo, la computadora) para calcular, capaz que no haga eso. Le tenemos que decir explícitamente que haga eso, escribiendo la subfunción $\text{raíz}(a, x, \epsilon)$ así:

$$\text{raíz}(a, x, \epsilon) = c := a/x; \text{si}(|x - c| < \epsilon) \text{ } x \text{ sino } \text{raíz}(a, \frac{x + c}{2}, \epsilon)$$

“ $c := a/x$ ” se llama *asignación*. Lo podemos considerar una abreviatura, “para el resto del cálculo de esta función c significa el valor que tiene a/x en este momento.” También se lo puede considerar una memoria, pero una local a la función; la asignación carece de validez cuando termine la función.

Ejercicio 9. Haga en Maple:

`a := 2.; x := 3.;` (no se olvide del punto decimal, que indica que se quiere aritmética real – si no todos los valores se dan como racionales). Luego invoque repetidamente la siguiente línea (para invocar de nuevo haga `↑` hasta que el cursor llegue de nuevo a la línea y haga `enter`):

`c := a/x; x := (x+c)/2.;`
terminando cuando $|c - x| < 0,000001$.

Ejercicio 10. Haga en Maple la función $raíz(a, \epsilon)$, usando la segunda versión de la subfunción $raíz(a, x, \epsilon)$. Se la hace subfunción poniendo su definición ($raíz1:=proc(a,x,epsilon)$ etc., llamándole $raíz1$ porque Maple, lamentablemente, no permite sobrecargar nombres) dentro de la definición de la función de $raíz(a, \epsilon)$. Invocar la función para probar. Luego haga $trace(raíz)$ e invocarla de nuevo.

Las asignaciones casi abren la puerta para la programación imperativa. Más adelante vamos a abrir esa puerta completamente. Pero por ahora seguimos con el punto de vista puramente funcional.

Para dos números naturales x, y , con $y \neq 0$, el *cociente* de x dividido y es el número de veces que se puede restar y de x y tener un saldo natural:

$$cociente(x, y) = \text{si}(x < y) \text{ 0 sino } 1 + cociente(x - y, y).$$

Otro algoritmo correcto pero malo (desde el punto de vista de la eficiencia). Y aquí otro igualmente correcto y malo:

$$resto(x, y) = \text{si}(x < y) \text{ } x \text{ sino } resto(x - y, y).$$

Vamos a ver mejores algoritmos para estas funciones, pero sirven para ilustrar un par de cosas más de la notación algorítmica. Notando que *cociente* y *resto* hacen la misma prueba, $x < y$, y hacen llamadas recursivas con los mismos argumentos, vemos que convendría combinar los dos algoritmos en uno, que llamaremos *cr*. $cr(x, y)$ nos va a entregar un par ordenado (c, r) de números naturales tales que $x = cy + r$ con $r < y$, es decir que c y r son el cociente y resto respectivamente de x dividido y . Las funciones no están restringidas a entregar sólo números o valores lógicos, sino pueden entregar estructuras, desde pares ordenados a sucesiones, matrices, conjuntos, etc. Una definición de $cr(x, y)$ para x, y naturales y $y \neq 0$:

```
cr(x, y) =
.  si(x < y) (0, x)
.  sino
.  .  (a, b) := cr(x - y, y)
.  .  (a + 1, b)
```

La cadena “ $(a, b) := (c, r)$ ” (siendo $(c, r) = cr(x - y, y)$) significa las dos asignaciones $a := c$ y $b := r$.

Ejercicio 11. (2) Realice $cr(x, y)$ en Maple. Pruébalo con ejemplos con $y < 6000$ primero. Haga $trace(cr)$ y pruebe con $y < 50$. Haga $untrace(cr)$. Haga $cr(40000, 7)$. ¿Cuál es el número máximo x tal que se puede hacer $cr(x, 7)$?

Ejercicio 12. (3) Realice en Maple el siguiente algoritmo:

```
cr2(a, b) = si(a < b) (0, a)
.  sino
.  .  (c, r) := cr2(a, 2b)
.  .  si(r < b) (2c, r) sino (2c + 1, r - b)
```

¿Qué calcula $cr2$?

Ejercicio 13. (5) ¿Porqué? Sugerencia: Pruebe primero que siempre termina. Pruebe correctitud por inducción sobre el número de llamadas recursivas.

Volvemos al algoritmo *raíz*, convirtiéndolo en un algoritmo de la matemática discreta: calculemos la parte entera de \sqrt{a} , que escribimos $\lfloor \sqrt{a} \rfloor$, empleando sólo números enteros (ahora el argumento a , por supuesto, tiene que ser natural):

```
raíz2(a) := raíz2(a, 3)
donde
.  raíz2(a, x) =
.  .  c := cociente(a, x)
.  .  si(|x - c| ≤ 1) mín(x, c) sino raíz2(a, cociente(x + c, 2))
```

Por ejemplo, para el cálculo de $raíz2(3000000, 3)$ que nos permite averiguar $\sqrt{3}$ hasta 3 decimales, los pares ordenados que se encuentran después de la asignación $c := \lfloor \frac{a}{x} \rfloor$ son $(3, 100000)$, $(500001, 5)$, $(250003, 11)$, $(125007, 23)$, $(62515, 47)$, $(31281, 95)$, $(15688, 191)$, $(7939, 377)$, $(4158, 721)$, $(2439, 1230)$, $(1834, 1635)$, $(1734, 1730)$, $(1732, 1732)$. Para el caso de $raíz2(3000000000, 3)$, son $(3, 100000000)$, $(500000001, 5)$, $(250000003, 11)$, $(125000007, 23)$, $(62500015, 47)$, $(31250031, 95)$, $(15625063, 191)$, $(781377, 383)$, $(390880, 767)$, $(195823, 1531)$, $(98677, 3040)$, $(50858, 5898)$, $(28378, 10571)$, $(19474, 15405)$, $(17439, 17202)$, $(17320, 17321)$. Note que $1732^2 \leq 3000000 < 1733^2$ y $17320^2 \leq 3000000000 < 17321^2$.

Ejercicio 14. Haga en Maple:

$a := 2000000$; $x := 3$; (preparándonos para $\lfloor \sqrt{2} \rfloor$ hasta 3 decimales). Repetir (ver ej. 9):
 $c = \text{cociente}(a, x)$; $x := \text{cociente}(x+a, 2)$; hasta $|x - c| \leq 1$.

Ejercicio 15.

- (a) (3) Demuestre que cuando $raíz2$ conteste, contesta correctamente (con $\lfloor \sqrt{a} \rfloor$).
- (b) (5) Demuestre que $raíz2$ siempre termina. Sugerencia: Demuestre que en con cada llamada recursiva $|x - c|$ se reduce en por lo menos 1.

LOS NÚMEROS NATURALES

1. Axiomas

La idea de un número natural se ve en la de una fila de palos. Aquí exhibo dos:

|||||

y ahora exhibo su suma, que se forma poniendo una fila al lado de la otra:

|||||

Para la multiplicación, $a \times b$, sean a y b así:

|||| |||

y ahora pongo una copia de b por cada palo en a :

||| ||| ||| ||| |||

y luego sumo todos (la ley asociativa de la adición es obvia):

|||||

Incluyo como número natural la fila vacía de palos:

Naturalmente, para hablar de este número natural necesito algún símbolo. Hace muchos siglos los indios decidieron representarlo con un huevo:

0

nadie sabe exactamente por qué. Algunos se quejan de que 0 no está conforme con la idea de *natural* y quieren hablar de “números enteros no negativos”, pero ¿de donde vienen los números enteros? Vienen de poner signos a los números naturales.

Ahora hacemos unas cuantas definiciones: $a < b$ significa que a tiene menos palos que b . $a \leq b$ significa que $b \not< a$, o sea que o $a < b$ o $a = b$.

Ahora podemos sacar algunas leyes: $0 \leq a$ para cualquier número a (por ahora con “número” queremos decir número natural.) $a + 0 = 0 + a = a$ para cualquier número a . $a \times (b + c) = a \times b + a \times c$ para números a, b, c cualesquiera, y la conmutatividad de $+$ y de \times , etc. Y la ley de ón:

Para cualquier par de números a, b , con $b \neq 0$, existe un único par c, r de números tal que

$$a = c \times b + r \text{ y } r < b$$

(nótese que sin tener 0 como natural no se cumpliría esta última.) La ley de división se ve con ir cada b palos en la fila a y metiendo un espacio, hasta que no quedan b palos más. Ese último grupo es r y el número de grupos de b palos es c .

Ahora, ¿queda claras todas las definiciones que hemos hecho? Y ¿tenemos demostración formal de esas leyes, y otras que queremos emplear? Por ejemplo, ¿qué quiere decir “ $a = b$ ”? ¿Que a tiene el mismo número de palos que tiene b ? Pero ésa no puede ser la respuesta porque hemos dicho que *número* quiere decir *fila de palos*, y “ a tiene la misma fila de palos que tiene b ” no transmite información. Podemos hablar de conectar con lápiz cada palo de a con un palo de b poniendo en correspondencia biunívoca los dos conjuntos de palos, pero ya la cosa se complica. Además, no puedo decir que dos filas de palos son iguales –si fueran exactamente

iguales tendrían que ocupar el mismo espacio en el mismo tiempo, y entonces serían una sola fila. Sólo podemos hablar de la *equivalencia* de dos filas de palos, en que tengan el mismo número de palos, pero allí estamos usando el término *número*, que es lo que queremos definir.

Claro, la idea de las operaciones con palos es bastante clara, y podríamos anunciar como axiomas todas las leyes que son tan claras a la intuición, y deducir cosas más sutiles razonando formalmente con éstas. Bueno, ¿cuál va a ser esta colección de leyes fundamentales? Peano sugirió las siguientes leyes, conocidas ahora como los *axiomas de Peano*:

1. 0 es un número.
2. Si a es un número entonces $a^|$ es un número (pronunciado “a prima”. Es el *sucesor* de a .)
3. Si a y b son números, $a^| = b^|$ si y sólo si $a = b$. (Un número no tiene dos *predecesores*.)
4. Si a es un número entonces $a^| \neq 0$. (0 no tiene predecesor.)
5. Si S es un conjunto de números que cumple:
 - $0 \in S$
 - Para cada $a \in S$, también $a^| \in S$
 entonces S contiene a todos los números.

De allí podemos demostrar formalmente todas las leyes que mencionamos, y más.

En el último capítulo vamos a ver que no todo lo que es verdadero de los números naturales puede demostrarse partiendo de estos axiomas, pero nada de eso ha hecho falta a los teóricos de números hasta ahora.

Pero quiero enfatizar que no es que los axiomas vienen de la montaña en tabletas de piedra para que podamos empezar sacando teoremas, sino vienen de lo que ya sabemos de las filas de palos. Vamos a definir por ejemplo adición formalmente y demostrar axiomáticamente que $a + b = b + a$ pero no por eso vamos a estar más seguros de ese hecho. Si los axiomas no fueran capaces de desarrollar eso tendríamos que agregar otro axioma, porque ya sabemos que es verdad, no por medio de un sistema formal, sino porque ¡vemos claramente que así son los palos!

Si creemos a Kronecker que Dios nos dió los números naturales, y el resto de la matemática es obra del hombre, de estos 5 axiomas, y por tanto de los palos, depende toda la matemática. Los números enteros son números naturales con signo (reconociendo que $-0 = 0$), los números racionales son pares ordenados de enteros, los números reales son límites de sucesiones de racionales, los números complejos son pares ordenados de reales, ...

La matemática discreta trata de problemas expresables en términos de enteros. Empezamos con números naturales. No es el objetivo tratar todo axiomáticamente, pero queremos indicar cómo se puede hacer eso. Se puede hacer las definiciones de las funciones $+$ y \times de forma tal que es posible demostrar formalmente las leyes que hemos mencionado arriba.

Hagamos en dos líneas una definición de adición:

- Para todo número a , $a + 0 = a$.
- Para todos números a, b , $a + b^| = a^| + b$.

Por ejemplo, $0^| + 0^| = 0^{||} + 0^| = 0^{|||} + 0 = 0^{||||}$, y allí tenemos una demostración de que $2 + 2 = 4$. Y allí están las filas de palos y la suma consiste en mover todos los palos de una fila al lado de la otra, aunque el único algoritmo que tenemos los mueve uno por uno.

Vamos a demostrar la ley conmutativa para la suma. Primero,

LEMA 2.1. *Para todo a, b , $a + b^| = (a + b)^|$.*

Demostración. Por inducción matemática (axioma 5) sobre b . Sea $S = \{b : \text{para todo } a, a + b^| = (a + b)^|\}$. El objetivo es mostrar que S es todos los números. $0 \in S$ porque para cualquier a , $a + 0^| = a^| + 0 = a^| = (a + 0)^|$, las primeras dos igualdades por la definición de $+$ y también la última porque $a = a + 0$. Dado $b \in S$, ¿necesariamente sigue que $b^| \in S$? Ahora $a + b^{||} = a^| + b^| = (a^| + b^|)^|$ porque $b \in S$. Pero $a^| + b = a + b^|$ por la definición de $+$, que nos

da finalmente $a + b^{\parallel} = (a + b^{\parallel})^{\parallel}$, mostrando que $b^{\parallel} \in S$. Entonces por el axioma 5 todo número pertenece a S . \square

LEMA 2.2. *Para todo a , $0 + a = a$.*

Demostración. Sea

$$S = \{a : 0 + a = a\}.$$

$0 \in S$ porque $0 + 0 = 0$ (definición de $+$). Si $a \in S$ entonces $0 + a^{\parallel} = (0 + a)^{\parallel}$ por lema 2.1, y $0 + a = a$ porque $a \in S$, y sigue que $0 + a^{\parallel} = a^{\parallel}$, es decir, $a^{\parallel} \in S$. Por axioma 5, S contiene todos los números. \square

Ahora sea $S = \{b : a + b = b + a \text{ para todo } a\}$. $0 \in S$ porque $a + 0$ y $0 + a$ ambos son a por definición de $+$ y lema 2.2 respectivamente. Sea $b \in S$. Entonces para todo a , $a + b^{\parallel} = (a + b)^{\parallel}$ por lema 2.1, que es $(b + a)^{\parallel}$ porque $b \in S$, que es $b + a^{\parallel}$ por lema 2.1, que es $b^{\parallel} + a$ por la definición de $+$. Entonces $b^{\parallel} \in S$. Por el axioma 5, S es todos los números, y así queda demostrada la ley conmutativa para $+$. \square

DEFINICIÓN 2.3. Para dos números a, b , $a \leq b$ si existe un número x tal que $a + x = b$. El número x se llama $b - a$. Se dice $a < b$ si $a \leq b$ y $a \neq b$.

Vamos a utilizar mucho la siguiente forma del principio de inducción matemática.

TEOREMA 2.4. *(Buen orden de los números naturales) Todo conjunto no vacío de números tiene un miembro mínimo, es decir, un miembro m tal que $m \leq a$ para cualquier miembro a del conjunto.*

Demostración. Sea $S = \{n : \text{todo conjunto de números que contenga } n \text{ contiene un miembro mínimo}\}$. $0 \in S$ porque para cualquier número a , $a + 0 = a$ así que $0 \leq a$ por definición. Ahora sea $n \in S$, vamos a ver si implica que $n^{\parallel} \in S$. Sea C un conjunto con $n^{\parallel} \in C$. Queremos asegurar que C tiene un miembro mínimo. Si $0 \in C$ entonces sí tiene un miembro mínimo. Entonces supongamos que $0 \notin C$. Entonces todo $a \in C$ es b^{\parallel} para algún b (ver ejercicio 5). Formemos el conjunto D restando 1 de cada miembro de C , es decir, $D = \{a : a^{\parallel} \in C\}$. Es claro que $n \in D$ porque $n^{\parallel} \in C$. Entonces por la hipótesis de inducción D tiene un miembro mínimo m . Por supuesto sospechamos que m^{\parallel} sea miembro mínimo de C . Para confirmar éso, sea $a \in C$. Entonces $a = b^{\parallel}$ para algún $b \in D$. Ahora $m \leq b$, por tanto (ver ejercicio 6) $m^{\parallel} \leq b^{\parallel} = a$. \square

Ejercicios 2.1

1. (1) Demuestre la ley asociativa para $+$, que siempre $(a + b) + c = a + (b + c)$. (Sea $S = \{c : \text{para todo } a, b, (a + b) + c = a + (b + c)\}$ y aplicar lema 2.1.)
2. (2) Definiendo
 - $0 \times b = 0$
 - $a^{\parallel} \times b = a \times b + b$
 demuestre la ley distributiva a la izquierda, $a \times (b + c) = a \times b + a \times c$. Sugerencia: sea $S = \{a : \text{para todo } b, c, a \times (b + c) = a \times b + a \times c\}$.
3. (2) Partiendo del ejercicio 2, demuestre la ley conmutativa para \times .
4. (1) Partiendo del ejercicio 2, demuestre la ley asociativa para \times .
5. (1) Demostrar, sin usar principio del buen orden, que para todo número $a \neq 0$, $a = b^{\parallel}$ para algún b .
6. (1) Demostrar que $a \leq b \Leftrightarrow a^{\parallel} \leq b^{\parallel}$ (no puede usar principio de buen orden).

Creo que ésto alcanza para indicar cómo los axiomas de Peano son suficientes para desarrollar la teoría de números. De aquí en adelante usaremos muchas otras propiedades, como por ejemplo las leyes de cancelación para la suma y el producto y leyes de exponenciación, sin demostrarlas desde los axiomas, pero sí se los puede demostrar.

De hecho, podemos demostrar todo teorema que aparece en un libro o revista de la teoría de números. Pero veremos después con el teorema de Gödel que no se puede demostrar todo con los axiomas de Peano.

2. División

TEOREMA DE LA DIVISIÓN. Para todos números a, b con $b \neq 0$, existe un único par c, r de números tal que $a = c \times b + r$ y $r < b$.

Demostración. La existencia de por lo menos un tal par estará asegurado por el algoritmo $cr(a, b)$ del capítulo 1 si demostramos que ese algoritmo es correcto. Para demostrar eso, notemos primero que hay dos maneras de contestar mal: contestar algo incorrecto, o no contestar nada, es decir, perderse en el pozo de siempre pedir otra invocación. Entonces queremos demostrar

∗: Para todo a, b con $b > 0$, $cr(a, b)$ contesta (c, r) tal que $a = cb + r$ y $r < b$.

Si hay un contraejemplo (a, b) a (\ast) sea uno con a lo más chico posible; es decir, para todo $a' < a$ y $b' > 0$, (a', b') no es un contraejemplo a (\ast) . No puede ser que $a = 0$ porque en ese caso $a = 0 \times b + a$ y $a < b$ ya que $b > 0$. Siendo $a \neq 0$, $cr(a, b)$ invoca $cr(a - b, b)$. Siendo $b > 0$, es $a - b < a$, entonces $(a - b, b)$ no es contraejemplo a (\ast) , es decir $cr(a - b, b)$ contesta (c, r) tal que $a - b = cb + r$ donde $r < b$. Tenemos que concluir que $a = b + cb + r$ lo que equivale a $a = (c + 1)b + r$. Ya que $r < b$ y el algoritmo en este punto contesta $(c + 1, b)$, está contestando correctamente, es decir, (a, b) no es contraejemplo a (\ast) . Pero si existe un contraejemplo, existe uno con a lo mínimo posible, entonces tiene que ser que no existe ningún contraejemplo, es decir (\ast) se cumple. \square

Este algoritmo es muy ineficiente (probar el cálculo de $cociente(10000, 7)$). Remediamos esto ahora.

Nosotros trabajamos con números que se expresan como $d_k \cdots d_0$, representando el número $\sum_{i=0}^k d_i \times 10^i$, que nos permite multiplicar por 10 fácilmente. Para aprovechar esto, conviene el siguiente algoritmo:

```

cr10(a, b) =
  si (a < b) (0, a)
  sino
    (c, r) := cr10(a, 10b)
    (d, s) := cr(r, b)
    (10c + d, s)

```

El funcionamiento de este algoritmo depende de que si $a = c \times 10b + r$ con $r < 10b$, y $r = db + s$ con $s < b$, entonces $a = c \times 10b + db + s = (10c + d)b + s$, y además, dirigiéndonos a su eficiencia, $d < 10$, poniendo límite al tiempo que requiere el cálculo de cr . Ahora no tenemos problema con el cociente y resto de 10000, 7. Para mantener claro el cálculo mientras lo hacemos a mano, hacemos una tabla con una fila para cada invocación de $cr10$, con los argumentos y las variables locales r, c, d, s :

a	b	c	r	d	s	resultado
10000	7	142	60	8	4	1428, 4
10000	70	14	200	2	60	142, 60
10000	700	1	3000	4	200	14, 200
10000	7000	0	10000	1	3000	1, 3000
10000	70000					0, 10000

Nótese que en la fila i , (c_i, r_i) es el resultado en la fila $i + 1$, $(d_i, s_i) = cr(r_i, b_i)$, y $c_i b_{i+1} + r_i = 10000$, y el resultado de la fila i es $cr(a_i, b_i)$.

En el ejercicio 1 se pide llenar la tabla para $cr10(98713, 11)$. Vamos a ver cómo hacer esto con la ayuda de Maple, para el caso 10000 dividido 7. Ponemos $k:=1; a[k]:=10000; b[k]:=7$; y hacemos $k:=k+1; a[k], b[k]:=a[k-1], 10*b[k-1]$; hasta que $b[k]>a[k]$. Entonces hacemos $kmax:=k; coc[k], res[k]:=0, a[k]$; y repetimos lo siguiente hasta que $k=1$:

```
k:=k-1; c[k], r[k]:=coc[k+1], res[k+1]; d[k], s[k]:=cr(r[k], b[k]);
coc[k], res[k]:=10*c[k]+d[k], s[k];
```

Finalmente recolectamos la tabla mediante

```
for i from 1 to kmax do print(a[i], b[i], c[i], r[i], d[i], s[i], coc[i], res[i]) od;
```

Aunque $cr10$ conviene para los que representan sus números en el sistema decimal, no despreciemos a los que prefieren otro sistema, digamos base B con $B > 1$:

```
crB(a, b, B) =
  si (a < b) (0, a)
  sino
    (c, r) := crB(a, Bb, B)
    (d, s) := cr(r, b)
    (Bc + d, s)
```

Por supuesto, si B es muy grande probablemente conviene usar mejor algoritmo que cr para calcular $cr(r, b)$. Esto ocurre cuando se hace aritmética de grandes números, que vemos más adelante.

Ejercicios 2.2

- (1) Llenar la tabla para $cr10(98713, 11)$.
- (1) Haga $a, b := \text{rand}(), 1 + \text{rand}() \bmod 10^5$ y llene la tabla para $cr10(a, b)$. Notese que a viene al azar de $\{0 \dots 10^{12} - 1\}$ y b de $\{1 \dots 10^7\}$, así que el cociente probablemente tiene 5 dígitos.
- (2) Demuestre que $crB(a, b, B)$ funciona. Sugerencia: inducción sobre el número de recursiones.
- (1) Usando la definición para dos números naturales a, b que $a \leq b$ si $b = a + n$ para algún número natural n , demuestre que si a divide a $b \neq 0$ entonces $a \leq b$.
- (4) Para cada aserción, si es verdadera demuéstrela y si no muestre un contraejemplo:
 - $\text{resto}(a + kb, b) = \text{resto}(a, b)$, $\text{cociente}(a + kb, b) = \text{cociente}(a, b) + k$.
 - $\text{resto}(a + d, b) = \text{resto}(a, b) + \text{resto}(d, b)$, $\text{cociente}(a + d, b) = \text{cociente}(a, b) + \text{cociente}(d, b)$.
 - $\text{resto}(a + d, b) = \text{resto}(\text{resto}(a, b) + \text{resto}(d, b), b)$.
 - $\text{resto}(ad, b) = \text{resto}(a, b)\text{resto}(d, b)$.
 - $\text{resto}(ad, b) = \text{resto}(\text{resto}(a, b)\text{resto}(d, b), b)$.
- (1) Aplique el algoritmo $cr10$ a los siguientes argumentos: $(300, 7)$, $(2541, 5)$, $(1001, 7)$, $(78964, 5)$.
- (1) Calcule $\text{resto}(2137^8, 7)$. (No es necesario obtener 2137^8 si aplica el resultado de uno de estos ejercicios.)
- (3) Es un hecho que para todo número racional positivo $r < 1$ existe un conjunto S de números naturales tal que $r = \sum_{i \in S} \frac{1}{i}$. Un ejercicio más adelante pide una demostración de esto, pero aquí solo vamos a sugerir mediante unos ejemplos un algoritmo para sacar tal conjunto, pedir expresar el algoritmo, y aplicarlo a otros ejemplos. Estos son los ejemplos que sugieren el algoritmo:
 - $\frac{2}{5} = \frac{1}{3} + (\frac{2}{5} - \frac{1}{3}) = \frac{1}{3} + \frac{1}{15}$
 - $\frac{7}{17} = \frac{1}{3} + (\frac{7}{17} - \frac{1}{3}) = \frac{1}{3} + \frac{4}{51} = \frac{1}{3} + \frac{1}{13} + (\frac{4}{51} - \frac{1}{13}) = \frac{1}{3} + \frac{1}{13} + \frac{1}{663}$
 - $\frac{8}{17} = \frac{1}{3} + (\frac{8}{17} - \frac{1}{3}) = \frac{1}{3} + \frac{7}{51} = \frac{1}{3} + \frac{1}{8} + (\frac{7}{51} - \frac{1}{8}) = \frac{1}{3} + \frac{1}{8} + \frac{5}{408} = \frac{1}{3} + \frac{1}{8} + \frac{1}{82} + (\frac{5}{408} - \frac{1}{82}) = \frac{1}{3} + \frac{1}{8} + \frac{1}{82} + \frac{1}{16728}$.

Expresa el algoritmo sugerido y aplíquelos a $2/3$, $77/120$, $1231/1430$, $6/19$.

9. (4) Demuestre el algoritmo del ejercicio anterior.
10. (1) Decida para cada una de las siguientes aserciones si es verdadera o falsa para los enteros, y justifique la respuesta: a) $a|a$. b) $a|-a$. c) si $a|b$ entonces $b|a$. d) $a|0$. e) Si $a|b$ y $b|c$ entonces $a|c$. f) Si $a|b$ y $b|a$ entonces $a = b$. g) Si $a|bc$ entonces o $a|b$ o $a|c$. h) Si $a|b$ entonces $a^2|b^2$.

3. Primos

DEFINICIÓN 2.5. Un número n es *primo* si $n > 1$ y sus únicos factores son 1 y n .

Los primeros 10 primos son 2,3,5,7,11,13,17,19,23,29.

Todo número $n > 1$ es divisible por algún primo. Porque, sea $S = \{n : n > 1 \text{ y ningún primo es factor de } n\}$. Si S no es vacío, sea n su miembro mínimo. Este n no es primo porque no tiene factor primo. Entonces n tiene un factor k con $1 < k < n$, $n = n'k$. Por la minimalidad de n en S , $k \notin S$, y por tanto k tiene un factor primo p , $k = k'p$. Pero entonces $n = n'k'p$; esto dice que $n \notin S$, contradicción que surge de suponer que S no es vacío.

TEOREMA 2.6. *Hay infinitos primos.*

Demostración. (Euclides). Supongamos que hay exactamente k primos, p_1, \dots, p_k . Sea n el producto de todos, $n = p_1 \times \dots \times p_k$. Entonces $\text{resto}(n+1, p_i) = 1$ para cada uno de esos p_i , es decir, ningún p_i es factor de $n+1$. Pero acabamos de demostrar que todo número mayor que 1 tiene un factor primo, por tanto $\{p_1, \dots, p_k\}$ no puede ser el conjunto de todos los primos. \square

Los algoritmos para descubrir si un número es primo han ocupado la atención de matemáticos por siglos. ¿Es 999 primo? No, es evidentemente divisible por 3. ¿Es 997 primo? No es divisible por 3 (la suma de sus dígitos no lo es), ni por 5, ni por 7, ni por 11 (aquí empiezo a usar una calculadora). Momentito: ¿por qué salto los números pares? Porque 997 no es par, y si fuera divisible por un número par, su árbol de factores tendría 2. ¿Por qué salto 9? Porque no es primo, y ya he probado sus factores, y si 9 fuera factor, también lo sería todo factor de él. Entonces sólo voy a probar primos para divisibilidad. Sigo con 13, 17, 19, 23, 29, 31. Ninguno es factor de 997. ¿Sigo con 37? No. Porque $37^2 > 997$, y si fuera $997 = 37x$ tendría que ser $x < 37$; si no, si $x \geq 37$, $37x \geq 37^2$ que es mayor que 997. Es decir, si hay un factor mayor o igual que 37, habrá otro menor que 37, pero ya sé que no hay porque he probado todos los primos menores que 37. Entonces podemos decir: para probar si un número n es primo basta probar todo primo p tal que $p^2 \leq n$. Para números menores que 1000 esto me sirve bien porque no me obliga a ir más allá que 31, y sé casi de memoria los 11 primos hasta 31: 2,3,5,7,11,13,17,19,23,29,31. Si supiera de memoria los primos hasta 100 me servirían, con calculadora, para probar, en la misma forma, la primalidad de números hasta 10 mil. Si necesito probar números del orden de 100000, tendría que saber de memoria los primos menores que 316. Hay 65 (lo calculé con la función `ithprime` de Maple que da el i -ésimo primo). Si no sé estos de memoria entonces voy a usar más tiempo verificando si 177, por ejemplo, es primo, que simplemente dividiendo por él. Pero las computadoras son buenas para saber cosas de memoria. No es nada cargar una máquina con los primeros 65 primos, y un programa que prueba primalidad de un número menor que 100000 probando los 65 primos uno tras otro. Si hablamos de primos menores que un millón, nuestro amigo 997 es primo número 168, así que la idea de cargar una tabla de n primos para poder decidir primalidad hasta n^2 no está tan mal.

Pero hay una constante competencia para ver quien puede encontrar el primo más grande. El actual campeón llenaría un libro gordo con dígitos, ya que el 23 de Agosto de 2008 fue descubierto en UCLA con 12.978.989 cifras decimales. Es $2^p - 1$, con p primo e igual a 43.112.609¹

¹Todos los campeones por varias décadas han sido de la forma $2^p - 1$ con p primo, es decir, *primos de Mersenne*. Este p es el número 45 de los primos de Mersenne conocidos, y con eso UCLA ganó un premio de 100.000 dolares por ser el primero en descubrir un primo de más de 10 millones de dígitos. El número 46, con

Hasta ahora no hay método general para probar la primalidad de cualquier número que sea significativamente mejor en sentido práctico que el que hemos enunciado, es decir, probar cada posible divisor primo menor que la raíz cuadrada. El nuevo algoritmo AKS, al que vamos a dar un vistazo más adelante, gana en cierto sentido teórico, pero hasta ahora no práctico. Hay métodos más rápidos para decidir con poca probabilidad de error si un número es primo, y es interesante notar que la función *isprime* de Maple usa uno de estos (*type(x, prime)*), que dice *false* cuando *x* no es natural, invoca *isprime* cuando lo es: no dice nada al respecto con *nextprime*, pero Maple demora más en *nextprime*(10^{500}) que en probar los impares mayor que 10^{500} con *isprime* hasta encontrar uno. Formalmente, el algoritmo más simple para determinar primalidad es:

```
primo(n) = si (n < 2) no sino primo(n, 2)
donde
  primo(n, d) =
    si ( $d^2 > n$ ) sí
    sino si (resto(n, d) = 0) no sino primo(n, d + 1)
```

Tal vez el lector detecte una ineficiencia en esto: una vez que sale no divisible por 2, ¿para qué vamos a probar los candidatos pares? Determinamos si es impar primero, y probamos solamente divisores impares, y el cálculo debería reducirse en la mitad más o menos. Invitamos al lector a escribir el algoritmo así.

Volviendo al tema de emplear un conjunto de *n* primos, ¿como engendramos tal conjunto? Una posibilidad sería este algoritmo *primosHasta* que contesta el conjunto de todos los primos que no exceden su argumento:

```
primosHasta(n) = P(n, {}, 2)
donde
  P(n, C, d) =
    si (d > n) C
    sino si (primo(d)) P(n, C ∪ {d}, d + 1)
    sino P(n, C, d + 1)
```

Como en el caso de *primo*, se puede mejorar esto bastante poniendo 2 en el conjunto primero y visitando sólo números impares después. Pero para primos hasta *n* un tal Eratóstenes tuvo una idea genial. Empezamos con el conjunto {2, ..., *n*} marcando el primer número, 2, como primo y borrando todos los otros múltiplos de 2, dejando esto, para el caso *n* = 31:

2 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31

destacando el 2 para indicar que es primo. Ahora el primer número no marcado después de 2, el 3, es primo. Entonces lo marcamos y borramos todos sus otros múltiplos:

2 3 5 7 11 13 17 19 23 25 29 31

De nuevo, el primer número no marcado es primo, entonces lo marcamos y borramos todos sus otros múltiplos (solo el 25):

2 3 5 7 11 13 17 19 23 29 31

De nuevo el primer número no marcado es primo, pero además su cuadrado, 49, es mayor que *n* = 31, entonces no queda nada más que primos.

Si fuera *n* = 190 veríamos, cambiando la notación para que quepa en la página, empezando con lo que queda después de borrar múltiplos de 2, 3 y 5:

2|3|5|7|11|13|17|19|23|29|31|37|41|43|47|53|59|61|67|71|73|77|79|83|89|91|97|101|103|107|109|113|119|
121|127|131|133|137|139|143|149|151|157|161|163|167|169|173|179|181|183|187

y luego múltiplos de 7:

2|3|5|7|11|13|17|19|23|29|31|37|41|43|47|53|59|61|67|71|73|79|83|89|97|101|103|107|109|113|

p = 35,156,667, fue descubierto 2 semanas después y dió 11.185.272 dígitos, perdiendo el premio, que estuvo abierto casi una década.

121|127|131|137|139|143|149|151|157|161|163|167|169|173|179|181|183|187

y luego múltiplos de 11:

2|3|5|7|11|13|17|19|23|29|31|37|41|43|47|53|59|61|67|71|73|79|83|89|97|101|103|107|109|113|127|
131|137|139|149|151|157|161|163|167|169|173|179|181|183

y luego múltiplos de 13:

2|3|5|7|11|13|17|19|23|29|31|37|41|43|47|53|59|61|67|71|73|79|83|89|97|101|103|107|109|113|127|
131|137|139|149|151|157|161|163|167|173|179|181|183

y esos son todos los primos hasta 190, porque no habrá múltiplos de 17 puesto que $17^2 > 190$.

Este procedimiento se llama la *criba de Eratóstenes*. Lo definimos formalmente en términos de $criba(P, C, n)$ cuyos tres argumentos son: un conjunto P de primos, un conjunto C de números $\leq n$ y el final de la búsqueda n :

$criba(n) = criba(\{\}, \{2, \dots, n\}, n)$

donde

$criba(P, C, n) =$
 $d :=$ el elemento mínimo de C
si $(d^2 > n)$ $P \cup C$
sino $criba(P \cup \{d\}, C - \text{ todos los múltiplos de } d, n)$

Con cada invocación de $criba(P, C, n)$ se cumplen las siguientes:

- El elemento mínimo d de C es primo
- P es el conjunto de todos los primos menores que d .
- C es $\{2, \dots, n\} -$ todos los múltiplos de miembros de P .

Estas se cumplen con la primera invocación, hecha por $criba(n)$. Suponga que se cumple para alguna invocación que engendra otra. Sea $C' = C -$ múltiplos de d . Sea e el elemento mínimo de C' . Algún primo $p > d$ es factor de e . Pero $p \in C$ y $p \leq e$, por tanto $p = e$, mostrando que el nuevo argumento cumple la primera propiedad. Ahora $P \cup \{d\}$ es el conjunto de todos los primos $q \leq d$. Para la segunda propiedad, falta asegurar que no hay primo p con $d < p < e$. Pero si hubiera tal primo p , p no es múltiplo de ningún elemento de P , y $p \leq n$ porque $p < e$, y ya que no puede ser múltiplo de e , $p \in C'$, pero esto contradice que e sea elemento mínimo de C' . La tercera es consecuencia de la hipótesis de inducción y que C' se forma sacando todos los múltiplos de miembros de P y luego todos los múltiplos de d de eso, habiendo empezado con $\{2, \dots, n\}$.

Ejercicios 2.3

1. (2) Modifique el algoritmo *primo* para que evite probar números pares, salvo 2.
2. (4) Programe *criba* en Maple. *Sugerencia:* Haga en Maple
 $n:=2000; S:= \{\text{seq}(i, i=2..n)\}; P := \{\};$
y después iterar esta línea:
 $d := S\{1\}; P := P \text{ union } \{d\}; S := S \text{ minus } \{\text{seq}(i*d, i=1..\text{floor}(n/d))\};$
tantas veces que parece necesario (la teoría la indica). Diseñe el programa en base de los resultados de esto.
3. (5) Diseñe un algoritmo (sin programarlo en Maple) que emplea una lista de todos los primos menor que 1000, para sacar todos los primos entre n y $n + k$, donde $1000 \leq n$ y $n + k < 1000000$. *Sugerencia:* use el mismo principio que *criba*, sacando $n \bmod p$ para cada p en la lista, y eliminando múltiplos de p en el intervalo indicado.
4. (1) Haga un árbol de factores para $7!$, y muestre su descomposición en primos.
5. (1) Encuentre el primer número primo mayor que 1000.
6. (3) Encuentre a mano los primos entre 1000 y 1100 de acuerdo a las siguientes indicaciones:

- a) Ejecute la criba de Eratóstenes entre 1 y algún valor chico de N adecuadamente seleccionado.
- b) Usando estos primos, aplique la idea de la criba de Eratóstenes al conjunto $\{1000, \dots, 1100\}$.
7. 1) ¿Si se multiplican los primeros n primos y se agrega 1 al resultado se obtiene un nuevo primo?
8. (2) Encuentre un método para encontrar n números compuestos consecutivos. Sugerencia: Comience a buscar con $n!$.
9. (2) Calcule $n^2 + n + 41$ para los n natural entre 1 y 20. ¿Qué observa sobre los resultados? ¿Ocurre siempre este fenómeno? Demuéstrelo o dé el primer contraejemplo.
10. (1) Alguien me dijo que la función definida así:

$$f(n) = nt(n) + 2(1 - t(n))$$
donde $t(n) = \text{resto}(M(n-1, n)^2, n)$ **y**
 $M(n, m) = \text{si } (n=0) \text{ 1 sino } \text{resto}(nM(n-1, m), m)$
 engendraba sólo primos, y todos los primos. Pruébela para unos cuantos valores de $n > 0$. Más adelante veremos qué está ocurriendo con este algoritmo.
11. (1) Encuentre una sucesión aritmética no constante que produce sólo primos, o demuestre que no puede existir.
12. (3) Use el método con que Euclides demostró que hay infinitos primos para demostrar que el n -ésimo primo no puede ser mayor que $2^{2^{n-1}}$. Sugerencia: considere el primer n que no cumple.

4. Máximo común divisor

DEFINICIÓN 2.7. Dados dos números a, b no ambos 0, el *máximo común divisor* de a y b es el máximo divisor común de los dos.

El orden correcto en castellano sería “divisor común máximo” y algunos autores hispánicos y traductores escriben así, pero la mayoría sigue el orden del inglés, *maximum common divisor*.

Un algoritmo para calcular ésto es

$mcd(a, b) =$
 $d := \min(a, b); \text{ si } (d = 0) \text{ máx}(a, b) \text{ sino } mcd(a, b, d)$
donde
 $mcd(a, b, d) = \text{si } (\text{resto}(a, d) = 0 \text{ y } (\text{resto}(b, d) = 0)) \text{ d sino } mcd(a, b, d - 1)$

lo que dice probar todos los d desde $\min(a, b)$ hasta 1, en ese orden, y aceptar el primero que sea divisor común. Pero Euclides hace más de dos milenios descubrió un método más rápido:

El algoritmo de Euclides

$mcd(a, b) = \text{si } (b = 0) \text{ a sino } mcd(b, \text{resto}(a, b))$

Se puede palpar el funcionamiento de ésto con Maple poniendo los dos argumentos en a, b y repitiendo mientras $b \neq 0$

$$a, b := b, a \bmod b$$

La razón por la que el algoritmo de Euclides funciona es que los factores comunes de a y b son exactamente los mismos que los de b y $\text{resto}(a, b)$. Porque, sea $a = cb + r$ con $r < b$. Si d es factor común de a y b , $a = md$ y $b = nd$, entonces $md = cnd + r$ o sea $r = (m - cn)d$, así que d es factor común de b y r . Y si d es factor común de b y r , $b = nd$ y $r = md$, entonces $a = cnd + md = (cn + m)d$, así que d es factor común de a y b . Si los dos conjuntos de factores comunes son los mismos, claro que los máximos comunes divisores son iguales.

Algunas veces se define el mcd como aquel divisor común que es divisible por todos los divisores comunes, una definición que no corresponde a su nombre. Aquí lo tenemos como

teorema: El número con esa propiedad tiene que ser el máximo común divisor por el ejercicio 3. Recíprocamente, sea $g = \text{mcd}(a, b)$, y sea d un divisor común de a, b . Procedemos por inducción sobre b . Si $b = 0$ entonces $g = a$ y d es divisor de $a = g$. Sea b tal que para todo $b' < b$ y todo a' , cualquier divisor común de a', b' es divisor de $\text{mcd}(a', b')$. Hemos visto que los divisores comunes de a y b son exactamente los de b y $r = \text{resto}(a, b) < b$. Por eso, la hipótesis de inducción, y el hecho que hemos visto que $\text{mcd}(a, b) = \text{mcd}(b, r)$, cualquier divisor común de a, b divide a g .

Ejercicios 2.4

1. (1) Palpe el algoritmo de Euclides con Maple como se dijo recientemente, poniendo `_seed:=287; a, b := rand(), rand()` (en Maple, $a \bmod b$ da el resto de a dividido b).
2. (2) Con Maple, contar para mil pares de números al azar (`rand()`), cuántos tienen su $\text{mcd} > 1$, y cuál es el máximo de los mil mcd y exprese ese último como producto de primos.
3. (1) Demuestre que un divisor común de a y b que es divisible por todo divisor común de a y b es el máximo común divisor de a y b .
4. (2) La función $\text{mcm}(a, b)$, el menor común múltiplo de a y b , se define como el primer número que es múltiplo de ambos. Demuestre que

$$\text{mcm}(a, b) = \frac{ab}{\text{mcd}(a, b)}.$$

5. (1) Demuestre que $\text{mcd}(ka, kb) = k \times \text{mcd}(a, b)$;
 6. (4) Use el ejercicio anterior para demostrar que el siguiente algoritmo calcula $\text{mcd}(a, b)$:
 $\text{mcd2}(a, b) =$
si ($a = 0$) b **sino si** ($b = 0$) a **sino**
sea $ca, ra = \text{cr}(a, 2)$; $cb, rb = \text{cr}(b, 2)$ (por cualquier algoritmo)
si ($ra = 0$)
si ($rb = 0$) $2 \times \text{mcd2}(ca, cb)$
sino $\text{mcd2}(ca, b)$
sino
si ($rb = 0$) $\text{mcd2}(a, cb)$
sino $\text{mcd2}(\text{máx}(a, b) - \text{mín}(a, b), \text{mín}(a, b))$
 7. (1) Calcular $\text{mcd}(a, b)$ a mano siguiendo el algoritmo de Euclides para $(a, b) = (17, 3)$, $(3, 17)$, $(121, 22)$, $(999, 99)$, $(1005, 15)$, $(9871, 794)$, $(377, 233)$.
 8. (1) Demostrar que si p es primo, $m > 0$, y p no es factor de b , entonces $\text{mcd}(pm, b) = \text{mcd}(m, b)$.
-

5. El Teorema Fundamental de la Aritmética

Si un número n no es primo, es $n = n_0 \times n_1$ donde $n_0, n_1 > 1$. Si n_0 no es primo es $n_0 = n_{00} \times n_{01}$ donde $n_{00}, n_{01} > 1$ y si n_1 no es primo es $n_1 = n_{10} \times n_{11}$ donde $n_{10}, n_{11} > 1$, y así sucesivamente hasta encontrar todos primos. En ese momento habremos sacado $n = p_1 \times \dots \times p_k$ donde cada p_i es primo. Ese resultado llamamos una *factorización en primos*.

Otra persona empezando con el mismo n puede sacar otra factorización $n = q_1 \times \dots \times q_j$ distinta de la primera. Por ejemplo

$$72 = 6 \times 12 = (3 \times 2) \times (2 \times 6) = (3 \times 2) \times (2 \times (3 \times 2))$$

y

$$72 = 18 \times 4 = (2 \times 9) \times (2 \times 2) = (2 \times (3 \times 3)) \times (2 \times 2).$$

Una factorización en primos de 72 es $3 \times 2 \times 2 \times 3 \times 2$, y otra es $2 \times 3 \times 3 \times 2 \times 2$. Notamos primero que cada una tiene 5 primos, pero más: Si los primos se ponen en orden no decreciente, las dos factorizaciones son idénticas: $2 \times 2 \times 2 \times 3 \times 3$, o más compactamente, $2^3 \times 3^2$. El sentido

del teorema fundamental es que hay una, y en ese sentido sólo una, factorización en primos de un número $n > 1$. Formalmente:

TEOREMA 2.8. (*factorización en primos*). *Cada número mayor que 1 tiene una factorización en primos. Esa factorización es única en el sentido que si p_1, \dots, p_k y q_1, \dots, q_j son primos, con $p_i \leq p_{i+1}$ y $q_i \leq q_{i+1}$ para cada i , y $p_1 \times \dots \times p_k = q_1 \times \dots \times q_j$, entonces $k = j$ y $p_i = q_i$ para cada i .*

La demostración de esto depende del *Lema de Euclides* (que era un lema en el contexto en que lo usó Euclides, pero que es un importante *teorema* de la teoría de números). El Lema de Euclides es consecuencia de otro teorema que tiene muchas aplicaciones. Este teorema y la demostración del Lema en base de ello veremos después. Lo que necesitamos aquí es un corolario del Lema, que anunciamos y demostramos ahora.

LEMA 2.9. (*Euclides*). *Si p es primo y divide a un producto ab , entonces o p divide a a o p divide a b .*

COROLARIO 2.10. *Si p es primo y divide a $q_1 \times \dots \times q_n$ donde cada q_i es primo, entonces $p = q_i$ para algún i .*

Demostración. Por inducción sobre n . Si $n = 1$ la hipótesis dice $p = q_1$ no más y no queda nada para demostrar. Si $n > 1$ y $p \neq q_1$ entonces por el Lema de Euclides p divide a $q_2 \times \dots \times q_n$. Entonces si la aserción se cumple para $n - 1$, $p = q_i$ para algún $i > 1$, es decir, si se cumple para $n - 1$ entonces o $p = q_1$ o $p = q_i$ para algún $i > 1$. Entonces se cumple para n . \square

Con esto podemos sacar la

DEMOSTRACIÓN del teorema de la factorización en primos. Para su existencia, suponga que hay números sin factorización en primos, y sea n el elemento mínimo de ese conjunto no vacío. Entonces $n = ab$ donde ambos de a y b son mayores que 1, y por tanto menores que n . Por la minimalidad de n , cada uno de a y b tienen factorizaciones en primos. Pero entonces su producto forma una factorización en primos, que es una contradicción porque n es un número sin factorización en primos. La única causa posible de esta contradicción es la asunción de la existencia de números sin factorización en primos.

Para la unicidad, sea $p_1 \times \dots \times p_k = q_1 \times \dots \times q_j$ un contraejemplo con $k \leq j$ y k lo menor posible. Si $p_1 < q_1$ entonces por el corolario al Lema de Euclides p_1 es uno de los q_i , lo cual es imposible porque $q_1 \leq q_i$ para cada i , y ya que $p_1 < q_1$ eso dice que $p_1 < q_i$ para cada i . El mismo argumento muestra que no puede ser $q_1 < p_1$. Entonces $p_1 = q_1$ y concluimos que $p_2 \times \dots \times p_k = q_2 \times \dots \times q_j$. Por la minimalidad de k , $p_2 \times \dots \times p_k = q_2 \times \dots \times q_j$ no constituye un contraejemplo. Es decir, $j = k$, y $p_i = q_i$ para todo i , $1 < i \leq k$. Falta allí sólo decir que $p_1 = q_1$, pero a eso ya lo tenemos, así que el supuesto contraejemplo no lo es, una contradicción implicada por la asunción de contraejemplos. \square

Queda para demostrar el Lema de Euclides. Es una consecuencia de un teorema que es debido a Euclides pero no lleva su nombre porque no lo enunció como teorema sino propuso un algoritmo para calcular los números cuyo existencia es la esencia del teorema. Con su acostumbrado desprecio para los algoritmos los matemáticos puros no estaban cómodos con aceptarlo como demostración, pero nosotros sí. El teorema es este:

TEOREMA 2.11. *Si a, b son números naturales no ambos 0 entonces existen números enteros s, t tales que $sa + tb = \text{mcd}(a, b)$.*

A este teorema vamos a decir “el teorema st ”. El algoritmo que lo demuestre sí lleva el nombre de Euclides:

EL ALGORITMO EXTENDIDO DE EUCLIDES.

$m, b) =$
si $(b = 0)$ $(1, 0)$
sino
sea $(c, r) = cr(a, b)$ **y** $(s, t) = st(b, r)$
 $(t, s - ct)$

La validez de este algoritmo, es decir que produce un par de números s, t que cumple el teorema st , se demuestra por inducción sobre el número de llamadas recursivas de st . Primero, eso es finito porque cada nueva llamada recursiva tiene el segundo argumento menor que la anterior porque $resto(a, b) < b$. Si no hay ninguna llamada recursiva entonces $b = 0$, y en ese caso $(1, 0)$ es una respuesta correcta porque $1 \times a + 0 \times 0 = a = mcd(a, 0)$. Si hay una llamada recursiva y el total de llamadas recursivas es $n + 1$, entonces esa primera llamada recursiva tiene n . Si esa última contesta (s, t) entonces el anterior contesta $(t, s - ct)$ y la cuestión es si $ta + (s - ct)b = mcd(a, b)$. Ahora, $a = bc + r$, o sea $r = a - bc$, y por tanto, si toda invocación del algoritmo da resultado correcto cuando hay n llamadas recursivas, es $sb + tr = mcd(b, r)$, o sea $sb + t(a - bc) = mcd(a, b)$ (recuerde que $mcd(a, b) = mcd(b, r)$), o sea $ta + (s - tc)b = mcd(a, b)$. Así hemos visto que la correctitud del algoritmo cuando hay n recursiones implica su correctitud para $n + 1$ recursiones.

Así queda demostrada el algoritmo st y con eso el teorema st .

Es interesante notar otra demostración del teorema st . Dados a, b naturales, no ambos 0, sea

$$C = \{sa + tb : s, t \text{ enteros, } sa + tb > 0\}.$$

Sea m el elemento mínimo de C (C no es vacío porque contiene $a + b > 0$) y sea $g = mcd(a, b)$. Todo divisor común d de a, b divide a m porque consiste en la suma de dos múltiplos de d . Para saber que $m = g$, falta saber si es divisor común de a, b . Suponga que m no es factor de a . Entonces existen c, r tal que $a = cm + r$ con $0 < r < m$. Entonces es $r = a - cm = a - c(sa + tb)$ para algunos enteros s, t , o sea $r = (1 - cs)a - ctb$, dando que $r \in C$, pero m es el elemento mínimo de C . Entonces sí m es factor de a . Similarmente, m es factor de b . Entonces, por la propiedad que vimos después de la definición de mcd , $m = g$.

Nótese que cuando el algoritmo llega a $b = 0$ puede contestar $(1, n)$ para cualquier entero n porque $1 \times a + n \times 0 = a = mcd(a, 0)$, y que cada uno de estos da respuesta distinta al primer nivel. El ejercicio 13 afirma que así se consiguen todos los s, t que hay.

Ahora para demostrar el Lema de Euclides, supongamos que p no divide a a . Entonces, siendo p primo, es $mcd(p, a) = 1$, y entonces por el teorema st existen enteros s, t tales que $sp + ta = 1$. Multiplicando esta ecuación por b sale $spb + tab = b$, y ya que ab y por tanto tab es múltiplo de p , b es la suma de dos múltiplos de p y entonces b es múltiplo de p . Así vemos que si p no divide a a tiene que dividir a b .

Con esto, la demostración del teorema fundamental está completa.

Ejercicios 2.5

- (1) Calcule $st(a, b)$ para $(a, b) = (12, 16), (4, 10), (9, 24), (14, 65), (28, 130), (108, 188), (3214, 303)$ y verifique cada resultado. No es necesario mostrar los detalles del cociente y resto.
- (2) ¿Para cuáles valores de k se cumple que, para todo par de naturales a, b , $a) mcd(a, b) = mcd(a, b + ka); b) mcd(a, b) = mcd(a, a + kb)$.
- (2) Escriba en notación algorítmica un procedimiento recursivo que recibe como argumentos los naturales a y p , y encuentra el máximo natural n y un natural k tales que $a = k \cdot p^n$.
- (2) Escriba un procedimiento para encontrar la descomposición en primos de cualquier natural a . El procedimiento debe contestar una lista $\{(p_1, n_1), \dots, (p_k, n_k)\}$ tal que $a = p_1^{n_1} \cdots p_k^{n_k}$ con p_1, \dots, p_k primos y $p_1 < \cdots < p_k$.

5. (3) ¿Qué pasa con $st(2a, 2b)$? ¿ $st(3a, 3b)$? ¿ $st(ka, kb)$? ¿Por qué?
6. (1) ¿Verdadera o falsa? a^2 divide a b^2 si y solo si a divide a b .
7. (1) Demuestre que $mcd(ka, b) = mcd(a, b)$ si k, b son coprimos.
8. (2) Demostrar: Si a y b no son ambos 0, $\frac{ab}{mcd(a, b)}$ es el mínimo común múltiplo de a y b .
9. (2) ¿Verdadera o falsa? Si r es racional, cada a_i es entero, y $r^n + a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + \dots + a_0 = 0$ entonces r es entero.
10. (1) Demostrar: Si a y b son coprimos y ab es un cuadrado entonces a y b son cuadrados.
11. (3) Caracterice todos los triángulos rectos cuyos lados son números naturales. Sugerencia: demuestre que si $x^2 + y^2 = z^2$ y x, y, z no tienen factor común entre los tres, entonces x, y no pueden ser ambos impares (demuestre y aplique el lema: $resto(a^2, 4)$ es siempre 0 o 1.) Entonces sea $y = 2w$ y considere que $w^2 = \frac{z^2 - x^2}{4}$. Aplique el ejercicio 10 para concluir que para algunos naturales m, n , coprimos y no ambos impares, es $z = m^2 + n^2$, $x = m^2 - n^2$, y $y = 2mn$.
12. (2) Sean $a = p_1^{n_1} \dots p_k^{n_k}$ y $b = p_1^{m_1} \dots p_k^{m_k}$ dos enteros positivos (algunos m_i y/o n_i podrían ser 0). Muestre que:
 - a) (2) $a|b$ si y sólo si $\forall i \in \{1, \dots, k\}, n_i \leq m_i$.
 - b) (2) $mcd(a, b) = p_1^{\min(n_1, m_1)} \dots p_k^{\min(n_k, m_k)}$.
13. (3) Demuestre que cualquier s, t que cumple $sa + tb = mcd(a, b)$ se consigue por el algoritmo extendido de Euclides modificado para contestar $(1, n)$ en el último nivel.

6. Aritmética Modular

6.1. Módulo. Si uno trata de calcular $resto((n-1)!, n)$ para n chico no hay problema. Pero para $n = 10000$ ni Maple puede contener un número tan grande como $9999!$. Pero $resto(ab, n) = resto(a \times resto(b, n), n)$ (veremos enseguida por qué). Entonces, cuando llegamos a $7! = 5040$, calculamos $resto(8 \times 5040, 10000) = 320$ y sabemos que $resto(8! \times 9 \times 10 \times \dots \times 999, 10000) = resto(320 \times 9 \times \dots \times 999, 10000)$, y que el último es $resto(resto(320 \times 9, 10000) \times 10 \times \dots, 9999, 10000) = resto(2880 \times 10 \times \dots, 9999, 10000)$. Y ahora calculamos $resto(10 \times 2880, 10000) = 8800$ y el problema se reduce a $resto(8800 \times 11 \times \dots \times 9999, 10000)$, lo cual con $resto(8800 \times 11, 10000) = 6800$ se reduce a $resto(6800 \times 12 \times \dots \times 999, 10000)$. Claro, todavía faltan unas 9988 iteraciones, lo que no quisiéramos hacer a mano, pero Maple puede hacer esas iteraciones en una fracción de segundo porque sólo involucra números de 5 dígitos o menos. El programa en Maple es

```
prod := 1 : for m from 2 to n - 1 do prod := (m * prod) mod n od
```

que deja el resultado en *prod*. Probando esto, va a ver que para $n = 10000$ hasta 10006 da 0, y para $n = 10007$ da 10006. Lo dejo para ponderar este hecho. Después vamos a ver qué significa.

Un ejemplo en que solo interesa el resto es cuando trabajamos con ángulos. Si giro a la derecha 200 grados, y luego 120 más, y luego 70 más, he hecho un giro de $resto(390, 360) = 30$ grados. Decimos que “390 es equivalente a 30 módulo 360”. Es una situación en que 390 es indistinguible de 30.

En el cuentakilómetros de un auto, 1300000 es indistinguible de 300000. En una computadora con aritmética de 16 bits, $65546 = 2^{16} + 10$ es indistinguible de 10. Pero sí sabemos que por más adiciones, sustracciones y productos que hacemos, el número que resulta es exactamente equivalente modulo 2^{16} al resultado completo.

La palabra *módulo* fue usada por primera vez por Gauss, en *Disquisiciones*. Viene de un verbo latino que significa “medir”, pero Gauss seguía a Euclides quien, como dijimos, disfrazaba la teoría de números como geometría, y decía que un segmento era una *medida* de otro si el otro era varias copias del primero, es decir, su longitud era múltiplo exacto del primero.

La modelación matemática de esas situaciones en que sólo percibimos el resto módulo n es uno de los temas claves de la matemática discreta.

6.2. Aritmética modular: una relación de congruencia.

DEFINICIÓN 2.12. Se dice de dos números enteros a, b y un número natural $n > 1$ que $a \equiv_n b$ si $a - b$ es un múltiplo de n .

$a \equiv_n b$ se lee “ a es congruente con b módulo n ”.

Eso es equivalente a decir que $\text{resto}(a, n) = \text{resto}(b, n)$ si tenemos cuidado de definir $\text{resto}(a, n)$ como el r del par único de enteros (c, r) con c entero y $0 \leq r < |n|$ tal que $a = cn + r$, que no está de acuerdo con algunas computadoras cuando a, b no son ambos naturales, así que mejor que adoptemos la definición dada. La relación \equiv_n es una *relación de equivalencia*, concepto que vamos a encontrar muchas veces, y que definimos ahora:

DEFINICIÓN 2.13. Una *relación de equivalencia* es una relación binaria sobre un conjunto C tal que

1. (Reflexiva) Para todo $a \in C$, $a \equiv a$.
2. (Simétrica) Para todo $a, b \in C$, $a \equiv b$ implica que $b \equiv a$.
3. (Transitiva) Para todo $a, b, c \in C$, $a \equiv b$ y $b \equiv c$ implica que $a \equiv c$.

La idea de una relación de equivalencia es que aunque los elementos de C son en realidad distintos, por alguna razón no podemos distinguir entre algunos; nos parecen iguales. Por ejemplo, un grupo en que hay algunos pares de gemelos idénticos: su madre los puede distinguir pero ella no está. O algunos bloques de juguete: algunos cubos, esferas, pirámides y conos de varios colores. En una fotografía en blanco y negro no podemos distinguir un cono rojo de un cono azul. Claro está que si no puedo distinguir entre a y b es igualmente válido decir que no puedo distinguir entre b y a . Y si no puedo distinguir ni entre a y b ni entre b y c , es razonable suponer que no voy a distinguir entre a y c .

Llamamos $[a]$ al conjunto de todos los elementos de C equivalentes a a : $[a] = \{b \in C : b \equiv a\}$. Es claro que $C = \bigcup_{a \in C} [a]$. Además, para cada $a, b \in C$, si $a \equiv b$ entonces $[a] = [b]$, sino $[a] \cap [b]$ es vacío. Así los $[a]$ forman una *partición* de C (ver ejercicio 1).

Si hubiéramos definido $a \equiv_n b$ cuando $\text{resto}(a, n) = \text{resto}(b, n)$ sería obvio que es una relación de equivalencia, porque siempre lo es cuando se define en términos de una propiedad compartida entre a y b . Para demostrar que las dos definiciones son equivalentes, necesitaríamos el teorema de la división para los enteros; por eso preferimos demostrarlo directamente de la definición dada:

1. $a \equiv_n a$ porque $a - a = 0 = 0 \times n$.
2. Si $a \equiv_n b$ entonces $a - b = kn$ para algún k entero; entonces $b - a = -kn$ así que $b \equiv_n a$.
3. Suponga que $a \equiv_n b$ y $b \equiv_n c$. Entonces para algún k_1, k_2 es $a - b = k_1n$ y $b - c = k_2n$.
Entonces $a - c = a - b + b - c = k_1n + k_2n$, un múltiplo de n .

La relación \equiv_n es también una relación de *congruencia* respecto de $+$ y también respecto de \times , concepto que vamos a reencontrar varias veces:

DEFINICIÓN 2.14. Sea \equiv una relación de equivalencia sobre un conjunto C y sea \circ una operación binaria sobre C . Se dice que \equiv es una *relación de congruencia respecto de \circ* si para todos $a, b, c, d \in C$,

$$a \equiv b \text{ y } c \equiv d \text{ implica que } a \circ c \equiv b \circ d.$$

Veamos que \equiv_n es de congruencia respecto de $+$: $(a + c) - (b + d) = (a - b) + (c - d)$, la suma de dos múltiplos de n y por tanto un múltiplo de n .

Veamos que \equiv_n es una relación de congruencia respecto de \times : $ac - bd = a(c - d + d) - (b - a + a)d = a(c - d) + (b - a)d$, y ya que $c - d$ y $b - a$ son ambos múltiplos de n , $a(c - d) + (b - a)d$ también lo es.

Quiere decir, por ejemplo, que podemos hacer sumas y sustracciones y productos todo el día en una computadora cuya aritmética es módulo 2^k y el resultado va a ser exacto módulo 2^k .

Si vemos los enteros bajo el lente de \equiv_n , vemos solamente n objetos distintos, que conviene nombrar $0, 1, 2, \dots, n-1$. La clase de equivalencia de 0 , $[0]$, es todos los múltiplos de n . La clase de equivalencia de i , $0 \leq i < n$ es $\{cn + i : c \text{ entero}\}$. Veamos las tablas de adición y de multiplicación para $n = 7$ y $n = 8$:

+	0	1	2	3	4	5	6	×	0	1	2	3	4	5	6	
0	0	1	2	3	4	5	6	0	0	0	0	0	0	0	0	
1	1	2	3	4	5	6	0	1	0	1	2	3	4	5	6	
2	2	3	4	5	6	0	1	2	0	2	4	6	1	3	5	
3	3	4	5	6	0	1	2	3	0	3	6	2	5	1	4	\mathbb{Z}_7
4	4	5	6	0	1	2	3	4	0	4	1	5	2	6	3	
5	5	6	0	1	2	3	4	5	0	5	3	1	6	4	2	
6	6	0	1	2	3	4	5	6	0	6	5	4	3	2	1	

+	0	1	2	3	4	5	6	7	×	0	1	2	3	4	5	6	7	
0	0	1	2	3	4	5	6	7	0	0	0	0	0	0	0	0	0	
1	1	2	3	4	5	6	7	0	1	0	1	2	3	4	5	6	7	
2	2	3	4	5	6	7	0	1	2	0	2	4	6	0	2	4	6	
3	3	4	5	6	7	0	1	2	3	0	3	6	1	4	7	2	5	\mathbb{Z}_8
4	4	5	6	7	0	1	2	3	4	0	4	0	4	0	4	0	4	
5	5	6	7	0	1	2	3	4	5	0	5	2	7	4	1	6	3	
6	6	7	0	1	2	3	4	5	6	0	6	4	2	0	6	4	2	
7	7	0	1	2	3	4	5	6	7	0	7	6	5	4	3	2	1	

Los nombres $\mathbb{Z}_7, \mathbb{Z}_8$ son los que se acostumbran a usar. Es decir, \mathbb{Z}_n es el sistema formado por las clases de equivalencia de \equiv_n sobre \mathbb{Z} , en donde definimos $[a] + [b] = [a + b]$ y $[a] \times [b] = [a \times b]$.

Para facilitar la construcción de estas tablas nótese que, en la tabla de $+$, cada fila es una rotación de la fila anterior. Y en la tabla de \times , nótese que se empieza poniendo 0 en la primera columna de la fila a , y se procede a sumar a a cada columna (módulo n por supuesto), porque aquí, como en los enteros, se cumple $a(j + 1) = aj + a$.

6.3. Inversos y la resolución de congruencias. Nótese que la tabla de \times de \mathbb{Z}_7 tiene todos los números $1, 2, 3, 4, 5, 6$ en algún orden en las columnas de los números no 0. Eso significa que podemos dividir en \mathbb{Z}_7 , es decir resolver la ecuación $ax = b$ para cada a, b distintos de 0, porque podemos resolver $ax = 1$, (llamamos a^{-1} a este x) porque $b = (aa^{-1})b = a(a^{-1}b)$.

Pero en \mathbb{Z}_8 no podemos dividir cualquier no nulo por otro: por ejemplo, examinado la fila de 2 en la tabla no aparece 1; tampoco 3 o 5. Pero $2 \times 4 = 0$. Podríamos saber eso sin construir la tabla, porque sumando 2 cualquier número x de veces da un número par y por tanto no un $8k + 1$. Pero algunos tienen inversos: examinando la fila de 5, cada número de $1, 2, 3, 4, 5, 6, 7$ aparece una vez. De hecho, para cada a impar, $a^2 = 1$.

TEOREMA 2.15. *En \mathbb{Z}_n , un elemento a distinto de 0 tiene inverso si y sólo si n y a son coprimos. El inverso, cuando existe, es único.*

Demostración. Si $a = k_1p$ y $n = k_2p$ para algún primo p , entonces todo múltiplo de a tiene p como factor. Pero 1 más un múltiplo de p no es un múltiplo de p , y al decir que $\text{resto}(ma, n) = 1$ estamos diciendo que $ma = cn + 1$ para algún c , ma es múltiplo de p y $cn + 1$ no es múltiplo de p porque cn lo es.

Por el otro lado, sean a y n coprimos. Por el teorema st , existen enteros s y t tales que $sa + tn = 1$. Entonces $sa - 1$ es un múltiplo de n , o sea que $sa \equiv_n 1$. Existe x , $0 \leq x < n$, tal que

$s \equiv_n x$ porque sabemos que $\mathbb{Z} = \bigcup_{i=0}^{n-1} [i]$. Este x es inverso de a porque $xa \equiv_n sa$ siendo \equiv_n una

relación de congruencia respecto de \times . (Más simplemente, $x = \text{resto}(s, n)$, pero en el caso de s negativo descubrimos que no hemos dado oficialmente una definición del resto en ese caso.)

Para la unicidad, suponga que x e y son ambos inversos de a en \mathbb{Z}_n . Entonces $xa = 1$ (igualdad relativa a \mathbb{Z}_n) y $ya = 1$, o sea $xa = ya$ en \mathbb{Z}_n . Pero $ya = ay$. Multiplicando por x , $xax = xay$, o sea $(xa)y = (xa)x$, o sea $x = y$ en \mathbb{Z}_n . \square

Los siguientes corolarios son inmediatos:

COROLARIO 2.16. *En \mathbb{Z}_n , la ecuación $ax = b$ tiene solución si a y n son coprimos. La solución es única cuando existe.*

COROLARIO 2.17. *En \mathbb{Z}_p para un primo p , cualquier ecuación $ax = b$ admite solución única x mientras $a \neq 0$ en \mathbb{Z}_p .*

COROLARIO 2.18. *En \mathbb{Z}_p con p primo, si $ab = 0$ entonces o $a = 0$ o $b = 0$.*

Demostración. Si $b \neq 0$ multiplicar por b^{-1} (teorema) y sale $a = 0$.

6.4. Congruencias simultáneas. Sean m, n coprimos. Entonces siempre existe una solución x de las congruencias simultaneas $x \equiv_m a$ y $x \equiv_n b$. Por ejemplo, para $x \equiv_2 1$ y $x \equiv_3 2$ pruebo 0, 1, 2, 3, 4, 5 y veo que $x = 5$ cumple. De hecho, vamos a ver que podemos encontrar una solución probando los números desde 0 a $mn - 1$, pero hay un algoritmo mejor aplicando el algoritmo extendido de Euclides:

Una solución a la primera congruencia es a . Ahora buscamos un k que cumple $a + km \equiv_n b$, o sea un k tal que $km \equiv_n b - a$. Ya que m, n son coprimos, el incógnito k se puede encontrar por el corolario 2.17. El número $a + km$ cumple las dos congruencias. Podemos tomar k entre 0 y $n - 1$, y $km < nm$, y podemos suponer que $0 \leq a < m$, así que podemos tener $0 \leq a + km < nm$.

Para un algoritmo que saque la solución, falta sólo un algoritmo para el inverso de a en \mathbb{Z}_n cuando a, n son coprimos, pero por la demostración del teorema anterior vemos que el algoritmo extendido de Euclides hace eso.

Concluimos esta sección con un teorema que aclara el ejercicio 10, permitiendo demostrar que $f(n)$ va a engendrar todos los primos y sólo ellos.

TEOREMA 2.19. $(n - 1)! \equiv_n n - 1$ si y sólo si n es primo, salvo el caso $n = 4$.

Demostración. Supongamos primero que n no es primo. Entonces $n = mq$ con $m, q > 1$. Si $m \neq q$, siendo ambos menor que n , su producto, n , es factor de $(n - 1)!$, así que $(n - 1)! \equiv_n 0$. Si $n = m^2$, puesto que hemos excluido el caso $n = 4$, podemos asegurar que m y $2m$ son ambos menores que n y por tanto factores de $(n - 1)!$, que hace que por tanto su producto $2m^2$ también; luego $m^2 = n$ también, así que de nuevo $(n - 1)! \equiv_n 0$.

Supongamos que n es primo. Entonces cada elemento salvo 0 de \mathbb{Z}_n tiene inverso único. Los elementos no 0 de \mathbb{Z}_n son $1, \dots, n - 1$. 1 es su propio inverso y $n - 1$ (que es -1 en \mathbb{Z}_n) también. Ningún otro es su propio inverso, porque dado $a^2 = 1$, es $a^2 - 1 = 0$, luego $(a - 1)(a + 1) = 0$ y por corolario 2.18, o $a - 1$ o $a + 1$ es 0, es decir, a es 1 o -1 . Entonces, podemos aparear los números $2, \dots, n - 2$ cada uno con su propio inverso, y ya que el inverso es único no repetimos nada, y tenemos en \mathbb{Z}_n que $(n - 1)! = n - 1$. \square

Ejercicios 2.6

- (1) Sea \equiv una relación de equivalencia sobre un conjunto U . Demostrar:
 - $x \equiv y$ si y sólo si $[x] = [y]$.
 - Para todos $x, y \in U$, o $[x] = [y]$ o $[x] \cap [y]$ es vacío.
 - $\bigcup_{x \in U} [x] = U$.
- Sea \equiv una relación de congruencia respecto de una operación binaria \circ . Para dos subconjuntos V, W de U decimos $V \circ W = \{x \circ y : x \in V, y \in W\}$. Demuestre que cuando \equiv es \equiv_n , $y \circ$ es una de $\{+, \times\}$, $[x] \circ [y] = [x \circ y]$ para todos $x, y \in \mathbb{Z}$.
- Demuestre la ley distributiva para \mathbb{Z}_n .

4. Muestre las tablas de adición y multiplicación para \mathbb{Z}_{10} y \mathbb{Z}_{11} .
5. Dé un algoritmo para calcular a^{-1} en \mathbb{Z}_n cuando a y n son coprimos.
6. Dé un algoritmo que resuelve $ax = b$ en \mathbb{Z}_n cuando a y n son coprimos.
7. Resuelva los que se pueden: $15x = 4$ en \mathbb{Z}_{19} , $3x = 1$ en \mathbb{Z}_5 , $4x = 1$ en \mathbb{Z}_6 , $4x = 1$ en \mathbb{Z}_7 , $4x = 3$ en \mathbb{Z}_9 , $4x = 3$ en \mathbb{Z}_{23} .
8. Demuestre que en \mathbb{Z}_n , para todo a existe un x tal que $a + x = 0$. Este x se llama $-a$. Demuestre que $(-a)(-b) = ab$.
9. Si p, q son coprimos, si el par ordenado de enteros (a, b) designa el primer número natural x tal que $x \equiv_p a$ y $x \equiv_q b$,
 - a) ¿cuántos números hay en el conjunto $\{(a, b) : a, b \text{ entero}\}$?
 - b) ¿Quién es $(a, b) + (c, d)$?
 - c) ¿Quién es $(a, b)(c, d)$?
10. Demuestre que si $q_1, q_2, q_3 > 1$ son coprimos en pares, entonces dados números enteros a_1, a_2, a_3 existe x tal que $x \equiv_{q_i} a_i$ para $i = 1, 2, 3$.
11. Encuentre el primer natural x tal que $x \equiv_5 3$, $x \equiv_8 4$, $x \equiv_9 5$.
12. *Teorema Chino del Resto* Generalizar el resultado del ejercicio 10.
13. Encuentre el x natural más chico tal que
 - a) $x \equiv_5 1$ y $x \equiv_{13} 5$.
 - b) $x \equiv_{13} 5$ y $x \equiv_{17} 5$.
 - c) $x \equiv_{13} 5$ y $x \equiv_{16} 6$.
14. Para $q_1 = 3, q_2 = 4, q_3 = 5$, calcule la terna que representa cada número entre 0 y 59 ($= q_1 q_2 q_3 - 1$), pero siga esta sugerencia: 0 es claramente representado por $(0, 0, 0)$. Dada la representación de $n < 59$ es (a, b, c) , la de $n + 1$ se saca de $(a + 1, b + 1, c + 1)$.
15. Sobre números enteros, defina $m \equiv n$ cuando $|m| = |n|$. Verifique que es una relación de equivalencia. ¿Es una relación de congruencia respecto de $+$? ¿Respecto de \times ?
16. ¿Verdadero o falso? $ax \equiv_n b$ si y solo si b es un múltiplo de $\text{mcd}(a, n)$.
17. A la luz del teorema 2.19, analice el algoritmo del ejercicio 10 de la sección de números primos.

7. Grupos

En \mathbb{Z}_n podemos sumar y también restar, es decir, resolver la ecuación $a + x = b$, incluido el caso $b = 0$. Si n es un primo p , como hemos visto, podemos decir lo correspondiente para la multiplicación: hay un elemento neutro, 1, tal que $1 \times a = a \times 1 = a$ para todo a en \mathbb{Z}_p , y para todo $a \neq 0$ en \mathbb{Z}_p existe $a^{-1} \neq 0$ tal que $a \times a^{-1} = a^{-1} \times a = 1$. Tan frecuentemente ocurre esta situación con un conjunto (posiblemente infinito) y una operación binaria que sus propiedades forman una especialidad que se llama la *teoría de grupos*.

DEFINICIÓN 2.20. Un *grupo* es un conjunto G con una operación binaria \circ que cumplen las siguientes propiedades:

1. $a \circ b \in G$ para todo a, b en G .
2. La ley asociativa: para todo $a, b, c \in G$, $(a \circ b) \circ c = a \circ (b \circ c)$.
3. Hay un elemento *neutro* e en G , es decir, para todo a en G , $a \circ e = e \circ a = a$.
4. Cada elemento a en G tiene un *inverso* a^{-1} en G tal que $a \circ a^{-1} = a^{-1} \circ a = e$.

En el ejemplo de \mathbb{Z}_p , para adición \circ es $+$, e es 0 y " a^{-1} " es $-a$; para multiplicación de elementos distintos de 0 \circ es \times , e es 1 y a^{-1} es a^{-1} . (Para cualquier duda si $+$ y \times hereden en \mathbb{Z}_p las leyes de asociatividad que tienen en \mathbb{Z} , ver el ejercicio 10.)

Los números naturales no forman un grupo ni con adición ni con el producto, por falta del inverso. Los números racionales forman un grupo respecto de $+$ y los racionales no 0 forman un grupo respecto de \times .

Fíjese que en la definición de grupo no hemos insistido en la conmutatividad de la operación \circ : puede ser que un grupo tenga elementos a y b tal que $a \circ b \neq b \circ a$. Un ejemplo es el grupo de las seis permutaciones de $\{1, 2, 3\}$ (una permutación es una función biyectiva de $\{1, \dots, n\}$ en sí mismo): si f lleva $(1, 2, 3)$ a $(2, 3, 1)$ y g lleva $(1, 2, 3)$ a $(2, 1, 3)$ entonces $f \circ g$ lleva $(1, 2, 3)$ a $(2, 3, 1)$ ($(f \circ g)(1) = f(g(1)) = f(2) = 3$, etc.), pero $g \circ f$ lo lleva a $(3, 1, 2)$.

Pero aquí vamos a encontrar sólo grupos conmutativos.

Un *subgrupo* de un grupo G con operación \circ es un subconjunto H de G tal que H forma un grupo respecto de \circ . Es inmediato que el neutro y los inversos en H son los mismos que en G .

Por ejemplo, las potencias de cualquier elemento de $\mathbb{Z}_p - \{0\}$ forman un subgrupo de $\mathbb{Z}_p - \{0\}$: en \mathbb{Z}_7 las potencias de 2 son 2, 4 y 1 (¿cuál es la tabla de multiplicación?)

Verifiquemos que las potencias de a en \mathbb{Z}_p forman un subgrupo: $a^i a^j = a^{i+j}$, así que es cerrado bajo el producto. Para descubrir que hay un $a^k = 1$, examine a^2, a^3, \dots . Tiene que haber una repetición, porque es un conjunto finito, entonces para algún i, j con $i < j$ tiene que ser $a^i = a^j$. Esto nos da $a^i a^{j-i} = a^j = a^i$, así que por la ley de cancelación en grupos, $a^{j-i} = 1$, así que el conjunto aspirante contiene 1: con $k = j - i$ es $a^k = 1$. De eso se ve que el grupo es exactamente $\{a * i : 0 \leq i < k\}$ si definimos $a^0 = 1$. Entonces si $0 \leq i < k$, el inverso de a^i es a^{k-i} . Así vemos que las potencias de a forman un subgrupo.

Decimos *orden* de un grupo G el número de elementos de G . Para un elemento a de G , decimos *orden de a* al orden del grupo de potencias de a . Para subconjuntos S, T de G , decimos $ST = \{st : s \in S, t \in T\}$, y para un elemento a de G , aS (o Sa) = $\{a\}S$ (o $S\{a\}$).

Cuando un grupo con operación \circ está bajo discusión se suele cortar “ $a \circ b$ ” a ab .

TEOREMA 2.21. *Para un subgrupo H de un grupo G la relación $b \in aH$ es una relación de equivalencia*

Demostración. $a \in aH$ porque el neutro está en H . Si $b \in aH$ quiere decir que $b = ah$ para algún $h \in H$. Multiplicando por h^{-1} a la derecha es $bh^{-1} = a$, así que $a \in bH$. Si $b \in aH$ y $c \in bH$ entonces para algún $h_1, h_2 \in H$ es $b = ah_1$ y $c = bh_2$, así que $c = ah_1h_2$; siendo $h_1h_2 \in H$, $c \in aH$. \square

Notemos que la clase de equivalencia de a para la relación antes definida es $[a] = aH$.

TEOREMA 2.22. *Para un grupo finito, el orden de cualquier subgrupo divide al orden del grupo.*

Demostración. Si H tiene m elementos entonces cualquier aH tiene m elementos, porque si $ah_1 = ah_2$, entonces multiplicando por a^{-1} resulta $h_1 = h_2$. Ya que las clases de equivalencia forman una partición de G , G es la unión de algún número k subconjuntos disjuntos que tienen m por elementos cada uno; por tanto el orden de G es km , es decir k por el orden de H . \square

En el caso del grupo multiplicativo de \mathbb{Z}_p , el orden de G es $p - 1$. Entonces el orden multiplicativo de cualquier elemento es factor de $p - 1$. Por ejemplo, en \mathbb{Z}_{11} sólo tenemos elementos de orden multiplicativo 1, 2, 5 y 10.

La relación de equivalencia definida por $b \in aH$ también es una relación de congruencia respecto de la operación del grupo cuando el grupo es conmutativo: si $a = bh_1$ y $c = dh_2$ con $h_1, h_2 \in H$, entonces $ac = bh_1dh_2 = bdh_1h_2$ por conmutatividad (y asociatividad) así que ac es equivalente a bd y es una relación de congruencia. Pero conmutatividad no es una condición necesaria.

Cuando la relación es de congruencia, las clases de equivalencia forman un grupo respecto de la operación $[a][b] = [ab]$, que se llama *grupo cociente de G modulo H* y se escribe G/H ; en ese caso H se llama *subgrupo normal*. El neutro de G/H es el mismo conjunto $H = [e]$ donde e es el neutro. Asociatividad es heredada del grupo G . El inverso de $[a]$ es $[a^{-1}]$.

Cuando un grupo de orden n tiene elemento de orden n se llama *cíclico*. Un grupo cíclico de orden n se comporta como el grupo aditivo de \mathbb{Z}_n , porque $a^i a^j = a^{(i+j) \bmod n}$. Se dice que tal grupo es *isomorfo* a \mathbb{Z}_n :

DEFINICIÓN 2.23. Un grupo G se dice *isomorfo* a un grupo H si existe una función biyectiva f de G en H tal que $f(a)f(b) = f(ab)$ (donde el lado izquierdo se refiere a la operación de H y el de la derecha a la de G) para todos a, b en G .

La relación de ser isomorfos tiene las propiedades reflexiva, simétrica y transitiva, pero tenemos que abstenernos de declararla relación de equivalencia porque no existe un conjunto de todos los grupos (ver la *paradoja de Russell* en el capítulo 7.)

En el caso de un grupo cíclico de orden n con un elemento a de orden n , la función f es $f(a^i) = i$ (definimos $a^0 = 1$).

Ejercicios 2.7

1. En el grupo de $+$ de los enteros, sea H el grupo de los múltiplos de n .
 - a) Verifique que es un subgrupo.
 - b) Defina la relación de equivalencia engendrada por este subgrupo (es como se hizo en la teoría, sólo cambiando $+$ por \circ).
 - c) ¿Es esta relación de equivalencia una relación de congruencia respecto de $+$? ¿La respuesta depende de n ?
 - d) ¿El grupo cociente es isomorfo a algún grupo conocido?
2. Para un grupo G , $[a, b]$ denota el subgrupo más chico de G que contenga a y b .
 - a) En el grupo de los enteros bajo $+$, ¿cómo son $[2, 4], [9, 12], [3, 5], [17, 19], [30, 36]$?
 - b) Caracterice $[a, b]$ para los enteros a, b .
3. (4) Para cada inciso encuentre el a descrito en el grupo multiplicativo del \mathbb{Z}_p dado, muestre su grupo cociente, y muestre su isomorfismo con un grupo conocido.
 - a) Sea a un elemento de orden 10 en \mathbb{Z}_{31} .
 - b) Para un b de orden 16 en \mathbb{Z}_{17} , sea $a = b^4$.
 - c) Para un b de orden 18 en \mathbb{Z}_{19} , sea $a = b^3$.
4. ¿Los números $a + b\sqrt{2}$ con a, b enteros forman un grupo respecto de $+$? ¿Los no nulos respecto de \times ?
5. ¿Los números $a + b\sqrt{2}$ con a, b racionales forman un grupo respecto de $+$? ¿Los no nulos respecto de \times ?
6. Demuestre que en un grupo conmutativo G , el conjunto $\{a^k : a \in G\}$ es un subgrupo de G para cada k . Para $k = 3$, identifique este subgrupo del grupo aditivo de \mathbb{Z}_7 , y también para los grupos multiplicativos de los no nulos de \mathbb{Z}_{11} y \mathbb{Z}_{13} .
7. Sea G un grupo con neutro 1, y a un elemento de orden n en G . Demuestre que:
 - $n = \min \{k > 0 : a^k = 1\}$.
 - Para t entero positivo, $a^t = 1$ si y sólo si t es múltiplo de n .
8. Demostrar que para cualquier a de un grupo, el orden de a^i divide al orden de a .
9. Sea G un grupo y $a \in G$. Sea n el orden de a . Si m es factor de n entonces G tiene un elemento de orden m .
10. Demuestre que si \equiv es una relación de equivalencia sobre un conjunto C , y es de congruencia respecto de la operación binaria \circ sobre C , entonces podemos extender \circ a las clases de equivalencia $[a]$ de \equiv , definiendo $[a][b] = [a \circ b]$, y que la nueva operación es asociativa.

8. Cuerpos

\mathbb{Z}_p con p primo es un *cuerpo*:

DEFINICIÓN. Un *cuerpo* es un conjunto con dos operaciones binarias $+$ y \times tales que forma un grupo conmutativo con $+$, y, llamando 0 al neutro de ese grupo, los elementos distintos de 0 forman un grupo conmutativo respecto de \times , y en que además se cumple la ley distributiva $a \times (b + c) = a \times b + a \times c$.

Se puede deducir en cualquier cuerpo que $a \times 0 = 0$: $a = a \times 1 = a \times (1 + 0) = a \times 1 + a \times 0 = a + a \times 0$. Ya que $a = a + a \times 0$, $a \times 0$ debe ser 0. En forma semejante, $0 \times a = 0$. Y de allí que $(-a) \times b = -(a \times b)$: $((-a) \times b) + a \times b = (-a + a) \times b = 0 \times b = 0$, y por tanto $(-a \times b) = -(a \times b)$.

Los números reales forman un cuerpo. También los números racionales y los números complejos. Los enteros, no: forman un grupo respecto de $+$ pero los no nulos, salvo 1, no tienen inversos.

8.1. Polinomios sobre cuerpos. Todos entendemos una expresión como $3x^3 - 2x^2 + x + 5$ como una cierta función f de los reales a los reales y que podemos calcular el valor de $f(r)$ para cualquier real r dado reemplazando r por x y haciendo las operaciones indicadas.

Es claro que podemos entender el mismo concepto donde los x , y los coeficientes, vienen del cuerpo de los racionales o del de los complejos o del \mathbb{Z}_p o de cualquier cuerpo, finito o infinito.

En cuerpos raros como los que veremos, hay algunas propiedades raras de polinomios. Por ejemplo, en un cuerpo finito vamos a ver que toda función del cuerpo en sí mismo es expresable como un polinomio. Sabemos que el cuerpo de los racionales no tiene $\sqrt{2}$. Algunos cuerpos finitos tienen y otros no: \mathbb{Z}_3 no tiene: allí $2^2 = 1^2 = 1$, $0^2 = 0$. Tampoco \mathbb{Z}_5 : $2^2 = 3^2 = 4$, $4^2 = 1$. \mathbb{Z}_7 sí tiene: $3^2 = 2$. Pero \mathbb{Z}_5 tiene una cosa que \mathbb{Z}_7 puede envidiar: una $\sqrt{-1}$.

Decir que un cuerpo no tiene \sqrt{a} es equivalente a decir que el polinomio $x^2 - a$ no tiene raíz en el cuerpo. Si un cuerpo C no tiene raíz del polinomio $x^2 - a$ podemos inventar una, pronunciando el decreto " $s^2 = a$ " y formando el conjunto $\{a + bs : a, b \in C\}$, que resulta ser otro cuerpo (ver ejercicio). Esto es lo que hacemos a los reales para formar el cuerpo de los complejos: declaramos por fiat que $i^2 = -1$ y formamos el conjunto $\{a + bi : a, b \text{ real}\}$. Pero podemos tomar el cuerpo de los racionales, declarar por fiat que $s^2 = 2$, y formar el cuerpo $\{a + bs : a, b \text{ racional}\}$, que es más que los racionales y menos que los reales. O podemos tomar $\{a + bs : a, b \in \mathbb{Z}_3\}$, un cuerpo de 9 elementos. Algunas de estas cosas vamos a explorar en los ejercicios.

Pero por ahora nos vamos a concentrar en algunas propiedades compartidas por los polinomios sobre todos los cuerpos.

Tenemos que definir formalmente la adición $p + q$ y la multiplicación $p \times q$ de dos polinomios p y q . Para esto tenemos que ver un polinomio como una sucesión de elementos del cuerpo. Conceptualmente esto es una sucesión finita $[a_n, a_{n-1}, \dots, a_0]$ con $a_n \neq 0$, que representa el polinomio $a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$, o sea $\sum_{i=0}^n a_i x^i$. Hay un inconveniente aquí: en que así estamos obligados a representar un polinomio como $x^{10} - 1$ mediante $[1, 0, 0, 0, 0, 0, 0, 0, 0, 1]$, y desde ese punto de vista sería mejor representarlo como una sucesión de pares ordenados $[[a_1, p_1], \dots, [a_k, p_k]]$, representando el polinomio $\sum_{i=1}^k a_i x^{p_i}$, pero las definiciones son más fáciles de hacer con la primera representación, y aún más fáciles si se obliga a incluir todos los 0 a la izquierda:

DEFINICIÓN 2.24. Un *polinomio* p sobre un cuerpo C es una sucesión infinita p_0, p_1, \dots , o sea una función de los naturales al cuerpo, en que sólo un número finito de los p_i son distintos de 0, es decir, o $p_i = 0$ para todo i o existe un número g , que llamamos el *grado* de p , tal que $p_g \neq 0$ y $p_i = 0$ para todo $i > g$. El *valor* de p para un elemento a de C es $\sum_{i=0}^{\infty} p_i a^i = \sum_{i=0}^g p_i a^i$. La *suma* $p + q$ de dos polinomios p y q se define por $(p + q)_i = p_i + q_i$. El *producto* $p \times q$ se define por $(p \times q)_i = \sum_{j+k=i} p_j q_k$.

Dejamos al lector como ejercicios que para todo $a \in C$, $(p + q)(a) = p(a) + q(a)$ y $(p \times q)(a) = p(a) \times q(a)$.

NOTACIÓN. Para $a \in C$, por el polinomio a queremos decir el polinomio p tal que $p_0 = a$ y $p_i = 0$ para todo $i > 0$. Para dos polinomios p y q , se dice $p < q$ si o $p = 0$ y $q \neq 0$ o ninguno de los dos es 0 y el grado de p es menor que el grado de q . Por el símbolo x^n queremos decir el polinomio p tal que $p_n = 1$ y $p_i = 0$ para todo $i \neq n$. (Entonces $x \times p$ designa el polinomio q tal que $q_0 = 0$ y $q_{i+1} = p_i$ para todo i .)

TEOREMA 2.25. (de la división para polinomios) *Dados dos polinomios p, q sobre un cuerpo C con $q \neq 0$, existen dos polinomios c, r sobre C con $r < q$ tal que $p = c \times q + r$.*

COMENTARIO. El par c, r es único en cierto sentido, pero técnicamente no lo es si C es finito por ser $x^m = x^{\text{resto}(m, n-1)}$ donde $n = |C|$. Por ahora sólo necesitamos la existencia.

Demostración. Consiste en demostrar que el siguiente algoritmo $D(p, q)$ produce el par enunciado:

$D(p, q) =$
si ($p < q$) ($0, p$)
sino
si (p y q tienen el mismo grado g) ($p_g/q_g, p - (p_g/q_g) \times q$)
sino
 $(c, r) := D(p, x \times q)$
 $(c1, r1) := D(r, q)$
 $(x \times c + c1, r1)$

Cuando $p < q$ es claro que $p = 0 \times q + p$ y $p < q$, así que la respuesta $(0, p)$ es correcta. Si ambos tienen grado g , claramente $p = (p_g/q_g) \times q + p - (p_g/q_g) \times q$ y entonces la correctitud de la repuesta depende solamente de que $p - (p_g/q_g) \times q < q$. Ahora $q = q_g x^g + \dots + q_0$, entonces $(p_g/q_g) \times q = p_g x^g + \dots + p_g q_0$, así que restando esto de p deja un polinomio de grado menor que g o 0; la respuesta es correcta. En el tercer caso, el grado de q es menor que el grado de p . Procedemos por inducción sobre esa diferencia en grados. Hemos visto que cuando es 0, el resultado es correcto. Suponga que el resultado es correcto cuando esa diferencia es n , y suponga que el grado de p excede el de q por $n + 1$. Ahora el grado de $x \times q$ es 1 más que el grado de q , por tanto la diferencia entre el grado de p y el de $x \times q$ es n ; por tanto (c, r) es correcto, es decir que $p = c \times (x \times q) + r$ con $r < x \times q$. Ahora, o $r < q$ o r tiene el mismo grado de q , y en cualquiera de los dos casos $(c1, r1)$ es correcto, lo que quiere decir que $r = c1 \times q + r1$ con $r1 < q$. Entonces $p = c \times (x \times q) + c1 \times q + r1 = (x \times c) \times q + c1 \times q + r1 = (x \times c + c1) \times q + r1$ con $r1 < q$, así que la respuesta $(x \times c + c1, r1)$ es correcta. \square

Un elemento a del cuerpo C es una raíz del polinomio p sobre C si $p(a) = 0$.

TEOREMA 2.26. *Un polinomio de grado n sobre un cuerpo no tiene más que n raíces en el cuerpo.*

Demostración. Por inducción sobre n . Un polinomio de grado 0 claramente no tiene ninguna raíz, así que se cumple para $n = 0$. Suponga que se cumple para n , y sea p un polinomio de grado $n + 1$. Si p no tiene ninguna raíz en C , entonces está cumpliendo. Suponga entonces que $p(a) = 0$. Según el teorema de la división, existen polinomios c, r con $r < x - a$ tales que $p = c \times (x - a) + r$. Para que $r < x - a$, r es algún polinomio b con $b \in C$. Pero $x - a = 0$ cuando $x = a$, así que $p(a) = b$; luego $b = 0$. El grado de c es n ; entonces, por la hipótesis de inducción, c no tiene más que n raíces. Pero cualquier raíz $d \neq a$ de p tiene que ser raíz de c porque $p(d) = c(d) \times (d - a)$, y ya que $d - a \neq 0$, $c(d) = 0$. Entonces el máximo número de raíces que puede tener p es $n + 1$, las a lo sumo n raíces de c junto con a . \square

8.2. Existencia de elementos primitivos.

TEOREMA 2.27. *Todo cuerpo finito tiene un elemento primitivo.*

Para la demostración de esto, necesitamos

LEMA 2.28. *En cualquier cuerpo finito, si hay un elemento de orden multiplicativo m y otro de orden multiplicativo n , entonces hay un elemento de orden multiplicativo $\text{mcm}(m, n)$.*

Cómo demostrar el Teorema a partir del Lema: Sea n el orden del cuerpo. Entonces el grupo multiplicativo tiene orden $n - 1$. Sea m el máximo orden multiplicativo entre los elementos no 0. Entonces el orden r de cualquier elemento es factor de m porque no puede ser mayor que el máximo m , pero si $r < m$ y si r no dividiere a m , entonces $\text{mcm}(m, r) > m$, pero, según Lema

2.28, habría un elemento de ese orden, lo que no es posible porque m es máximo. Entonces para cada elemento x distinto de 0, $x^m - 1 = 0$. Así, el polinomio $x^m - 1$ tiene $n - 1$ raíces. Pero un polinomio de grado m no puede tener más que m raíces, y por tanto $n - 1 \leq m$. Ningún elemento puede tener orden mayor que $n - 1$, así que $m = n - 1$. \square

Para demostrar el Lema 2.28, necesitamos unos lemas sencillos:

LEMA 2.29. *Si un grupo tiene un elemento de orden n y m es factor de n entonces existe un elemento de orden m .*

Demostración. Sea a de orden $n = mk$. Entonces $(a^k)^m = 1$, y para $1 \leq j < m$, siendo $kj < km = n$, $(a^k)^j \neq 1$. \square

LEMA 2.30. *En un grupo finito conmutativo, si elementos a, b tienen órdenes m, n respectivamente y m, n son coprimos entonces el orden de ab es mn .*

Demostración. Sea $(ab)^i = 1$. Por la conmutatividad, $a^i b^i = 1$ y por tanto $a^i = b^{-i}$, o sea, $a^i = (b^{-1})^i$. Siendo a^i un miembro del subgrupo de potencias de a , y siendo m el orden de ese subgrupo, el orden de a^i es divisor de m . Pero a^i también es miembro del subgrupo de potencias de b , cuyo orden es n , y por tanto el orden de a^i también es divisor de n . Pero el máximo común divisor de m y n es 1, por tanto el orden de a^i es 1, por tanto $a^i = 1$. En forma exactamente igual, $b^i = 1$. Entonces i es múltiplo de m y de n , pero mn es el menor común múltiplo de m y n , y $(ab)^{mn} = a^{mn} b^{mn} = (a^m)^n (b^n)^m = 1$. Entonces el orden de ab es mn . \square

LEMA 2.31. *En un grupo finito conmutativo, sean m_1, \dots, m_k los órdenes de los elementos a_1, \dots, a_k respectivamente, con m_i coprimo con m_j siempre que $i \neq j$. Entonces el orden de $a_1 \cdots a_k$ es $m_1 \cdots m_k$.*

Demostración. Por inducción sobre k . Es obvio que se cumple para $k = 1$. Sea k cualquier número para el cual se cumple, y sean a_1, \dots, a_{k+1} y m_1, \dots, m_{k+1} como en la hipótesis. Entonces el orden de $a_1 \cdots a_k$ es $m_1 \cdots m_k$. Por el Lema 2.30, como m_{k+1} es coprimo con $m_1 \cdots m_k$, el orden de $(a_1 \cdots a_k) a_{k+1}$ es $(m_1 \cdots m_k) m_{k+1}$. \square

Demostración del Lema 2.28. Sean a y b elementos del cuerpo con órdenes m, n respectivamente. Sea p_1, \dots, p_k primos tales que

$$m = p_1^{e_1} \cdots p_k^{e_k} \text{ y } n = p_1^{f_1} \cdots p_k^{f_k}.$$

(Por ejemplo, si $m = 3 \times 5$ y $n = 25 \times 7$ entonces $m = 3^1 \times 5^1 \times 7^0$ y $n = 3^0 \times 5^2 \times 7^1$.) Sea para cada i , $g_i = \max(e_i, f_i)$. Es claro que

$$mcm(m, n) = p_1^{g_1} \cdots p_k^{g_k}.$$

Cada $p_i^{g_i}$ divide o a m o a n , y entonces, por el Lema 2.29, para cada i existe un elemento c_i en el cuerpo con orden $p_i^{g_i}$. Por el Lema 2.31, entonces $c_1 \cdots c_k$ tiene orden $p_1^{g_1} \cdots p_k^{g_k}$, que como hemos observado es $mcm(m, n)$. \square

Ejercicios 2.8

1. (1) Encuentre un primo p y un número que no tiene raíz cuadrada en \mathbb{Z}_p .
2. (2) Encuentre un primo $p > 2$ tal que todo miembro de \mathbb{Z}_p tiene raíz cuadrada, o diga por qué no se lo puede encontrar.
3. (1) Encuentre un primo p tal que el polinomio $x^2 + x + 1$ tiene raíz en \mathbb{Z}_p , y otro primo que no.
4. (2) ¿Vale la fórmula cuadrática para las raíces de un polinomio de segundo grado en \mathbb{Z}_p , si p es primo?
5. (1) Saque cociente y resto en \mathbb{Z}_{71} de $x^4 + x^2 + 1$ dividido $x^2 + 1$.
6. (2) Saque cociente y resto en \mathbb{Z}_{11} y en \mathbb{Z}_{13} de $x^{10} + 10$ dividido $x^2 + x + 2$.
7. (1) En \mathbb{Z}_p , con p primo, ¿cuántos elementos tienen raíz cuadrada?

8. (5) Investigue las raíces cúbicas en \mathbb{Z}_p , p primo:
 - a) ¿Cuáles elementos tienen raíz cúbica en $\mathbb{Z}_5, \mathbb{Z}_7, \mathbb{Z}_{11}, \mathbb{Z}_{13}, \mathbb{Z}_{17}, \mathbb{Z}_{19}$?
 - b) ¿Un cuerpo puede tener exactamente 2 raíces cúbicas de 1?
 - c) Demuestre que si un cuerpo tiene tres raíces cúbicas de 1 entonces cualquier elemento con raíz cúbica tiene tres.
 - d) Demuestre si algún elemento tiene más que una raíz cúbica entonces hay un $a \neq 1$ tal que $a^3 = 1$.
 - e) ¿Qué pasa con raíces cúbicas en \mathbb{Z}_p , p primo, cuando $p - 1$ es divisible por 3, y cuando no es divisible por 3? Demuestrelo.
9. (3) ¿Qué concluye, del resultado del problema 6, de la existencia de raíces de $x^2 + x + 2$ en \mathbb{Z}_{11} y \mathbb{Z}_{13} ? Aplicando la fórmula cuadrática, encuentre las dos raíces en el cuerpo que las tiene. Si tiene un programa en la computadora que saca cociente y resto de polinomios, ¿cómo lo emplearía para determinar si una ecuación cuadrática dada tiene solución en \mathbb{Z}_p ? ¿si un elemento dado de \mathbb{Z}_p tiene raíz cuadrada?
10. (4) Consideramos la famosa $\sqrt{-1}$ en el cuerpo \mathbb{Z}_p , p primo. Ud. recordará que el cuerpo de los reales no lo tiene, y esa carencia molestó tanto a los matemáticos que inventaron los números complejos. Pero fíjese que \mathbb{Z}_5 tiene $\sqrt{-1}$: $2^2 = 4$ y $3^2 = 4$ en \mathbb{Z}_5 , y 4 equivale a -1 en \mathbb{Z}_5 .
 - a) Determine cuáles de los siguientes tienen $\sqrt{-1}$: $\mathbb{Z}_3, \mathbb{Z}_7, \mathbb{Z}_{11}, \mathbb{Z}_{13}, \mathbb{Z}_{17}, \mathbb{Z}_{19}, \mathbb{Z}_{23}, \mathbb{Z}_{29}$.
 - b) Piense en las respuestas al inciso anterior. ¿Tiene alguna conjetura sobre cuál es la condición sobre p que determina si \mathbb{Z}_p tiene $\sqrt{-1}$ o no?
 - c) ¿Puede demostrar la conjetura anterior?
11. (3) Hagamos el mismo truco que hicieron para inventar los números complejos: Agregamos un i tal que $i^2 = -1$, teniendo o no \mathbb{Z}_p un $\sqrt{-1}$, y consideramos el conjunto de elementos $\{a + bi : a, b \in \mathbb{Z}_p\}$, e imponemos las leyes distributivas y asociativas a éstos, implicando que $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$.
 - a) Para \mathbb{Z}_3 , ¿estos elementos forman un cuerpo?
 - b) Para \mathbb{Z}_5 , ¿estos elementos forman un cuerpo? (Sugerencia: si $r^2 = -1$ entonces el polinomio $x^2 + 1 = (x - r)(x + r)$. Pero el valor del producto de dos polinomios es el producto de los valores de los polinomios factores, y un cuerpo no puede tener divisores de 0.)
 - c) Demuestre que estos elementos forman un cuerpo si \mathbb{Z}_p no tiene $\sqrt{-1}$. (Sugerencia: La cuestión clave es la existencia de inversos. ¿Qué necesitamos para que, dados a y b en \mathbb{Z}_p , se pueda encontrar x, y en \mathbb{Z}_p tales que $(a + bi)(x + yi) = 1$?
 - d) Demuestre que estos elementos no forman un cuerpo si \mathbb{Z}_p tiene $\sqrt{-1}$. (Sugerencia: ¿Qué puede impedir lo que se hizo en el inciso anterior)?
12. (3) Hagamos con $\sqrt{2}$ lo mismo que hicimos con $\sqrt{-1}$. Es decir, agregamos un elemento $r \notin \mathbb{Z}_p$ tal que $r^2 = 2$, y consideramos el conjunto $\{a + br : r \in \mathbb{Z}_p\}$. Conteste las mismas preguntas que en el ejercicio 10.
13. (1) Calcular $\text{resto}(17^{6000}, 19)$.
14. (2) Encuentre todas las raíces del polinomio $x^3 - x^2 + x - 1$ en:
 - a) \mathbb{Z}_{13} ,
 - b) \mathbb{Z}_{19} .
15. (2) Encuentre x, y en \mathbb{Z}_{521} tales que se cumplen en \mathbb{Z}_{521} las ecuaciones simultáneamente:

$$x + y = 1$$

$$3x - 2y = 0$$

16. (3) Encuentre un primo $p > 3$, cuando sea posible, tal que en \mathbb{Z}_p el polinomio $x^4 - x^2 - 2$ tenga: a) ninguna raíz, b) exactamente una raíz, c) exactamente dos raíces, d) exactamente tres raíces, e) exactamente cuatro raíces, f) más de cuatro raíces. En cada caso, de no existir tal p , indique brevemente por qué; en cada caso de existir, indique cuáles son las raíces.

17. (2) Demuestre que si las operaciones binarias $+$ y \times cumplen la ley distributiva $a \times (b + c) = a \times b + a \times c$ sobre un conjunto C , y si cada una es compatible con una relación de equivalencia \equiv cuyas clases de equivalencia son $[a] = \{b \in C : b \equiv a\}$, entonces se cumple la ley distributiva cuando extendemos las operaciones a las clases de equivalencia así:

$$[a] + [b] = [a + b]; \quad [a] \times [b] = [a \times b].$$

Es decir que siempre $[a] \times ([b] + [c]) = [a] \times [b] + [a] \times [c]$.

18. (3) Demuestre que en un cuerpo finito C , toda función de C en C es polinómica.

9. Más sobre primalidad y factorización

¿Se puede determinar que un número no es primo sin sacarle un factor? Sabemos que si n es primo, entonces para cualquier $a \in \mathbb{Z}_n$, $a^{n-1} = 1$. Esto es *el pequeño teorema de Fermat*, uno de sus más famosos teoremas. Extraña algo que Fermat, en el mismo año que publicó ese teorema, en una carta a Mersenne conjeturaba que $2^{2^n} + 1$ era siempre primo, que se cumplía para $n = 1, 2, 3, 4$ ($n = 4$ requiere división por 54 primos), pero no había conseguido demostrarlo para $n = 5$. (Hay 6542 primos p tal que $p^2 \leq 2^{2^5} + 1$, pero para restringir las divisiones de prueba a éstos, tendríamos que engendrarlos, que aún con la criba parece formidable.) Pero según el teorema del mismo Fermat, si $2^{32} + 1$ es primo entonces $3^{2^{32}} \equiv_{2^{32}+1} 1$. Pero con 32 multiplicaciones modulo $2^{32} + 1$ llegamos a $3^{2^{32}} \equiv_{2^{32}+1} 3029026160$, no 1. (Si pedimos a Maple $3^{(2^{32})} \bmod (2^{32} + 1)$ registra un desborde aritmético. Tenemos que hacer $x, n := 3, 0$; y repetir $x, n := x \cdot 2 \bmod (2^{32} + 1), n + 1$; hasta que veamos $n = 32$. Empezamos cumpliendo la condición $x = 3^{2^n}$, condición que se preserva bajo la operación dada.) Claro que Fermat no tenía Maple en su computadora pero no debe ser más que media hora de trabajo hacer eso a mano, y Fermat hubiera podido saber por su propio teorema que $2^{32} + 1$ no era primo.

Entonces sí es posible saber que un número no es primo sin producir un factor, y entonces es muy posible, muchos matemáticos dirían probable, que es mucho más fácil determinar si un número es primo o no que factorizarlo.

Pero no es 100 % segura esta suposición, y hasta que se demuestre que sí, tenemos que estar un poco nerviosos cada vez que mandamos el número de una tarjeta de crédito por internet. Veremos por qué en la próxima sección.

Hasta agosto de 2002 la mayoría de los matemáticos hubiera conjeturado que el problema de la primalidad no era polinómico en el número de dígitos, es decir, para una computadora dada no hay polinomio $p(n)$ tal que la máquina puede decidir la primalidad de cualquier número de n dígitos en menos de $p(n)$ segundos. (Se supone que si hubiera tal polinomio, otra computadora que es 10 veces más rápida haría esa decisión en menos de $p(n)/10$ segundos).

Pero en agosto de 2002 tres matemáticos indios, el Dr. Manindra Agrawal y sus alumnos Neeraj Kayal y Nitin Saxena (“AKS”) aumentaron el nerviosismo de los que hacen comercio en el internet y los criptógrafos: descubrieron un algoritmo para decidir primalidad en tiempo polinómico.

Hay que entender que eso no automáticamente lo hace superior a algoritmos existentes para todas las aplicaciones. Por ejemplo si alguien descubre un algoritmo para calcular una función $f(n)$ en menos de n^2 segundos, y yo estoy en la misma computadora usando un algoritmo que demora x^n segundos donde $x = 1,013911386$, no necesariamente tengo que abandonar mi algoritmo exponencial por el polinómico, porque puede ser que mi aplicación necesita $f(n)$ solo para $n < 1000$. Ahora, $x^{500} = 1000$ y $500^2 = 250000$: mi algoritmo calcula $f(500)$ en menos de 17 segundos, mientras ¡el nuevo sólo garantiza hacerlo en 69 horas! El primer n en que el nuevo algoritmo gana es $n = 1000$, donde los dos algoritmos están demorando entre 11 y 12 días. Desde luego, de ahí en adelante n^2 gana contundentemente: $1050^2 = 1,10$ millones, $x^{1050} = 1,99$ millones, $1100^2 = 1,21$ millones, $x^{1100} = 3,98$ millones.

Algo parecido ocurre con el algoritmo AKS: no empieza a ganar a los viejos algoritmos hasta números tan grandes que ninguna computadora con ningún algoritmo puede terminar durante la vida del operador. Pero la historia de los nuevos algoritmos es que los investigadores los mejoran, y ya han mejorado el algoritmo AKS por un factor de más de un millón, pero todavía le falta otro factor de 100000 para ser competitivo.

Hay muchas aplicaciones para primos grandes, y veremos una en la criptografía más adelante. Pero la búsqueda de primos más y más grandes ha sido un fin en sí mismo, una competición entre matemáticos que ya estaba en pleno desarrollo en el siglo 17 pero que existía entre los griegos antiguos como evidencia la criba de Eratóstenes. Pero Mersenne y Fermat encontraron primos que no podían soñar los griegos, y continuamente en estos 4 siglos se ha encontrado, sobre los trabajos de ellos, primos de millones de dígitos, siendo el último descubierto en la Universidad de California en Los Angeles, el 23 de Agosto de 2008: tiene 12978989 dígitos, pero tiene una expresión más compacta: $2^{43112609} - 1$.

En los últimos años, existió una motivación nueva para buscar primos grandes: había un premio para el que encuentre el primer primo de 10 millones de dígitos o más. Los científicos de UCLA lo ganaron, y ahora el desafío se renovó: ganará ciento cincuenta mil dólares de premio aquel que encuentre el primer primo con más de cien millones de dígitos. Ya que estamos en trece millones, no hay que perder tiempo.

En 1588, el primo más grande publicado era de 6 dígitos, saliendo dos en ese año, $2^{17} - 1$ y $2^{19} - 1$. En el siglo 17 Mersenne conjeturó la primalidad de varios primos de ese tipo $2^p - 1$ con p primo, que se llaman ahora *primos de Mersenne*, pero no demostró ninguno y se equivocó en varios. Euler demostró que $2^{31} - 1$ (10 dígitos) era primo, y en 1867 salió uno de 13 dígitos. En 1876 Lucas encontró el primo $2^{127} - 1$, que probablemente es el primo más grande descubierto con sólo papel y lápiz, y que duró como campeón hasta 1951 cuando $(2^{148} + 1)/17$ (44 dígitos) salió. Entra en el escenario en ese año la computadora programable, y nuevos campeones empezaban a salir uno tras otro, cinco en 1952, otro en 1957, cuatro por década entre 1960 y 1990, siete en los 90, y ya ocho más desde 2000. El número de dígitos del primo campeón ha crecido a un promedio de 178000 por año desde 1951 (el gráfico es casi lineal, ver el sitio web “The Prime Pages”).

De estos 34 campeones, 32 han sido primos de Mersenne. Un número de Mersenne es siempre potencial candidato por el siguiente

TEOREMA. Un número $2^n - 1$ es primo si y sólo si n es primo impar y $L_{n-2} = 0$ donde L es la sucesión definida por

$$L_0 = 4, \quad L_{m+1} = L_m^2 - 2 \bmod (2^n - 1).$$

La demostración de ésto usa sólo principios simples de la teoría de números, como hemos visto, pero muchos en cadena, que ocupan tres páginas en Knuth, tomo 2, y no lo reproducimos aquí, excepto para decir que si n no es primo, $n = uv$ con $1 < u, v$, entonces por la identidad $x^k - 1 = (x - 1)(1 + x + x^2 + \dots + x^{k-1})$ es $(2^u)^v - 1 = (2^u - 1)(1 + 2^u + \dots + 2^{u(v-1)})$, es decir $2^{uv} - 1$ es divisible por $2^u - 1$. Por ejemplo, $2^7 - 1$ es primo porque $L_0, L_1, L_2, L_3, L_4, L_5 = 4, 14, 67, 42, 111, 0$. Y éste es el algoritmo, introducido por Lucas (1876) y mejorado por Lehmer (1930), que se usa ahora para buscar primos grandes.

El punto de partida para el algoritmo AKS es una generalización del teorema de Fermat a polinomios: si n es coprimo con a entonces n es primo si y sólo si

$$(x - a)^n \equiv_n (x^n - a) \text{ para todo natural } x,$$

lo cual es equivalente a decir (ver ejercicio) que simplificando los dos polinomios lo más posible tomando los coeficientes mod n , y aprovechando $x^n \equiv_n x$ para todo x, n , los dos polinomios son iguales. Es una caracterización elegante de números primos pero no muy útil para cálculos porque el cálculo de $(x - a)^n$ es muy largo. Pero si se pudiera trabajar con los restos módulo un polinomio como $x^r - 1$ con r suficientemente chico, el cálculo se achica.

Ahora, si n es primo y si r y n son coprimos con a entonces

$$(x - a)^n \equiv x^n - 1 \pmod{(x^r - 1, n)}$$

donde $p(x) \equiv q(x) \pmod{(s(x), n)}$ quiere decir que tomando los coeficientes \pmod{n} , los restos de p, q divididos por s son iguales. ¿Para cuáles a, r se puede declarar la recíproca? La respuesta llegó con el siguiente

TEOREMA. (Agrawal, Kayal, Saxena). Sea $s \leq n$. Suponga que los primos q, r cumplen: $q \mid r - 1$, $n^{(r-1)q} \not\equiv 0, 1 \pmod{r}$, y

$$\binom{q + s - 1}{s} \geq n^{2\lfloor \sqrt{r} \rfloor}.$$

Si para todo $1 \leq a < s$ tenemos

- (i) a es coprimo con n , y
- (ii) $(x - a)^n \equiv x^n - a \pmod{(x^r - 1, n)}$

entonces $n = p^k$ para algún primo p y $k > 1$.

La existencia de q, r, s que cumplan la hipótesis de este teorema originalmente requirió el uso de la teoría analítica de números, y AKS, no sintiéndose satisfechos con su propia demostración sin entender a fondo el paso que apeló a esa teoría, se rindieron en el intento, diciendo Agrawal que la teoría analítica de números era demasiado densa. Después salió una pequeña variación del teorema que permitió eliminar el uso de esa teoría.

El teorema, y los límites establecidos por los q, r, s condujo directamente al algoritmo AKS:

```

primo( $n$ ) =
  si ( $n = a^b$  para algún  $n > a, b > 0$ ) no
  sino
    para  $r$  desde 2 hasta  $n$  haga
      si ( $\text{mcd}(n, r) \neq 1$ ) no
      sino
        si ( $r$  es primo mayor que 2)
           $q :=$  el factor más grande de  $r - 1$ 
          si ( $q > 4\sqrt{r} \log n$  y  $n^{(r-1)q} \not\equiv 1 \pmod{r}$ ) break
        si (existe  $a$ ,  $1 \leq a \leq 2\sqrt{r} \log n$ , tal que  $(x - a)^n \not\equiv (x^n - a) \pmod{(x^r - 1, n)}$ ) no
        sino sí

```

Es polinómico probar si n es una potencia, y la demostración es elemental pero de varias páginas.

10. Criptografía

Suponga que el agente 007 quiere recibir mensajes del agente 011 por correo electrónico pero teme que los mensajes pueden ser interceptados por el hacker enemigo Rasputnik. Los mensajes no requieren Microsoft Word ni Tex – puro ASCII, con un alfabeto de tamaño 2^8 , basta. Ahora, un mensaje de k caracteres de ASCII se puede considerar un número escrito en la base 2^8 , o sea un número escrito en binario de $8k$ dígitos binarios (“bits” por el inglés “binary digits”). Entonces cada mensaje es un número, y 007 quiere que el número sea codificado según una clave que Rasputnik no conoce. Un simple mapeo biyectivo de los números menores que 2^8 en sí mismos no basta, porque es fácil de romper en base de las frecuencias conocidas de las letras en castellano. Además, quiere cambiar el código de vez en cuando y quiere indicar por correo electrónico cual código quiere que se use. Entonces le gustaría especificar por correo electrónico que cada mensaje M sea codificado como $f(M)$; después 007 en su casa va a aplicarle f^{-1} para entender el mensaje. Pero debería ser difícil para Rasputnik calcular f^{-1} aún conociendo f . Al mismo tiempo, no debería ser difícil para 011 calcular $f(M)$.

Por ejemplo, Sea p un primo grande, y a un elemento primitivo de \mathbb{Z}_p . Sea $f(M) = a^M \pmod{p}$. f es una biyección de \mathbb{Z}_p en sí mismo. La dificultad con esto es que f^{-1} va a ser tan difícil para

007 como para Rasputnik. Claramente si $p = 2^{232582657} - 1$, no va a ser satisfactorio multiplicar por a hasta que aparezca $f(M)$.

Otro ejemplo: sea $f(M) = M^2 \bmod n$. Ahora el receptor de $k = f(M)$ tiene que calcular \sqrt{k} en \mathbb{Z}_n . Esto es difícil para algunos n , pero no si n es un primo equivalente mod 4 a 3: $n = 4j + 3$. Según el pequeño teorema de Fermat, $k^{4j+3} = k \bmod (4j + 3)$, entonces $k^{4(j+1)} = k^2$, entonces $k^{2(j+1)} = \pm k$, pero no puede ser $-k$ porque en ese caso tendríamos un $M^2 = k$ y un $x^2 = -k$ en \mathbb{Z}_n , entonces $(\frac{M}{x})^2 = -1$, pero este \mathbb{Z}_n no puede tener $\sqrt{-1}$ (ver ej. 1), o sea $(k^{j+1})^2 = k$, y concluimos que $M = \pm k^{j+1} = \pm k^{(n+1)/4} \bmod n$, para la decodificación de $f(M)$, y seguro que uno de estos no tiene sentido. Pero esto es fácil para Rasputnik también.

10.1. El método de Rabin. Pero suponga que $n = pq$ donde ambos de p y q son equivalentes a 3 mod 4. Ahora 011 manda $k = M^2 \bmod pq$. Es decir, $M^2 = jpq + k$, así que $M^2 \equiv_p k$ y $M^2 \equiv_q k$, y por tanto (ver ejercicio 1) $M \equiv_p \pm k^{\frac{p+1}{4}}$ y $M \equiv_q \pm k^{\frac{q+1}{4}}$. Se pueden armar entonces 4 sistemas de dos congruencias simultáneas, entre cuyas soluciones debe elegir 007. Es decir, 007 tiene que elegir entre “reunión esta noche a las 10 en lo de 013”, “45b;#\\$bububa”, “&&!zxvb,?37#d@” y “ser_o_no_ser, eso es la gzornenplatz”. No tendrá dificultad en saber cuál de éstos es el mensaje. Rasputnik, en cambio, no sabe p y q , aún conociendo el método de codificación. Este método es debido a Rabin.

Pero, ¿Rasputnik tiene necesariamente que hacer la factorización para sacar esas raíces cuadradas en \mathbb{Z}_n ? Lo que podemos demostrar es que sacando esas raíces cuadradas, con un poco más de trabajo sacamos p y q , es decir, hacemos la factorización, y por tanto calcular las raíces es el equivalente computacional de hacer la factorización. Porque, supongamos que M tiene las cuatro raíces cuadradas distintas. Por lo que hemos visto hace poco, sabemos que tienen que ser $\pm a, \pm b$ para algún a, b . Ahora, $a^2 = b^2$ en \mathbb{Z}_{pq} , por tanto $a^2 - b^2 = 0$ en \mathbb{Z}_{pq} , por tanto $(a - b)(a + b) = 0$ en \mathbb{Z}_{pq} . Ninguno de esos factores es 0 en \mathbb{Z}_{pq} , entonces por lo menos uno no es coprimo con pq ($st = kpq \implies p|s$ o $p|t$). Suponemos que $p|a - b$. Entonces q no divide a $a - b$ (porque implicaría que $pq|a - b$ o sea que $a - b = 0$ en \mathbb{Z}_{pq}). Entonces tenemos que $p = \text{mcd}(a - b, pq)$, y ya que q no divide a $a - b$, $q|a + b$ y concluimos que $q = \text{mcd}(a + b, pq)$. Entonces, sacando a, b Rasputnik sólo necesita sacar $\text{mcd}(a - b, n)$ y $\text{mcd}(a + b, n)$ para saber p y q . Sacar mcd es fácil, entonces si Rasputnik tiene un método con que es fácil sacar las raíces, le es fácil factorizar pq . Según *Wikipedia*, en el artículo sobre criptografía, éste es el único método en que la decodificación es demostrablemente tan difícil como la factorización.

10.2. El método RSA. El método RSA, por sus inventores R. Rivest, A. Shamir y L. Adleman, también emplea aritmética modulo $n = pq$ donde p y q son primos distintos. También figuran dos números d y e que especificaremos en seguida. 007 manda por correo a 011 la función f para la codificación del mensaje M :

$$f(M) = M^e \bmod n.$$

Así que manda por vía pública n y e , que pueden ser interceptados por Rasputnik. 007 decodifica con

$$f^{-1}(m) = m^d \bmod n.$$

Para saber cuáles son e y d , necesitamos otro teorema. Una generalización del teorema pequeño de Fermat. Primero notamos que para cualquier n , los números de $\{1, \dots, n\}$ coprimos con n forman un grupo bajo multiplicación mod n : si $ab = kn + r$ y r no es coprimo con n entonces $kn + r$ es múltiplo de algún primo divisor de n , y ese primo tendría que dividir o a a o a b , así que a y b no son ambos coprimos con n . Dado a coprimo con n , ya sabemos que podemos resolver $ax = 1 \bmod n$ notando que existen r, s tales que $ra + sn = 1$, así que $r \bmod n$ sería dicho inverso. El orden de ese grupo tiene el nombre especial $\varphi(n)$. Como en todo grupo, el orden de cualquier elemento divide al orden del grupo, entonces el nuevo teorema es

Si a es coprimo con n entonces $a^{\varphi(n)} = 1 \bmod n$.

Consecuentemente para a coprimo con n tenemos $a^{\varphi(n)+1} = a \bmod n$. Ahora, el n que nos interesa aquí es $n = pq$, donde, como dijimos, p y q son ambos primos y distintos. Vamos a ver que en este caso $a^{\varphi(n)+1} = a \bmod n$ para todo a en \mathbb{Z}_n , coprimo o no con n . Eso es porque en este caso los únicos no coprimos con $n = pq$ en \mathbb{Z}_n son 0, los $q-1$ múltiplos de p y los $p-1$ múltiplos de q ; por tanto $\varphi(n) = n-1-(q-1)-(p-1)$. Simplificando, $\varphi(n) = (p-1)(q-1)$. Ahora, $p^{q-1} = 1 \bmod q$, y si $1 \leq k < q$ entonces $k^{q-1} = 1 \bmod q$, y por tanto $(kp)^{(q-1)(p-1)} = k^{(q-1)(p-1)} p^{(q-1)(p-1)} = 1 \bmod q$, lo que quiere decir que $(kp)^{\varphi(n)} = sq + 1$ para algún s . Multiplicando por kp en los dos lados da $(kp)^{\varphi(n)+1} = skpq + kp = kp \bmod n$. Exactamente el mismo tratamiento da para $1 \leq k < p$, $(kq)^{\varphi(n)+1} = kq \bmod n$. Y por supuesto $0^{\varphi(n)+1} = 0$.

Ahora, sea d cualquier número coprimo con $\varphi(n)$, y sea e su inverso en $\mathbb{Z}_{\varphi(n)}$. Entonces $(M^d)^e = M^{de} = M^{k\varphi(n)+1}$ para algún k . Falta demostrar por inducción sobre k que esto es M . Lo acabamos de demostrar para $k = 1$. Si se cumple para algún k entonces $M^{(k+1)\varphi(n)+1} = M^{k\varphi(n)} M^{\varphi(n)+1} = M^{k\varphi(n)} M = M^{(k+1)\varphi(n)+1} = M \bmod n$.

Rasputnik sólo puede robar n y d ; para saber e tiene que factorizar n .

Pero si descubre cómo factorizar en tiempo polinómico, los secretos de los agentes, y tu número de tarjeta, peligran en la web.

10.3. Aritmética de números grandes. Una computadora viene equipada de aritmética modulo, típicamente, 2^{32} . Puesto que 2^{10} es aproximadamente 1000, 2^{30} es aproximadamente mil millones, 10^9 , y 2^{32} es 4×10^9 , un número de diez dígitos, cuando en los cálculos que acabamos de considerar nos encontramos con números de miles, aún millones, de dígitos. Tenemos que trabajar con números representados como cadenas de dígitos, pero no estamos restringidos a la base 10, que hemos inventado porque tenemos 10 dedos, sino en una base B que conviene a la computadora, que puede ser, por ejemplo, $B = 2^{32}$. Entonces representamos un número como una sucesión d_k, \dots, d_0 donde $0 \leq d_i < B$ para cada i , representando el número $\sum_{i=0}^k d_i \times B^i$. Podemos poner ésto en un algoritmo:

$val(d, B) = \text{si}(d = \emptyset) \ 0 \ \text{sino} \ d_0 + B \times val(resto(d))$

donde $resto([d_k, \dots, d_0])$ se define como la sucesión d_k, \dots, d_1 . (Si $k = 0$ entonces $resto(d) = \emptyset$, la sucesión vacía.) Y el inverso de val es

$rep(n, B) = \text{si} \ (n = 0) \ \emptyset \ \text{sino} \ (c, r) := cr(n, B); r, rep(c, B)$.

Aquí $r, rep(c, B)$ significa la sucesión d tal que $d_0 = r$ y $resto(d) = rep(c, B)$. Se debe entender que d_k, \dots, d_0 denota la misma sucesión que d_0, \dots, d_k , porque denotan la misma función $d : \{0, \dots, k\} \rightarrow \{0, \dots, B-1\}$. En la computadora, no hay izquierda y derecha.

Podemos decir que $val(rep(n, B), B) = n$, pero no necesariamente $rep(val(d, B), B) = d$, porque si d tiene algunos 0 a la izquierda, por ejemplo $d_k = d_{k-1} = 0$, van a desaparecer en $rep(val(d, B), B)$, porque rep no produce ningún 0 a la izquierda.

Ahora, dadas dos cadenas d, e , ¿cómo vamos a engendrar la cadena que representa la suma $val(d) + val(e)$? Parece fácil responder: $rep(val(d) + val(e))$. Pero eso no nos ayuda en la computadora porque presupone que la función val existe en la máquina, y entonces tiene que producir alguna representación de $val(d)$, pero d es la única representación que conocemos. Aquí un algoritmo:

$d + e = suma(d, e, 0)$

donde

$suma(d, e, \text{carreo}) =$

si $(d = \emptyset)$ **si** $(\text{carreo} = 0)$ e **sino** $aumentar(e)$

sino si $(e = \emptyset)$ **si** $(\text{carreo} = 0)$ d **sino** $aumentar(d)$

sino

si $(d_0 + e_0 + \text{carreo} < B)$ $(d_0 + e_0 + \text{carreo}), suma(resto(d), resto(e), 0)$

sino $(d_0 + e_0 + \text{carreo} - B), suma(resto(d), resto(e), 1)$

donde

$aumentar(d) =$

si $(d = \emptyset)$ $[1]$

sino

si ($d_0 = B - 1$) 0, *aumentar*(*resto*(d))

sino ($d_0 + 1$), *resto*(d)

Si $B = 2^l$ donde l es la longitud de palabra de la máquina, el paso $d_0 + e_0 + \text{carreo}$ se hace directamente con la aritmética incorporada en la máquina, que da el resultado modulo B y avisa si hubo desborde.

Para multiplicación, hay que hacer una subfunción que multiplica por un solo dígito. La multiplicación de la computadora da doble precisión, es decir dos dígitos base B . (Esto corresponde a memorizar la tabla de multiplicación en la escuela primaria.) Se corre d a la izquierda con 0, d . En lugar de multiplicar por cada uno de k dígitos y sumar los k resultados, sumamos cada resultado a la acumulación de los anteriores en el momento que esté listo.

División se define casi igual que para los números puros. Sean d, e cadenas en la base B :

$crB(d, e, B) =$

si ($d < e$) (\emptyset, d)

sino

(c, r) := $crB(d, [0, e], B)$

($c1, r1$) := $cr(r, e)$

($[0, c] + c1, r1$)

Hay dos problemas pendientes aquí. Uno es como se hace " $d < e$ ". Lo dejamos como ejercicio. El otro es, ¿como se hace $cr(r, e)$? En una máquina con aritmética binaria (hubo hace unos 60 años una máquina ternaria, pero hoy en día hay que ir a un museo para ver una de esas), B va a ser una potencia de 2, $B = 2^k$ y la máquina hace suma, diferencia, producto y cociente-resto modulo B . Pero una sucesión de m números menores que 2^k escritos en binario puede ser interpretado como un número de mk bits (dígitos binarios) y por tanto tratado con el algoritmo $crB(a, b, 2)$, o equivalentemente $cr2(a, b)$. Ya que el número de iteraciones en $cr2(a, b)$ es limitado por $\lceil \log_2(a/b) \rceil$, o sea, el menor número k tal que $a/b \leq 2^k$. Se lleva a cabo $cr2$ con comparación, adición y multiplicación por 2.

La demostración formal de que estos algoritmos son polinómicos en el número de dígitos es medio tediosa, pero pensándolo un poco es creíble.

Hay muchos problemas de números grandes cuya solución polinómica depende de que el cálculo del mcd sea polinómico. Ver Knuth, tomo 2 para una demostración de ésto.

Ejercicios 2.10

- Sea p primo congruente con 3 módulo 4, y suponga que k tiene raíz cuadrada en \mathbb{Z}_p . Muestre que las raíces cuadradas de k en \mathbb{Z}_p son $\pm k^{\frac{p+1}{4}}$.
- Considere el alfabeto $\{a, b, c, d, \dots, l, m, n, \tilde{n}, \dots, r, s, \dots, z, \}$ con 28 elementos (el espacio en blanco cuenta), asigne a espacio el valor 0, a a el 1, y así sucesivamente hasta z con valor 27. Con esto, cualquier mensaje de texto sobre ese alfabeto se puede ver como un número expresado en la base 28. Por ejemplo, el mensaje 'hola' corresponde al número 188.497.
 - Haga un programa que recibe un mensaje de hasta 15 caracteres sobre el alfabeto y obtiene el número que le corresponde.
 - Para $p = 71397866609$ y $q = 71397866609$, usando el método de Rabin codifique el mensaje 'el curso' y decodifique el mensaje 63576193649455.
 - Con los mismos $p = 20511251$ y $q = 20511319$, (este p, q es suficiente para 10 caracteres no más) codifique el mensaje 'el curso' en el sistema RSA con $d = 1009$, y decodifique el mensaje 73701686838974.
 - Discutir la factibilidad de mandar mensajes de valor mayor que pq , suponiendo que el receptor tiene en cuenta esa posibilidad.
- Usando MAPLE, expresar en base 2^{16} el número 10^{20} .

4. Escriba un algoritmo que recibe dos cadenas c y d de dígitos en base B , y contesta 1 si el número representado por la primera cadena es menor que el representado por la segunda, y 0 en otro caso. Programe su algoritmo en MAPLE.
-

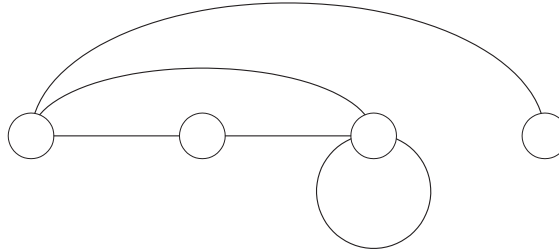
TEORÍA DE GRAFOS

1. Grafos

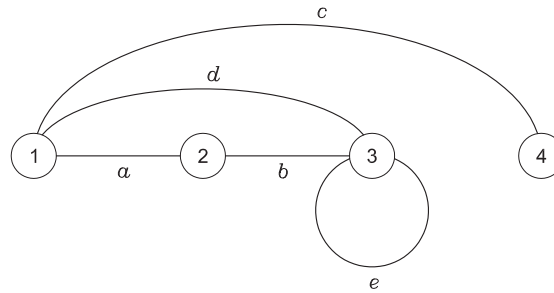
En una red de comunicación, no es necesario que toda estación pueda comunicarse directamente con otra, puesto que las estaciones pueden actuar de posta para un mensaje entre otras dos estaciones. Si una estación, o una línea de datos, deja de funcionar, queremos saber si la red queda conexas, es decir, si todas las estaciones que siguen funcionando pueden comunicarse entre sí. Para preguntas como ésta, no nos interesa la ubicación física de las estaciones, sino sólo su conectividad, y es así que surge la noción matemática de *grafo*, que es simplemente unos *nodos* con algunas conexiones que se llaman *aristas*. Una arista puede conectar dos nodos, o, como en algunas aplicaciones, un nodo consigo mismo. Una arista está anclada en sus dos extremos a nodos, o posiblemente al mismo nodo en los dos extremos. Formalmente:

DEFINICIÓN 3.1. Un *grafo* es (N, A, P) donde N es un conjunto de *nodos*, A es un conjunto de *aristas*, y P es una función de las aristas tal que cada $P(a) = \{p, q\}$ donde p, q son nodos (posiblemente con $p = q$, así que podemos decir que $P(a)$ es un conjunto de 1 o 2 elementos). Cuando G es un grafo, G_N denota sus nodos, G_A sus aristas, y G_P su función de aristas.

Generalmente pensamos en un grafo como un dibujo:



pero es difícil hacer algoritmos sobre dibujos. Si ponemos nombres a los nodos y a las aristas así:



este dibujo corresponde al grafo G en donde $G_N = \{1, 2, 3, 4\}$, $G_A = \{a, b, c, d, e\}$, $P(a) = \{1, 2\}$, $P(b) = \{2, 3\}$, $P(c) = \{1, 4\}$, $P(d) = \{1, 3\}$, $P(e) = \{3\}$. Claro que la representación formal es medio engorrosa, pero lo que importa ahora es el concepto, y buscamos estructuras de datos más limpias después.

Una arista a con $P(a) = \{p\}$, como la arista e del ejemplo, se llama *rulo*. Los rulos son irrelevantes en las aplicaciones que consideramos aquí pero los teoremas que consideramos son válidos si se permiten rulos o no, así que no queremos prohibirlos.

Es más amigable decir “el nodo p está conectado a la arista a ” que decir, aunque más compactamente, “ $p \in P(a)$ ”; por eso elegimos el primero muchas veces.

Un nodo que no es conectado a ninguna arista se llama *nodo aislado*. No interesan mucho nodos aislados; la única atención que los damos es prohibirlos donde molestan.

2. Caminos y conectividad

Una noción fundamental es la de un camino. Un camino empieza en algún nodo, y sigue por aristas visitando varios nodos. Formalmente:

DEFINICIÓN 3.2. Un *camino* en un grafo $G = (N, A, P)$ es una sucesión (posiblemente vacía) $a = a_1 \dots a_n$ de aristas tal que existe una sucesión $p = p_0 \dots p_n$ de nodos tal que para cada a_i , $P(a_i) = \{p_{i-1}, p_i\}$. La sucesión p se llama un *recorrido* del camino a . Se dice que a *empieza* en p_0 y *termina* en p_n .

Claro que no cualquier sucesión de aristas es un camino, porque puede ser que no exista recorrido. Por ejemplo, en el grafo anterior la sucesión c, b no es un camino. La sucesión a, b sí es, y su recorrido es 1, 2, 3. La sucesión a, a tiene dos recorridos: 1, 2, 1 y 2, 1, 2. La sucesión de a solo tiene dos recorridos, 1, 2 y 2, 1. Para cualquier nodo p , la sucesión de sólo p es recorrido del camino vacío. Dado un camino y un nodo p , hay a lo sumo un recorrido de ese camino que empieza con p .

Algunos autores prohíben que un camino repita aristas (pero dejan que sus recorridos repitan nodos). Para algunas aplicaciones nuestras más adelante conviene hablar de caminos que repiten aristas.

DEFINICIÓN 3.3. Los nodos p, q de un grafo son *conexos*, o p es *conexo* con q , si hay un camino que empieza en p y termina en q .

Nótese que la relación de *conexos* es de equivalencia. El camino vacío conecta cualquier nodo consigo mismo (si ya estoy en p , entonces para llegar a p no hago nada).

DEFINICIÓN 3.4. Un grafo es *conexo* si cualquier par de sus nodos son conexos.

El primer algoritmo que vamos a ver va a decidir si dos puntos dados son conexos. Empezamos con una descripción informal. Supongamos que los residentes de Grafilandia sólo saben del nodo donde viven y los nodos vecinos, es decir, unidos a su nodo por una arista. Si estamos en el nodo p , y le preguntamos “¿Cómo llegamos a p ?”, contestan “Ya está Ud. en p ”, es decir, es el camino vacío el que nos lleva a p : no hace falta hacer nada. Si preguntamos “¿Cómo llegamos a q ?”, si q es vecino, dicen “Tómese aquella arista”, pero si q no es vecino dicen “¡No sé!” Pero lo que sí podemos hacer es pedirle que nos diga quiénes son todos los vecinos, y sus números de teléfono. Empezamos a llamar a estos vecinos por teléfono, y guardamos una lista de los nodos llamados. Al residente de cada nodo que contesta el teléfono le preguntamos quiénes son sus vecinos, y para cada uno que no está en la lista lo llamamos para pedirle la lista de sus vecinos y después lo agregamos a nuestra lista de nodos llamados. Si alguna vez nos dicen que q es un vecino, sabemos que p y q son conexos. Entonces, si p y q no son conexos, nunca en ningún nodo nos van a decir que q es un vecino. Pero no podemos seguir llamando para siempre porque el número de nodos es finito y estamos cuidando de no llamar dos veces a ningún nodo.

Se procede así: Sean p, q los puntos. Si $p = q$, ya está, la respuesta es **sí**. Si no, buscamos una arista que une p con cualquier otro nodo que no sea p , digamos r_1 . Si no hay tal arista, es claro que no se puede conectar p con q , y la respuesta es **no**. Si hay y $r_1 = q$, ya está, la respuesta es **sí**. Si no, buscamos una arista que une el conjunto $\{p, r_1\}$ con cualquier otro punto fuera de ese conjunto, es decir, que une uno de los nodos de ese conjunto con un punto fuera de él, digamos r_2 . Otra vez, si no hay, la respuesta es **no**. Si hay y $r_2 = q$, ya está, la respuesta es **sí**. Sino, buscamos una arista que une el conjunto $\{p, r_1, r_2\}$ con cualquier punto fuera de ese conjunto. La situación general, entonces, es que tenemos un conjunto C de nodos en que p es conexo con cualquier nodo de C , y q está fuera de C . Buscamos una arista que une C con cualquier punto fuera de C , y si no hay, p no es conexo con q (lo que vamos a demostrar formalmente cuando tengamos el algoritmo formal). Si hay, si apunta a q terminamos con **sí**,

y si apunta a otro nodo r lo agregamos a C y repetimos. Cada vez que no hay respuesta, C se agranda, así que tenemos que conseguir respuesta tarde o temprano.

Ahora reducimos esto a sus esenciales. Tenemos una lista de nodos que hemos “visitado”, pero dejemos de decirle *lista*, porque una lista ordena sus elementos y aquí no importa el orden: tenemos un conjunto de nodos. Elegimos *cualquier* vecino de uno de los nodos del conjunto que no está en el conjunto. Si no hay, q no es conexo con p . Si hay, y es q , entonces q sí es conexo con p , y si no es q lo agregamos al conjunto y repetimos.

La información que hay que entregar a la función que acabamos de describir es el grafo, el conjunto C de nodos, y el nodo objetivo q . Su llamada inicial, de parte de un algoritmo que llamamos *conexos*, es con $C = \{p\}$, y a esta función ayudante de *conexos* la llamamos *extender*. Para simplificar su expresión definimos que q es *vecino* de p si hay una arista a tal que $P(a) = \{p, q\}$. (¿Cuándo es p vecino de p , técnicamente según esta definición? ¿Siempre? ¿A veces? ¿Nunca?)

$conexos(G, p, q) = \mathbf{si} (p = q) \mathbf{sí} \mathbf{sino} \ extender(G, \{p\}, q)$

donde

$extender(G, C, q) =$

si hay $r \in C$ con vecino $s \notin C$

si ($s = q$) **sí sino** $extender(G, C \cup \{s\}, q)$

sino no

La clave es la propiedad de que todo nodo en C es conexo con p , y que $q \notin C$. Esta propiedad es obvia en la primera invocación, hecha por *conexos*, porque esa invocación es con $C = \{p\}$, y no ocurre a menos que $q \neq p$. Ahora, cualquier invocación en que se cumple esta propiedad y hace una nueva invocación, se cumple también en la nueva invocación. Para ver esto, considere la invocación $extender(G, C, q)$ donde todo nodo de C es conexo con p , y $q \notin C$. Hay otra invocación sólo si se encuentra el vecino $s \notin C$ de un nodo de C , y además con $s \neq q$, porque si $s = q$ no hay nueva invocación. La nueva invocación es $extender(G, C \cup \{s\}, q)$. Cada nodo en C es conexo con p , como está dado, y s es conexo con p por ser conexo con un nodo de C , que a su vez es conexo con p ; conclusión: cada nodo de $C \cup \{s\}$ es conexo con p . Y $q \notin C \cup \{s\}$ por no estar en C ni ser igual a s .

Entonces, si *extender* contesta **no**, es que no encontró el buscado vecino s . En ese caso no hay arista entre C , que contiene p , y $N - C$, que contiene q . Si contesta **sí** es porque q es conexo con algún nodo de C , y por tanto con p . El algoritmo queda demostrado.

El lector puede estar inquieto sobre el hecho de que este algoritmo no es determinista en su ejecución; frecuentemente nos ofrece una elección: *cualquier* vecino de *cualquier* nodo en C . Si el algoritmo pusiera restricciones sobre esas elecciones estaría mal hecho, porque ninguna restricción es necesaria. Claro que en la realización se va a elegir determinísticamente por algún criterio, sea por orden numérico o por ubicación en una estructura de datos, o aún por propiedades especiales del sistema representado por el grafo, pero *nada de eso es parte de la esencia del algoritmo*.

Alguien puede pensar que después de semejante trabajo para establecer que p, q son conexos, que lástima que no podemos exhibir el camino. ¿Tenemos que modificar mucho el algoritmo *conexos* para que engendre el camino? Resulta que no: falta nada más que mantener una función que para cada nodo en C , salvo p , da la arista que lo hizo entrar en C . A esta función la llamamos *portador*. En la primera invocación es vacía ya que p no entra por ninguna arista: p no tiene portador. Este algoritmo, ya que estamos pidiéndole un camino, falla si no hay camino. Cuando determina que p, q son conexos, contesta con un camino de p a q por medio de ir de q a p , teniendo el portador de todos estos nodos salvo el de p (porque no hay), y colecciona estas aristas al revés.

Para definir formalmente este algoritmo, conviene un nuevo mecanismo notacional. Aquí tenemos la función *portador* definida en datos, cuando hasta ahora las funciones han sido definidas por algoritmos. No hay nada raro en definir una función como una tabla: por ejemplo,

la guía de teléfono define una función de sus abonados a sus números de teléfono. Este tipo de datos, como cualquier otro, necesita de algunas operaciones básicas. Una es la invocación de f con el argumento x , que escribimos $f(x)$. La otra que usaremos aquí es la operación que cambia la función para un argumento dado. Supongamos que tenemos una función f , y queremos cambiarla para que $f(5) = 4$, y puede ser o no ser que $f(5)$ esté definida. Puede ser que $f(5)$ esté definida como 8, y queremos que sea 4, o que $f(5)$ no esté nada definida y queremos que sea 4. Pero en realidad estamos queriendo definir una nueva función g que es casi igual que f y difiere únicamente en la cuestión de $g(5)$: queremos que $g(5)$ sea 4, pero que para cualquier $x \neq 5$ $g(x)$ esté definida sólo si $f(x)$ está definida, y en caso que sí, que $g(x) = f(x)$. Llamamos a esta g como $f(5 \rightarrow 4)$.

En general tenemos esta definición formal:

DEFINICIÓN 3.5. Para una función f con dominio D , con valores en el conjunto U , y dos objetos x e y , $f(x \rightarrow y)$ es la función g cuyo dominio es $D \cup \{x\}$ con $g(x) = y$, y para todo $z \neq x$ en D , $g(z) = f(z)$.

Ahora podemos escribir el algoritmo para el camino del nodo p al nodo q en el grafo G . El subalgoritmo *extender* va construyendo una función *portador* que da para cada nodo r encontrado, salvo p , el nodo predecesor de r que hizo entrar a su vecino r . El valor final de *portador*, el que se forma cuando por fin el nodo destino q se encuentra, junto con q es suficiente para que el subalgoritmo *construir* construya el camino de p a q :

```
camino( $G, p, q$ ) =
  si ( $p = q$ ) el camino vacío
  sino extender( $G, \{p\}$ , función vacía,  $q$ )
donde
extender( $G, C, portador, q$ ) =
  si hay arista  $a$  con  $P(a) = \{s, r\}$  donde  $r \in C$  y  $s \notin C$ 
    si ( $s = q$ ) construir( $q, portador(q \rightarrow a)$ )
    sino extender( $G, C \cup \{s\}, portador(s \rightarrow a), q$ )
y donde
construir( $q, portador$ ) =
  si existe  $a$  tal que  $portador(q) = a$ 
    sea  $p \in P(a) - \{q\}$ 
    construir( $p, portador$ ) $a$ 
  sino  $\{\}$ 
```

La alternativa de “**si hay**” no se da; el algoritmo en ese caso no produce nada, falla. ¿Quiere decir que la computadora que lo implementa se va a colgar? No, no más que una persona que ejecuta el algoritmo se va a colgar, en lugar de informar gentilmente que el algoritmo falló.

Este algoritmo, basado en *conexos*, también es no determinista. En este caso su no determinismo es más importante porque afecta el resultado en caso de que haya más de un camino entre p y q . Siempre produce un camino cuando hay, pero cuál produce depende de cómo se hacen las elecciones que se ofrecen.

Considérese el problema de determinar si un grafo es conexo o no. Mirando la definición, parecería que tendríamos que invocar *conexos*(p, q) para cada par de nodos distintos p, q , que sería $n(n-1)/2$ invocaciones de *conexos*. Pensándolo un poco, hace falta una sola invocación de un algoritmo que es una leve modificación de *conexos*: uno que investiga si p, q son conexos, donde p es cualquier nodo y q es un nodo agregado, que no es un nodo de G . Claro que *conexos* tiene que contestar **no** en este caso, pero la modificamos para contestar el tamaño de C en el momento de decidir para **no**. Porque en ese momento C tiene todos los nodos conexos con p , y G es conexo si y sólo si estos son todos los nodos de G . Probablemente el número de nodos de G es accesible en algún lado (¡pero depende de la estructura de datos otra vez!), y simplemente comparamos ésto con el número de nodos en C .

Consideremos un momento la eficiencia de estos algoritmos. Primero notamos que piden siempre una búsqueda: un nodo con ciertas propiedades, entre ellas que no esté en un cierto conjunto. Pero la eficiencia con que determinamos si un objeto está o no está en un conjunto depende mucho de la estructura de datos con que representamos el conjunto, así que no podemos saber mucho de la eficiencia de estos algoritmos antes de ver algo sobre cómo están realizados.

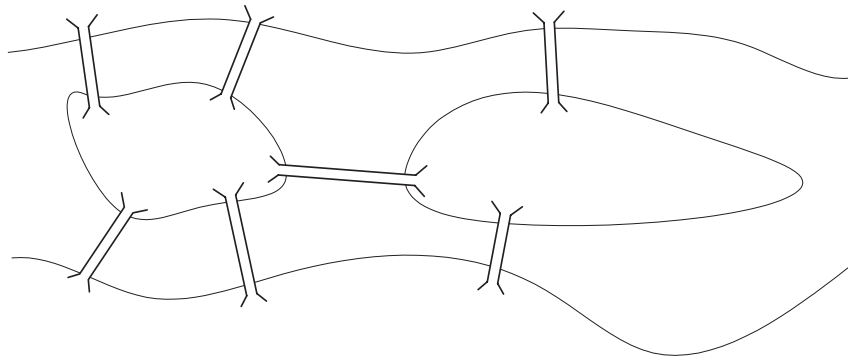
Pero algo podemos saber. Sabemos por ejemplo que no va a haber más de n invocaciones a *extender*, donde n es el número de nodos en G . Entonces la eficiencia reside en la de *extender*.

En resumen, el algoritmo abstracto puede indicar algo de la eficiencia con que puede trabajar, pero puede depender mucho de cómo se lo realiza también.

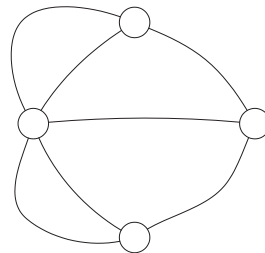
3. Los puentes de Königsburg y ciclos de Euler

Leonard Euler, uno de los grandes matemáticos de la historia, vivía en Königsburg, y los colegas de él se divertían discutiendo un problema: cómo hacer una caminata por la ciudad cruzando cada uno de sus siete puentes una y sólo una vez, y volver al punto de partida.

Un plano de la ciudad mostraba algo como se ve en la figura.



Los puentes de Königsburg...



...y su grafo equivalente

Naturalmente hemos simplificado bastante el plano de Königsburg del siglo 18 porque sólo importan los puentes, las islas, y las dos bandas del río. Pero Euler quería simplificar aún más. El decía que era un problema del grafo en la figura, y que no importaba si los nodos eran islas o bandas o qué. Y se puso a pensar en la generalidad de esos curiosos objetos, que ahora llamamos *grafos*, de los cuales había producido un ejemplo, y en el problema de encontrar un camino que pasaba por cada arista y volvía al nodo de partida.

DEFINICIÓN 3.6. Un camino no nulo cuyo recorrido empieza y termina en el mismo nodo y que no repite ninguna arista se llama *ciclo*.

Un ciclo que contiene todas las aristas del grafo se llama ahora *ciclo de Euler* porque Euler resolvió ese primer problema con que fundó la teoría de grafos.

Y resultó que no existía un ciclo de Euler para los puentes de Königsburg. La razón era que el grafo tiene nodos de grado impar. El *grado* de un nodo es el número de aristas no nulos conectados al nodo. La idea es que si imaginamos una puerta en cada extremo de cada arista no

culo, para entrar y salir del nodo, el grado es el número de puertas en el nodo. Más formalmente, y más generalmente para usar en la demostración:

DEFINICIÓN 3.7. El *grado* de un nodo p en un conjunto C de aristas de un grafo $G = (N, A, P)$ es el número de aristas $a \in C$ que no son rulos tal que $p \in P(a)$. El *grado de p en el grafo G* es el grado de p en A .

TEOREMA 3.8. *Un grafo sin nodos aislados tiene un ciclo de Euler si y sólo si es conexo y cada nodo tiene grado par en el grafo.*

Comentamos que el único motivo para excluir nodos aislados es que el ciclo de Euler no implica que el grafo es conexo si hay nodos aislados. Pero si todo nodo tiene grado par sí hay ciclo de Euler si los nodos no aislados y sus aristas forman un grafo conexo. En la demostración vamos a usar esta

LEMA 3.9. *Dado un ciclo de un grafo, cada nodo del grafo tiene grado par en el conjunto de aristas usadas en el ciclo.*

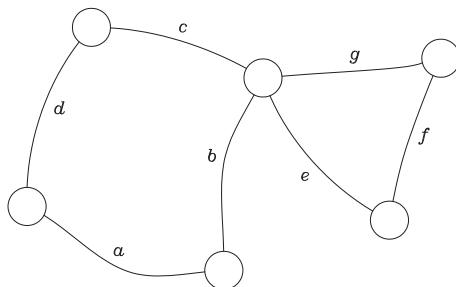
Demostración. Sea $c = c_1 \dots c_n$ un ciclo con recorrido $r_0 \dots r_n$ (así que la sucesión de nodos y aristas en el ciclo es $r_0 c_1 r_1 c_2 r_2 \dots c_n r_n$). Si toda arista en el ciclo es rulo, cada nodo tiene grado 0 (par) en $\{c_1, \dots, c_n\}$. Si no, sea $c_{i_1} \dots c_{i_m}$ la subsecuencia de todos los nodos no rulos de c . sea $r_0, r_{i_1}, \dots, r_{i_m} r_0$ la correspondiente subsucesión del recorrido r . Es claro que el camino sale de r_0 por c_{i_1} y entra de nuevo en c_{i_m} : dos aristas no rulos para r_0 . Para cualquier j , $1 \leq j \leq m$, el camino entra en r_{i_j} por $c_{i_{j-1}}$ y sale de r_{i_j} por c_{i_j} : dos más aristas no rulos para r_{i_j} . Claramente, se puede asignar así un número par de nodos no rulos a cada nodo del grafo (siendo ese número 0 cuando el nodo no ocurre en el recorrido r). \square

Demostración del teorema 3.8. Si hay un ciclo de Euler, por el lema 3.9 cada nodo tiene grado par en el grafo porque todas las aristas del grafo están en el ciclo.

El recíproco va a ser consecuencia de demostrar que cuando se cumple la hipótesis que cada nodo tiene grado par entonces el algoritmo *cicloEuler* que se da más adelante tiene que producir un ciclo de Euler.

Primero veamos la idea de este algoritmo. Estamos dado un grafo conexo en que cada nodo está conectado a una arista y cada nodo tiene grado par. Elegimos cualquier nodo y construimos un ciclo sobre ese nodo (es decir, un ciclo con recorrido que empieza y termina con ese nodo). Repetimos lo siguiente hasta usar todas las aristas:

rep: Elegir un nodo en el recorrido del ciclo o camino nulo que tiene una arista no usada todavía. Construir un ciclo sobre ese nodo empezando con esa arista y usando solo aristas no usados todavía. Obtener nuevo ciclo insertando ese ciclo en el ciclo dado sobre el nodo dado (ver figura).



La inserción del ciclo efg en el ciclo $abcd$, dando el ciclo $abefgcd$.

A esto le falta los detalles de “construir un ciclo” e “insertarlo en el ciclo dado”. Atendemos primero al anterior:

Habiendo determinado que un grafo $G = (N, A, P)$ es conexo y que todo nodo tiene grado par en G , llamamos $cicloEuler(G)$ donde $cicloEuler(G) = extender(\emptyset, A)$

donde

$extender(c, B) =$

si $(B = \emptyset)c$

sino

sea p un nodo de un recorrido de c tal que hay $a \in B$ con $p \in P(a)$.

sea $P(a) = \{p, q\}$.

sea $(d, D) = ciclo(a, q, B - \{a\})$

(“ a ” significa aquí el camino que consiste de la sola arista a)

sea e el resultado de insertar d en c donde ocurre p

$extender(e, D)$

y

$ciclo(c, q, B) =$

si (hay $a \in B$ tal que $P(a) = \{q, s\}$ para algún s)

$ciclo(ca, s, B - \{a\})$

sino (c, B)

DEFINICIÓN 3.10. Decimos que un par ordenado (c, B) donde c es una sucesión de aristas y B un conjunto de nodos es *extendible* si c es un ciclo o nulo y B es el conjunto de todas las aristas del grafo no usadas en c .

Como consecuencia del lema 3.9, si cada nodo tiene grado par y (c, B) es extendible entonces cada nodo del grafo tiene grado par en B .

Primero queremos demostrar

(*) Para una invocación de $extender(c, B)$ con (c, B) extendible, si la invocación $ciclo(c, q, B) = (d, D)$ entonces d es un ciclo y $D = B - \{\text{todas las aristas de } d\}$.

Una consecuencia de (*) es que cada vez que se invoca $extender(c, B)$ con B no vacío y (c, B) extendible resulta en una nueva invocación $extender(c', B')$ con (c', B') extendible y con B' un subconjunto propio de B . Y eso demuestra el algoritmo porque implica que en alguna iteración la invocación va a ser $extender(c, \emptyset)$ con (c, \emptyset) extendible, que implica que c es ciclo de Euler.

Para demostrar (*) tenemos que contender con una sucesión de llamadas recursivas de $ciclo$. Queremos demostrar que para cada una de esas llamadas a $ciclo$ su terna de argumentos tiene la propiedad que llamamos *ciclable* definida así:

DEFINICIÓN 3.11. Sea $extender(c, B)$ la última invocación de $extender$ y sea p, a como en el algoritmo. Una terna (d, q, C) se dice *ciclable* si

1. d es un camino de p a q que no repite aristas.
2. Toda arista de d está en B y $C = B - \{\text{todas las aristas de } d\}$.
3. Cada nodo $\notin \{p, q\}$ del grafo tiene grado par en C .
4. Si $q = p$ el grado de q en C es par.
5. Si $q \neq p$ entonces cada uno de p, q tiene grado impar en C .

Ahora si la última llamada $ciclo(d, q, C)$, la que encuentra que no hay arista en C conectada a q , tiene (d, q, C) ciclable podemos concluir que $q = p$, implicando que d es un ciclo que empieza y termina en p . Y eso junto con punto (2) de la definición de *ciclable* y el hecho que c, B es extendible implican que en la próxima llamada $extender(e, D)$, (e, D) es extendible.

Entonces tenemos que demostrar que cada llamada de $ciclo$ es con una terna de argumentos ciclable. Lo demostramos por inducción matemática: es ciclable en la primera llamada, y para cada llamada en que es ciclable, si engendra otra llamada el argumento es ciclable también. Y eso va a completar la demostración del teorema.

La primera llamada a $extender$ es $extender(\emptyset, A)$, y este argumento es claramente extendible. Lo que vamos a demostrar es que una sucesión de invocaciones de $ciclo$ que surgen de una llamada a $extender$ con argumentos extendibles, tienen argumentos ciclables. Entonces considere

$extender(c, B)$ con (c, B) extendible. Sean p, a, q como en el algoritmo. La invocación de interés es $ciclo(a, q, B - \{a\})$. Chequeamos cada punto de la definición de *ciclable* para verificar que $(a, q, B - \{a\})$ es *ciclable*:

1. a es una arista con $P(a) = \{p, q\}$, así que el camino con la sola arista a es de p a q . Claramente no repite aristas porque hay una sola.
2. Es consecuencia directa del hecho que (c, B) es extendible.
3. Por el lema 3.9 todo nodo tiene grado par en el conjunto de aristas en el ciclo c . Ya que B es el conjunto de todas las aristas del grafo no usadas en c , y cada nodo tiene grado par en el grafo, tiene que tener grado par en B . Ya que $P(a) = \{p, q\}$, cada nodo $r \notin \{p, q\}$ tiene el mismo grado en $B - \{a\}$ como en B , que es par porque (c, B) es extendible.
4. Si $q = p$ entonces a es rulo, y ya que p tiene grado par en B tiene grado par en $B - \{a\}$.
5. Si $q \neq p$ entonces a no es rulo, por tanto ya que tienen grado par en B tienen que tener grado impar en $B - \{a\}$ porque cada uno está conectado a a .

Entonces esta primera llamada de *ciclo* tiene argumento *ciclable*.

Ahora veremos que si una llamada de $ciclo(d, q, C)$ con (d, q, C) *ciclable* produce la nueva invocación $ciclo(da, s, C - \{a\})$, con a, s como en la definición de *ciclo*, entonces $(da, s, C - \{a\})$ es *ciclable*. Pasamos por los puntos de la definición de *ciclable*:

- (1) a une los nodos q y s . Siendo d un camino de p a q , da es un camino de p a s . d no repite aristas, y a no puede ser arista de d pues (d, q, B) es *ciclable*.
- (2) Toda arista de d está en B y $C = B - \{\text{las aristas de } d\}$, pues d, q, C es *ciclable*. $a \in C \subset B$, así que todas las aristas de da están en B . Ahora $C - \{a\} = B - \{\text{aristas de } d\} - \{a\} = B - \{\text{aristas de } da\}$. Se cumple este punto.

Para propiedades 3,4,5 podemos suponer que a no es rulo, es decir que $s \neq q$, porque si a es rulo entonces $C - \{a\}$ tiene las mismas aristas no rulos que C . Entonces $(da, q, C - \{a\})$ va a ser *ciclable* porque (d, q, C) lo es.

- (3) Supongamos que $r \notin \{p, s\}$. Tenemos que demostrar que r tiene grado par en $C - \{a\}$.
 - *Caso* $q = p$. r tiene grado par en C . Pero $r \notin \{q, s\}$, por tanto r tiene el mismo grado en $C - \{a\}$ como en C : par.
 - *caso* $q \neq p$. Entonces q y p tienen grado impar en C . Ya que a no es rulo q adquiere grado par en $C - \{a\}$. Entonces si $r = q$ tiene grado par en $C - \{a\}$; si $r \neq q$, ya que $r \neq s$, $r \notin \{q, s\}$ y por tanto tiene el mismo grado en $C - \{a\}$ como en C : par porque $r \notin \{p, q\}$.

- (4) Si $s = p$ entonces ya que a no es rulo el grado de p en $C - \{a\}$ es distinto de su grado en C . Ahora $p \neq q$ porque sino a sería rulo y por tanto p tiene grado impar en C , y por tanto par en $C - \{a\}$.

- (5) Supongamos que $s \neq p$. Entonces $s \notin \{p, q\}$ porque si fuera $s = q$, a sería rulo. Por tanto s tiene grado par en C , y entonces impar en $C - \{a\}$ porque s está conectado al no rulo a . En cuanto del grado de p en $C - \{a\}$, si $p = q$ entonces p tiene grado par en C pero está conectado a la arista no nulo a y por tanto tiene grado impar en $C - \{a\}$; y si $p \neq q$ tiene grado impar en C pero en ese caso $p \notin \{q, s\}$, es decir no está conectado a la arista no rulo a , y por tanto sigue con grado impar en $C - \{a\}$. Así se cumple (5) para que sea *ciclable* $(da, s, C - \{a\})$.

Esto completa la demostración del teorema. \square

Un algoritmo es implícito en esa demostración, pero requeriría especificar el subalgoritmo “insertar sobre un nodo q ”. Es más sencillo hacer la inserción del nuevo ciclo mientras estamos engendrandolo, como se explica en el ejercicio 7.

4. Estructuras de datos para grafos

Expresiones como “ d es vacío” y “ $c(i \rightarrow d_1)$ ” no se entienden por un compilador. Un lenguaje como *Maple* se acerca a esta riqueza de expresión pero los algoritmos sobre grafos son notorios

por largos en su ejecución y los procesadores como el de *Maple* son lerdos – el precio inevitable de la riqueza de expresión.

Entonces tenemos que usar un lenguaje como *C* o *Pascal*, y enseñarle cómo trabajar con conjuntos y funciones definidas en datos.

Definimos una *estructura de datos* como una representación de un tipo de datos en términos de otros. Observamos que los números están representados por estructuras de datos en el hardware de una computadora. Ahora nos toca representar conjuntos.

Hay muchas maneras de representar un conjunto, que se adecuan a distintos tipos de problemas, y antes de terminar vamos a ver varias, pero ahora, para introducir el concepto, vamos a ver una sola. Todo lenguaje de programación nos facilita sucesiones (alias arreglo, array) y se puede representar un conjunto de objetos de tipo T como una sucesión sobre T , es decir, una sucesión s en que cada s_i es un objeto de T . Entendemos que un objeto t está en el conjunto si $s_i = t$ para algún s_i .

Tenemos que saber el largo de s para poder determinar si el t dado está en el conjunto. Entonces dedicamos una variable n a decirnos cuántos objetos hay en el conjunto en un momento dado.

Primero vamos a definir funciones que trabajan con estas representaciones, haciendo como siempre libre uso de recursión. Después, reconociendo los problemas de memoria y tiempo de ejecución que trae el uso de recursión, vamos a expresarlas nuevamente en forma imperativa. Las funciones que vamos a ver para el algoritmo del ciclo de Euler son más complicadas que las que hemos visto, y cuando sacamos la recursión son más complicadas aún. La idea es que la construcción de un programa se divide en etapas, empezando con la expresión matemática pura, progresando a representar la expresión matemática con otra que representa algunos tipos de datos en otros más cercanos a los que son suministrados por el lenguaje de programación (en algunos casos conviene efectuar esto en dos o más etapas, cada una acercándose más al lenguaje de programación), luego transformando las funciones en pseudocódigo imperativo, y finalmente traduciendo el pseudocódigo en código.

Primero entonces las funciones $x \in C$, $C \cup x$ (abreviatura de $C \cup \{x\}$) y $C - x$, donde el conjunto C se representa por una sucesión s de largo n .

```

 $t \in (s, n) =$ 
  si (hay  $i$  con  $1 \leq i \leq n$  y  $t = s_i$ ) sí sino no
 $(s, n) \cup t =$ 
  si  $t \in (s, n)$   $(s, n)$ 
  sino  $(s(n+1 \rightarrow t), n+1)$ 
 $(s, n) - t =$ 
  si (hay  $i$  con  $1 \leq i \leq n$  y  $t = s_i$ )  $(s(j \rightarrow s_{j+1}$  para  $i \leq j < n), n-1)$ 
  sino  $(s, n)$ 

```

Ahora veamos el pseudocódigo imperativo, que usa $\{\dots\}$ en lugar de **begin...end** como delimitadores de bloques, y, más importante, ni usa delimitadores de bloques pues se puede mostrar el bloque por la sangría. Seguimos usando s_i para el i -ésimo término de la sucesión s , porque es pseudocódigo y por tanto un vehículo de comunicación entre seres humanos. Pero si el nivel de los subscriptos llega a ser pesado, nos reservamos el derecho de usar la forma $s(i)$ (la forma $s[i]$ usada por muchos lenguajes es para simplificar el trabajo de los que hacen compiladores, y no tiene ninguna ventaja sintáctica). El comando *contestar e* entrega el valor e al usuario como valor de la función que invocó, y por tanto termina la actividad de la subrutina, que devuelve control al punto donde ocurrió la llamada.

Aquí es muy importante aclarar que las variables se pasan por nombre, es decir, cualquier cambio que las hace se efectúa en el espacio del usuario.

```

 $t \in (s, n)$ 
  para  $i = 1$  hasta  $n$ 
    si  $(t = s_i)$  contestar sí

```

contestar no (no hace falta “sino” porque no se puede llegar aquí si la condición anterior era falsa)

Para las operaciones de agregar un elemento a un conjunto y la de sacar uno, en cada aplicación tenemos que pensar con cuidado si podemos arreglarla con la operación $C = C \cup \{t\}$, que sobrescribe la representación de C en la memoria, o si vamos a necesitar el conjunto viejo de nuevo. En las aplicaciones que tenemos a mano, como en la gran mayoría de las aplicaciones, es suficiente representar $C = C \cup \{t\}$, y $C = C - \{t\}$.

```

agregar( $x, s, n$ )
  si( $x \notin (s, n)$ )
     $n = n + 1$ ;  $s_n = x$  (Acuérdese que estos cambios se efectúan en el espacio del usuario.)
sacar( $x, s, n$ )
  para( $i = 1$  hasta  $n$ )
    si( $s_i = x$ )
      para( $i$  hasta  $n - 1$ )  $s_i = s_{i+1}$ 
   $n = n - 1$ 

```

Para la representación de una función, notamos primero que una función es un conjunto de pares ordenados. Pero aunque la representación va a ser similar a la de un conjunto, hay una diferencia importante en que este conjunto no puede tener dos pares (x, y) y (x, z) con $y \neq z$. Además no nos sirven mucho las operaciones que hemos visto para conjuntos; necesitamos representar $f = f(x \rightarrow y)$, $f = sacar(x, C)$, que quiere decir remover x del dominio de definición de f si es que está allí, y, muy importante, la evaluación $f(x)$. En lugar de representar la función directamente como conjunto de pares ordenados, preferimos aquí simplificar la sintaxis teniendo dos sucesiones x, y que significan $f(x_i) = y_i$.

Primero veamos la expresión matemática de estas funciones:

```

( $x, y, n$ )( $t$ ) = (hablamos de  $f(t)$  pero  $f$  es representada por  $(x, y, n)$ )
  si ( $x_i = t$  para algún  $i$ ,  $1 \leq i \leq n$ ) contestar  $y_i$ 
  (no hay un “sino” – si no está definida, falta un valor para alguna expresión)
( $x, y, n$ )( $t \rightarrow u$ ) =
  si ( $x_i = t$  para algún  $i$ ,  $1 \leq i \leq n$ ) ( $x, y(i \rightarrow u), n$ )
  sino ( $x(n + 1 \rightarrow t), y(n + 1 \rightarrow u), n + 1$ )

```

Ahora vemos las representaciones de estas operaciones en pseudocódigo:

```

cambiar( $x, y, n, t, u$ )
  para( $i = 1$  hasta  $n$ )
    si( $x_i = t$ )
       $y_i = u$ 
  volver
 $n = n + 1$ 
 $x_n = t$ 
 $y_n = u$ 
eval( $x, y, t$ )
  para( $i = 1$  hasta  $n$ )
    si( $x_i = u$ ) contestar  $y_i$ 
  error

```

Apliquemos estas consideraciones al ciclo de Euler. ¿Cómo vamos a representar el grafo? No hace falta representar explícitamente los conjuntos de nodos y aristas, sino sólo la función P , tomando las aristas como números naturales $1, 2, \dots$ y los nodos también como un conjunto de números naturales (no necesariamente consecutivos en esta aplicación). Para representar P usamos dos sucesiones $N1$ y $N2$ de largo igual al número n de aristas en el grafo, e interpretamos que $P(i) = \{N1_i, N2_i\}$.

```
cicloEuler( $N1, N2, n$ ) =
```

```

    extender({},  $\emptyset(i \rightarrow i \text{ para } 1 \leq i \leq n), n)$ 
donde
    extender( $c, l, s, m$ ) = ( $c$  es un ciclo,  $B$ , representado por  $(s, m)$ , las aristas no en  $(c, l)$ )
    si ( $m = 0$ ) ( $c, l$ )
    sino
        sea  $r$  un recorrido del camino  $c$ 
        sea  $a \in (s, m)$  tal que para algún  $i \leq \text{largo}(r)$ ,  $N1_a = r_i | N2_a = r_i$ 
        si  $N1_a = r_i$  sea  $q = N2_a$  sino sea  $q = N1_a$ 
        sea  $(d, D) = \text{ciclo}(\{a\}, r_i, q, (s, m) - \{a\})$ 
        extender(insertar( $d, c, i$ ),  $D$ )
y
    ciclo( $c, p, q, s, m$ ) =
    si ( $p = q$ ) ( $c, s, m$ )
    sino
        sea  $a \in (s, m)$  con  $N1_a = q | N2_a = q$ 
        si ( $N1_a = q$ ) sea  $r = N2_a$  sino sea  $r = N1_a$ 
        ciclo( $ca, p, r, (s, m) - \{a\}$ )

```

Ahora el pseudocódigo. Pasamos todos los argumentos por variables globales, es decir, por memoria visible a todas las rutinas (no es imprescindible arreglarlo así, pero evita la duda sobre si los argumentos se pasan por nombre o valor). En las variables globales tenemos un camino C con su largo $\text{largo}C$, que será el ciclo producido hasta ahora por *extender*, otro camino c con su largo $\text{largo}c$ que será el camino producido hasta ahora por *ciclo*, la sucesión s y un natural m tal que (s, m) representa el conjunto B de aristas, y las sucesiones $N1$ y $N2$ representando el grafo como antes.

Ponemos otros dos argumentos de *ciclo* en las variables globales p y q , siendo el nodo buscado por *ciclo* y el nodo actual de *ciclo* respectivamente.

Ahora podemos definir

```

ciclo()
    mientras( $q \neq p$ )
        para( $i = 1$  hasta  $m$ )
            si ( $N1_{s_i} = q$ )  $q = N2_{s_i}$ 
            sino si ( $N2_{s_i} = q$ )  $q = N1_{s_i}$ 
            sino ir a proxI
            largoc+ = 1
             $c_{\text{largoc}} = i$ 
            sacar( $i, s, m$ )
            ir a proxInvocación
        proxI:
        proxInvocación:
        si( $i > m$ ) no hay ciclo de Euler, falló

```

Para *extender*, tenemos que buscar la arista como en la especificación, poner valor correspondiente en p , copiar el vecino a q , llamar a *ciclo*, insertar el ciclo en el ciclo grande, y repetir hasta que $m = 0$, que significa que B es vacío.

Ponemos el ciclo grande en la variable global C , de largo $\text{largo}C$. Empezamos con $\text{largo}C = 0$.

Para más eficiencia también hacemos lo que pensamos al principio: ponemos en $(r, \text{largo}R)$ el recorrido del camino en C ($\text{largo}R$ necesariamente es $\text{largo}C + 1$ pero ofusca un poco usarlo) y que es más conveniente engendrar r en la rutina *insertar*.

```

cicloEuler()
    para( $i = 1$  hasta  $n$ )  $s_i = i$ 
     $m = n$  //está listo el argumento B de extender

```

```

    largoC = 0
    r1 = N11 (cualquier nodo es recorrido del camino vacío)
    largoR = 1
    extender()
    extender()
    mientras(m > 0)
        para(i = 1 hasta largoR)
            para (j = 1 hasta m)
                si(N1j = ri)
                    q = N2j
                sino si(N2j = ri)
                    q = N1j
                sino ir a proxj
                p = ri
                c1 = i
                largoc = 1
                sacar(j, s, m)
                ciclo()
                insertar(i)
                ir a nuevaInvocación
            proxj:
        ¡fracaso!
    nuevaInvocación: (marcando el fin del bucle de mientras)

```

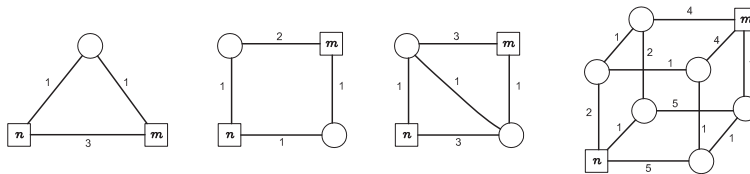
Dejamos *insertar* como ejercicio. No se olvide de que tiene que insertar el recorrido en r , $largoR$ también.

5. Camino mínimo

Hay un montón de problemas en que cada arista tiene un costo de algún tipo, como por ejemplo distancia, o tiempo, o precio del boleto de avión.

DEFINICIÓN 3.12. Un *grafo con costos* es (N, A, P, c) donde (N, A, P) es un grafo y c es una función de las aristas a los números reales no negativos.

El *costo* de un camino a_1, a_2, \dots, a_n es, desde luego, $\sum_1^n c(a_i)$. Consideramos ahora el problema de encontrar el camino de mínimo costo entre dos nodos n y m . En tres de los grafos de la figura es fácil, pero en el otro no lo es, y se ve que hace falta un buen algoritmo.



Análogamente a lo que hicimos con $conexos(n, m)$ y $camino(n, m)$, hacemos primero un algoritmo que evalúa el costo del camino mínimo entre n , m , y luego recuperamos el camino agregando una función *portador*, para llegar al *algoritmo de Dijkstra*.

El principio del algoritmo de Dijkstra es éste: Supongamos que tengo un conjunto C de nodos, y para cada nodo p en C , el costo $d(p)$ de un camino mínimo de n a p . Si $m \in C$, entonces $d(m)$ es el resultado que buscamos, y si no, elegimos una arista con un pie en C y el otro afuera, es decir, una arista a tal que $P(a) = \{p, q\}$ con $p \in C$ y $q \notin C$. Hasta allí igual que en el algoritmo *conexos*, pero aquí imponemos otra condición sobre esa arista a : que $d(p) + c(a)$ sea el mínimo posible.

Ahora el costo de un camino mínimo entre n y q es $d(p) + c(a)$, que podemos ver así: Sea a_1, \dots, a_k un camino entre n y q de costo menor que $d(p) + c(a)$. Sea i el primer número en

que el recorrido de a_1, \dots, a_i termina en un nodo no en C . Entonces $P(a_i) = \{r, s\}$ con $r \in C$ y $s \notin C$. El costo del camino a_1, \dots, a_{i-1} entre n y r no puede ser menor que $d(r)$, y el costo de a_1, \dots, a_i es este costo más $c(a_i)$. Concluimos que $d(r) + c(a_i)$ es menor o igual que el costo de a_1, \dots, a_i , que desde luego es menor o igual que el costo de a_1, \dots, a_k , que supuestamente es menor que $d(p) + c(a)$. Pero mire lo que esto nos está diciendo: que $d(r) + c(a_i) < d(p) + c(a)$, contradiciendo la elección de a .

Esto nos dice que si tenemos cualquier conjunto C con los costos de los mínimos caminos entre n y cada uno de sus nodos, podemos agregarle un nodo q y saber $d(q)$, y por tanto cumplir la condición para agregar otro nodo r y saber $d(r)$, etc., hasta finalmente, por lo menos si m es conexo con n , agregar el nodo m y saber $d(m)$.

Pero, ¿donde conseguimos el primer conjunto C que nos conduce a esta feliz conclusión? Fácil: $C = \{n\}$. $d(n) = 0$, el costo del camino vacío que va de n a n .

Ahora podemos expresar el algoritmo:

$costo1(n, m) = costo1(\{n\}, m, \emptyset(n \rightarrow 0))$

donde

$costo1(C, m, d) =$

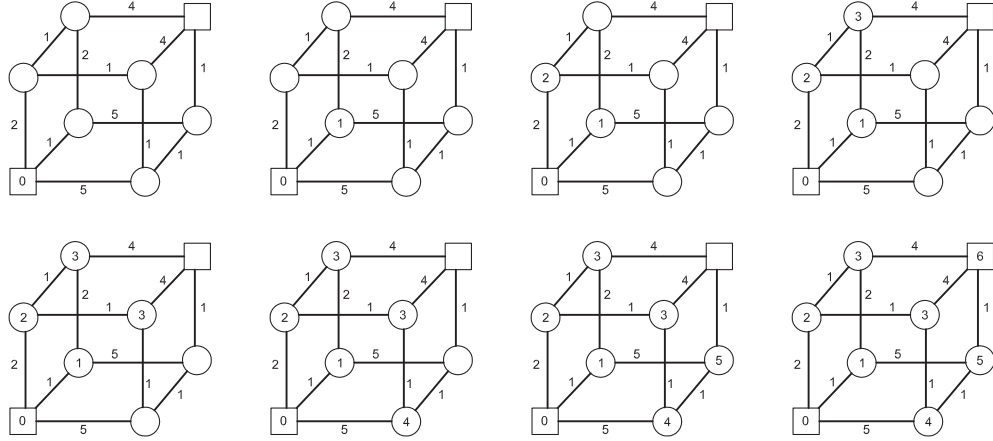
sea a una arista tal que $P(a) = \{p, q\}$ con $p \in C$ y $q \notin C$

y tal que $d(p) + c(a)$ es lo más pequeño posible

si ($q = m$) $d(p) + c(a)$

sino $costo1(C \cup \{q\}, m, d(q \rightarrow d(p) + c(a)))$

La figura muestra diagramáticamente los argumentos en las sucesivas llamadas a *costo1* para el grafo de la derecha en la figura previa. Los nodos con un número escrito adentro son los que pertenecen a C , y el número es el valor de d para ese nodo.



Notamos que además de inscribir el costo en el nodo, podríamos haber puesto una flecha que indica la arista que trajo el nodo. Así el diagrama permitiría construir el camino mínimo. Agregamos a los argumentos de *costo1* la función, inicialmente vacía, que da, para cada nodo traído por otro nodo a C , la arista por la cual fue traído. La idea es la misma que la que se usó en el algoritmo *camino*, con la función indicando la arista respectiva. Éste es el argumento *portador* del algoritmo ayudante *extender*, que cuando termina entrega esa función para que *construir*, el mismo que se usó en *camino*, extraiga el camino mínimo:

$caminoMinimo(G, n, m) =$

si ($m = n$) $\{\}$

sino $extender(G, \{n\}, \text{función vacía}, m, \emptyset(n \rightarrow 0))$

donde

$extender(G, C, portador, m, d) =$

sea a una arista tal que $P(a) = \{p, q\}$ con $p \in C$ y $q \notin C$

y tal que $d(p) + c(a)$ es lo más pequeño posible

si $(q = m)$ *construir* $(m, \text{portador}(m \rightarrow a))$
sino *extender* $(G, C \cup \{q\}, \text{portador}(q \rightarrow a), m, d(q \rightarrow d(p) + c(a)))$

En muchas aplicaciones, es útil tener una matriz M tal que para dos nodos m, n cualesquiera, $M_{m,n}$ es el costo del camino mínimo entre m y n . Para un grafo de N nodos uno podría pensar que esto se hace por $N(N-1)/2$ invocaciones al algoritmo *costo*. Pero el *algoritmo de Floyd* lo hace un poco más sencillamente. Veamos.

Sea V la matriz tal que $V_{n,n} = 0$ para todo nodo n , y para nodos distintos m, n , que están ligados por alguna arista, $V_{m,n} = \min\{c(a) : P(a) = \{m, n\}\}$. Es decir, es el mínimo de los costos de las aristas ancladas en m y n , y para nodos distintos m, n que no están ligados por ninguna arista, sea $V_{m,n}$ igual a algún número grande, como la suma de los costos de todas las aristas, que designamos “ ∞ ”.

Podemos decir que V nos da el costo mínimo de caminos de largo menor o igual a 1, cuando hay tal camino, y ∞ si no hay. Ahora considere la matriz V^2 , y el motivo para decirle “ V^2 ” va a aparecer enseguida, definida por

$$V_{m,n}^2 = \min\{V_{m,p} + V_{p,n} : p \text{ un nodo}\}$$

V^2 da el costo mínimo de caminos de largo menor o igual que 2. Le llamamos V^2 por analogía con el cuadrado de una matriz M :

$$M_{i,j}^2 = \sum_{1 \leq k \leq N} M_{i,k} M_{k,j}$$

En lugar de \sum , V^2 tiene \min , y en lugar de multiplicación, suma. Si definimos así el “producto” de dos matrices M y L sobre pares de nodos:

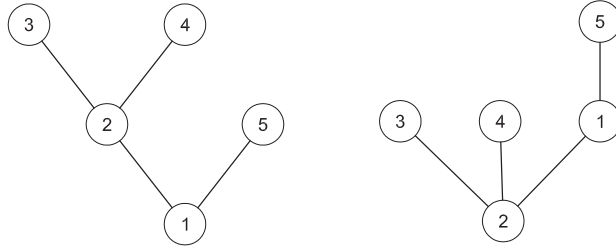
$$(ML)_{m,n} = \min\{M_{m,p} + L_{p,n} : p \text{ un nodo}\}$$

entonces $V_{m,n}^k$ es el costo del camino mínimo de largo menor o igual que k , cuando hay, y ∞ si no hay.

El algoritmo de Floyd consiste en sacar las potencias de V hasta que no cambien más.

6. Árboles

Un *árbol* es un grafo conexo sin ciclos. La razón por la que se llama árbol es que lo podemos dibujar eligiendo algún nodo como la “raíz” y dibujándolo al fondo, dibujar todos los vecinos un nivel más arriba, todos los nuevos vecinos de ellos un nivel más arriba que eso, y así sucesivamente, como en la figura, que muestra el mismo árbol eligiendo dos nodos distintos para ser raíz.



Un ejemplo típico en donde aparece la importancia de un árbol es el siguiente. En una red de comunicación, queremos asegurar la conectividad de la red como grafo, para que ningún nodo quede desconexo con otro. Pero los nodos pueden actuar de posta, recibiendo y pasando adelante mensajes destinados a otro nodo. Se decide hacer especialmente confiables algunas conexiones, para que mientras sigan funcionando éstas, la red siga conexa. Desde luego estas conexiones especiales van a costar más, y por tanto no vamos a poner más que las necesarias. Entonces, las conexiones especiales van a formar un árbol, porque si hay un ciclo hay dos caminos entre algún par de nodos, y podemos reemplazar alguna conexión especial con una ordinaria.

Este árbol se llama además *árbol cubridor* porque sus aristas son todas aristas del grafo original, y lo mantiene conexo. He aquí un algoritmo para construir un árbol cubridor:

$\text{árbolCubridor}(G) = \text{cub}(\{p\}, \{\})$

donde

$\text{cub}(C, B) =$ (C un conjunto de nodos, B un conjunto de aristas)

si (C es todos los nodos) B

sino $\text{cub}(C \cup \{q\}, B \cup \{a\})$

donde a es una arista con $P(a) = \{p, q\}$ con $p \in C$ y $q \notin C$

¿Cómo sabemos que el conjunto de aristas entregado a *árbolCubridor* por *cub* va a formar un árbol cubridor? Por demostrar que, en cada llamada $\text{cub}(C, B)$, el grafo (C, B, P) (los puristas pueden leer “la restricción de P a B” en lugar de P si quieren) es un árbol; como sigue: En la primera llamada, con $C = \{p\}$ y B vacío, esto claramente se cumple (¿hay ciclo? ¿es conexo?). Supongamos que en una llamada $\text{cub}(C, B)$, (C, B, P) es un árbol. La próxima llamada, si hay, es a $\text{cub}(C \cup \{q\}, B \cup \{a\})$. Es claro que la conectividad de (C, B, P) implica la conectividad de $(C \cup \{q\}, B \cup \{a\}, P)$. La cuestión es si hay ciclo. Si hay ciclo, tiene que intervenir la nueva arista a y el nuevo nodo q, sino sería un ciclo de (C, B, P) . Pero en el recorrido de un ciclo, todo nodo tiene dos aristas, pero en el grafo $(C \cup \{q\}, B \cup \{a\}, P)$ q tiene una sola arista. Lista.

Ahora sea G un grafo con costos, y busquemos un árbol cubridor de costo mínimo, es decir, un árbol cubridor tal que la suma de los costos de sus aristas es lo mínimo posible.

Un algoritmo (el de Prim), se construye modificando el algoritmo *árbolCubridor* para que en lugar de elegir cualquier a con un pie en C y el otro fuera de C, se elige la arista de menos costo con un pie adentro y el otro afuera:

$\text{árbolCubridorMin}(G) = \text{cub}(\{p\}, \{\})$

donde

$\text{cub}(C, B) =$ (C un conjunto de nodos, B un conjunto de aristas)

si (C es todos los nodos) B

sino $\text{cub}(C \cup \{q\}, B \cup \{a\})$

donde a es una arista con $P(a) = \{p, q\}$ con $p \in C$ y $q \notin C$ con $c(a)$ el mínimo posible

La primera cosa que se nos ocurre para demostrar esto es adaptar la demostración del algoritmo *árbolCubridor* que acabamos de ver. Eso sería demostrar que si llamamos a $\text{cub}(C, B)$ con (C, B, P) siendo un árbol cubridor mínimo para el grafo que resulta de restringir G a C, entonces la próxima llamada, si hay, preservaría la misma propiedad. Pero un intento a eso resulta engorroso.

La clave es demostrar que si existe un árbol cubridor mínimo para G que contiene todas las aristas de B, entonces existe uno que tiene todas las aristas de $B \cup \{a\}$.

Necesitamos primero este teorema.

TEOREMA 3.13. (El teorema de la cadena de margaritas.) Sea T un grafo con n nodos. Las siguientes afirmaciones son equivalentes, es decir, o el grafo T goza de todas las seis propiedades, o de ninguna.

- (a) T es un árbol.
- (b) T no tiene ciclos y tiene $n - 1$ aristas.
- (c) T es conexo y tiene $n - 1$ aristas.
- (d) T es conexo, pero sacando cualquier arista queda no conexo.
- (e) Existe exactamente un camino sin repetición entre dos nodos cualesquiera.
- (f) T no tiene ciclos, pero agregando cualquier nueva arista, se formará un ciclo.

Para demostrar éso, vamos a demostrar que (a) implica (b) implica (c) implica (d) implica (e) implica (f) implica (a).

Dado (a). T no tiene ciclos por definición de árbol. Para mostrar que tiene $n - 1$ aristas, consideramos $\text{árbolCubridor}(T)$. Tiene $n - 1$ aristas porque agrega un nodo para cada arista, empezando con un nodo y ninguna arista, y no se va a colgar porque T, árbol, es conexo. Pero

si cubre a T , es T porque T no puede tener ninguna otra arista, porque agregando una arista a un grafo conexo se forma un ciclo.

Dado (b). Para (c) hay que demostrar que T es conexo. Si no lo es, lo dividimos en componentes conexos T_1, \dots, T_k , y sea n_i el número de nodos de T_i . Cada T_i es un árbol porque es conexo y no tiene ciclos (porque T no los tiene). Entonces T_i tiene $n_i - 1$ aristas, aplicando (a) implica (b). Claro que si $k > 1$ esto no nos da suficiente aristas.

Dado (c). Supongamos que, sacando una arista, el grafo T' que resulta es conexo. Entonces tiene n nodos. Pero siendo conexo le aplicamos el algoritmo para el árbol cubridor, y vamos a ver $n - 1$ iteraciones para agregar los $n - 1$ nodos que faltan, y entonces vamos a ver $n - 1$ aristas, pero T' tiene $n - 2$ aristas.

Dado (d). Si hay dos caminos entre dos nodos, sacamos una arista de uno de esos caminos (que no esté en el otro camino) y el grafo queda conexo, contradiciendo (d).

Dado (e). Entonces no hay ciclo porque daría dos caminos entre dos nodos. Si agrego una arista entre nodo n y nodo m entonces el camino que existía entre n y m y esta arista me dan un ciclo.

Dado (f). Para saber (a) falta sólo saber que T es conexo. Si no es conexo, hay dos nodos n y m que no tienen camino entre sí. Entonces si agrego una arista entre n y m no creo un ciclo, contradiciendo (f). \square

Ahora demostramos el algoritmo del árbol cubridor mínimo. Vamos a demostrar que, para cada llamada de $cub(C, B)$ tal que existe un árbol cubridor mínimo con todas las aristas de B , la próxima llamada, $cub(C \cup \{q\}, B \cup \{a\})$, tiene la misma propiedad, donde $P(a) = \{p, q\}$ con $p \in C$ y $q \notin C$. Sea T un árbol cubridor mínimo, que tiene todas las aristas de B . Estas aristas tienen el conjunto C de nodos. Si resulta $a \in T$ no hay nada para demostrar. Supongamos que $a \notin T$. Sea π el camino en T desde q a p . Sea $\pi = \pi_1 b \pi_2$ donde π_1 tiene recorrido en el complemento de C de q a $q' \notin C$, $P(b) = \{q', p'\}$ con $p' \in C$, y π_2 es de p' a p . Por la elección de a hecho por el algoritmo, $c(a) \leq c(b)$. Sea U el grafo con los nodos de T (que son los de G), y, siendo A_T las aristas de T , sean las aristas de U , $U_A = A_T - \{b\} \cup \{a\}$. Es claro que U tiene $n - 1$ aristas y n nodos. Si resulta que U es conexo, por el teorema de la cadena de margaritas va a ser un árbol, y por tanto un árbol cubridor porque sus nodos son los de G . Además el costo de U no es mayor que el de T porque $c(a) \leq c(b)$, así que U será el árbol cubridor mínimo buscado para la próxima recursión.

Entonces nos dirigimos a la conectividad de U . Hay un camino en U de p' a q' , a saber $\pi = \pi_2 a \pi_1$. Sea r y s dos nodos. Si el camino en T no usa b es un camino de U . Si usa b , reemplazamos b con a o π o π^R , dependiendo de como el camino cruza b , para tener un camino en U de r a s .

Ya que la primera llamada de cub , $cub(\{p\}, \{\})$ satisface esta propiedad, todas las llamadas la cumplen, especialmente la última. Listo.

7. Redes de Transporte

Dado un grafo, si ponemos una flecha en cada arista tenemos lo que se llama un grafo dirigido. En el problema que vamos a estudiar, no hace falta considerar la posibilidad de tener dos aristas de un nodo p a un nodo q así que podemos simplificar la notación designando cada arista con nodo origen p y nodo destino q por (p, q) . También excluimos la posibilidad de una arista (p, p) . Entonces podemos nombrar un camino con solo nombrar su recorrido.

Se dice que un grafo dirigido G es *conexo* si es conexo considerado como grafo no dirigido, es decir que el grafo no dirigido (N, A, P) con N igual a los nodos de G , A igual a las aristas de G , y para cada arista (p, q) , $P(p, q) = \{p, q\}$, es conexo.

DEFINICIÓN 3.14. Una *red de transporte* es un grafo dirigido conexo con una función w de las aristas a los números reales no negativos, que tiene exactamente un nodo α tal que no hay ninguna arista (p, α) y exactamente un nodo ω tal que no hay ninguna arista (ω, p) .

La red representa capacidad de flujo de algo – mensajes, petróleo, tráfico vehicular – y se quiere calcular el flujo máximo de que es capaz, dado que el flujo en cada arista (p, q) no puede exceder a su capacidad $w(p, q)$, y que todo el flujo entrando en un nodo tiene que ser igual en cantidad a todo el flujo que sale del nodo, exceptuando la fuente α y el sumidero ω . Se considera que la fuente produce la cosa que fluye y el sumidero la consume.

Más formalmente:

DEFINICIÓN 3.15. Un *flujo* de una red de transporte es una función ϕ de las aristas a los números reales no negativos tal que

1. Para cualquier arista (p, q) , $\phi(p, q) \leq w(p, q)$.
2. Para cualquier nodo $p \notin \{\alpha, \omega\}$, $\sum_{\text{aristas } (q,p)} \phi(q, p) = \sum_{\text{aristas } (p,q)} \phi(p, q)$

Por ejemplo, consideramos la red con nodos $\{\alpha, a, b, c, d, \omega\}$ con aristas

$$(\alpha, a), (\alpha, c), (a, b), (c, b), (c, d), (d, a), (b, \omega), (d, \omega)$$

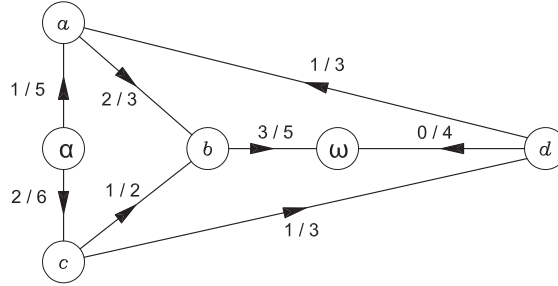
con capacidad definida por

$$w(\alpha, a) = 5, w(\alpha, c) = 6, w(a, b) = 3, w(c, b) = 2, w(d, a) = 3, w(b, \omega) = 5, w(d, \omega) = 4, w(c, d) = 3$$

y flujo definida por

$$\phi(\alpha, a) = 1, \phi(\alpha, c) = 2, \phi(a, b) = 2, \phi(c, b) = 1, \phi(d, a) = 1, \phi(b, \omega) = 3, \phi(d, \omega) = 0, \phi(c, d) = 1.$$

La red está dibujada en la figura, con la anotación $\phi(p, q)|w(p, q)$ cerca de cada arista (p, q) .



Extendemos ϕ a cualquier par p, q de nodos diciendo $\phi(p, q) = 0$ si (p, q) no es arista de la red.

Parece que debe ser $\sum_p \phi(\alpha, p) = \sum_p \phi(p, \omega)$ ya que no se acumula materia en ningún nodo intermedio. Pero la demostración formal de esto es medio delicada. En anticipación de su demostración formal, definimos el flujo en R resultante del flujo ϕ como

$$\phi(R) = \sum_{\text{nodos } p} \phi(p, \omega).$$

DEFINICIÓN 3.16. Sean C y D dos conjuntos disjuntos de nodos. $\phi(C, D) = \sum_{p \in C, q \in D} \phi(p, q)$.

Notación Por $-C$ queremos decir el conjunto de todos los nodos de R que no pertenecen a C .

LEMA 3.17. Sea C un conjunto de nodos que no contiene ni α ni ω . Entonces $\phi(C, -C) = \phi(-C, C)$.

Demostración. Para $|C| = 1$ la aserción es parte de la definición del flujo en una red de transporte. Sea $|C| = n$ para algún n tal que la aserción se cumple para todo conjunto de menos de n nodos. Sea $C = C_1 \cup C_2$ donde C_1 y C_2 son disjuntos y no vacíos. Por la hipótesis de inducción, $\phi(C_1, -C_1) = \phi(-C_1, C_1)$, pero $\phi(C_1, -C_1) = \phi(C_1, -C) + \phi(C_1, C_2)$ porque $-C_1 = -C \cup C_2$. Similarmente, $\phi(-C_1, C_1) = \phi(-C, C_1) + \phi(C_2, C_1)$. Entonces la hipótesis de inducción nos dice

$$\phi(C_1, -C) + \phi(C_1, C_2) = \phi(-C, C_1) + \phi(C_2, C_1)$$

Similarmente, la hipótesis de inducción aplicada a C_2 nos dice

$$\phi(C_2, -C) + \phi(C_2, C_1) = \phi(-C, C_2) + \phi(C_1, C_2)$$

Sumando las dos igualdades anteriores miembro a miembro y simplificando, resulta

$$\phi(C_1, -C) + \phi(C_2, -C) = \phi(-C, C_1) + \phi(-C, C_2)$$

o sea

$$\phi(C, -C) = \phi(-C, C)$$

y la inducción se extiende a C . \square

Un resultado de este lema, tomando C como $-\{\alpha, \omega\}$, es que $\sum_p \phi(\alpha, p) = \sum_p \phi(p, \omega)$, o sea $\phi(R)$, porque el primero es $\phi(\alpha, \omega) + \phi(\{\alpha, \omega\}, -\{\alpha, \omega\})$ y el segundo es $\phi(\alpha, \omega) + \phi(-\{\alpha, \omega\}, \{\alpha, \omega\})$.

Otro resultado es éste, que vamos a citar después:

TEOREMA 3.18. *Si C es un conjunto que contiene α y no contiene ω entonces $\phi(C, -C) - \phi(-C, C) = \phi(R)$.*

Demostración. Sea $C_1 = C - \{\alpha\}$ y $C_2 = -C - \{\omega\}$. Entonces $\phi(-C, C) = \phi(C_2, C_1)$. Ahora $\phi(C, -C) = \phi(\alpha, \omega) + \phi(\{\alpha\}, C_2) + \phi(C_1, C_2) + \phi(C_1, \{\omega\})$, así que

$$\phi(C, -C) - \phi(-C, C) = \phi(\alpha, \omega) + \phi(\{\alpha\}, C_2) + \phi(C_1, \{\omega\}) + (\phi(C_1, C_2) - \phi(C_2, C_1)),$$

y ahora calculamos $\phi(C_1, C_2) - \phi(C_2, C_1)$ aplicando el Lema a C_1 . Igualando $\phi(C_1, -C_1)$ y $\phi(-C_1, C_1)$ tenemos

$$\phi(C_1, \{\omega\}) + \phi(C_1, C_2) = \phi(\{\alpha\}, C_1) + \phi(C_2, C_1)$$

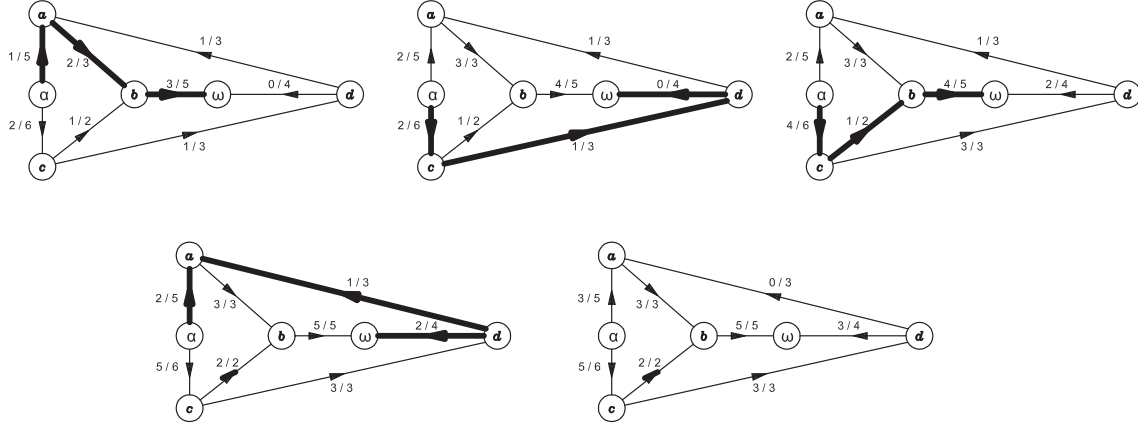
de donde

$$\phi(C_1, C_2) - \phi(C_2, C_1) = \phi(\{\alpha\}, C_1) - \phi(C_1, \{\omega\}).$$

Al sustituir eso queda:

$$\phi(C, -C) - \phi(-C, C) = \phi(\alpha, \omega) + \phi(\{\alpha\}, C_2) + \phi(\{\alpha\}, C_1) = \phi(R). \square$$

Volvemos al ejemplo de la figura, donde podemos percibir cómo es un algoritmo para sacar el flujo máximo posible, es decir un flujo ϕ tal que $\phi(R) \geq \phi'(R)$ para cualquier flujo ϕ' , en cuya demostración el resultado anterior es clave. La nueva figura muestra los cuatro pasos de un algoritmo que vamos a formalizar después. El primer paso está ilustrado con repetir la figura original pero con el camino $\alpha ab\omega$ destacado, que es lo que vamos a definir como *camino abierto*. Se elige cualquier camino como ése.



Si aumentamos el flujo en cada arista de este camino en 1, y los flujos en otras aristas quedan como están, el resultado sigue siendo un flujo, porque cada uno de los nodos a, b tiene 1 más de entrada y 1 más de salida, y así que condición 2 de flujos se cumple, y en ninguno de las tres aristas del camino es el nuevo flujo mayor que la capacidad de la arista. El segundo dibujo muestra esos cambios en ϕ y destaca otro camino abierto $\alpha cd\omega$. Podemos hacer lo mismo con este camino, salvo que aquí podemos aumentar el flujo en 2 en cada arista. Mostramos el resultado de los dos aumentos en el tercer dibujo, donde también destacamos el camino α, c, b, ω . Vemos que de la misma manera podemos aumentar el flujo en 1 más, y el resultado se ve en el

cuarto dibujo. Tenemos ahora un flujo total de 7. Buscamos otro camino desde α a ω en que podemos aumentar el flujo en la misma forma. Pero no hay. Todos los caminos desde α a ω pasan por alguna arista cuyo flujo es igual a su capacidad. Necesitaríamos un camino tal que $\min(\{w(p, q) - \phi(p, q) : (p, q) \text{ una arista del camino}\}) > 0$, pero no hay. Pero ahora miremos al “camino” α, a, d, ω , que es el recorrido de un camino en el grafo no dirigido obtenido por borrar las flechas indicadoras de dirección. Usamos las comillas porque no es camino en el grafo dirigido: va a contramano en la arista (d, a) . En este camino podemos sacar el 1 de flujo que va por la arista (d, a) , compensando eso con aumentar por 1 el flujo en (α, a) y en (d, ω) así preservando propiedad 2 para los nodos a y b . El resultado es un flujo de 8, que se ve en el quinto dibujo.

Observe lo que nos permitió un aumento en el último camino: En la arista en que el camino iba a contramano, el flujo era positivo, y en cada arista en que el flujo iba por mano correcta, el flujo era estrictamente menos que la capacidad de la arista.

Entonces, busquemos otro camino así en el grafo no dirigido. Pero resulta ahora que no hay: cada camino desde α a ω en el grafo no dirigido o bien va por mano correcta por una arista cuyo flujo iguala su capacidad, o bien a contramano por una arista cuyo flujo es 0. Vamos a ver que en este caso el flujo de la red es el máximo posible. Formalmente:

Sea R una red de transporte con capacidad w , flujo ϕ , fuente α y sumidero ω .

Sea G el grafo no dirigido derivado de R , es decir, G es el grafo no dirigido cuyas aristas y nodos son las aristas y nodos de R , y cuya función de nodos P se define por $P((p, q)) = \{p, q\}$.

Nos pueden interesar algunos caminos que van a contramano una sola vez, o varias veces. Si no hay un par de nodos con aristas en ambas direcciones sería suficiente especificar sólo el recorrido de nodos para indicar tal camino, pero en el caso general habría que juntar al recorrido un indicador si vamos contramano o no en cada punto. Por eso hablamos de un par, el recorrido p_1, \dots, p_k y la sucesión de indicadores d_1, \dots, d_{k-1} , superfluos para los (p_i, p_{i+1}) que tienen arista en una sola dirección, pero necesario para indicar cuál arista tomamos si hay dos.

DEFINICIÓN 3.19. Un *camino no dirigido* de R es un par (r, d) donde r es el recorrido p_1, \dots, p_k de un camino de G y d es una sucesión d_1, \dots, d_{k-1} de 0 y 1 tal que si $d_i = 0$ entonces (p_i, p_{i+1}) es una arista de R ((p_{i+1}, p_i) puede ser o no ser arista), y si $d_i = 1$ entonces (p_{i+1}, p_i) es una arista de R . Un *camino abierto* de R es un camino no dirigido (r, d) cuyo recorrido no repite nodos con la siguiente propiedad para $i \in \{1, \dots, k-1\}$:

1. Si $d_i = 0$ entonces $\phi(p_i, p_{i+1}) < w(p_i, p_{i+1})$, y
2. Si $d_i = 1$ entonces $\phi(p_{i+1}, p_i) > 0$.

Es decir, podemos pasar por mano correcta sólo si la arista puede recibir algo más de flujo, y podemos pasar a contramano sólo si la arista tiene algo de flujo para quitarle.

Un camino abierto (r, d) es *completo* si r empieza con α y termina con ω .

Dado un camino abierto completo en R se puede engendrar un nuevo flujo ϕ' para R tal que $\phi'(R) > \phi(R)$ como sigue: Sea

$$\Delta = \min(\min_{d_i=0}(w(p_i, p_{i+1}) - \phi(p_i, p_{i+1})), \min_{d_i=1}(\phi(p_{i+1}, p_i))).$$

Para cada i , si $d_i = 0$ entonces $\phi'(p_i, p_{i+1}) = \phi(p_i, p_{i+1}) + \Delta$, y si $d_i = 1$ entonces $\phi'(p_{i+1}, p_i) = \phi(p_{i+1}, p_i) - \Delta$.

Si la función ϕ' resulta ser un flujo (definición 3.15), entonces $\phi'(R) > \phi(R)$ porque ninguna arista (p, ω) puede ser contramano, entonces la que está en el camino abierto completo tiene un aumento positivo.

Vamos entonces a verificar que ϕ' es un flujo.

Cuando $d_i = 0$, $\Delta \leq w(p_i, p_{i+1}) - \phi(p_i, p_{i+1})$, y cuando $d_i = 1$, $\Delta \leq \phi(p_{i+1}, p_i)$; en el primer caso $\phi'(p_i, p_{i+1}) \leq w(p_i, p_{i+1})$ y en el segundo caso $w(p_{i+1}, p_i) \geq \phi(p_{i+1}, p_i) > \phi'(p_{i+1}, p_i) \geq 0$, entonces ϕ' cumple con condición (1) de la definición 3.15 y la restricción básica que un flujo es

no negativo. La condición (2) se cumple porque en cada i , $1 < i < k$, consideramos las cuatro posibilidades:

- $d_{i-1} = d_i = 0$. Entonces hay un aumento de Δ entrando en p_i y un aumento de Δ saliendo de p_i .
- $d_{i-1} = 0$ y $d_i = 1$. Hay un aumento de Δ entrando en p_i y una reducción de Δ entrando en p_i desde el otro lado.
- $d_{i-1} = 1$ y $d_i = 0$. Hay una reducción de Δ entrando en d_i y un aumento de Δ entrando en p_i desde el otro lado.
- $d_{i-1} = d_i = 1$. Hay una reducción de Δ saliendo en p_i y una reducción de Δ entrando de p_i .

En cada caso hay un aumento neto de 0 del flujo por p_i .

Entonces ϕ' es un flujo y $\phi'(R) > \phi(R)$.

Tenemos que demostrar que si seguimos aumentando caminos abiertos completos así hasta que no hay más, el flujo que encontramos así es máximo. Para aclarar esto, ayuda considerar el algoritmo formal, que es similar al algoritmo *conexo*(p, q) en que busca un camino no dirigido desde α hasta ω pero que, en adición, ese camino sea abierto. Entonces habiendo llegado acumulado el conjunto C y la información de predecesores para poder construir un camino abierto desde ω hasta cualquier punto p de C , busca una arista no dirigida abierta desde algún $p \in C$ hasta algún $q \notin C$ es decir una arista o bien por mano correcta con $\phi(p, q) < w(p, q)$ o bien a contramano con $\phi(q, p) > 0$, y va dejando el mínimo de estos valores encontrados en el camino, para que cuando al fin llega a ω pueda construir el nuevo flujo a lo largo del camino encontrado.

$flujoMax(R, w) =$

$\phi := \emptyset$

para cada arista (i, j) de R $\phi := \phi((i, j) \rightarrow 0)$

$flujoMax(R, w, \phi)$

donde

$flujoMax(R, w, \phi) =$

$\phi1 := caminoAbierto(R, w, \phi, \{\alpha\}, \emptyset(\alpha \rightarrow \infty), \emptyset, \emptyset)$

si $(\phi1 = \emptyset)$ ϕ

sino $flujoMax(R, w, \phi1)$

donde

$caminoAbierto(R, w, \phi, C, \Delta, pred, d) =$

si hay arista (p, q) con $(p \in C \text{ y } q \notin C \text{ y } \phi(p, q) < w(p, q))$ o
 $(q \in C \text{ y } p \notin C \text{ y } \phi(p, q) > 0))$

si $(p \in C)$ (mano correcta)

$\Delta q := \min(\Delta(p), w(p, q) - \phi(p, q))$

si $(q = \omega)$ $nuevo\phi(R, \phi, \Delta q, pred(\omega \rightarrow p), d(q \rightarrow 0), \omega)$

sino $caminoAbierto(R, w, \phi, C \cup \{q\}, \Delta(q \rightarrow \Delta q), pred(q \rightarrow p), d(q \rightarrow 0))$

sino (contramano)

$caminoAbierto(R, w, \phi, C \cup \{p\}, \Delta(p \rightarrow \min(\Delta(q), \phi(p))), pred(p \rightarrow q), d(p \rightarrow 1))$

sino \emptyset

donde

$nuevo\phi(R, \phi, \Delta q, pred, d, p) =$

si $(pred(p)$ no está definido) ϕ

sino

si $(d(p) = 0)$ $nuevo\phi(R, \phi((pred(p), p) \rightarrow \phi((pred(p), p) + \Delta q), \Delta q, pred, d, pred(p))$

sino $nuevo\phi(R, \phi((pred(p), p) \rightarrow \phi((pred(p), p) - \Delta q), \Delta q, pred, d, pred(p))$

Si este algoritmo contesta algo, es correcto porque considere cómo es el conjunto C al momento de no encontrar arista abierta de C a $-C$. Para cada arista (p, q) con $p \in C$ y $q \in -C$, $\phi(p, q) = w(p, q)$, y para cada arista (p, q) con $q \in C$ y $p \notin C$, $\phi(p, q) = 0$. El

conjunto C de nodos alcanzados por el algoritmo antes de fallar en la búsqueda de una arista abierta contiene α y no contiene ω , y por el teorema $\phi(C, -C) - \phi(-C, C) = \phi(R)$. Pero $\phi(-C, C) = 0$, pues sino habría una arista abierta, y $\phi(C, -C)$ es el máximo posible para C , puesto que cada arista (p, q) de C a $-C$ tiene $\phi(p, q) = w(p, q)$. Es decir, para cualquier flujo ϕ' , $\phi'(C, -C) - \phi'(-C, C) \leq \phi(C, -C) - \phi(-C, C)$, pero según el teorema el primero es $\phi'(R)$ y el segundo es $\phi(R)$, así que $\phi'(R) \leq \phi(R)$ para todo flujo ϕ' , es decir ϕ da el flujo máximo posible.

Falta una consideración: ¿siempre termina este algoritmo? Cada iteración produce un flujo algo más grande que el previo. Si todos los $w(p, q)$ son enteros no negativos, eso garantiza la terminación porque no hay incrementos menores que 1. Y, ¿si son racionales no necesariamente enteros? Ver ejercicio.

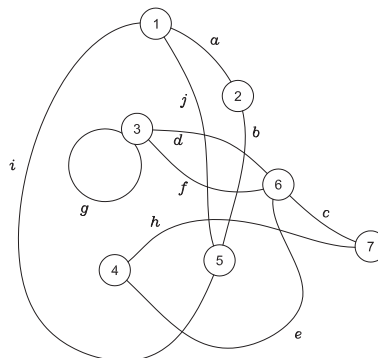
Pero no tenemos ejercicio para demostrar la terminación del algoritmo si los $w(p, q)$ pueden ser irracionales. La razón es que, sorprendentemente, no siempre termina. Más sorprendentemente aún, aunque cada iteración produce un flujo estrictamente mayor que el anterior, y esta sucesión está acotada superiormente (por la suma de los $c(p, w)$, por ejemplo), y por tanto se acerca a un límite, ¡ese límite no es necesariamente el flujo máximo! Para los detalles ver el trabajo de Ford y Fulkerson (citado abajo) cuyos nombres lleva el algoritmo. Estos detalles son algo densos para el nivel que buscamos aquí.

Pero una modificación del algoritmo de Ford y Fulkerson, el algoritmo de Edmonds y Karp, sí termina. Ese algoritmo busca el camino abierto más corto – con el mínimo posible de nodos – de α hasta ω , alguna especie del algoritmo de Dijkstra con cada arista con costo 1, y por tanto tiene que trabajar bastante más. Claro, alguien que sigue el algoritmo de Ford y Fulkerson puede por casualidad optar por la elección de aristas conforme con Edmonds y Karp, pero el algoritmo permite cualquier elección y con $w(p, q)$ irracionales la elección puede resultar mal.

Ahora, da un poco que pensar en cuál es el verdadero significado de esto para la implementación del algoritmo en una computadora digital, puesto que todos los números allí son racionales. Aún Maple o Mathematica usan números racionales para $\pi, e, \sqrt{2}$, etc. Podemos pedir un número muy grande de dígitos, pero por más dígitos que tengan son racionales. La solución de esta aparente paradoja es que se puede simular cualquier proceso involucrando números irracionales al grado de realismo que uno quiere con aproximaciones suficientemente exactas de esos irracionales. Esto quiere decir que si uno quiere elegir números que hace Ford-Fulkerson confundirse en la computadora, ¡puede!

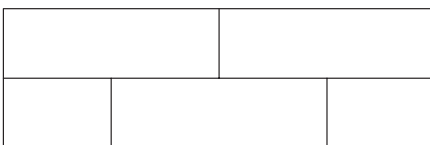
Ejercicios

- (1) Dibuje el grafo con aristas $\{a, b, c, d, e, f\}$, nodos $\{1, 2, 3, 4, 5\}$ y función de aristas P que se define así:
 $P(a) = \{1, 2\}$, $P(b) = \{2, 3\}$, $P(c) = \{2, 4\}$, $P(d) = \{1, 5\}$, $P(e) = \{2, 5\}$, $P(f) = \{5\}$.
- (1) Especifique formalmente un grafo que tiene el siguiente dibujo:



- (1) Ejecute el algoritmo *conexos*(2, 5) para el grafo del problema 1.
- (2) Construya un algoritmo para determinar si un grafo es conexo.

5. (1) Aplique el algoritmo del problema 4 a los grafos de los problemas 1 y 2.
6. (3) La solución del “problema del cartero”, en que se busca un camino de mínima distancia que atraviesa todas las aristas, depende de aparear los nodos de grado impar, y por tanto es fundamental saber que el número de esos nodos es par. Demuestre esto. (Sugerencia: Considere un grafo contraejemplo con el mínimo de aristas posible, y el grafo que resulta de remover cualquier arista de éste (*¿Hay una arista para sacar?*))
7. (4) El algoritmo para el ciclo de Euler estaría más eficiente en ejecución si en lugar de insertar el ciclo engendrado por *ciclo*, que el mismo programa que engendra el ciclo también hace la inserción. Haga ese algoritmo. Sugerencia: Sea r el recorrido de c , y sea $q = r_i$. Entonces, siendo n el largo de la sucesión c , $c_i \dots c_n c_1 \dots c_{i-1}$ es un ciclo de r_i a r_i . Por tanto ca es un camino de r_i a q y por tanto $(ca, q, B - \{a\})$ es habilitado para ser argumento de *ciclo*, con la consecuencia de que *ciclo* sirve como está para subalgoritmo, con un leve cambio en *extender*.
8. (3) Dé una condición necesaria y suficiente para que exista un camino, no necesariamente un ciclo, en un grafo sin nodos aislados, que usa cada arista exactamente una vez. (Tal camino se llama *camino de Euler*, sea o no un ciclo.)
9. (2) Encuentre la solución al acertijo de cortar con un trazo continuo los dieciséis segmentos rectos de la siguiente figura, exactamente una vez.

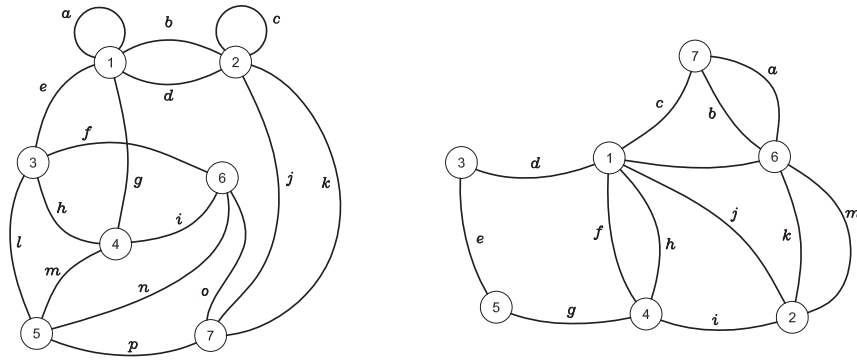


10. (3) Un grafo con costos está representado por un arreglo M de números naturales de dimensión $N \times N$. M_{ij} es el costo de una arista entre el nodo i y el nodo j . Se puede suponer que $M_{ii} = 0$ para todo i y que $M_{ij} = M_{ji}$ para todo i, j .

Represente el algoritmo de Dijkstra para el camino mínimo entre dos nodos dados, usando esta estructura de datos y otros arreglos, y variables, de números naturales. Haga determinista el algoritmo (por ejemplo, si el algoritmo pide cualquier elemento de un cierto conjunto, haga una búsqueda en la estructura de datos que representa ese conjunto).

11. (3) A continuación se define tres grafos en forma de matriz M donde $M(x, y)$ es el número de aristas entre los nodos x e y , y dos grafos mediante dibujos. Para cada grafo, decida si tiene ciclo de Euler, y si tiene, muestre su recorrido. Después, decida si tiene camino de Euler y muestre su recorrido si tiene. Justifique cada respuesta que es “no”.

	A	B	C	D	E	F	G		1	2	3	4	5	6	7		1	2	3	4	5	6	7
A	0	1	1	1	1	0	0	1	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0
B	1	0	0	1	0	0	0	2	1	0	1	0	0	0	1	2	1	0	1	0	0	1	0
C	1	0	0	1	0	1	1	3	0	1	0	1	0	0	1	3	1	1	0	1	0	1	0
D	1	1	1	0	1	1	1	4	0	0	1	0	1	0	1	4	0	0	1	0	2	0	1
E	1	0	0	1	0	1	1	5	0	0	0	1	0	1	1	5	1	0	0	2	0	1	0
F	0	0	1	1	1	0	1	6	1	0	0	0	1	0	1	6	1	1	1	0	1	0	0
G	0	0	1	1	1	1	0	7	1	1	1	1	1	1	0	7	0	0	0	1	0	0	0



12. (1) ¿Un grafo puede tener un ciclo de Euler y un camino de Euler que no es un ciclo? Justifique la respuesta.
13. (2) Sea $R(G, p, q, C)$ un algoritmo sobre un grafo G cuyos nodos están representados por números, un par p, q de sus nodos (no necesariamente distintos), y un conjunto C de sus nodos, que contesta un par (c, B) donde c es una sucesión de nodos de G y B es un conjunto de nodos de G , con la siguiente definición (observe que se pide una elección determinista de un nodo, cuando puede haber varios con la propiedad dada):

$$R(G, p, q, C) =$$

si $(p = q)$ (p, C) (donde p se refiere a la sucesión cuyo único término es p)

sino

sea r el primer nodo numéricamente de G conectado a p por una arista de G , con $r \notin C$, si hay

sea $(c, B) = R(G, r, q, C \cup \{r\})$

si $(c$ es vacía) $R(G, p, q, B)$

sino (pc, B)

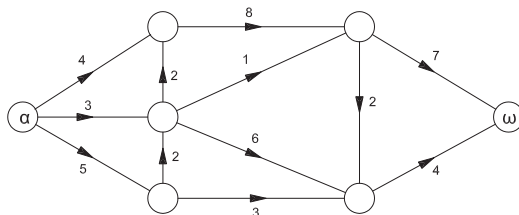
si no hay tal r (vacía, C)

- a) Aplique $R(G, 7, 3, \emptyset)$ al grafo G cuya matriz de incidencia (la matriz M tal que $M(p, q)$ es el número de aristas que conectan p y q) es lo siguiente:

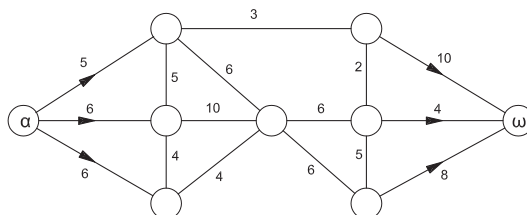
	1	2	3	4	5	6	7
1	0	0	0	0	1	0	0
2	0	0	0	1	1	0	1
3	0	0	0	0	1	0	0
4	0	1	0	0	0	0	0
5	1	1	1	0	0	1	0
6	0	0	0	0	1	0	0
7	0	1	0	0	0	0	0

- b) Diga para qué sirve $R(G, p, q, \emptyset)$.

14. (2) Para cada una de las siguiente proposiciones sobre grafos con costos sin rulos, conteste: ¿Es verdad para todos tales grafos, o no? Si contesta que no de un contraejemplo. Si contesta que sí, demuéstrela usando cualquier teorema y/o algoritmo que se dio en la materia.
- a) Si una arista tiene costo menor o igual que el costo de cualquier otra arista del grafo entonces hay un árbol cubridor mínimo que contiene esa arista.
- b) Si una arista conectada al nodo n tiene costo menor o igual que cualquier otra arista del grafo, entonces para cualquier otro nodo m hay un camino mínimo entre n y m que contiene esa arista.
15. (1) Calcular un flujo máximo para la siguiente red de transporte.

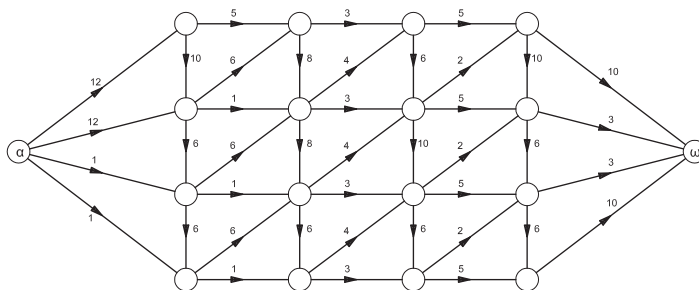


16. (2) En el siguiente dibujo las aristas sin flechas indican la capacidad en cada dirección es decir cada uno representa dos aristas con la misma capacidad, una en cada dirección.



Encuentre un flujo máximo.

17. (2) Encuentre y muestre un flujo máximo para la siguiente red de transporte.



18. (5) Demuestre que cualquier nodo p tal que no existen caminos dirigidos desde α hasta p y desde p hasta ω puede ser eliminado de la red, junto con todas sus aristas, sin afectar el valor del flujo máximo. (Sugerencia: Considere separadamente los casos en que falta el camino hasta ω y en que falta el camino desde α . En el primer caso, sea C el conjunto de todos los nodos accesibles desde p .)
19. (2) Demuestre que el algoritmo para el flujo máximo termina cuando los $w(p, q)$ son racionales.
20. (2) Con el “producto” ML de dos matrices definida como en la discusión del algoritmo de Floyd, comente el siguiente esquema: $M_1 = M$, $M_{k+1} = M_k^2$ para todo $k \geq 1$. Calcule M_k hasta que $M_{k+1} = M_k$. ¿Para qué sirve ese último M_k ? Comente sobre la eficiencia del esquema.
21. (1) Demuestre: En un grafo con n nodos, si dos nodos son conexos entonces hay un camino de largo (número de aristas) menor que n entre los dos. Si es grafo con costos, el camino mínimo entre dos nodos conexos tiene menos que n aristas.
22. (2) Dado un camino mínimo entre dos nodos de un grafo con costos, ¿existe necesariamente un árbol cubridor mínimo que contiene todas las aristas de ese camino?
23. (4) Sea G un grafo conexo con costos c , con a lo sumo una arista entre dos nodos dados. Dada la matriz N tal que N_{ij} es el costo del camino mínimo entre i y j , ¿es suficiente información para construir un camino mínimo entre cualesquier par de nodos? ¿Es suficiente para construir la matriz original M , de los costos de las aristas?

AUTÓMATAS

Por *autómata* en la matemática discreta entendemos una especie de computadora abstracta que dada una entrada produce una salida, es decir, calcula alguna función. Generalmente la entrada es una cadena de símbolos de un alfabeto¹, y la salida también, posiblemente sobre otro alfabeto.

Vamos a estudiar en este capítulo tres clases de autómatas, que se llaman *autómatas finitos*, *máquinas secuenciales*, y *máquinas de Turing*, y después, en el capítulo de lenguajes, *máquinas de pila*. Las primeras dos son de memoria finita, y las otras dos de memoria ilimitada pero de una finita cantidad de información en cualquier momento durante su computación.

Podríamos observar que una computadora electrónica, por mucha memoria RAM y disco duro que tenga, es un autómata finito, pero sí la cantidad de memoria nos permite para la mayoría de propósitos seguir como si fuera ilimitada. Pero no es nada difícil definir en unas cuantas líneas una función que agota toda la memoria de cualquier computadora enseguida, como por ejemplo $A(2, 4, 6)$ (A para Ackermann) donde $A(m, p, n)$ se define así para $m, p, n \geq 0$:

$$\begin{aligned} A(m, 0, 0) &= m \\ A(m, 1, 0) &= 0 \\ A(m, p+2, 0) &= 1 \\ A(m, 0, n+1) &= A(m, 0, n) + 1 \\ A(m, p+1, n+1) &= A(m, p, A(m, p+1, n)) \text{ para todo } m, n, p \geq 0 \end{aligned}$$

Aquí p es una operación, digamos op_p , y $m \ op_p \ n = m \ op_{p-1} \ m \ op_{p-1} \ m \dots op_{p-1} \ m$ (n veces). Entonces se puede ver que op_0 es adición, op_1 es multiplicación, op_2 es exponenciación: $A(m, 2, n) = m^n$. op_3 es una torre de exponentes, $A(m, 3, 3) = m^{m^m}$. En general op_{p+1} se obtiene de op_p en la misma forma que exponenciación se obtiene de multiplicación y multiplicación de adición. (Hemos definido $A(0, p, 0) = 1$, para todo $P > 1$, mientras 0^0 por ejemplo queda sin definición en la matemática, pero nuestro interés es como crece de rápido A .)

Sugerimos que tome unos minutos para programar esto en Maple, a ver que pasa con $A(2, 3, 4)$. Claro, si lo programamos diciendo que 0 es adición, 1 es multiplicación y 2 es exponenciación, todas operaciones incorporados a Maple, y vamos a la recursión solo para $P > 2$ podemos hacer $A(2, 3, 4)$ sin problema (y no presenta demasiada dificultad a mano). $A(2, 3, 5)$ es algo sorprendente. ¿Y $A(2, 3, 6)$? ¿ $A(2, 4, 4)$?

Así que conviene ver cuáles son las limitaciones verdaderas de la memoria finita.

NOTACIÓN. Sea Σ un alfabeto. La cadena vacía se llama ϵ para no confundirla con el conjunto vacío \emptyset de cadenas. Dadas dos cadenas c_1 y c_2 , c_1c_2 significa la concatenación de las dos. Si $a \in \Sigma$, cuando escribimos a puede significar o simplemente la letra a o la cadena de longitud 1 cuya única letra es a , y cuál es será claro del contexto. Por ejemplo, si $a \in \Sigma$ y c es una cadena sobre Σ , ac significa la concatenación de la cadena a con c . Si C_1 y C_2 son dos conjuntos de cadenas, C_1C_2 significa el conjunto $\{c_1c_2 : c_1 \in C_1 \text{ y } c_2 \in C_2\}$. Si C es un conjunto de cadenas, $C^0 = \{\epsilon\}$, $C^{k+1} = CC^k$ para todo natural k , y $C^* = \cup_{i=0}^{\infty} C^i$, es decir, una concatenación de 0 o más palabras de C . Para una cadena c , con $|c|$ queremos decir la longitud de c .

¹Cuando decimos *alfabeto*, técnicamente es simplemente un conjunto finito, pero lleva la connotación de que sus miembros son símbolos que son legibles en algún sentido.

1. Autómatas finitos y lenguajes regulares

DEFINICIÓN 4.1. Un *autómata finito* A sobre un alfabeto Σ es $(K, \Sigma, \delta, p_0, F)$ donde K es un conjunto finito de *estados*, δ es una función de $K \times \Sigma$ en K , p_0 es un estado (el estado *inicial*), y F es un conjunto de estados (los estados *finales*). Extendemos la función δ al dominio $K \times \Sigma^*$, siendo Σ^* el conjunto de todas las cadenas finitas sobre los símbolos de Σ , diciendo

- $\delta(q, \epsilon) = q$
- Para c en Σ^* y a en Σ , $\delta(q, ac) = \delta(\delta(q, a), c)$.

El conjunto *aceptado por* A es $\{c \in \Sigma^* : \delta(p_0, c) \in F\}$, y se llama $C(A)$.

DEFINICIÓN 4.2. Para un alfabeto Σ , un subconjunto de Σ^* es *regular* si es $C(A)$ para algún autómata A sobre Σ .

EJEMPLO 4.3. Sea $\Sigma = \{0, 1\}$. Sea $M = \{c \in \Sigma^* : c \text{ representa en binario un múltiplo de } 7\}$. Entonces M es regular. El autómata que acepta M tendrá 7 estados, que llamamos $\{0, 1, 2, 3, 4, 5, 6\}$, y queremos que para todo c , $\delta(0, c) = \text{resto}(\text{val}(c, 2), 7)$ (val definido como en la parte de aritmética de grandes números). Esto se va a cumplir si definimos $\delta(n, 0) = \text{resto}(2n, 7)$ y $\delta(n, 1) = \text{resto}(2n + 1, 7)$. Entonces la tabla para definir δ sería:

	0	1	2	3	4	5	6
0	0	0	2	4	6	1	3
1	1	1	3	5	0	2	4

donde $\delta(p, i)$ se ve en la fila i , columna p . El estado inicial es 0, y $F = \{0\}$. Así el autómata acepta la cadena nula ϵ .

EJEMPLO 4.4. Sea $M = \{c \in \{0, 1\}^*, |c| \geq 4 : \text{los últimos 4 símbolos de } c \text{ representan un número primo en binario}\}$. Entonces M es regular. En la memoria es necesario guardar sólo las últimas 3 entradas, que hace necesario estados $\{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111\}$ pero además necesitamos los estados finales que llamamos $\{p_2, p_3, p_5, p_7, p_{11}, p_{13}\}$, 21 estados en total. La tabla para esta δ :

	ϵ	0	1	00	01	10	11	000	001	010	011	100	101	110	111	p2	p3	p5	p7	p11	p13
0	0	00	10	000	010	100	110	000	p2	100	110	000	010	100	110	100	110	010	110	110	010
1	1	01	11	001	011	101	111	001	p3	p5	p7	001	p11	p13	111	p5	p7	p11	111	p7	p11

El estado inicial es ϵ .

Después de este ejemplo, sería natural preguntar si el conjunto de todos los primos puede ser regular. La respuesta es no, pero la demostración es un poco engorrosa para contarla aquí. Pero no es sorprendente cuando consideramos que ningún autómata puede aceptar $\{0^n 1^n : n \text{ natural}\}$. La demostración de éso es sencilla: tendría que aceptar una cadena $0^n 1^n$ donde n es mayor que el número de estados, así que algún estado tiene que ocurrir dos veces, y si borramos todos los 0 entre las dos ocurrencias ésas, el autómata tiene que aceptar esa nueva cadena porque termina en el mismo estado, pero ahora hay menos 0 que 1.

Es claro que la intersección de dos conjuntos regulares es regular, y lo mismo con la unión y el complemento (ver ejercicio 2).

No es tan inmediato que M^R es regular siempre que M es regular, donde C^R significa el conjunto formado por todas las cadenas de C escritas al revés. Tampoco es tan inmediato que si M y N son regulares, entonces MN lo es. Una herramienta teórica muy útil en estos y otros problemas es el *autómata finito no determinista*, que puede ofrecernos, para un estado y entrada dados, una elección entre varios posibles nuevos estados, y esto puede ocurrir varias veces durante la lectura de la cadena de entrada. El autómata acepta la cadena de entrada si es posible hacer todas estas elecciones de manera tal de llegar a un estado final al terminar la lectura. A primera vista, parecería demasiado fácil: seguro que estos autómatas van a aceptar algún conjunto que los autómatas finitos deterministas no son capaces de distinguir. Entonces es algo sorprendente que no: siempre hay un autómata determinista que acepta el mismo conjunto. Veamos esto formalmente.

DEFINICIÓN 4.5. Un *autómata finito no determinista* (afnd) es $(K, \Sigma, \delta, p_0, F)$ donde K, Σ, p_0 y F son como en la definición de autómatas finitos y δ es una función de $K \times \Sigma$ a los subconjuntos de K . Para una cadena c de longitud k de Σ^* y un estado p de K , una *computación de p para c* es una sucesión de estados $p = p_0, \dots, p_k$ tal que $p_i \in \delta(p_{i-1}, c_i)$ para todo $i = 1, \dots, k$, y en tal caso decimos que $(p, c) \vdash_A p_k$. Se dice que A *acepta* $c \in \Sigma^*$ si $(p_0, c) \vdash_A q$ para algún $q \in F$.

Veamos que fácil es demostrar que M^R es regular cuando M es regular, una vez que sabemos que los afnd aceptan sólo conjuntos regulares. Sea M el conjunto aceptado por el autómata finito $A = (K, \Sigma, \delta, p_0, F)$. Definimos el afnd $B = (K_B, \Sigma, \delta_B, p_{0B}, F_B)$ donde $K_B = K \cup \{p_{0B}\}$ (se supone que $p_{0B} \notin K$), $F_B = \{p_0\}$ si $p_0 \notin F$ y $F_B = \{p_0, p_{0B}\}$ si $p_0 \in F$, $\delta_B(p_{0B}, a) = \{p : \delta(p, a) \in F\}$ para cada $a \in \Sigma$, $\delta_B(q, a) = \{p \in K : \delta(p, a) = q\}$ para cada $(q, a) \in K \times \Sigma$.

Ahora, suponga que B acepta c . Si $c = \epsilon$ entonces tiene que ser que $p_{0B} \in F_B$, luego $p_0 \in F$ así que A acepta $\epsilon = \epsilon^R$ también. Si $c \neq \epsilon$, sea $c = c_1 \dots c_k$. Existe una sucesión de estados q_0, \dots, q_k tal que cada $q_i \in \delta_B(q_{i-1}, c_i)$, $i = 1, \dots, k$, y con $q_k \in F_B$. Pero esto implica que $\delta(q_i, c_i) = q_{i-1}$ para $i = 2, \dots, k$, y para $i = 1$ implica que $\delta(q_i, c_1) = r$ para algún $r \in F$. Ya que $q_k \in F_B$ tiene que ser que $q_k = p_0$ porque $\delta(q, a)$ nunca incluye a p_{0B} . Es claro entonces que $\delta(p_0, c_k \dots c_1) = r \in F$, es decir, A acepta c^R .

Recíprocamente, suponga que A acepta c . Si $c = \epsilon$, debe ser $p_0 \in F$, así que $p_{0B} \in F_B$ y por tanto B acepta $c^R = \epsilon$ también. Si $c \neq \epsilon$, sea $c = c_1 \dots c_k$. Existe una sucesión de estados $p_0 = q_0, \dots, q_k \in F$ tal que $q_i = \delta(q_{i-1}, c_i)$ para $i = 1, \dots, k$. Entonces para $i = 1, \dots, k$ es $q_{i-1} \in \delta_B(q_i, c_i)$, y siendo $q_k \in F$, $q_k \in \delta_B(p_{0B}, c_k)$. Por tanto p_{0B}, q_k, \dots, q_0 es una sucesión de estados de B que verifica que B acepta $c_k \dots c_1 = c^R$.

Así vemos que para toda cadena c , c es aceptada por A si y sólo si c^R es aceptada por B .

Para la teoría que sigue, será conveniente extender δ a $K \times \Sigma^*$ para los afnd como hicimos para los autómatas finitos: Definimos $\delta(p, \epsilon) = \{p\}$; y para $a \in \Sigma$ y $c \in \Sigma^*$ definimos inductivamente $\delta(p, ac) = \bigcup \{\delta(q, c) : q \in \delta(p, a)\}$.

LEMA 4.6. Sea $A = (K, \Sigma, \delta, p_0, F)$ un afnd. Para todo $p, q \in K$ y $c \in \Sigma^*$, $\delta(p, c) = \{q : (p, c) \vdash_A q\}$.

Demostración. Por inducción sobre $|c|$. Si $c = \epsilon$, es p el único $q \in \delta(p, \epsilon)$. La única sucesión de estados que cumple la definición de $(p, \epsilon) \vdash_A p$ es la sucesión $\{p\}$, que sí lo cumple porque empieza y termina en p y la única condición adicional no puede ser incumplida porque no hay (q_{i-1}, q_i) .

Suponga que algún c cumple para todo estado p , $\delta(p, c) = \{q : (p, c) \vdash_A q\}$. Vamos a demostrar que esto implica que para todo estado p y todo a en Σ , $\delta(p, ac) = \{q : (p, ac) \vdash_A q\}$. Ahora,

$$\delta(p, ac) = \text{bigcup} \{\delta(q, c) : q \in \delta(p, a)\} = \bigcup \{ \bigcup \{r : (q, c) \vdash_A r\} : q \in \delta(p, a) \}.$$

pero es claro de las definiciones que

$$(q, c) \vdash_A r \& q \in \delta(p, a) \text{ si y solo si } (p, ac) \vdash_A r. \square$$

TEOREMA 4.7. El conjunto aceptado por un autómata finito no determinista es aceptado por algún autómata finito determinista.

Demostración. Sea $A = (K, \Sigma, \delta, p_0, F)$ un autómata finito no determinista. Definimos el autómata finito determinista $B = (K_B, \Sigma, \delta_B, p_{0B}, F_B)$ como sigue: K_B es el conjunto de los subconjuntos de K . $\delta_B(S, a) = \bigcup \{\delta(q, a) : q \in S\}$. $p_{0B} = \{p_0\}$. $F_B = \{S \subset K : S \cap F \neq \emptyset\}$. Es fácil demostrar por inducción que para cada $S \subset K$ y $c \in \Sigma^*$

$$\delta_B(S, c) = \bigcup \{\delta(p, c) : p \in S\}.$$

La base de la inducción, para evitar cualquier confusión, debería ser para $|c| = 0$ y $|c| = 1$, que se demuestran directamente de las definiciones. Suponiendo que vale para todo c con $|c| = k > 0$,

sea $c = c_1 \cdots c_{k+1}$. Sabemos que $\delta_B(S, c_1 \cdots c_{k+1}) = \delta_B(\delta_B(S, c_1), c_2 \cdots c_{k+1})$. Por la hipótesis de inducción, esto es $\bigcup \{\delta(r, c_2 \cdots c_{k+1}) : r \in \delta_B(S, c_1)\}$. Ya que $\delta_B(S, c_1) = \bigcup \{\delta(p, c_1) : p \in S\}$, podemos expresar $\delta_B(S, c_1 \cdots c_{k+1})$ como $\{q : \exists p \in S \exists r \in \delta(p, c_1) \text{ tal que } q \in \delta(r, c_2 \cdots c_{k+1})\}$. Pero $\delta(p, c) = \bigcup \{\delta(r, c_2 \cdots c_{k+1}) : r \in \delta(p, c_1)\}$, así que $\bigcup \{\delta(p, c) : p \in S\} = \{q : \exists p \in S \exists r \in \delta(p, c_1) \text{ tal que } q \in \delta(r, c_2 \cdots c_{k+1})\}$ también.

Ahora, B acepta c si y sólo si $\delta_B(\{p_0\}, c)$ contiene un estado final. Pero $\delta_B(\{p_0\}, c) = \delta(p_0, c)$ según lo demostrado recién, lo cual, según el último Lema, es $\{q : (p_0, c) \vdash_A q\}$. Concluimos que B acepta c si y sólo si $(p_0, c) \vdash_A q$ para algún estado final q , es decir si y sólo si A acepta c . \square

1.1. La caracterización de Kleene. La caracterización más importante de los conjuntos regulares sobre Σ , que por muchos autores se da como su definición, es la de Kleene:

DEFINICIÓN 4.8. Sea Σ un alfabeto y sea $\mathbf{K}(\Sigma)$, o simplemente \mathbf{K} cuando Σ está entendido, la mínima familia de subconjuntos de Σ^* que satisface las siguientes propiedades:

1. Todo conjunto finito pertenece a \mathbf{K} .
2. Si C y D pertenecen a \mathbf{K} , entonces también le pertenecen $C \cup D$, CD y C^* .

Dicho en forma más compacta, \mathbf{K} es la familia más chica de conjuntos de Σ^* que incluye todo conjunto finito y es cerrada bajo unión, concatenación y la operación C^* .

Los minimalistas axiomáticos sacarían la redundancia de la primera diciendo que $\{a\} \in \mathbf{K}$ para cada $a \in \Sigma$, y $\emptyset \in \mathbf{K}$.

TEOREMA 4.9. La familia de conjuntos regulares sobre Σ es $\mathbf{K}(\Sigma)$.

Demostración. En el conjunto de ejercicios se encuentran los de la clausura de los conjuntos regulares bajo unión, concatenación, y C^* , y que los finitos son regulares, que se demuestra mediante autómatas no deterministas. Ya que \mathbf{K} es la mínima familia así clausurado, está incluida en los conjuntos regulares.

Para la otra inclusión, definimos primero que un sistema de ecuaciones simultáneas sobre el conjunto de incógnitas $\{B_q : q \in J\}$

$$\{B_q = A_q \cup (\bigcup_{p \in J} A_{qp} B_p) : q \in J\}$$

es *bien formado* si cada coeficiente A_q o A_{qp} está en $\mathbf{K}(\Sigma)$ y ningún A_{qp} contiene ϵ .

Ahora sea $A = (K, \Sigma, \delta, p_0, F)$ un autómata finito. Para cada estado q sea $B_q = \{c : \delta(q, c) \in F\}$. Para cada $q \in K$ se cumple la ecuación:

$$B_q = A_q \cup \bigcup_{a \in \Sigma} a B_{\delta(q, a)},$$

donde $A_q = \{\epsilon\}$ si $q \in F$ y $A_q = \emptyset$ si $q \notin F$.

Es claro que este sistema de ecuaciones es bien formado.

Esas son $|K|$ ecuaciones con las $|K|$ incógnitas $\{B_q : q \in K\}$. Si podemos resolver el sistema tendremos B_{p_0} , y veremos que pertenece a \mathbf{K} .

Entonces vamos a demostrar que dado un sistema de ecuaciones bien formado con $n > 1$ variables podemos derivar otro sistema equivalente con $n - 1$ variables, eliminando cualquiera de las variables que queremos. Esto nos permite llegar a un sistema bien formado de una sola ecuación que dice que B_{p_0} es igual a una expresión formada por uniones, concatenaciones y operaciones $*$ de conjuntos de \mathbf{K} , es decir, que $B_{p_0} \in \mathbf{K}$.

Vamos a proceder a resolverlas por algo parecido a la eliminación Gaussiana. Pero las constantes son conjuntos. Estos conjuntos son de \mathbf{K} . Al hacer la eliminación, tenemos que asegurar que las nuevas constantes siguen siendo de \mathbf{K} , y aseguramos esto notando que formamos las nuevas constantes con unión, concatenación, y la operación $*$. Encontramos un requerimiento técnico más: cada constante que se concatena con una incógnita no tiene que contener ϵ .

Formalmente, vamos a demostrar la siguiente aserción:

Dado un sistema bien formado sobre J , con $|J| > 1$,
para cada $q \in J$ existe un sistema bien formado sobre $J - \{q\}$.

Es decir, que podemos eliminar cualquier q . Para demostrar ésto, necesitamos:

LEMA 4.10. Si S , A y T son subconjuntos de Σ^* , $\epsilon \notin A$ y $S = T \cup AS$, entonces $S = A^*T$.

Demostración. Si T es vacío el único conjunto que puede cumplir $S = T \cup AS$ es $S = \emptyset$, en cuyo caso $S = A^*T$ también. Suponga que T no es vacío. Demostramos primero por inducción sobre n que $A^nT \subset S$. Se cumple para $n = 0$ porque en este caso la aserción es simplemente que $T \subset S$, conclusión inmediata dada $S = T \cup AS$. Suponga que se cumple para algún n , $A^nT \subset S$. De $S = T \cup AS$ sabemos que $AR \subset S$ para cualquier $R \subset S$, así que $A(A^nT) = A^{n+1}T \subset S$. Así queda demostrado por inducción que $A^nT \subset S$ para todo n .

De eso concluimos que $A^*T \subset S$.

Para la otra inclusión, si $S - A^*T$ no es vacío, sea c una cadena de longitud mínima en $S - A^*T$. De $S = T \cup AS$, o $c \in T$ o $c \in AS$, pero $c \notin T$ porque $T \subset A^*T$ (siendo $\epsilon \in A^*$), entonces $c \in AS$, así que $c = de$ con $d \in A$ y $e \in S$. Ahora $d \neq \epsilon$ porque $\epsilon \notin A$ por hipótesis, así que $|e| < |c|$. Por la minimalidad de $|c|$ en $S - A^*T$, siendo $e \in S$, $e \in A^*T$. Pero esto conduce a una contradicción porque entonces $c \in AA^*T \subset A^*T$. La única manera para evitar esta contradicción es admitir que $S - A^*T$ es vacío, es decir, que $S \subset A^*T$. \square

Sea q el estado que decidimos eliminar. Si A_{qq} es vacío ya tenemos B_q en función de los demás. Si A_{qq} no es vacío, poniendo $T = A_q \cup (\bigcup_{p \in J - \{q\}} A_{qp}B_p)$, tenemos $B_q = T \cup A_{qq}B_q$. Por el Lema, $B_q = A_{qq}^*T$, expresando B_q en términos de las otras B_p . Al sustituir eso en el lugar de B_q en cada ecuación para B_p para $p \in J - \{q\}$ tenemos nuevas ecuaciones

$$B_p = A_p \cup A_{pq}A_{qq}^*T \cup \bigcup_{r \in J - \{q\}} A_{pr}B_r = A'_p \cup \bigcup_{r \in J - \{q\}} A'_{pr}B_r.$$

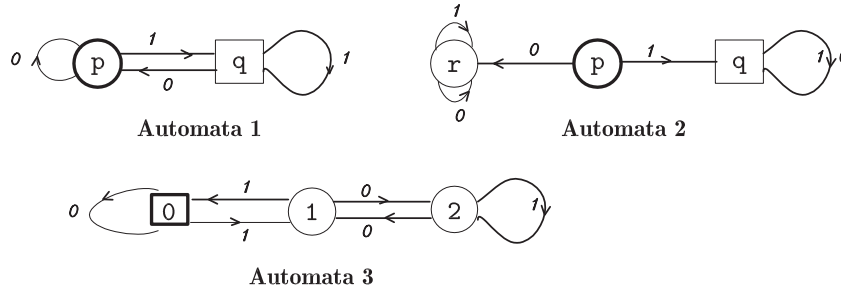
Vamos a ver cuánto vale el coeficiente A'_{pr} de B_r en ésto.

$$A_{pq}A_{qq}^*T = A_{pq}A_{qq}^*(A_q \cup (\bigcup_{r \in J - \{q\}} A_{qr}B_r))$$

así que $A'_{pr} = A_{pq}A_{qq}^*A_{qr} \cup A_{pr}$. Esto muestra dos cosas: puesto que A_{pr} , A_{qr} , A_{qq} y A_{pq} son de $\mathbf{K}(\Sigma)$, A'_{pr} es de $\mathbf{K}(\Sigma)$. Además, $\epsilon \notin A'_{pr}$ porque $\epsilon \notin A_{pq}$ y por tanto tampoco en $A_{pq}A_{qq}^*A_{qr}$, y $\epsilon \notin A_{pr}$. Finalmente, las nuevas cadenas para acompañar a A_p en A'_p vienen de $A_{pq}A_{qq}^*T$, y son de $A_{pq}A_{qq}^*A_q$, así que $A'_p = A_p \cup A_{pq}A_{qq}^*A_q$, que es de $\mathbf{K}(\Sigma)$ puesto que A_p , A_{pp} , A_{qq} y A_q lo son. (Es interesante notar que $\epsilon \in A'_p$ si y sólo si $\epsilon \in A_p$, y así siguiendo hasta el primer conjunto de ecuaciones, en donde $A_p = \epsilon$ o \emptyset , según $p \in F$ o no.)

Aunque podemos elegir los B_q por eliminación Gaussiana en cualquier orden, nos conviene dejar B_{p_0} para el último, cuando expresando B_{p_0} “en términos de las demás incógnitas” significa expresarle como sólo la constante A_{p_0} de la última iteración, es decir, como un conjunto de $\mathbf{K}(\Sigma)$. \square

Ilustramos este teorema a través de algunos ejemplos. Consideremos los autómatas sobre $\{0, 1\}$ dados en la siguiente figura, donde seguimos la convención de que los estados finales son representados por cuadrados y el estado inicial más negro.



El autómata 1 acepta las cadenas que terminan en 1. Las ecuaciones son $B_p = 0B_p \cup 1B_q$ y $B_q = \epsilon \cup 1B_q \cup 0B_p$, y de la segunda sacamos $B_q = 1^*(\epsilon \cup 0B_p)$, que al sustituir en la primera da $B_p = 0B_p \cup 11^*(0B_p \cup \epsilon) = (0 \cup 11^*0)B_p \cup 11^*$, y finalmente $B_p = C(A) = (0 \cup 11^*0)^*11^*$. ¿Estaba esperando $\{0, 1\}^*1$, que escribiríamos de inmediato sin el algoritmo? Pero no se presenta este algoritmo como para sacar la expresión regular más bonita y compacta posible, sino como para sacar alguna expresión regular y así demostrar el teorema de Kleene. Pero en resolver las ecuaciones no estamos obligados a seguir el procedimiento de la demostración: en este caso podemos observar que enchufando la primera ecuación en la segunda vemos $B_q = \{\epsilon\} \cup B_p$ y enchufando esto de vuelta en el primero y reordenando vemos $B_p = \{1\} \cup \{0, 1\}B_p$, y de allí el lema nos da $B_p = \{0, 1\}^*1$.

A veces el procedimiento saca la expresión más compacta, como por ejemplo, en el caso del autómata 2, que acepta el revés del autómata 1, es decir, todas las cadenas que empiezan con 1. Ahora las ecuaciones son $B_p = 1B_q \cup 0B_r$, $B_r = \{0, 1\}B_r$, y $B_q = 1B_q \cup 0B_q \cup \epsilon = \{1, 0\}B_q \cup \epsilon$, dando $B_q = \{1, 0\}^*\epsilon = \{1, 0\}^*$, y $B_r = \{0, 1\}^*\emptyset = \emptyset$. Sustituyendo en B_p da $B_p = 1\{1, 0\}^*$; más compacto no se puede esperar.

El autómata 3 acepta las cadenas sobre $\{0, 1\}$ que, interpretadas como números representados en el sistema binario, son divisibles por 3. Por ejemplo, las de longitud 5 son 00000, 00011, 00110, 00101, 01100, 01111, 10010, 10101, 11000, 11011, 11110. El autómata tiene que saber en cuál miembro de \mathbb{Z}_3 está: si ha recibido un número i , y ahora viene un 0, va a tener $2i \pmod{3}$, y si recibe 1 es $2i + 1 \pmod{3}$, así que podemos llamar 0, 1, 2 a los tres estados, y $\delta(i, a) = 2i + a \pmod{3}$. Por eso el gráfico del autómata 3. Las ecuaciones son $B_0 = 0B_0 \cup 1B_1 \cup \epsilon$, $B_1 = 0B_2 \cup 1B_0$, $B_2 = 0B_1 \cup 1B_2$. Sustituyendo B_1 en B_0 y B_2 :

$$\begin{aligned} B_0 &= 0B_0 \cup 1(1B_0 \cup 0B_2) \cup \epsilon = (0 \cup 11)B_0 \cup 10B_2 \cup \epsilon \\ B_2 &= 0(1B_0 \cup 0B_2) \cup 1B_2 = (00 \cup 1)B_2 \cup 01B_0 \end{aligned}$$

Poniendo $B_2 = (00 \cup 1)^*01B_0$ y sustituyendo en la nueva ecuación para B_0 :

$$B_0 = 0B_0 \cup 1(1B_0 \cup 0(00 \cup 1)^*01B_0) \cup \epsilon = (0 \cup 11 \cup 10(00 \cup 1)^*01)B_0 \cup \epsilon,$$

y de allí sacamos finalmente

$$B_0 = (0 \cup 11 \cup 10(00 \cup 1)^*01)^*.$$

Sabemos que si no hemos hecho un error manipulativo esto sí es correcto. Pero podemos ver directamente que es correcto pensando así: para aceptar c , o $c = \epsilon$ o $c \in 0B_0$ o $c \in 1D1B_0$ donde D consiste de todas las cadenas que llevan el estado 1 al estado 1 sin pasar por el estado 0. Por inspección del autómata $D = (01^*0)^*$. Entonces $B_0 = 0B_0 \cup 1(01^*0)^*B_0 \cup \epsilon = (0 \cup 1(01^*0)^*)B_0 \cup \epsilon$, y por el Lema $B_0 = (0 \cup 1(01^*0)^*)^*$ (concatenado con ϵ pero eso es como multiplicar por 1). Pensando un poco se ve que $1(01^*0)^*1 = 11 \cup 10(00 \cup 1)^*01$ (por ejemplo, $1(0110)(01110)1 = (10)(1100111)(01)$).

Hemos mencionado cosas, por ejemplo clausura bajo concatenación, que son más difíciles de ver por la caracterización por autómata. Pero, ¿quiere demostrar directamente por la caracterización de Kleene la clausura bajo complementación de \mathbf{K} ?

1.2. Máquinas Secuenciales. Ahora vamos a ver un tipo de autómeta de memoria finita que particiona Σ^* en más de dos conjuntos, mientras el autómeta finito particiona en 2. Lo hacen por emitir una salida, que depende de la entrada, que viene de un alfabeto finito, y el estado.

DEFINICIÓN 4.11. Una *máquina secuencial* es $(K, \Sigma, \Delta, \delta, \lambda, p_0)$ donde K es un conjunto finito de *estados*, Σ y Δ son alfabetos finitos, δ es una función $K \times \Sigma \rightarrow K$, λ es una función $K \times \Sigma \rightarrow \Delta$, y p_0 , el *estado inicial*, es en K .

Extendemos δ a $K \times \Sigma^* \rightarrow K$ como para el autómeta finito, y λ a $K \times \Sigma^* \rightarrow \Delta^*$ en forma semejante:

$$\begin{aligned}\lambda(q, \epsilon) &= \epsilon \\ \lambda(q, ac) &= \lambda(q, a)\lambda(\delta(q, a), c) \text{ para todo } a \in \Sigma, c \in \Sigma^*.\end{aligned}$$

Por ejemplo, definimos una máquina secuencial que emite $i \in \{0, 1, 2\}$ cuando la cadena de entradas en $\{0, 1\}$, interpretada como número representado en binario, es $i(\text{mod}3)$. Los estados son 0,1,2. En la tabla T que sigue, $T_{ij} = (q, a)$ significa que $\delta(j, i) = q$ y $\lambda(j, i) = a$:

	0	1	2
0	0,0	2,2	1,1
1	1,1	0,0	2,2

Es decir, δ es igual a la del autómeta 3 de un ejemplo anterior que aceptaba múltiplos de 3, y $\lambda(i, j) = \delta(i, j)$ para todo i, j .

Es fácil demostrar, y no muy sorprendente, que para cualquier máquina secuencial $(K, \Sigma, \Delta, \delta, \lambda, p_0)$ y cada $a \in \Delta$ el conjunto $\{c \in \Sigma^* : \lambda(p_0, c) \text{ termina en } a\}$ es regular. Casi recíprocamente, para cada conjunto regular R sobre Σ hay una máquina secuencial con $\Delta = \{0, 1\}$ tal que $R - \{\epsilon\} = \{c \in \Sigma^* : \lambda(p_0, c) \text{ termina en } 1\}$ (lamentablemente, no hay forma de hacer que $\lambda(q, \epsilon)$ termine en 1).

La primera propiedad es un caso especial de la pregunta, para un conjunto regular R dado sobre Δ , si $\{c : \lambda(q, c) \in R\}$ es regular. La respuesta es sí: Sea $A = (K_A, \Delta, \delta_A, p_{0A}, F_A)$ un autómeta con $C(A) = R$, y sea $M = (K, \Sigma, \Delta, \delta, \lambda, p_0)$ la máquina secuencial. Sea B el autómeta $(K_B, \Sigma, \delta_B, p_{0B}, F_B)$, donde $K_B = K \times K_A$, $\delta_B((p, q), a) = (\delta(p, a), \delta_A(q, \lambda(p, a)))$, $p_{0B} = (p_0, p_{0A})$. B quiere aceptar $c \in \Sigma^*$ si A acepta $\lambda(q, c)$, entonces manda c a M carácter por carácter, observando cada salida que resulta de M y mandándola a A . Cada vez que A dice “sí, acepto”, B acepta. Entonces $F_B = \{(p, q) \in K_B : q \in K_A\}$.

2. Máquinas de Turing

A.M. Turing fue un pionero de la computación, tanto en la ingeniería como en la teoría. En los años 30 participó en la búsqueda de la formalización matemática de la noción de computabilidad, de función computable. Había varias definiciones que pretendían captar esta noción, que después de mucha discusión fueron demostradas ser equivalentes. Una era la *máquina de Turing*.

La máquina de Turing es de estados finitos pero trabaja con una cinta que tiene escrita en cualquier momento dado una cantidad finita de información, pero que puede ser aumentada por la máquina en una forma controlada. Entonces la memoria es finita pero ilimitada. La memoria tiene dos partes: una parte finita que constituye los estados internos de la máquina, que es como las de los autómetas finitos y las máquinas secuenciales, y una cinta infinita en las dos direcciones en que está escrita una cantidad finita de información, en el sentido que todas menos un conjunto finito de casillas tiene escrito “blanco”. La máquina, posicionada en una casilla de esta sucesión, lee el carácter, escribe allí uno nuevo de su alfabeto finito, dependiendo del estado interno y el carácter leído, entra en un nuevo estado, que depende del estado y del carácter leído, y se mueve una casilla a la derecha o a la izquierda, dependiendo del estado y el carácter leído. Si no hay instrucciones acerca de este carácter y estado, la máquina para.

La máquina puede tener la tarea de aceptar o rechazar una cadena sobre un alfabeto finito dado, o calcular una función de uno o más argumentos naturales.

Formalmente:

DEFINICIÓN 4.12. Una *máquina de Turing* es $M = (K, \Sigma, \Delta, \beta, \delta, p_0, p_f)$ donde K es un conjunto finito de *estados*, Σ y Δ son alfabetos finitos con $\Sigma \subset \Delta$, $\beta \in \Delta - \Sigma$ (el “blanco”), δ es una función de un subconjunto de $(K - p_f) \times \Delta \rightarrow K \times \Delta \times \{I, D\}$, p_0 y p_f son estados (el estado inicial y el estado final respectivamente). Una *configuración* de M es (t, q, i) donde t es una sucesión infinita en ambas direcciones sobre Δ , es decir una función $\mathbb{Z} \rightarrow \Delta$, con sólo finitos valores distintos de β , es decir en donde existen m, n tal que $t(i) = \beta$ para todo $i < m$ y todo $i > n$, $q \in K$, e $i \in \mathbb{Z}$ (la posición de la cabeza de lectura). De dos configuraciones (t, q, i) y (s, p, j) se dice que $(t, q, i) \rightarrow (s, p, j)$ si $\delta(q, t_i)$ está definida e igual a (p, a, d) donde

- $s_i = a$ y $s_k = t_k$ para todo $k \neq i$,
- si $d = I$ entonces $j = i - 1$ y si $d = D$ entonces $j = i + 1$.

Se define la relación \vdash sobre configuraciones como la mínima relación transitiva que incluye \rightarrow . Una configuración (t, p, i) es un *paro* si $\delta(p, t_i)$ no está definido. Se dice que M *acepta* una cadena $c \in \Sigma^*$ si la configuración $(t, p_0, 1) \vdash (s, p_f, i)$ para algún s, i donde $t_k = \beta$ para todo $k \leq 0$ y para todo $k > |c|$ y $t_1 \dots t_{|c|} = c$. Se dice que M *rechaza* c si $(t, p_0, 1) \vdash (s, p, i)$ donde (s, p, i) es un paro y $p \neq p_f$.

Se dice que M *acepta* un conjunto $C \subset \Sigma^*$ si acepta cada cadena $c \in C$ y rechaza cada cadena $c \in \Sigma^* - C$.

Observe que para que M acepte C , M tiene que parar en algún momento para cada c , sea $c \in C$ o no. Si hay alguna cadena que lo hace seguir para siempre, no acepta ningún conjunto. Nos tiene que dar siempre una respuesta a la pregunta ¿ $c \in C$?

Notemos (t, p_f, i) siempre es un paro, porque hemos excluido (p_f, a) del dominio de δ . Para aceptar c , la máquina tiene que parar en p_f . Para rechazar c , tiene que parar en un estado $p \neq p_f$.

EJEMPLO 4.13. Una máquina de Turing que acepta las cadenas de $\{0, 1\}^*$ en que el número de caracteres 1 es igual al número de caracteres 0. Según las reglas, tenemos que recibir una configuración $(t, p_0, 1)$ con $c \in \Sigma^*$ y con $t_0 t_1 \dots t_{|c|} t_{|c|+1} = \beta c \beta$. El plan es el siguiente: la máquina busca el primer carácter a la derecha que no es 2 (inicialmente será el primer carácter de c , si hay, pero ésta es la operación que proyectamos para p_0 y la repetimos). Si encuentra un β primero, va a p_f . Si encuentra un 0 lo cambia a 2 y busca un 1 a la derecha. Si encuentra β primero esta vez, para, sino cambia el 1 a 2 y busca a la izquierda a β , y cuando lo encuentra se mueve uno a la derecha sin cambiarlo y entra en estado p_0 . Si consigue cambiar en 2 todas las casillas 1 hasta $|c|$, acepta. Entonces $K = \{01, 0, 1, \beta, p_f\}$ que significan “buscar a la derecha por el primer carácter en $\{0, 1\}$ ”, “buscar el primer 0 a la derecha”, “buscar el primer 1 a la derecha”, y “buscar el primer β a la izquierda”. $\Delta = \{0, 1, 2, \beta\}$ y $\Sigma = \{0, 1\}$. El estado inicial es 01 y la tabla de transiciones (fila i , columna j es $\delta(i, j)$ a menos que sea “paro”, que es equivalente a dejarlo en blanco excepto que queremos implicar que el paro es intencional) es:

	0	1	2	β
01	1, 2, D	0, 2, D	01, 2, D	p_f, β, I
0	$\beta, 2, I$	0, 1, D	0, 2, D	paro
1	1, 0, D	$\beta, 2, I$	1, 2, D	paro
β	$\beta, 0, I$	$\beta, 1, I$	$\beta, 2, I$	01, β, D

Cuando Turing empezó a investigar la naturaleza de estos autómatas, primero le interesaba hacer las operaciones aritméticas comunes, adición, multiplicación, división, y luego cosas más complicadas, hasta sospechar que su alcance iba a ser todas las funciones que intuitivamente diríamos computables. Luego, después de divulgar, le decían “el cálculo lambda puede hacer cualquier función que hacen tus máquinas”, o “cualquiera de tus máquinas calcula una función recursiva general”, y lo mismo para sistemas Post y otros. Pero en cada caso, resultó la recíproca:

para cualquier función del cálculo lambda, hay una máquina de Turing que la hace, etc. Todos los intentos independientes de formalizar la noción de computabilidad resultaron equivalentes, y de allí salió la llamada *tesis de Church-Turing*, que todos representaban la idea de computabilidad.

Un ejemplo que nos lleva un paso en esa dirección:

EJEMPLO 4.14. Una máquina de Turing que acepta las cadenas de 1^* cuya longitud es un número primo. La máquina empieza sobre el primer carácter de c en una cinta $\beta^\infty 1^n \beta^\infty$, y se puede trabajar dentro del segmento $\beta 1^n \beta$. Vamos probando divisores $2, 3, 4, \dots, n-1$ para establecer primalidad – no paramos en $\lceil \sqrt{n} \rceil$, es más fácil recorrer todos los demás divisores que engendrar d^2 y compararlo con n – no nos interesa eficiencia, sino mostrar que las máquinas Turing pueden hacer la función. Cuando estamos por probar un divisor d , tenemos una configuración con $\beta 2^d 1^{n-d} \beta$ y repetimos lo siguiente: cambiar el primer 2 a 3, ir al primer 1 y cambiarlo al 4, a menos que se llega a β antes, en cuyo caso d no es factor de n . Si cambiamos todos los 2 a 3, cambiando un 1 a 4 cada vez, volvemos cambiando todos los 3 a 2 y todos los 4 a 1 hasta encontrar β . Luego vamos al final de los 2, que va a ser seguido por un 1, y cambiamos el 1 a 2. En este momento, si el carácter a la derecha es β , hemos terminado, aceptamos porque n es primo.

Especificamos la máquina de Turing con el siguiente programa:

si ($t = \beta$) rechazar sino $2 \rightarrow$. sino ($t \neq \beta$)
si($t = \beta$) rechazar sino $t \leftarrow$. . $4 \leftarrow$
d+	. . mientras($t \neq \beta$) $t \leftarrow$
mientras ($t = 2$) $2 \rightarrow$. . ir a cancelar
$2 \rightarrow$	sino ($t \neq 2$)
si($t = \beta$) aceptar	. mientras($t \notin \{1, \beta\}$) $t \rightarrow$
mientras($t \neq \beta$) $t \leftarrow$. si($t = \beta$) rechazar
cancelar	. sino
$\beta \rightarrow$. . $1 \leftarrow$
mientras($t = 3$) $3 \rightarrow$. . mientras($t = 4$) $4 \leftarrow$
si($t = 2$)	. . mientras($t = 3$) $2 \leftarrow$
. $3 \rightarrow$. . ir a cancelar
. mientras($t \notin \{1, \beta\}$) $t \rightarrow$	
. si($t = \beta$)	
. . $\beta \leftarrow$	
. . mientras($t \neq \beta$)	
. . . si($t = 4$) $1 \leftarrow$	
. . . sino si($t \in \{2, 3\}$) $2 \leftarrow$	
. . $\beta \rightarrow$	
. . ir a d+	

La máquina ejecuta el programa. Su estado es la línea de código que está ejecutando. La instrucción fundamental es $a \leftarrow$ o $a \rightarrow$, donde $a \in \Delta$, y las flechas significan “mover a la izquierda” y “mover a la derecha” respectivamente. También usamos sentencias condicionales, si(condición) código1 sino código 2, y mientras(condición) código. Todo esto es trasladable directamente, aunque tediosamente, a una tabla de transiciones (la parte más molesta es inventar nombres para todos los estados). En el programa, hemos usado la notación t para significar el carácter t_i de la configuración actual (t, q, i) . Usamos “ir a etiqueta” donde *etiqueta* es la etiqueta de alguna línea de código, escrita a la izquierda antes de la línea. $\Sigma = \{1\}$ y $\Delta = \{\beta, 1, 2, 3, 4\}$.

Las máquinas de Turing con alfabeto de trabajo $\Delta = \{0, 1, \beta\}$ pueden representar todas las máquinas de Turing. Bosquejamos esto para una máquina con $\Delta = \{a, b, c, d, \beta\}$, y la idea se ve. Codificamos a, b, c, d en binario, $a = 00, b = 01, c = 10, d = 11$, así que las cadenas de $\{a, b, c, d\}^*$ están representadas en forma única por las cadenas de longitud par en $\{0, 1\}^*$. Ahora suponga

que la máquina con $\Delta = \{a, b, c, d, \beta\}$ tiene estados p, q con $\delta(p, a) = (q, b, D)$. La nueva máquina M tendrá estados $p, q, p0, qb, qb0$ (entre otros). $\delta_M(p, 0) = (p0, 0, D)$. $\delta_M(p0, 0) = (qb, 0, I)$. $\delta_M(qb, t) = (qb0, 0, D)$ para $t \in \{0, 1\}$, y $\delta_M(qb0, t) = (q, 1, D)$ para $t \in \{0, 1\}$. En otras palabras, qb escribe el código de b , se mueve dos veces a la derecha, y termina en el estado b , cualquier que sea el contenido de las dos casillas que atraviesa. Los otros movimientos se hacen en la misma manera.

2.1. La Máquina de Turing Universal. Se puede hacer una máquina de Turing con $\Sigma = \{0, 1\}$ y que, dada la especificación de una máquina de Turing M sobre esos alfabetos, y una entrada en $\{0, 1\}^*$, codificados adecuadamente, simula la acción de M sobre esa entrada. M se codifica como una cadena en $(1^+(((01^+)(0\{1, 11, 111\})(0\{1, 11\} \cup \{\epsilon\})^3 0))^+)$ donde enumeramos los estados de M como q_1, \dots, q_k y la subcadena

$$1^n(01^{p_0}01^{a_0}01^{d_0})(01^{p_1}01^{a_1}01^{d_1}) \dots (01^{p_\beta}01^{a_\beta}01^{d_\beta})$$

significa que $\delta_M(q_n, 0) = (q_{p_0}, e, d)$ donde $e = 0, 1, \beta$ según $a_0 = 1, 2, 3$ respectivamente, y $d = I, D$ según $d_0 = 1, 2$ respectivamente, reglas similares para (p_1, a_1, d_1) y $(p_\beta, a_\beta, d_\beta)$ para $\delta_M(q_n, 1)$ y $\delta_M(q_n, \beta)$ respectivamente. Si algún $p_i = 0$ entonces los correspondientes a_i y d_i son 0, significando que el valor de δ_M correspondiente no está definido. Esta especificación está seguida por $\beta 1^s \beta$ donde s indica el estado inicial q_s , y después viene la entrada en $\{0, 1\}^*$ seguida por otro blanco. La máquina universal U empieza poniendo $(01)^s \beta$ a la izquierda de todo, los 0 intercalados para facilitar comparaciones, y siempre durante la simulación la cadena $\beta\{01, 11\}^k \beta$ va a significar que el estado actual de la simulación de M es q_k (los 11 son para marcar dónde va la comparación de k con un estado de la especificación). La cinta que ve M es simulada por una cadena de $(\{0, 1\}\{0, 1, \beta\})^+$, el primer carácter 1 sólo cuando la actual configuración de M en la simulación ubica la cabeza de lectura en esa casilla. Cada vez que M quiere ir a la casilla a la izquierda de ésta, hay que correr todo dos casillas a la derecha y meter un β en la nueva casilla. Un movimiento de M se simula por buscar el estado actual en la especificación, ubicar la terna correspondiente al carácter actual, escribir en la cinta simulada el nuevo carácter, y escribir a la izquierda el nuevo estado.

Sabiendo que hay una máquina U de Turing capaz de simular cualquier máquina de Turing, podemos pensar en hacerla simular a sí misma simulando a sí misma. Pero eso no es posible, porque la cinta inicial tendrá infinita información.

Pero si U es tan inteligente, ¿no puede haber una máquina U' que, dada la especificación e de una máquina M y su entrada c , decide si M para alguna vez cuando recibe c ? Si existe esta U' , podemos tener otra máquina U'' que, dado nada más que la especificación de M , construye la cinta inicial correspondiente a la especificación de M seguida por la especificación de M , y entra en una submáquina equivalente a U' . Ahora U'' va a contestar, dada la especificación de una máquina M , si M para o no enfrentada con su propia especificación.

2.2. Una función no computable. Hagamos un pequeño cambio en la máquina U'' . Dijimos que tenía una submáquina equivalente a U' . Para cada estado final allí, lo cambiamos para leer e ir a la derecha sin parar. Llamamos a esta máquina \bar{U} . Ahora, $\bar{U}(M)$ para si M no para presentada con su propia especificación. Lo mismo que U'' , pero si $M(M)$ para, U'' para en un estado final, y consecuentemente \bar{U} no para.

Entonces $\bar{U}(M)$ para si y sólo si $M(M)$ no para.

¿Qué pasa entonces con $\bar{U}(\bar{U})$?

$\bar{U}(\bar{U})$ para si y sólo si $\bar{U}(\bar{U})$ no para.

Repasando las observaciones que nos condujeron a esta contradicción, el culpable era la máquina U' , supuestamente capaz de decidir si una máquina M iba a poder decidir sobre una entrada c . Eso se llama *el problema de la parada*, y es el primer *problema indecidible* conocido. Desde su descubrimiento por Turing, se ha descubierto un montón más, algunos de los cuales vamos a ver aquí.

Ejercicios de Autómatas

1. (1) Demuestre que la familia de conjuntos regulares sobre un alfabeto Σ es cerrada bajo intersección y complemento respecto de Σ .
2. (3) Demuestre que todo conjunto finito es regular, y que la familia de conjuntos regulares es cerrada bajo unión, concatenación y la operación $*$, y por tanto incluye todo $\mathbf{K}(\Sigma)$. Autómatas no deterministas pueden ser útiles en este ejercicio.
3. (1) Haga un autómata finito que acepta cadenas en $\{0, 1\}^*$ que interpretados como números expresados en binario son divisibles por 9.
4. (2) Haga un autómata finito que acepta cadenas en $\{0, 1\}^*$ que son divisibles por 2 o por 3. Por ejemplo, las cadenas de 4 dígitos que acepta son 0000, 0010, 0011, 0100, 0110, 1000, 1001, 1010, 1100, 1110, 1111. Sugerencia: hágalo como una unión.
5. (3) Haga una máquina secuencial con $\Sigma = \{0, 1\} \times \{0, 1\}$ y $\Delta = \{0, 1\}$ tal que, para toda cadena no nula en Σ^* , $\lambda((a_0, b_0), \dots, (a_n, b_n)) = \text{rep}(\sum_{i=0}^n a_i 2^i + \sum_{i=0}^n b_i 2^i) \bmod (2^n, 2)$.
6. (4) Sea A un autómata finito. Definimos para $p, q \in K$, $p \equiv_i q$ si para cada $c \in \Sigma^*$ con $|c| \leq i$, $\delta(p, c) \in F \Leftrightarrow \delta(q, c) \in F$. Es claro que \equiv_i es una relación de equivalencia. Sea π_i la partición de K inducida por \equiv_i . Demuestre que para todos estados p, q :
 - a) $p \equiv_0 q$ si y sólo si $p \in F \Leftrightarrow q \in F$
 - b) para todo $i > 0$, $p \equiv_i q$ si y sólo si $p \equiv_{i-1} q$ y $\delta(p, a) \equiv_{i-1} \delta(q, a)$ para cada $a \in \Sigma$
 Definimos $p \equiv q$ si para cada $c \in \Sigma^*$, $\delta(p, c) \in F \Leftrightarrow \delta(q, c) \in F$. Sea π_i la partición de K inducida por \equiv_i y sea π la inducida por \equiv . Demuestre que si $\pi_i = \pi_{i+1}$ entonces $\pi_i = \pi$, y que $\pi_i = \pi_{i+1}$ para algún $i \leq |K|$.

Demuestre que así se puede construir un autómata B con el mínimo número posible de estados tal que $C(B) = C(A)$.

He aquí un ejemplo de cómo son esas π_i para un autómata que detecta divisibilidad por 6 en cadenas binarias. Los estados son $\{0, 1, 2, 3, 4, 5\}$, el estado inicial y único estado final es 0, y $\delta(i, j)$ para un estado i y entrada j se encuentra en la fila j , columna i de la siguiente tabla:

	0	1	2	3	4	5
0	0	0	2	4	0	2
1	1	1	3	5	1	3

En la siguiente, $[q]_i$ para un estado q y número i es la clase de q en π_i :

$$\begin{aligned}
 [0]_0 &= \{0\} \\
 [1]_0 &= \{1, 2, 3, 4, 5\} \\
 [1]_1 &= \{1, 2, 4, 5\} \\
 [1]_2 &= \{1, 4\} \\
 [1]_3 &= \{1, 4\} \\
 [2]_2 &= \{2, 5\} \\
 [2]_3 &= \{2, 5\} \\
 [3]_1 &= \{3\}
 \end{aligned}$$

y de allí se ve que $\pi_2 = \pi_3$ y entonces que $\pi = \{\{0\}, \{1, 4\}, \{2, 5\}, \{3\}\}$. La máquina equivalente reducida es

	0	1	2	3
0	0	0	2	1
1	1	1	3	2

7. (3) Construya un autómata finito mínimo para cada uno de los siguientes conjuntos. Sugerencia: se puede cortar el trabajo notando que si $p \equiv_i q$ para algún i y se sabe ya que $\delta(p, a) \equiv \delta(q, a)$ para todo $a \in \Sigma$ (usualmente porque son iguales) entonces $p \equiv q$.
 - a) Las cadenas de $\{0, 1\}^*$ que, interpretadas en binario, representan un múltiplo de 4.

b) Las cadenas de $\{0, 1\}^*$ que, interpretadas en binario, representan un múltiplo de 12. (Sugerencia: construirlo como la intersección de los múltiplos de 3 y de 4 para tener menos estados al principio).

c) Las cadenas de $\{a, l, r, o\}^*$ que terminan en una de las cadenas del conjunto $S = \{ala, alla, loro, rala, ralo, oro, ola, olla, olor, oral, orar, ora, rol, ara, aro, arar, rara, raro, lloa, llo, lloa, lloa\}$. Sugerencia: cada estado va a ser una cadena en $\{a, l, r, o\}^*$ de longitud no más de 4. Todas van a ser prefijos de alguna cadena en S . Los prefijos de S son

$$S \cup \{\epsilon, a, l, r, o, al, ar, lo, ra, ro, or, ol, all, oll, lor, ral, rar, olo, ll, llo, llor\}$$

(vea allí π_0).

8. (2) Sean $A = (K_A, \delta_A, \Sigma, p_{0A}, F_A)$, $B = (K_B, \delta_B, \Sigma, p_{0B}, F_B)$ dos autómatas finitos. Sea

$$C = (\{(p, q) : p \in K_A, q \in K_B\}, \delta_C, \Sigma, (p_{0A}, p_{0B}), \{(p, q) : p \in F_A, q \in F_B\})$$

donde $\delta_C((p, q), a) = ((\delta_A(p, a), \delta_B(q, a)))$ para cada $p \in A$, $q \in B$, $a \in \Sigma$. ¿Si A y B son reducidos (es decir, con el mínimo número de estados), es C necesariamente reducido? Justifique la respuesta.

9. (3) Haga una máquina de Turing que, presentada con una cinta $\beta^\infty 1^n 01^m 01^p \beta^\infty$, y ubicada sobre el primer 1 a la izquierda, acepta si $p = m \times n$ y rechaza si no. Use cualquier Δ que sea conveniente.
10. (2) Sea C el conjunto de cadenas sobre $\{0, 1\}$ que, leídas como binarios, representan un múltiplo de 3. Construir un autómata finito determinista que acepta C^R .
11. (2) Construya un autómata finito determinista que acepta C^R , donde C es el conjunto de cadenas binarias que no representan ni un múltiplo de 2 ni de 3. Nótese que no es necesario construir 2^6 estados, sino sólo los que surgen del estado $\{p_{0B}\}$.
12. (2) Si no tuviéramos tanta confianza en la capacidad del autor en hacer manipulaciones simbólicas sin error, dudaríamos que la expresión regular $(0 \cup 11^*0)^*11^*$ representa el conjunto $\{0, 1\}^*1$ aceptado por el autómata 1. Pero el autor mismo no tiene esa confianza y no lo creyó hasta demostrarlo directamente. Hágalo Ud.

GRAMÁTICAS

Imagínese que los usuarios de un sistema que Ud. mantiene piden que se incorpore una calculadora que hace sumas, restas, productos, cocientes y exponenciación con números enteros. (No están conformes con la calculadora del sistema operativo porque quieren ver toda la expresión en la pantalla antes de mandarla a evaluar.)

Su primer problema es la evaluación de expresiones; sin hacer eso bien no sirve cualquier cantidad de íconos bonitos. Y el primer problema con expresiones es decidir qué son y qué significan; hasta arreglar eso no vale la pena hacer ni una línea de código.

Tenemos que evaluar expresiones como $3 * (27 - 10) * (27 + 10) + 2^9$.

Las computadoras no vienen sabiendo que $3 + 5 * 7$ es 38 y no 56. Tengo una calculadora manual que cree que es 56, es decir, lo mismo que $(3 + 5) * 7$. Un mecanismo que se llama *gramática* lo hace claro:

- Una *expresión* consiste en uno o más *términos* unidos por $+$ o $-$.
- Un *término* consiste en uno o más *factores* unidos por $*$ o $/$.

El decir eso ya implica que evaluamos los términos antes de sumarlos, y que para evaluar un término tenemos que evaluar los factores y luego aplicarles las operaciones de multiplicar o dividir. Es decir que multiplicación y división tienen precedencia sobre suma y resta. Excepto en una expresión con paréntesis: $(4 + 5) * 3$ es 27; la suma se hace primero porque los paréntesis dicen “evalúe lo que encerramos nosotros antes de interactuar con el resto de la expresión.”

Además de la asociación de operadores de la misma precedencia, tenemos signos unarios (como en -5) y exponenciación. Claro que $-5 + 8$ no se interpreta como $-(5 + 8)$, es decir, un signo unario tiene precedencia sobre $+$ y $-$ binario. En la expresión $-2 * 3$ no importa mucho si se evalúa $(-2) * 3$ o $-(2 * 3)$; vamos a elegir el primero. ¿Qué me dice de -2^4 ? Es decir, tiene el $-$ unario precedencia sobre exponenciación? La convención es que no, $-2^4 = -16$, es decir se interpreta como $-(2^4)$; exponenciación tiene precedencia.

El mecanismo formal que se llama *gramática libre de contexto* aclara todas estas reglas de evaluación en una forma muy compacta y clara, que además facilita mucho la construcción de la calculadora.

La afirmación dada arriba, que una expresión es una suma de términos, deja abierta la cuestión de la asociatividad en la ausencia de paréntesis. Si decimos que una expresión es un término posiblemente seguido por $+$ o $-$ y una expresión, entonces sí estamos implicando una cierta asociatividad, pero no la que queremos, porque en “ $3 - 5 + 7$ ”, 3 sería el término y “ $5 + 7$ ” la expresión, y según eso la expresión pide restar $5 + 7$, 12, de 3, con resultado -9 . Lo que vamos a decir es que una expresión es un término posiblemente precedido por una expresión y un signo. La notación de gramáticas expresa esto así:

$$E \rightarrow T \mid E' + T \mid E' - T$$

Ya que la E de $E' + T$ puede ser otra $E' + T$, tenemos que concluir que una E puede ser $E' + T' + T$, y por tanto, ya que esa E puede ser también simplemente T , E puede ser $T' + T' + T$. También podemos decir que un término puede ser un factor posiblemente precedido por un término y un signo de $*$ o $/$. En la notación compacta de gramáticas:

$$T \rightarrow F \mid T' * F \mid T' / F$$

Antes de decir cuáles son las reglas para los factores F y lo demás, vamos a establecer formalmente qué es una gramática y considerar algunos ejemplos más sencillos. Luego volvemos a desarrollar la gramática para expresiones y examinar algunas de sus consecuencias.

1. Lenguajes libres de contexto

DEFINICIÓN 5.1. Una *gramática libre de contexto* consiste de un alfabeto particionado en *variables sintácticas* y *símbolos finales*, una variable sintáctica distinguida que se llama *símbolo inicial*, y unas *producciones* de la forma

$$V \rightarrow w$$

donde V es una variable sintáctica y w es una cadena sobre el alfabeto.

DEFINICIÓN 5.2. Un *árbol de derivación con letra inicial x y cadena final w* de una gramática se define así inductivamente:

- Cada x en el alfabeto es un árbol de derivación con letra inicial x y cadena final x .
- Si V es una variable sintáctica, $V \rightarrow c$ una producción con $c = c_1, \dots, c_k$ y A_1, \dots, A_k son árboles de derivación con letras iniciales c_1, \dots, c_k y cadenas finales d_1, \dots, d_k entonces $\langle V, A_1, \dots, A_k \rangle$ es un árbol de derivación con letra inicial V y cadena final $d_1 \dots d_k$. Se dice que V *engendra* $d_1 \dots d_k$.

Una *cadena terminal* es una cadena sobre los símbolos finales.

DEFINICIÓN 5.3. El *lenguaje* de una gramática es el conjunto de todas las cadenas terminales engendradas por su variable inicial.

Vamos a ejercitar estas definiciones sobre una gramática más sencilla que la de las expresiones pero no totalmente trivial:

Variable sintáctica y variable inicial: E .

Símbolos finales: $a, (,), [,]$.

Producciones: $E \rightarrow a, E \rightarrow (E), E \rightarrow [E], E \rightarrow EE$.

NOTACIÓN. El conjunto de cuatro producciones que acabamos de dar, puede abreviarse como

$$E \rightarrow a \mid (E) \mid [E] \mid EE$$

El lenguaje de esta gramática es todas las cadenas sobre $a, (,), [,]$ en que los paréntesis y corchetes hacen juego, pero en que encierren algún “contenido”, es decir, que no ocurran subcadenas $()$ ni $[]$.

Por ejemplo, $\langle E, (, (E),) \rangle$ es un árbol de derivación con letra inicial E y cadena final (E) . Lo mismo $\langle E, [, E,] \rangle$. Por la producción $E \rightarrow a$, $\langle E, a \rangle$ es un árbol de derivación con letra inicial E y cadena final a . Por tanto $\langle E, (, \langle E, a \rangle,) \rangle$ y $\langle E, [, \langle E, a \rangle,] \rangle$ son árboles de derivación con letra inicial E y cadenas finales (a) y $[a]$ respectivamente. Por la producción $E \rightarrow EE$ entonces el árbol $\langle E, \langle E, (, \langle E, a \rangle,) \rangle, \langle E, [, \langle E, a \rangle,] \rangle \rangle$ es un árbol de derivación con letra inicial E y cadena final $(a)[a]$, mostrando que esa cadena está en el lenguaje.

Se puede ver, o demostrar por inducción fácilmente, que para cualquier gramática, si hay un árbol de derivación con letra inicial V y cadena final $c_1 W c_2$ donde W es una variable sintáctica con producción $W \rightarrow d$, entonces hay un árbol de derivación con letra inicial V y cadena final $c_1 d c_2$. Es decir, si una variable V engendra una cadena con una variable W y aplicamos una producción $W \rightarrow d$ allí, la cadena que resulta está engendrada por V .

¿El lenguaje del ejemplo engendra todas las cadenas en que los paréntesis hacen juego? Primero vemos que engendra todas las cadenas a^n (n repeticiones de a). Si no engendra a^n tampoco engendra a^{n-1} , porque si engendra a^{n-1} es porque hay un árbol de derivación A con letra inicial E y cadena final a^{n-1} , pero ya hemos visto que hay un árbol B con letra inicial E y cadena final a : por lo tanto $\langle E, A, B \rangle$ es, por la producción $E \rightarrow EE$, un árbol de derivación con letra inicial E y cadena final a^n . Ahora vamos a ver cadenas con $($. Si es (c) con esos dos paréntesis haciendo juego, entonces los paréntesis hacen juego en c también, así que si E no

engendra (c) tampoco puede engendrar c , porque con el árbol A con que engendra c hacemos el árbol $\langle E, (, A,) \rangle$, que por la producción $E \rightarrow (E)$ es árbol de derivación, y tiene cadena final (c) . Una cadena que no empieza y termina con paréntesis o corchetes que hacen juego tiene que ser cd con c y d no nulos y con los paréntesis haciendo juego. Si E engendra las dos, entonces con los árboles A y B que las engendran construimos el árbol $\langle E, A, B \rangle$ que es árbol de derivación por la producción $E \rightarrow EE$ y tiene cadena final cd . Entonces se ve que, en cualquier caso, si hay una cadena en el conjunto no engendrada por E , hay una cadena más corta en el conjunto no engendrada por E , y eso es imposible porque ¿que hacemos cuando llegamos a longitud 1?

La importancia de los árboles de derivación es que nos ayudan a entender el significado de las cadenas. Si G es una gramática con símbolo inicial S y letras finales Σ , si hay $c \in \Sigma^*$ tal que hay dos árboles de derivación distintos con letra inicial S y cadena final c eso constituye una *ambigüedad* y decimos que la gramática es *ambigua*. La gramática del ejemplo es ambigua. La cadena $(a)(a)(a)$ es la cadena final del árbol de derivación

$$\langle E, \langle E, (, \langle E, a \rangle,) \rangle, \langle E, \langle E, \langle E, (, \langle E, a \rangle,) \rangle, \langle E, (, \langle E, a \rangle,) \rangle \rangle \rangle \rangle,$$

y también del árbol de derivación

$$\langle E, \langle E, \langle E, (, \langle E, a \rangle,) \rangle, \langle E, (, \langle E, a \rangle,) \rangle \rangle, \langle E, (, \langle E, a \rangle,) \rangle \rangle.$$

Claro que estos árboles se muestran más amablemente por dibujos, ya que son grafos (con la complicación adicional que las aristas saliendo de un nodo vienen ordenadas, y ese orden es de importancia fundamental.) He aquí una notación más compacta que los dibujos pero más amena que los nidos de \langle y \rangle . Los dos árboles de derivación que acabamos de ver se muestran así en esta notación:

$ \begin{array}{l} E \rightarrow \\ . \quad E \rightarrow \\ . \quad . \quad (\\ . \quad . \quad E \rightarrow a \\ . \quad . \quad) \\ . \quad E \rightarrow \\ . \quad . \quad E \rightarrow \\ . \quad . \quad . \quad (\\ . \quad . \quad . \quad E \rightarrow a \\ . \quad . \quad . \quad) \\ . \quad . \quad E \rightarrow \\ . \quad . \quad . \quad (\\ . \quad . \quad . \quad E \rightarrow a \\ . \quad . \quad . \quad) \end{array} $	$ \begin{array}{l} E \rightarrow \\ . \quad E \rightarrow \\ . \quad . \quad E \rightarrow \\ . \quad . \quad . \quad (\\ . \quad . \quad . \quad E \rightarrow a \\ . \quad . \quad . \quad) \\ . \quad . \quad E \rightarrow \\ . \quad . \quad . \quad (\\ . \quad . \quad . \quad E \rightarrow a \\ . \quad . \quad . \quad) \\ . \quad E \rightarrow \\ . \quad . \quad (\\ . \quad . \quad E \rightarrow a \\ . \quad . \quad) \end{array} $
--	--

Cada fila de una de estas representaciones termina o con \rightarrow o con un símbolo terminal. Los símbolos terminales leídos desde arriba hacia abajo muestran la palabra engendrada. También se puede mostrar un árbol no completo así, por ejemplo

$$\begin{array}{l}
 E \rightarrow \\
 . \quad E \rightarrow \\
 . \quad E \rightarrow
 \end{array}$$

Ya que no hay producción con el lado derecho nulo, la clave de si un árbol es completo es si cada línea que termina con \rightarrow está seguida por al menos una línea indentada un lugar más.

La siguiente gramática tiene el mismo lenguaje pero no es ambigua:

$$\begin{array}{l}
 E \rightarrow D \mid DE \mid F \\
 D \rightarrow aD \mid (E) \mid [E] \\
 F \rightarrow a \mid Fa
 \end{array}$$

La idea de esta gramática es que para engendrar una palabra p , si p no tiene ni paréntesis ni corchetes hay que empezar con $E \rightarrow F$ y claramente las derivaciones desde F no son ambiguas; y si los tiene, digamos $p = a^n(q)r$ tal que los dos paréntesis hacen juego y r no es nulo,

la derivación tiene que empezar con $E \rightarrow DE$ y después $D \rightarrow aD$ n veces (puede ser 0) y $D \rightarrow (E)$, que esta E engendra q y la segunda E engendra r , reduciendo el problema a palabras más cortas. Si r es nulo, la única diferencia es que estamos obligados a empezar con $E \rightarrow D$.

Ahora estamos en condiciones de atacar la gramática de expresiones. Pero omitimos el menos unario para dejarlo como ejercicio:

$$\begin{aligned} E &\rightarrow T \mid E' + ' T \mid E' - ' T \\ T &\rightarrow F \mid T' * ' F \mid T' / ' F \\ F &\rightarrow P \mid P' \wedge ' F \\ P &\rightarrow N \mid '(E)' \\ N &\rightarrow D \mid ND \\ D &\rightarrow '1' \mid '2' \mid '3' \mid '4' \mid '5' \mid '6' \mid '7' \mid '8' \mid '9' \mid '0' \end{aligned}$$

Aquí con P estamos pensando “primario”, con N , “número”, y con D , “dígito”. Por ejemplo, para derivar $(5 + 3) * (5 - 3)$ hacemos primero $E \rightarrow T \rightarrow T * F$. Con el F hacemos $F \rightarrow P \rightarrow (E)$, y con el T hacemos $T \rightarrow F \rightarrow P \rightarrow (E)$, así que hemos llegado a $(E) * (E)$. Ahora queremos hacer llegar la primera de estas dos E a $5 + 3$: $E \rightarrow E + T$, y para la T hacemos $T \rightarrow F \rightarrow P \rightarrow N \rightarrow D \rightarrow 3$. Con la E sigue una derivación similar terminando en 5. La derivación de T a $5 - 3$ es similar. Mostramos esquemáticamente toda la derivación así:

$$\begin{aligned} E &\rightarrow T \rightarrow \\ . \quad . \quad . \quad &T \rightarrow F \rightarrow P \rightarrow \\ . \quad . \quad . \quad . \quad & (\\ . \quad . \quad . \quad . \quad & E \rightarrow \\ . \quad . \quad . \quad . \quad . \quad & E \rightarrow T \rightarrow F \rightarrow N \rightarrow D \rightarrow 5 \\ . \quad . \quad . \quad . \quad . \quad & + \\ . \quad . \quad . \quad . \quad . \quad & T \rightarrow F \rightarrow N \rightarrow D \rightarrow 3 \\ . \quad . \quad . \quad . \quad &) \\ . \quad . \quad . \quad & * \\ . \quad . \quad . \quad & F \rightarrow P \rightarrow \\ . \quad . \quad . \quad . \quad & (\\ . \quad . \quad . \quad . \quad & E \rightarrow \\ . \quad . \quad . \quad . \quad . \quad & E \rightarrow T \rightarrow F \rightarrow P \rightarrow N \rightarrow D \rightarrow 5 \\ . \quad . \quad . \quad . \quad . \quad & - \\ . \quad . \quad . \quad . \quad . \quad & T \rightarrow F \rightarrow P \rightarrow N \rightarrow D \rightarrow 3 \\ . \quad . \quad . \quad . \quad &) \end{aligned}$$

Se puede demostrar que esta gramática no es ambigua. Pero mejor que una demostración es simplemente pensarlo bien, que permite una sola interpretación de cada expresión.

Pero la ambigüedad es tramposa. La gramática para el lenguaje Algol ya estaba publicada hacía unos dos años cuando Floyd mandó una carta a *Communications of the Association for Computing Machinery* indicando que tenía una ambigüedad. Irónicamente, ya se había hecho varios compiladores cuyos autores jarreglaron la ambigüedad sin darse cuenta! La ambigüedad fue la famosa “else colgante”:

<sentencia> \rightarrow **if** <condicion> <sentencia> **|** **if** <condicion> <sentencia> **else** <sentencia>

Entonces

“**if** $x < 5$ **if** $x > 2$ $x := 5$ **else** $x := 0$ ”

se podía interpretar como

“**if** $x < 5$ (**if** $x > 2$ $x := 5$ **else** $x := 0$)”

o como

“**if** $x < 5$ (**if** $x > 2$ $x := 5$) **else** $x := 0$ ”

En el primer caso si $x \geq 5$ no pasa nada, y en el segundo ¡todo $x \geq 5$ se anula!

Ahora vamos a construir una función que evalúa una cadena producida por la gramática. Otra vez omitimos el menos unario para dejarlo como ejercicio. La función y sus ayudantes

contestan un valor y la cadena que queda después de la cadena inicial que tiene ese valor. Hay una función para cada variable sintáctica, que busca la cadena inicial más larga engendrable por esa variable sintáctica y la evalúa.

```

evalE(e) =
.  (n, f) := evalT(e)
.  evalE1(n, f)
evalE1(n, e) =
.  si(e =  $\emptyset$  | no(e1 = ' + ' | e1 = ' - ')) (n, e)
.  sino
.  .  sea(m, f) = evalT(resto(e))
.  .  si(e1 = ' + ') evalE1((n + m, f))
.  .  sino evalE1((n - m, f))
evalT(e) =
.  (n, f) := evalF(e)
.  evalT1(n, f)
evalT1((n, e)) =
.  si(e =  $\emptyset$  | no(e1 = ' * ' | e1 = ' / ')) (n, e)
.  sino
.  .  sea(m, f) = evalF(resto(e))
.  .  si(e1 = ' * ') evalT1((n * m, f))
.  .  sino evalT1((n / m, f))
evalF(e) =
.  sea(n, f) = evalP(e)
.  si(f =  $\emptyset$  | f1 ≠ '^' (n, f))
.  sino
.  .  sea(m, g) = evalF(resto(f))
.  .  (nm, g)
evalP(e) =
.  si(e1 es dígito) evalN(e)
.  sino si(e1 = '^')
.  .  sea (n, f) = evalE(resto(e))
.  .  si(f1 = '^') (n, resto(f)) (sino error, y no decidimos todavía que hacemos con errores)
evalN(e) = evalN1(0, e)
evalN1(n, e) =
.  si(e =  $\emptyset$  | e1 no es dígito) (n, e)
.  sino evalN1(10n + valor(e1), resto(e))
valor('0') = 0, valor('1') = 1, etc.

```

Veamos el desarrollo de $evalE('3 * (9 - 5)')$. Para ello, cuando se llega a formar el valor que contesta una función, se pone ese valor en la línea en que fue invocada esa función.

```

evalE('3 * (9 - 5)') = evalE1(evalT('3 * (9 - 5)') = (12, ''))
.  evalT('3 * (9 - 5)') = (12, '') → evalE1()
.  .  evalF('3 * (9 - 5)') = (3, ' * (9 - 5)') → evalT1()
.  .  .  evalP('3 * (9 - 5)') = evalN(' * (9 - 5)') = (3, ' * (9 - 5)') → (n, f)
.  .  .  .  evalN('3 * (9 - 5)') = evalN1(0, ' * (9 - 5)') = (3, ' * (9 - 5)')
.  .  .  .  .  evalN1(0, '3 * (9 - 5)') = evalN1(3, ' * (9 - 5)') = (3, ' * (9 - 5)')
.  .  .  .  .  .  evalN1(3, ' * (9 - 5)') = (3, ' * (9 - 5)')
.  .  .  (n, f) = (3, ' * (9 - 5)')
.  .  evalT1((3, ' * (9 - 5)') = (12, ''))
.  .  .  evalF(' * (9 - 5)') = (4, '') → (m, f)
.  .  .  .  evalP(' * (9 - 5)') = (4, '') → (n, f)
.  .  .  .  .  evalE(' * (9 - 5)') = (4, '^') → (n, f)

```

```

. . . . . evalT('9 - 5') = (9, ' - 5') → evalE1()
. . . . . evalF('9 - 5') = (9, ' - 5') → evalT1()
. . . . . evalP('9 - 5') = (9, ' - 5') → (n, f)
. . . . . evalN('9 - 5') = (9, ' - 5')
. . . . . evalN1(0, '9 - 5') = (9, ' - 5')
. . . . . evalN1(9, ' - 5') = (9, ' - 5')
. . . . . evalT1(9, ' - 5') = (9, ' - 5')
. . . . . evalE1(9, ' - 5') = (4, ' ') → (m, f)
. . . . . evalT('5') = (5, ' ') → (m, f)
. . . . . evalF('5') = (5, ' ') → evalT1()
. . . . . evalP('5') = (5, ' ') → (n, f)
. . . . . evalN('5') = (5, ' ')
. . . . . evalN1(0, '5') = (5, ' ')
. . . . . evalN1(5, ' ') = (5, ' ')
. . . . . evalT1(5, ' ') = (5, ' ')
. . . . evalT1(12, '"') = (12, '"')
. . . . evalE1(12, '"') = (12, '"')

```

Hablemos un poco de la implementación de esto. Es claro que el argumento s y el componente t de la respuesta sean segmentos de la misma cadena en memoria global. Pero podemos suponer más de esa cadena: que la estamos leyendo desde un archivo, y nunca tenemos que rebobinar ese archivo: lo más que tenemos que hacer es guardar un operador u otro carácter de vez en cuando. Entonces vamos a suponer que estamos leyendo una cadena c y usamos una variable global i para indicar dónde estamos en la lectura de c . Cada vez que una función decide que el próximo carácter es suyo aumenta i en 1. Las funciones siguen contestando un par, un número y una cadena, pero contestan la cadena por medio de i . Suponemos un carácter fin que indica la terminación de c . La función principal $eval$ inicializa i , invoca a $evalE$, y después verifica que $evalE$ usó todo de c (aquí abreviamos $i = i + 1$ con $i++$).

$eval() =$	$evalF() =$
. $i=1$. si ($c_i = ' - '$)
. $n=evalE()$. . $i++$
. si ($c_i \neq fin$) error	. . $-evalF()$
. sino contestar n	. sino
$evalE() = evalE1(evalT())$. . $n = evalP()$
$evalE1(n) =$. . si ($c_i = \Lambda$)
. si ($c_i \in \{'+', '-'\}$)	. . . $i++$
. . . $op = c_i$. . . $n^{evalF()}$
. . . $i++$. . sino n
. . . $m = evalT()$	$evalP() =$
. . . $evalE1(\mathbf{si}(op = ' + ') \ n + m \ \mathbf{sino} \ n - m)$. si ($c_i = ' ('$)
$evalT() = evalT1(evalF())$. . $i++$
$evalT1(n) =$. . $n = evalE()$
. si ($c_i \in \{'*', '/'\}$)	. . si ($c_i = ')'$) $i++$; contestar n
. . . $op = c_i$. sino $evalN()$
. . . $i++$	$evalN() = evalN1(0)$
. . . $m = evalF()$	$evalN1(n) = \mathbf{si}(c_i \text{ no es dígito}) \ n$
. . . $evalE1(\mathbf{si}(op = ' * ') \ n * m \ \mathbf{sino} \ n/m)$	sino $m = 10n + c_i; \ i++; evalN1(m)$

2. Autómatas de Pila

Hay un tipo de memoria que es ilimitado pero con acceso restringido. Se llama *pila*. La restricción es que el autómata sólo puede ver el último carácter puesto: es como que se lo pone

arriba de una pila y para ver qué caracteres hay abajo hay que sacar el primero. La memoria finita interna del autómata sólo tiene lugar para un número finito de estos caracteres. Es algo sorprendente que, como veremos, este tipo de restricción caracteriza a los lenguajes libre de contexto.

DEFINICIÓN 5.4. Un *autómata de pila* M es $(K, \Sigma, \Delta, \rightarrow, p_0, Z_0)$ donde

- K es un conjunto finito de *estados*.
- Σ es un alfabeto finito (que compone las cadenas de entrada).
- Δ es un alfabeto finito (que compone las cadenas de la pila).
- \rightarrow es una relación de $K \times (\Sigma \cup \{\epsilon\}) \times \Delta$ a $K \times \Delta^*$.
- $p_0 \in K$ es el estado inicial.
- $Z_0 \in \Delta$ es el símbolo de pila inicial.

No termina la definición todavía; tenemos que decir qué es lo que hacen estos dispositivos. Una *configuración* de un autómata de pila es (q, c, w) donde $q \in K$, $c \in \Sigma^*$, $w \in \Delta^*$. Si $(p, a, Z) \rightarrow (q, w)$ decimos que $m = (p, a, Z) \rightarrow (q, w)$ es un *movimiento* de M , y para cualquier configuración $\mathbf{c} = (p, ac, Zv)$, decimos que m es *aplicable* a \mathbf{c} y que $m(p, ac, Zv) = (q, c, wv)$. Un *programa* P es una sucesión de movimientos, y definimos $P(\mathbf{c})$ para configuraciones \mathbf{c} inductivamente como sigue: si $|P| = 0$, P es *aplicable* a \mathbf{c} y $P(\mathbf{c}) = \mathbf{c}$. Si no, $P = mP'$ para algún movimiento m y programa P' , y decimos que P es *aplicable* a \mathbf{c} y que $P(\mathbf{c}) = P'(m(\mathbf{c}))$ si m es aplicable a \mathbf{c} y P' es aplicable a $m(\mathbf{c})$. Finalmente, el *lenguaje* de M es

$$L(M) = \{c \in \Sigma^* : \text{existe un programa } P \text{ tal que } P(p_0, c, Z_0) = (q, \epsilon, \epsilon) \text{ para algún estado } q\}$$

Vamos a usar la notación $\mathbf{c} \vdash \mathbf{d}$, o $\mathbf{c} \xrightarrow{M} \mathbf{d}$ cuando hay más de un autómata a la vista, para dos configuraciones \mathbf{c}, \mathbf{d} cuando existe un programa P tal que $P(\mathbf{c}) = \mathbf{d}$

¡Wow! Mejor ver un ejemplo ya. En general, los autómatas de pila son no deterministas, es decir que se puede encontrar una configuración \mathbf{c} a la que son aplicables dos o más movimientos distintos, por eso una relación \rightarrow en lugar de una función. Pero nuestro primer ejemplo es determinista:

- $K = \{p_0, q\}$.
- $\Sigma = \{0, 1\}$.
- $\Delta = \{0, 1, Z_0\}$.
- $(p_0, 0, Z_0) \rightarrow (p_0, 0), (p_0, 0, 0) \rightarrow (p_0, 00), (p_0, 1, 0) \rightarrow (q, \epsilon), (q, 1, 0) \rightarrow (q, \epsilon)$.

Este autómata acepta el conjunto $\{0^n 1^n : n > 0\}$ engendrado por la gramática $S \rightarrow 01|0S1$. Por ejemplo, si los cuatro movimientos se llaman respectivamente m_1, m_2, m_3, m_4 y $P = m_1 m_2 m_3 m_4$, entonces $P(p_0, 0011, Z_0) = (q, \epsilon, \epsilon)$.

Otro ejemplo: el conjunto de cadenas sobre $\{0, 1\}$ que contienen igual número de 1 que de 0. Una gramática es

$$S \rightarrow \epsilon | 0S1 | 1S0 | SS.$$

Esta vez incluimos el caso $n = 0$. Un autómata de pila que acepta esto: $\Sigma = \{0, 1\}$, $\Delta = \{S, 0, 1\}$, $K = \{p\}$, el símbolo inicial es S , y \rightarrow es la siguiente relación:

$$(p, 0, S) \rightarrow (p, 0S), (p, 1, S) \rightarrow (p, 1S), (p, \epsilon, S) \rightarrow (p, \epsilon), (p, 0, 0) \rightarrow (p, 00), (p, 1, 1) \rightarrow (p, 11), (p, 1, 0) \rightarrow (p, \epsilon), (p, 0, 1) \rightarrow (p, \epsilon).$$

El método del autómata es tener el exceso de 0 sobre 1 en la pila, si hay, como una fila de 0, o el exceso de 1 sobre 0, si hay, como una fila de 1. Si son iguales, la pila tiene una S sola. Si los movimientos se llaman $m_1, m_2, m_3, m_4, m_5, m_6$ respectivamente y $P = m_1 m_5 m_2 m_6 m_1 m_5 m_3$, entonces $P(011001) = (p, \epsilon, \epsilon)$.

Este ejemplo es el tema de un ejercicio después. Hay una cosa rara: el autómata es no ambiguo, es decir, cada cadena aceptada es aceptada por un solo programa, pero la gramática sí es ambigua.

Vamos a demostrar algunos lemas básicos que parecen auto-evidentes para el que tiene la idea fundamental del autómata de pila, como son las leyes conmutativas cuando modelamos

los números naturales con los axiomas de Peano. Los demostramos formalmente para asegurar que el modelo es lo que parece que es.

Una cosa sorprendente sobre los autómatas de pila es que un solo estado es suficiente para aceptar el conjunto aceptado por cualquier autómata de pila. Dos lemas y sus corolarios facilitan la demostración de ésto. El primero dice que si un programa borra una cadena de la pila, la puede borrar cuando es prefijo de una cadena más larga. El segundo dice que si borra una cadena de la pila, se lo puede dividir en segmentos que borran cada letra si fuera la única en la pila.

LEMA 5.5. *Si P es un programa de un autómata de pila y $P(p, c, w) = (q, \epsilon, \epsilon)$, entonces para todos $d \in \Sigma^*$ y $v \in \Delta^*$, $P(p, cd, wv) = (q, d, v)$.*

Demostración. Por inducción sobre $|P|$. Si $|P| = 1$, entonces $c = a \in \Sigma \cup \{\epsilon\}$ y P consiste en un solo movimiento $(p, a, Z) \rightarrow (q, \epsilon)$. Entonces $(p, ad, Zv) \vdash (q, d, v)$: se cumple para $|P| = 1$. Suponga que se cumple siempre cuando $|P| = n$, y considere un caso cuando $|P| = n + 1$. Entonces $P = mP'$ donde m es un movimiento $(p, a, Z) \rightarrow (r, u)$. $c = ac'$ y $w = Zw'$, y $m(p, ac', Zw') = (r, c', uw')$. $w' \neq \epsilon$ porque P' no es vacío ya que $|P| > 1$, y por la hipótesis de inducción, ya que $|P'| = n$ y $P'(r, c', uw') = (q, \epsilon, \epsilon)$, $P'(r, c'd, uw'v) = (q, d, v)$. Pero $m(p, cd, wv) = m(p, ac'd, Zw'v) = (r, c'd, w'v)$. Pero $P(p, cd, wv) = P'(m(p, cd, wv))$, por tanto $P(p, cd, wv) = (q, d, v)$ y la inducción está completa. \square

Por este lema y una simple inducción sobre k , se obtiene:

COROLARIO 5.6. *Si P_1, \dots, P_k son programas, $c_1, \dots, c_k \in \Sigma^*$, $w_1, \dots, w_k \in \Delta^*$ y para cada $i \in \{1, \dots, k\}$, $P_i(q_i, c_i, w_i) = (q_{i+1}, \epsilon, \epsilon)$, entonces para todo $d \in \Sigma^*$ y $w \in \Delta^*$, $P_1 \dots P_k(q_1, c_1 \dots c_k d, w_1 \dots w_k w) = (q_{k+1}, d, w)$.*

LEMA 5.7. *Sean w_1 y w_2 cadenas no nulas de Δ^* , P un programa y p un estado. Si $P(p, c, w_1 w_2) = (q, \epsilon, \epsilon)$, entonces existen c_1, c_2 con $c_1 c_2 = c$, programas P_1, P_2 con $P_1 P_2 = P$, y un estado r tal que $P_1(p, c_1, w_1) = (r, \epsilon, \epsilon)$ y $P_2(r, c_2, w_2) = (q, \epsilon, \epsilon)$.*

Demostración. Es claro que si la pila tiene $w w_2$ en algún momento de la computación, el próximo movimiento va a dejar w_2 como sufixo en la pila. Entonces en algún momento una configuración va a tener solo w_2 en la pila. El programa que hace esto lo haría igual sin la presencia constante de w_2 al final, y es P_1 . La entrada que consume es c_1 y el estado que produce es r . Lo que queda de c es c_2 y el programa que completa la tarea borrando w_2 y produciendo el estado q es P_2 . \square

Una simple inducción usando el lema anterior permite obtener el siguiente resultado.

COROLARIO 5.8. *Si $P(q_1, c, w_1 \dots w_k) = (q_{k+1}, \epsilon, \epsilon)$ entonces existen $c_1 \dots c_k \in \Sigma^*$ con $c_1 \dots c_k = c$, estados q_2, \dots, q_k , y programas P_1, \dots, P_k con $P_1 \dots P_k = P$, tal que para cada $i \in \{1, \dots, k\}$ $P_i(q_i, c_i, w_i) = (q_{i+1}, \epsilon, \epsilon)$.*

TEOREMA 5.9. *Si M es un autómata de pila, existe un autómata de pila N de un solo estado tal que $L(M) = L(N)$.*

Demostración. Sea $M = (K_M, \Sigma, \Delta_M, \xrightarrow{M}, p_0, Z_0)$.

Sea $N = (\{p_N\}, \Sigma, \{S\} \cup \{pZq : p, q \in K_M, Z \in \Delta_M\}, \xrightarrow{N}, S, p_N)$, donde \xrightarrow{N} se define así:

- $(p_N, \epsilon, S) \xrightarrow{N} (p_N, p_0 Z_0 q)$ para todo $q \in K_M$,
- $(p_N, a, pZq) \xrightarrow{N} (p_N, (q_1 Z_1 q_2)(q_2 Z_2 q_3) \dots (q_n Z_n q))$ para cada sucesión q_2, \dots, q_n de estados de M , siempre cuando $(p, a, Z) \xrightarrow{M} (q_1, Z_1 \dots Z_n)$.
- Si $(p, a, Z) \xrightarrow{M} (q, \epsilon)$ para algún p, q, a, Z , entonces $(p_N, a, pZq) \xrightarrow{N} (p_N, \epsilon)$.

Ahora $c \in L(N)$ si y sólo si $(p_N, c, S) \vdash^N (p_N, \epsilon, \epsilon)$ si y sólo si para algún $p_0 Z_0 q$, $(p_N, c, p_0 Z_0 q) \vdash^N (p_N, \epsilon, \epsilon)$. Entonces tenemos que demostrar que para todos q y c , $(p_N, c, p_0 Z_0 q) \vdash^N (p_N, \epsilon, \epsilon)$ si y sólo si $(p_0, c, Z_0) \vdash^M (q, \epsilon, \epsilon)$.

Demostremos eso demostrando que para toda pareja de estados $p, q \in K_M$, $(p, c, Z) \vdash^M (q, \epsilon, \epsilon)$ si y sólo si $(p_N, c, pZq) \vdash^N (p_N, \epsilon, \epsilon)$.

Para la dirección \Rightarrow , sea P un programa de M tal que $P(p, c, Z) = (q, \epsilon, \epsilon)$. Procedemos por inducción sobre $|P|$ para demostrar que $(p_N, c, pZq) \vdash^N (p_N, \epsilon, \epsilon)$. Si $|P| = 1$, el primer movimiento m de P es algún $(p, a, Z) \xrightarrow{M} (q, w)$, y para que eso sea aplicable a la configuración (p, c, Z) , tiene que ser que $c = ac'$, para alguna c' . Para que éste sea el único movimiento, tiene que ser $c' = w = \epsilon$. Por definición de N , $(p_N, a, pZq) \xrightarrow{N} (p_N, w)$. Entonces $(p_N, a, pZq) \vdash^N (p_N, \epsilon, \epsilon)$: se cumple para $|P| = 1$. Suponga ahora que para algún $n > 1$ se cumple en todos los casos que $|P| < n$, y considere un caso con $|P| = n$ donde $P(p, c, Z) = (q, \epsilon, \epsilon)$. Sea $P = mP'$ donde m es el movimiento $(p, a, Z) \xrightarrow{M} (q_1, w)$. Para que sea aplicable a (p, c, Z) , tiene que ser $c = ac'$ para alguna c' , y entonces $m(p, c, Z) = m(p, ac', Z) = (q_1, c', w)$. No puede ser $w = \epsilon$ porque el programa P no terminó con eso y no podría seguir con una pila vacía. Entonces $w = Z_1 \dots Z_k$ con $k > 0$, y tiene que ser que $P'(q_1, c', Z_1 \dots Z_k) = (q, \epsilon, \epsilon)$.

Por el corolario 5.8, existen programas P_1, \dots, P_k con $P_1 \dots P_k = P'$, cadenas $c_1, \dots, c_k \in \Sigma^*$ con $c_1 \dots c_k = c'$, y estados $q_2 \dots q_{k+1} = q$ tal que $P_i(q_i, c_i, Z_i) = (q_{i+1}, \epsilon, \epsilon)$. Cada $|P_i| < n$ porque la suma de los $|P_i|$ es $|P'| = n - 1$, así que por la hipótesis de inducción, $(p_N, c_i, q_i Z_i q_{i+1}) \vdash^N (p_N, \epsilon, \epsilon)$. Por el corolario 5.6, $(p_N, c_1 \dots c_k, q_1 Z_1 q_2 \dots q_k Z_k q_{k+1}) \vdash^N (p_N, \epsilon, \epsilon)$. Por la definición de N , el movimiento $m = (p, a, Z) \xrightarrow{M} (q_1, Z_1 \dots Z_k)$ hace que $(p_N, a, pZq) \xrightarrow{N} (p_N, q_1 Z_1 q_2 \dots q_k Z_k q_{k+1})$ donde $q_{k+1} = q$. Hemos visto entonces que $(p_N, c, Z) \vdash^N (p_N, c', q_1 Z_1 q_2 \dots q_k Z_k q) \vdash^N (p_N, \epsilon, \epsilon)$, completando la inducción: $L(M) \subset L(N)$.

Para la otra dirección, suponga que $(p_N, c, pZq) \vdash^N (p_N, \epsilon, \epsilon)$. Sea P un programa de N tal que $P(p_N, c, pZq) = (p_N, \epsilon, \epsilon)$. Demostremos por inducción sobre $|P|$ que $(p, c, Z) \vdash^M (q, \epsilon, \epsilon)$. Si $|P| = 1$, tiene que ser que $c = a \in \Sigma \cup \{\epsilon\}$, y el único movimiento de P es $(p_N, a, pZq) \xrightarrow{N} (p_N, \epsilon)$. Eso puede ser únicamente en el caso que $(p, a, Z) \xrightarrow{M} (q, \epsilon)$, así que $(p, c, Z) \vdash^M (q, \epsilon)$, y se cumple para $|P| = 1$. Suponga ahora que para algún $n > 1$ se cumple en todos los casos que $|P| < n$, y considere un caso con $|P| = n$. Sea $P = mP'$ donde m es el movimiento $(p_N, a, pZq) \xrightarrow{m} (p_N, q_1 Z_1 q_2 \dots q_k Z_k q_{k+1})$, con $k > 0$ porque P' no es vacía pero no puede ser aplicable si la pila es ϵ , y con $q_{k+1} = q$. Entonces $P'(p_N, c', q_1 Z_1 q_2 \dots q_k Z_k q) = (p_N, \epsilon, \epsilon)$. Por el corolario 5.8 existen P_1, \dots, P_n , programas de N con $P_1 \dots P_n = P'$, y cadenas c_1, \dots, c_k con $c_1 \dots c_k = c'$ tales que $P_i(p_N, c_i, q_i Z_i q_{i+1}) = (p_N, \epsilon, \epsilon)$ para cada i . Por el corolario 5.6, $(q_1, c_1 \dots c_k, Z_1 \dots Z_k) \vdash^M (q, \epsilon, \epsilon)$. Esto, junto con el primer movimiento $(p, a, Z) \xrightarrow{M} (q_1, z_1 \dots Z_k)$, muestra que $(p, c, Z) \vdash^M (q, \epsilon, \epsilon)$, completando la inducción: $L(N) \subset L(M)$.

Por lo tanto, $L(N) = L(M)$. \square

Ahora es casi inmediato el hecho central, que los autómatas de pila son equivalentes a las gramáticas libres de contexto. Primero, dado un autómata de pila de un solo estado, sus movimientos son $(p, \epsilon, Z) \rightarrow (p, w)$ y $(p, a, Z) \rightarrow (p, w)$ que corresponden a producciones $Z \rightarrow w$ y $Z \rightarrow aw$, y se ve que lo que hace el autómata es una derivación por izquierda, es decir, una sucesión de cadenas cada una de las cuales se obtiene de la anterior con aplicar una producción a la primera variable sintáctica de la anterior.

Segundo, dada una gramática, para cada producción $V \rightarrow cw$, en donde la primera letra de w no está en Σ^* , le damos a un autómata de un solo estado p el movimiento $(p, \epsilon, V) \rightarrow (p, W)$, y le equipamos con los movimientos $(p, a, a) \rightarrow (p, \epsilon, \epsilon)$. Así, lo único que puede hacer este M es aplicar una producción de G y después borrar todas las letras de Σ que esa producción puso a la izquierda, si están de acuerdo con sus próximas entradas; al terminar esta operación o borrar todo, completando una computación, o se encuentra con una variable sintáctica a la cual aplica de nuevo ese mismo procedimiento.

Pero eso es argumento intuitivo. Demostrar ésto formalmente es más largo.

Empezamos con una definición formal de una derivación por izquierda. Para abreviar, dada una gramática usamos Δ para denotar su conjunto de variables sintácticas.

DEFINICIÓN 5.10. Una *derivación por izquierda* (o *izquierdista*) de la gramática G es una sucesión w_1, \dots, w_n de cadenas de $(\Sigma \cup \Delta)^*$ tal que para cada $i \in \{1, \dots, n-1\}$, $w_i = c_i Z_i v_i$ con $c_i \in \Sigma^*$ y $Z_i \in \Delta$, $Z_i \rightarrow u_i$ y $w_{i+1} = c_i u_i v_i$. Decimos que w_1, \dots, w_n es una *derivación por izquierda* de w_1 a w_n con *programa izquierdista* $Z_1 \rightarrow u_1, \dots, Z_{n-1} \rightarrow u_{n-1}$.

Es claro que $u \vdash v$ si y sólo si existe una derivación por izquierda w_1, \dots, w_n con $w_1 = u$ y $w_n = v$.

Esta noción de *programa* es similar a la de autómatas de pila, pero estos programas sólo pueden ser aplicados en una derivación por izquierda.

Primero definimos una nueva notación: informalmente, $pre(w)$ (“prefijo de w ”) para $w \in \Delta$ significa la cadena que se obtiene copiando todas las letras de Σ al principio de w hasta encontrar una letra de Δ o el fin de w ; y $suf(w)$ (“sufijo de w ”) es lo que queda de w después de borrar $pre(w)$. Nótese que $pre(w)suf(w) = w$. Por ejemplo, si $\Sigma = \{0, 1\}$, $pre(10Z00) = 10$ y $suf(10Z00) = Z00$. Formalmente:

NOTACIÓN. Definimos $pre(w)$ y $suf(w)$ para todo $w \in (\Sigma \cup \Delta_G)^*$ como sigue: Para todo $c \in \Sigma^*$, $Z \in \Delta$, y $v \in (\Sigma \cup \Delta)^*$, $pre(c) = c$, $suf(c) = \epsilon$, $pre(cZv) = c$, $suf(cZv) = Zv$.

PROPOSICIÓN 5.11. Dada una gramática G libre de contexto, existe un autómata de pila M tal que $L(M) = L(G)$.

Demostración. Sea Δ el conjunto de variables sintácticas de G y Σ el conjunto de sus letras finales. Escribimos las producciones de G como $V \xrightarrow{G} w$. Definimos el autómata de pila

$$M = (\{p\}, \Sigma, \Delta \cup \Sigma, \xrightarrow{M}, p, S)$$

donde S es el símbolo inicial de G . Los movimientos de M son, para cada producción $V \xrightarrow{G} w$, $(p, \epsilon, V) \xrightarrow{M} (p, w)$. Además, para cada $a \in \Sigma$, $(p, a, a) \xrightarrow{M} (p, \epsilon)$.

Notamos primero que para cada $w \in (\Sigma \cup \Delta)^*$ existe un programa P_w de M , que consiste en aplicar movimientos $(p, a, a) \rightarrow (p, \epsilon)$ hasta borrar todo de $pre(w)$, de tal modo que

$$P_w(w) = suf(w).$$

Vamos a demostrar la siguiente proposición: Para cada $Z \in \Delta$ y cada $w \in (\Sigma \cup \Delta)^*$ tal que existe un programa izquierdista P tal que $P(Z) = w$,

$$(1) \quad (p, Z, w) \vdash^M (p, \epsilon, suf(w))$$

Esto es suficiente para demostrar que $L(G) \subset L(M)$, porque entonces en el caso $w \in \Sigma^*$ y $Z = S$ tenemos $suf(w) = \epsilon$ así que $(p, S, w) \vdash^M (p, \epsilon, \epsilon)$.

La idea de la demostración de (1) es que cada vez que tenemos un miembro de Δ arriba en la pila se le aplica el movimiento de M que corresponde a la correspondiente producción de P (recuerda que P es izquierdista), y si esto produce una pila con símbolos de Σ a la izquierda tienen que corresponder a un segmento inicial de lo que queda de la cadena de entrada; entonces se van comparando y borrando estos símbolos con los movimientos $(p, a, a) \rightarrow (p, \epsilon)$.

Formalmente, se procede por inducción sobre la longitud del programa P .

Si $|P| = 0$ entonces el programa de M deseado es P_w .

Suponga que la proposición se cumple para todo programa de G de longitud n , y sea P de longitud $n+1$, es decir $P = P'\pi$ donde P' es un programa de G de longitud n y π es una producción $V \rightarrow u$, y suponga que para algún $Z \in \Delta$ y $w \in (\Delta \cup \Sigma)^*$, $P(Z) = w$. Entonces $P'(Z) = u'Vw'$ donde $u' \in \Sigma^*$ (porque P representa una derivación izquierdista) y $w = u'uw'$. Por la hipótesis de inducción, existe un programa Q de M tal que $Q(p, Z, w) = suf(u'Vw') = Vw'$. Por la producción π , M tiene el movimiento $m = (p, \epsilon, V) \rightarrow (p, u)$ así que $(Qm)(w) = uw'$. Es claro que $suf(w) = suf(u'uw') = suf(uw')$. Pero $P_{uw'} = suf(uw')$ y

concluimos que $QmP_{suf(u'w)} = suf(w)$, completando la inducción, demostrando la proposición (1) y por tanto que $L(G) \subset L(M)$.

Recíprocamente, vamos a demostrar para todo $c \in \Sigma^*$, $w \in \Sigma \cup \Delta$ y $V \in \Delta$, que si $(p, c, V) \vdash^M (p, \epsilon, w)$, entonces $V \vdash^G cw$. Procedemos por inducción sobre el largo del programa P de M tal que $P(p, c, V) = (p, \epsilon, w)$. Si $|P| = 1$ entonces $P = (p, \epsilon, v) \xrightarrow{M} (p, w)$, entonces $c = \epsilon$ y $V \xrightarrow{G} w$: se cumple la aserción cuando $|P| = 1$. Suponga ahora que para algún $n \geq 1$ se cumple para todos los casos en que $|P| = n$, y consideremos un caso en que $|P| = n + 1$. Entonces $P = P'm$ donde m es un movimiento de M . Sea $c = c'a$ con $a = \epsilon$ si $m = (p, \epsilon, Z) \xrightarrow{M} (p, v)$ y $a \in \Sigma$ si $m = (p, a, a) \xrightarrow{M} (p, \epsilon, \epsilon)$. Sabemos en este último caso que $P'(p, c', V) = (p, \epsilon, aw)$. Por la hipótesis de inducción, $V \vdash^G c'aw$, pero $c'aw = cw$: la aserción se cumple para P . El otro caso es $m = (p, \epsilon, Z) \xrightarrow{M} (p, v)$. En este caso $P'(p, c, V) = Zu$ para algún $Z \in \Delta$ y $m(p, Zd) = (p, vd)$, así que $vd = w$. Por la hipótesis de inducción $V \xrightarrow{G} cZu$ y por lo tanto $V \xrightarrow{G} cvu = cw$. La inducción está completa.

El caso especial que $(p, c, S) \vdash^M (p, \epsilon, \epsilon)$, implicando que $S \vdash^G c$, muestra que $L(M) \subset L(G)$. \square

PROPOSICIÓN 5.12. *Dado un autómata de pila M , existe una gramática libre de contexto G tal que $L(G) = L(M)$*

Demostración. Podemos suponer sin perder generalidad que $\Delta_M \cap \Sigma$ es vacío (por ejemplo, reemplazando cada $a \in \Sigma$ que ocurre con carácter en la pila, con \hat{a} .) Por el teorema 5.9, podemos suponer además que M tiene un solo estado p .

Conviene cortar la notación: Ya que hay un solo estado, mención de el es superfluo así que en lugar de $(p, a, Z) \rightarrow (p, w)$ escribimos $(a, Z) \rightarrow w$ y en lugar de $(p, \epsilon, Z) \rightarrow (p, w)$ escribimos $Z \rightarrow w$.

La gramática que engendra el lenguaje de M tiene las siguiente producciones:

- Para cada movimiento $(a, Z) \rightarrow w$ con $a \in \Sigma$, la producción $Z \rightarrow aw$.
- Para cada movimiento $Z \rightarrow w$, la producción $Z \rightarrow w$.

Con esta estrecha relación entre producciones y movimientos (visto más claro con esta notación), se establece una estrecha relación entre computaciones del autómata y derivaciones izquierdistas de la gramática:

1. Si hay una computación del autómata que empieza con solo S en la pila, en que se ha leído c y se ha dado la configuración (p, d, w) entonces hay una derivación izquierdista empezando con S en que la cadena final es cw , donde $\text{pre}(c, w) = c$.
2. Recíprocamente, si hay una derivación izquierdista empezando con S con cadena final cw con $\text{pre}(cw) = c$ entonces hay una computación empezando con solo S en la pila en que se ha leído c y la configuración es (p, d, w) para algún d .

Demostramos (1) por inducción sobre el largo de la computación (o sea del programa que la produce). Si eso es nulo entonces no ha leído nada, y la derivación correspondiente es solo " S ", que es una derivación izquierdista con cadena final S . Supongamos que (1) se cumple para cualquier computación de largo n . Sea m un movimiento y P un programa de largo n . Preguntamos si el programa Pm cumple (1). $P(S)$ produce lee c y la configuración (p, d, Zw) . Si el último movimiento m es $(a, Z) \rightarrow v$ con $a \in \Sigma$, produce la configuración (p, d', vw) y entonces Pm ha leído ca . Por la hipótesis de inducción hay una derivación izquierdista con cadena final cZw . Por el movimiento m la gramática tiene una producción $Z \rightarrow av$. Aplicando esta producción tenemos una derivación izquierdista (porque $\text{pre}(cZw) = c$) con cadena final $cavw$, y $\text{pre}(cavw) = ca$, que es lo que Pm ha leído, y vw es lo que tiene el autómata en su pila. Si m es $Z \rightarrow v$ entonces la gramática tiene la producción $Z \rightarrow v$, que si aplico a la cadena final cZw llego a la cadena final cvw , y se cumple (1).

Demostremos (2) por inducción sobre el número de pasos en la derivación izquierdista. Y las consideraciones son casi iguales como para el (1) ya que hay una producción $Z \rightarrow av$ con $a \in \Sigma$ si y solo si hay un movimiento $(a, Z) \rightarrow v$ en el autómata, y hay una producción de la gramática $Z \rightarrow v$ con $v \in \Delta^*$ si y solo si hay un movimiento $Z \rightarrow v$ en el autómata, y las dos hacen casi la misma cosa. \square .

Las Proposiciones 5.11 y 5.12 nos dan

TEOREMA 5.13. *Los lenguajes de los autómatas de pila son los lenguajes libre de contexto.*

Observemos que las autómatas de pila no deterministas, a diferencia de los autómatas finitos, son más poderosos que los deterministas. Uno se convence rápidamente de ésto considerando el lenguaje $\{0^n 1^n 0^k : n, k > 0\} \cup \{0^n 1^k 0^k : n, k > 0\}$ (aunque la demostración formal es bastante engorrosa).

Frecuentemente, los autómatas de pila permiten simplificar un problema, por ejemplo, en demostrar que la intersección de un conjunto regular con un lenguaje libre de contexto es libre de contexto (ver ejercicio 5), o en encontrar una gramática no ambigua para el lenguaje sobre $\{0, 1\}$ en que 0 ocurre el mismo número de veces que 1 (ver ejercicio 3).

Otro ejemplo de la utilidad de los autómatas de pila es en el siguiente problema:

Problema. Construir un algoritmo para determinar, dadas dos gramáticas libre de contexto, si la intersección de los dos lenguajes es o no vacía. (Antes de intentarlo, lea lo que sigue.)

Vamos a ver que si uno construye tal algoritmo, lo puede convertir en un algoritmo para decidir el problema de la parada de una máquina de Turing. Sea $M = (K, \Sigma, \Delta, \beta, \delta, p_0, p_f)$ una máquina de Turing y sea c una entrada para M (es decir, $(\beta^\infty c \beta^\infty, p_0)$ es una configuración inicial). Podemos suponer que $K \cap \Delta$ es vacía. Vamos a representar una configuración de M por una cadena en $\Sigma^* K \Sigma^*$: cqd representa la configuración $\beta^\infty c d \beta^\infty$ con el cabezal posicionado inmediatamente a la derecha de la subcadena c , viendo el primer caracter de d si hay, y un β si $d = \epsilon$. Sea $z \notin K \cup \Delta$. Sea $L_1 = \{wzv : w, v \in \Delta^* K \Delta^*, w \xrightarrow{M} v^R\}$. Bosquejamos un autómata A de pila que acepta L_1 . Primero A convierte w en v^R en la pila, donde $w \xrightarrow{M} v$, después verifica que lo que sigue al z es v (repetiendo $(q, a, a) \rightarrow (q, \epsilon)$). La transformación a v^R se hace así: El símbolo de pila inicial de A es z . Se pasa todas las letras a la pila hasta encontrar $q \in K$. Sea a la letra que sigue a q . Si $a \neq z$ y $\delta(q, a) = (r, b, D)$ o $a = z$ y $\delta(q, \beta) = (r, b, D)$, entonces pasa br a la pila. Si $a \neq z$ y $\delta(q, a) = (r, b, I)$ o $r = z$ y $\delta(q, \beta) = (r, b, I)$, si c es la última letra de la pila, si $c = z$ se agrega $r\beta b$ a la pila; y si $c \neq z$, se reemplaza c en la pila con rcb . Se copia las demás letras hasta el final. Se ve que L_1 es aceptable por un autómata de pila, así que es libre de contexto. Sea $L_2 = \{w_1 z v_1 z w_2 z v_2 z \dots w_{k-1} z v_{k-1} z w_k : k > 0, w_i z v_i \in L_1^*\}$, o sea $L_2 = (L_1 z)^*(\Delta \cup K)^*$. Y sea $L_3 = \{w_1 z v_1 z w_2 z \dots v_{k-1} z w_k z v_k z : \text{cada } w_i \in (\Delta \cup K)^*, w_{i+1} = v_i^R \text{ para cada } i\}$. En $L_2 \cap L_3$ vemos $w_i \rightarrow_M v_i^R$ para cada i , y $w_{i+1} = v_i^R$ para cada i . Entonces $w_i \rightarrow w_{i+1}$ para cada i . Es decir, en esencia vemos una computación de M . Si además limitamos L_2 a tener w_1 la cadena que representa una configuración inicial con entrada c , y que w_k represente una parada de M , que es la intersección de L_2 con un conjunto regular, que llamamos L_4 , $L_4 \cap L_3$ no es vacío si y sólo si M para dada la entrada c . Para contestar el problema de la parada para esta M y c entonces, formamos los L_3 y L_4 correspondientes y aplicamos nuestro algoritmo para decidir si su intersección es vacía, si tenemos uno.

Ejercicios

- (3) Una calculadora tiene la siguiente sintaxis: Después de teclear un número, se teclea \leftarrow . Un número puede ser negativo. Las operaciones vienen después de los operandos, incluida la $'-'$ unitaria: *operando₁ operando₂ op*. Cualquier de los dos operandos puede ser compuesto a su vez por dos operandos y una operación, y así sin límite de profundidad. Por ejemplo, $3 \leftarrow 2 \leftarrow +$ se evalúa a 5, $3 \leftarrow 2 \leftarrow 4 \leftarrow \times +$ se evalúa a 11 ($3 + (2 \times 4)$). $3- \leftarrow 2 \leftarrow 4 \leftarrow \times +$ se evalúa a 5 ($-3 + (2 \times 4)$). En general, el binario $- \leftarrow$ quiere

decir “cambie el signo de la última expresión”. Entonces $7 \leftarrow 3 \leftarrow + - \leftarrow 14 \leftarrow +$ evalúa a 4 $(-(7 + 3) + 14)$.

Haga una gramática para describir este lenguaje, teniendo $+$, $-$, \times , \div y el $-$ unitario.

2. (4) Haga una función evaluadora para el lenguaje del ej. 1. (División da la parte entera).
3. (2) Haga una gramática para las expresiones regulares. La precedencia de operadores es $*$, concatenación, unión.
4. (2) Haga un automata de pila equivalente a la gramática

$$S \rightarrow \epsilon | 0S1 | 1S0 | SS$$

(cuyo language sobre $\{0, 1\}$ es las cadenas en que 1 y 0 aparecen igual número de veces, incluido 0.) Construir un autómata de pila equivalente a la gramática siguiendo a la letra la especificación de la demostración de la proposición 5.11. Demostrar que la gramática es ambigua. ¿Es ambiguo el autómata?

5. (4) El automata de pila usado en el texto para reconocer el lenguaje del ejercicio anterior, el con movimientos (se usa la abreviatura usado en la demostración de la proposición 5.12 y el reemplazo de 0,1 en la pila con $\hat{0}$, $\hat{1}$ respectivamente):

$$\begin{aligned} (0, S) &\rightarrow \hat{0}S, (1, S) \rightarrow \hat{1}S, S \rightarrow \epsilon \\ (0, \hat{0}) &\rightarrow \hat{0}\hat{0}, (1, \hat{1}) \rightarrow \hat{1}\hat{1}, (1, \hat{0}) \rightarrow \epsilon, (0, \hat{1}) \rightarrow \epsilon \end{aligned}$$

Haga una gramática equivalente siguiendo a la letra la construcción de la gramática en la demostración de la proposición 5.12. Demostrar que el automata es no ambiguo. ¿Es la gramática no ambigua?

6. (2) Modifique la gramática de la calculadora para incluir el $-$ unario. Cuidado: $-2 \wedge 10$ es $-(2^{10})$ y no $(-2)^{10}$.
7. (3) Modifica la función evaluadora según esa gramática.
8. (2) Demuestre que la intersección de un lenguaje libre de contexto y un conjunto regular es libre de contexto. Sugerencia: combine un autómata de pila con un autómata finito.
9. (2) Haga un autómata de pila que acepta

$$\{1^n + 1^m = 1^p : p = n + m\} \cup \{1^n - 1^m = 1^p : p = n - m\}$$

Ejemplos: $11111 + 111 = 11111111$, $11111 - 111 = 11$. Por supuesto, si la operación es $'-'$ y $m > n$ no se acepta la cadena.

10. (5) Hay lenguajes libre de contexto que son *inherentmente ambiguos*, es decir, que toda gramática libre de contexto que engendra el lenguaje es ambigua. Demuestre que un lenguaje es engendrado por alguna gramática no ambigua si y sólo si es aceptado por algún autómata de pila no ambiguo. Sugerencia: muestre que en la demostración del último teorema, el autómata derivado de la gramática es no ambiguo si la gramática lo es, y vice versa.

FUNCIONES GENERADORAS Y ECUACIONES DE DIFERENCIAS

1. Introducción

Existe un artificio que se llama *función generadora* para atacar a sucesiones definidas inductivamente. Decimos “artificio” porque no siempre hay justificación teórica por lo que se hace, sino sólo argumento de plausibilidad, pero no nos importa mucho éso, porque al final podemos demostrar el resultado por inducción.

Sea $a = a_0, a_1, a_2, \dots$ una sucesión infinita. Damos la siguiente definición sabiendo que su utilidad puede no ser aparente al principio:

DEFINICIÓN 6.1. La *función generadora* de la sucesión a es

$$a_0 + a_1z + a_2z^2 + a_3z^3 + \dots = \sum_{n \geq 0} a_n z^n.$$

Usamos la variable z porque tenemos que estar preparados para encontrar números complejos aquí. No hace falta resucitar sus conocimientos de reglas de convergencia y cosas así porque nos va a importar poco o nada la convergencia. Metamos un ejemplo ya:

Sea f la sucesión de Fibonacci, la gran abuela de todas las sucesiones inductivas, definida por $f_0 = 0$, $f_1 = 1$, y para $n > 1$, $f_n = f_{n-1} + f_{n-2}$. Los primeros de la sucesión entonces son 0, 1, 1, 2, 3, 5, 8, 13, 21. Sea $F = f_0 + f_1z + f_2z^2 + \dots$ su función generadora. Ahora, zF corre todos los términos un lugar a la derecha, reemplazando f_0 con 0, y z^2F los corre dos posiciones a la derecha, llenando los primeros dos lugares, dejados vacíos por la corrida, con 0. Ahora examinemos $r = F - zF - z^2F$:

F	0	z	z^2	$2z^3$	$3z^4$	$5z^5$	$8z^6$	$13z^7$. . .
$-zF$	0	0	$-z^2$	$-z^3$	$-2z^4$	$-3z^5$	$-5z^6$	$-8z^7$. . .
$-z^2F$	0	0	0	$-z^3$	$-z^4$	$-2z^5$	$-3z^6$	$-5z^7$. . .
r	0	z	0	0	0	0	0	0	. . .

y vemos que la diferencia es z :

$$F - zF - z^2F = z$$

o sea $(1 - z - z^2)F = z$, así que (plausiblemente) podemos concluir que

$$F = \frac{z}{1 - z - z^2}.$$

Ahora, hay una serie para ese cociente de polinomios. Si lo podemos sacar, igualamos sus términos con los términos f_i de F y tendremos una solución para la sucesión de Fibonacci. Resulta que $1 - z - z^2 = (1 - \varphi z)(1 - \widehat{\varphi} z)$ donde φ y $\widehat{\varphi}$ son las raíces del famoso polinomio $z^2 - z - 1$, ($\frac{1 \pm \sqrt{5}}{2}$: chequee que $\varphi + \widehat{\varphi} = \varphi\widehat{\varphi} = -1$). Entonces debemos poder encontrar A y B tales que

$$\frac{A}{1 - \varphi z} + \frac{B}{1 - \widehat{\varphi} z} = \frac{z}{1 - z - z^2}.$$

Una vez encontrados A y B , usamos la identidad

$$\sum_{n \geq 0} z^n = \frac{1}{1 - z}$$

(que si uno olvida se saca diciendo que si la suma converge a un valor S , tendría que ser que $S = 1 + zS$, pero si va a trabajar mucho con funciones generadoras mejor que memorice unas cuantas identidades básicas como ésa), y metiendo φz en uno y $\widehat{\varphi} z$ en el otro, en el lugar de z , tenemos

$$F = A \sum_{n \geq 0} \varphi^n z^n + B \sum_{n \geq 0} \widehat{\varphi}^n z^n$$

y para que esto sea igual a $F = \sum_{n \geq 0} f_n z^n$, tiene que ser que $f_n = A\varphi^n + B\widehat{\varphi}^n$.

Calculamos A y B notando que es necesario que $A(1 - \widehat{\varphi}z) + B(1 - \varphi z)$ tiene que ser igual como polinomio al polinomio z , que nos da las dos ecuaciones $A + B = 0$ y $-A\widehat{\varphi} - B\varphi = -1$, que dan primero $A(\varphi - \widehat{\varphi}) = 1$, que da $A = -1/\sqrt{5}$ y $B = 1/\sqrt{5}$. (Chequear con Maple haciendo `seq(A * \varphi^n + B * \widehat{\varphi}^n, n=0..30)`, habiendo puesto valores apropiados en los parámetros.)

Nuestras aplicaciones aquí son sistemas de dos o más ecuaciones de diferencia, como en el problema de los autos de alquiler, en que un auto retirado en Salta puede ser devuelto o en Salta o en Tucumán: Hay 700 autos entre los dos, y cada día se alquilan todos. Cada día 60 % de los autos en Salta se quedan en Salta y 40 % van a Tucumán, y cada día 70 % de los autos en Tucumán se quedan en Tucumán y 30 % van a Salta. Siendo S_n, T_n el número de autos en Salta y Tucumán respectivamente al principio del día n , se cumple el sistema de dos ecuaciones:

$$\begin{aligned} S_{n+1} &= 0,6S_n + 0,3T_n \\ T_{n+1} &= 0,4S_n + 0,7T_n. \end{aligned}$$

Podemos calcular cuantos términos deseamos de las sucesiones S y T , dados S_0 y T_0 , por ejemplo con el siguiente programa de Maple:

```
for i from 2 to nops(S) do Si = 0,6Si-1 + 0,3 Ti-1: Ti = 0,4Si-1 + 0,7 Ti-1 :
```

pero podemos tener fórmulas exactas para los S_n, T_n que suministran más información. Escribimos una tabla de las funciones generadoras como hicimos para la sucesión Fibonacci:

S	500	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
T	200	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
zS	0	500	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
zT	0	200	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}

Las ecuaciones, junto con la especificación de $S_0 = 500$ y $T_0 = 200$ dicen ni más ni menos que

$$\begin{aligned} S &= 500 + 0,6zS + 0,3zT \\ T &= 200 + 0,4zS + 0,7zT \end{aligned}$$

De aquí sacamos primero $T(1 - 0,7z) = 200 + 0,4S$, o sea $T = \frac{200+0,4S}{1-0,7z}$, y enchufando ésto para T en la primera ecuación nos da una ecuación en S y z solo, y el hecho de que se puede sumar, restar, multiplicar y dividir funciones racionales sin salir de ellas indica que vamos a poder, si tenemos mucho cuidado, resolver éso para S como función racional. Pero no demanda tanto cuidado entregar todo el problema a Maple, diciendo

```
solve({S = 500 + 0,6 * z * S + 0,3 * z * T, T = 200 + 0,4 * z * S + 0,7 * z * T}, {S, T})
```

y Maple contesta (redactado un poco para hacerlo más presentable)

$$S = -100 \frac{29z - 50}{3z^2 - 13z + 10}, \quad T = 400 \frac{2z + 5}{3z^2 - 13z + 10}.$$

Si podemos expandir éstos en series, tendríamos los términos de la función generadora, y por tanto la sucesión día por día del número de autos en Salta y Tucumán. Las raíces del denominador de S son $10/3$ y 1 (por la fórmula cuadrática si quiere, pero confieso que las pedí de Maple con `solve(denom(S))`). Entonces, el denominador de S es $3(z - 10/3)(z - 1)$. Dividiendo arriba y abajo por 3,

$$S = \frac{-\frac{100}{3}(29z - 50)}{(z - 10/3)(z - 1)}.$$

Ahora escribimos el denominador como $(10/3 - z)((1 - z))$ sin cambiar S y multiplicamos arriba y abajo por $3/10$, y sale

$$S = \frac{-10(29z - 50)}{(1 - \frac{3}{10}z)(1 - z)}.$$

Ahora busquemos A, B tales que

$$\frac{A}{1 - \frac{3}{10}z} + \frac{B}{1 - z} = S$$

lo que es equivalente a

$$A(1 - z) + B(1 - \frac{3}{10}z) = -10(29z - 50).$$

Igualando coeficientes de z^n ($n = 0, 1$) en los dos lados, da dos ecuaciones con las dos incógnitas. Estas son fáciles de resolver a mano, pero por su utilidad en casos más engorrosos preguntamos a Maple con

`solve({seq(coeff(A(1-z)+B(1-3/10*z), z, i) = coeff(-10(29z-50), z, i), i = 0..1)}, {A, B}))`.

El formato de solve en este caso es `solve({ecuaciones}, {incognitas})`, los dos argumentos son conjuntos, y formamos el conjunto de ecuaciones con `{seq()}`. Maple contesta $A = 200, B = 300$.

Ahora aplicamos la identidad

$$\frac{1}{1 - x} = \sum_{n \geq 0} x^n$$

que mencionamos antes. Así llegamos a escribir $S = \sum_{n \geq 0} 200(\frac{3}{10}z)^n + \sum_{n \geq 0} 300z^n = \sum_{n \geq 0} (200(\frac{3}{10})^n + 300)z^n$.

Así que el número de autos en Salta en el día n es $200(\frac{3}{10})^n + 300$. Y el de Tucumán es $700 - (200(\frac{3}{10})^n + 300) = 400 - 200(\frac{3}{10})^n$. Lo dejamos al lector demostrar esto por inducción matemática. (Se cumple para $n = 0$, queda demostrar, mediante el sistema de ecuaciones, que cuando se cumple para algún n entonces se cumple para $n + 1$. Confieso que calculé numéricamente los primeros 30 días usando las ecuaciones, y observando que estaban de acuerdo dentro de 10^{-6} con el resultado teórico, me sentí satisfecho.)

2. Funciones generadoras racionales

En todos los ejemplos que vamos a ver aquí, hay una función generadora F que es una función racional: $F = \frac{p}{q}$ donde p, q son polinomios. Sea c, r el cociente y resto de p dividido q , es decir, $p = cq + r$ donde r es o 0 o de menor grado que q . Entonces $F = c + \frac{r}{q}$. Sean s_1, \dots, s_k las raíces de q , posiblemente algunas repetidas, así que $q = a(z - s_1) \cdots (z - s_k)$ para algún número a . Escribimos

$$q = a(-1)^k(s_1 - z) \cdots (s_k - z) = (-1)^k s_1 \cdots s_k (1 - \frac{1}{s_1}z) \cdots (1 - \frac{1}{s_k}z)$$

y ahora podemos escribir, poniendo $t_i = \frac{1}{s_i}$

$$F = c + \frac{t_1 \cdots t_k (-1)^k r/a}{(1 - t_1 z) \cdots (1 - t_k z)}.$$

Suponga primero que todas las raíces son distintas. Entonces buscamos A_i, \dots, A_k tal que

$$\frac{A_1}{1 - t_1 z} + \cdots + \frac{A_k}{1 - t_k z} = \frac{(-1)^k (t_1 \cdots t_k) r/a}{(1 - t_1 z) \cdots (1 - t_k z)},$$

que es equivalente a

$$\sum_{i=1}^k A_i \prod_{j \neq i} (1 - t_j z) = (-1)^k (t_1 \cdots t_k) r/a.$$

Igualando coeficientes de z^n en los dos lados, da k ecuaciones en las k incógnitas. Pues $\frac{1}{1-t_j z} = \sum_{n \geq 0} t_j^n z^n$, esto nos da una sucesión cuyo n -ésimo término es

$$\sum_{i=1}^k A_i t_i^n.$$

Si $c = 0$, esto nos da la solución. Si $c = \sum_{i=1}^m c_i z^i$ con $c_m \neq 0$, entonces el n -ésimo término es

$$c_n + \sum_{i=1}^k A_i t_i^n \quad \text{si } n \leq m,$$

$$\sum_{i=1}^k A_i t_i^n \quad \text{si } n > m.$$

Si no todas las raíces son distintas, sean r_1, \dots, r_k las raíces de q , y para cada i sea m_i la multiplicidad de r_i , así que

$$q = a \prod_{i=1}^k (z - r_i)^{m_i}$$

para algún a . Entonces, con c, r el cociente y resto de p dividido q como antes, vamos a tener, usando $g = \sum_i m_i$ para el grado de q ,

$$F = c + \frac{\frac{1}{a}(-1)^g \prod_{i=1}^k r_i^{m_i} r}{\prod (1 - \frac{1}{r_i})^{m_i}}.$$

Ahora busquemos $\{A_{ij} : 1 \leq i \leq k, 0 \leq j \leq m_i - 1\}$ tal que, usando $t_i = \frac{1}{r_i}$ como antes,

$$\sum_{i=1}^k \frac{\sum_{j=0}^{m_i-1} A_{ij} z^j}{(1 - t_i z)^{m_i}} = \frac{\frac{1}{a}(-1)^g \prod r_i^{m_i} r}{\prod (1 - \frac{1}{r_i})^{m_i}},$$

que, como antes, equivale a

$$\sum_{i=1}^k \sum_{j=0}^{m_i-1} A_{ij} z^j \prod_{h \neq i} (1 - t_h z)^{m_h} = \frac{1}{a}(-1)^g \prod r_i^{m_i} r.$$

Ahora usamos la identidad $\frac{1}{(1-x)^c} = \sum_{n \geq 0} \binom{c+n-1}{n}$ para decir

$$\frac{1}{(1 - t_i z)^{m_i}} = \sum_{n \geq 0} \binom{m_i + n - 1}{n} (t_i z)^n.$$

Ahora, $\binom{n}{k} = \binom{n}{n-k}$, porque el número de subconjuntos de orden k elegidos de uno de orden n es igual que el número de subconjuntos de orden $n - k$ en ese conjunto. Por eso, $\binom{m_i+n-1}{n} = \binom{m_i+n-1}{m_i-1} = \frac{(m_i+n-1)(m_i+n-2) \cdots n+1}{(m_i-1)!}$ (porque $\binom{n}{k} = \frac{n(n-1)(n-2) \cdots (n-k+1)}{k!}$). Lo que importa aquí es que $\binom{m_i+n-1}{m_i-1}$ es un polinomio en n de grado $m_i - 1$, digamos $\sum_{k=0}^{m_i-1} c_{ik} n^k$, donde c_{ik} depende sólo de i y k , independiente de cualquier otra cosa, y por eso

$$\frac{1}{(1 - t_i z)^{m_i}} = \sum_{n \geq 0} \left(\sum_{k=0}^{m_i-1} c_{ik} n^k \right) (t_i z)^n.$$

Entonces

$$F = \sum_{n \geq 0} \sum_{j=0}^{m_i-1} A_{ij} z^j \left(\sum_{i=0}^{m_i-1} a_i n^i \right) (t_i z)^n.$$

2.1. Sistemas de funciones racionales. Sean T_1, \dots, T_k funciones generadoras, y para cada i , $1 \leq i \leq k$, sea

$$T_i = \sum_{j=1}^k a_{ij} T_j,$$

donde cada a_{ij} es una función racional en una variable z . Se puede eliminar cualquier T_i del sistema, expresándola en términos de las demás y sustituyendo esa expresión para cada ocurrencia de T_i en las expresiones para las demás. En el caso en que $a_{ii} = 0$, ya tenemos T_i expresada en términos de las demás y procedemos a la sustitución; si no, escribimos $T_i(1 - a_{ii}) = \sum_{j \neq i} a_{ij} T_j$, o sea

$$T_i = \frac{\sum_{j \neq i} a_{ij} T_j}{1 - a_{ii}},$$

y de nuevo tenemos T_i expresada en términos de las demás, con coeficientes racionales dado que esas funciones pueden multiplicarse, dividirse, sumarse y restarse sin salir de ellas.

Así reducimos el sistema de k ecuaciones a uno de $k - 1$, mientras $k > 1$. Es claro que podemos seguir reduciendo así hasta quedar con una sola función generadora expresada como una función racional.

No vamos a ver ejemplos de estos todavía porque usualmente su solución se encuentra más directamente con los métodos que consideramos ahora.

3. Ecuaciones de diferencia

Un caso especial de sistemas de funciones generadoras es un *sistema de ecuaciones de diferencia*, como las ecuaciones para la empresa de autos que vimos al principio, que definimos formalmente ahora. Sea x_1, \dots, x_k k sucesiones numéricas, $x_i = x_{i0}, x_{i1}, x_{i2}, \dots$. La idea es que cada x_{in} se calcula en términos de los m anteriores de cada x_j , es decir, para cada i, n

$$x_{in} = \sum_{j=1}^k \sum_{p=1}^m a_{ijp} x_{j(n-p)}.$$

(Probablemente la mayoría de los a_{ijp} sean 0.)

Ahora vemos cómo aplicar funciones generadoras a éstas. Convertimos el sistema de ecuaciones de diferencia primero en

$$F'_i = \sum_{a_{ijp} \neq 0} a_{ijp} F_j z^p$$

para cada i . Ahora F'_i es casi la función generadora para la sucesión x_i . Para ver qué pasa aquí, hay que imaginar la matriz que resulta si para cada a_{ijp} que es distinto de 0 corremos F_j p lugares a la derecha, así multiplicándola por z^p : Si $n \geq m$, la posición $n - p$ de F_j queda debajo de la posición n de F'_i , y al restar $\sum a_{ijp} F_j z^p$ de F'_i , una de las infinitas operaciones que se hace es restar $\sum a_{ijp} F_j z^p$ de F_{in} , y para las columnas n con $n \geq m$ esto da 0 porque en esas columnas es exactamente la ecuación de diferencia para x_{in} . En las columnas n con $n < m$, se supone que nuestro conocimiento del problema nos permite calcular cada x_{in} . Éstos son los valores iniciales. Ahora en estas columnas tenemos $x_{in} - \sum_{j=1}^k \sum_{p=0}^{m-1} a_{ijp} F_j z^p$ y llamamos a esta diferencia d_{in} . Ahora podemos decir que $F_i = F'_i + \sum_{p=0}^{m-1} d_{ip}$. Con esto tenemos un sistema de k ecuaciones vinculando las k funciones generadoras.

Por tanto, un problema que se expresa como un sistema de ecuaciones lineales de diferencias, se puede reducir a un sistema de funciones generadoras y que podemos resolver para cualquier de las funciones generadoras como función racional, y esto a una sola ecuación lineal de diferencias. Esto es útil porque hay métodos más directos y menos engorrosos para resolver éstas, que consideramos después.

Primero veamos un ejemplo. Consideramos una empresa con tres agencias de autos en tres ciudades, tal que cada día la fracción p_{ij} de los autos en la ciudad i se devuelven en la ciudad j , con $p_{i1} + p_{i2} + p_{i3} = 1$ para todo i . Un auto que va de la ciudad 1 a la ciudad 3 demora 2

días, igual que uno que va de la ciudad 3 a la ciudad 1. Otra complicación es que en la ciudad 2, 25 de los autos están en el taller y no van a estar disponibles hasta el próximo día, día 1. Hay solamente 75 autos disponibles en la ciudad 2 en el día 0, y 100 cada una en las otras dos. El cálculo numérico de esto, que deja en $c_i(n)$ el número de autos en la ciudad i al principio del día n , va así:

Inicialización: $c_1(0) := 100, c_3(0) := 100, c_2(0) := 75, c_1(1) := p_{11}c_1(0) + p_{21}c_2(0), c_3(1) := p_{13}c_3(0) + p_{33}c_3(0), c_2(1) := p_{12}c_1(0) + p_{22}c_2(0) + p_{32}c_3(0) + 25$.

Recursión: for i from 2 to 30 do

$$\begin{aligned} c_1(n) &:= p_{11}c_1(n-1) + p_{21}c_2(n-1) + p_{31}c_3(n-2) \\ c_2(n) &:= p_{12}c_1(n-1) + p_{22}c_2(n-1) + p_{32}c_3(n-1) \\ c_3(n) &:= p_{13}c_1(n-2) + p_{23}c_2(n-1) + p_{33}c_3(n-1) \end{aligned}$$

El correspondiente sistema de ecuaciones para las funciones generadoras F_1, F_2, F_3 es

$$\begin{aligned} F_1 &= p_{11}zF_1 + p_{21}zF_2 + p_{31}z^2F_3 + c_1(0) \\ F_2 &= p_{12}zF_1 + p_{22}zF_2 + p_{32}zF_3 + c_2(0) + 25z \\ F_3 &= p_{13}z^2F_1 + p_{23}zF_2 + p_{33}zF_3 + c_3(0) \end{aligned}$$

Por ejemplo, la segunda ecuación se verifica así:

potencia de z	z^0	z^1	z^2	z^3	...
F_1	$c_1(0)$	$c_1(1)$	$c_1(2)$	$c_1(3)$...
F_2	$c_2(0)$	$c_2(1)$	$c_2(2)$	$c_2(3)$...
F_3	$c_3(0)$	$c_3(1)$	$c_3(2)$	$c_3(3)$...
$p_{12}zF_1$	0	$p_{12}c_1(0)$	$p_{12}c_1(1)$	$p_{12}c_1(2)$...
$p_{22}zF_2$	0	$p_{22}c_2(0)$	$p_{22}c_2(1)$	$p_{22}c_2(2)$...
$p_{32}zF_3$	0	$p_{32}c_3(0)$	$p_{32}c_3(1)$	$p_{32}c_3(2)$...
$c_2(0)$	$c_2(0)$	0	0	0	...
$25z$	0	25	0	0	...
$\sum p_{i2}zF_i + c_2(0) + 25z$	$c_2(0)$	$\sum p_{i2}c_i(0) + 25$	$\sum p_{i2}c_i(1)$	$\sum p_{i2}c_i(2)$...

Para la primera ecuación necesitamos las filas para $p_{11}zF_1, p_{21}zF_2, p_{31}z^2F_3$, que es 0,0, $c_3(0), c_3(1), c_3(2), \dots$, y $c_1(0)$. Se verifica que la última fila de la columna de z^n coincide con el cálculo numérico de $c_i(n)$.

Con los p_{ij} elegidos al azar sujetos a la restricción $\sum_{j=1}^3 p_{ij} = 1$ para cada i (eligiendo cada p_{ij} de la distribución uniforme entre 0 y 1 y luego haciendo $p_{ij} := p_{ij} / \sum_{k=1}^3 p_{ik}$) salió, redondeado a tres dígitos,

$$p = \begin{bmatrix} .267 & .170 & .563 \\ .608 & .075 & .317 \\ .089 & .621 & .290 \end{bmatrix}.$$

Cuando aplicamos el algoritmo, definido arriba, para el cálculo numérico salen, redondeados a dos decimales, unos cuantos:

$c_1 = 100, 72.30, 94.91, 62.44, 78.54, 72.62, 80.44, 74.30, 76.72, 74.89, 76.76, 75.75, 76.27, 75.77, 76.13, \dots$
 $c_2 = 75, 109.72, 53.29, 86.19, 72.00, 84.83, 73.88, 79.37, 75.96, 79.13, 77.11, 78.18, 77.32, 77.98, 77.61, \dots$
 $c_3 = 100, 52.77, 106.38, 88.45, 106.41, 88.84, 96.88, 92.4, 97.25, 94.11, 95.57, 94.32, 95.35, 94.81, 95.15, \dots$

¿Ud. pregunta cómo puede haber 72,30 autos en la ciudad 1 el día 1? Podemos interpretarlo como una esperanza. Claro que el número de autos alquilados y el número de llevados a una ciudad dada en realidad van a ser números aleatorios, pero este modelo puede iluminar un poco la realidad sin capturarla completamente. Cuando estudiemos autómatas estocásticos veremos un modelo más realista.

Cuando damos a Maple las ecuaciones de las tres funciones generadoras con estos parámetros, después de una redacción para hacer volver a este mundo los coeficientes astronómicos dados inexplicablemente por Maple, y redondeando un poco, salen las soluciones

$$\begin{aligned} F1 &= \frac{70,53z^4 - 295,95z^3 + 312,21z^2 + 910,00z + 10000}{0,38z^5 - 26,75z^4 + 7,67z^3 - 18,10z^2 - 63,20z + 100} \\ F2 &= \frac{-125,27z^5 - 375,80z^4 + 3841,11z^3 - 2962,85z^2 + 6232,50z + 7500}{0,38z^5 - 26,75z^4 + 7,67z^3 - 18,10z^2 - 63,20z + 100} \\ F3 &= \frac{855,76z^4 + 1933,43z^3 + 5943,26z^2 - 1042,50z + 10000}{0,38z^5 - 26,75z^4 + 7,67z^3 - 18,10z^2 - 63,20z + 100} \end{aligned}$$

En la próxima sección vamos a ver una fórmula que predice con exactitud el resultado del cálculo numérico, resolviendo los problemas de una sola ecuación implicados por las expresiones de las F_i como funciones racionales.

3.1. Aplicación a la teoría de automatas. Hay aplicaciones en que uno quiere saber, dado un automata finito cuyo conjunto aceptado es R , cuántas cadenas de longitud n hay en R . Planteamos el siguiente sistema de ecuaciones de diferencia, donde $N_q(n)$ es el número de cadenas c de longitud n tales que $\delta(q, c)$ es un estado final:

$$\{N_q(n) = \sum_{a \in \Sigma} N_{\delta(q,a)}(n-1) : q \in K\} \cup \{N_q(0) = 0 : q \in K - F\} \cup \{N_q(0) = 1 : q \in F\}$$

Veremos unas aplicaciones de esto después.

4. Una sola ecuación de diferencia

En ésta sección vamos a ver técnicas especiales para sistemas con una sola ecuación de diferencia, y vamos a caracterizar más completamente la relación entre éstos y las correspondientes funciones generadoras.

Sea x una sucesión infinita tal que

$$\sum_{i=0}^d a_i x_{n-i} = 0 \quad (1)$$

para todo $n \geq d$, donde $d > 0$ es un número natural y los a_i son números que en general pueden ser complejos, con $a_0 \neq 0$. La ecuación dice en esencia que para $n \geq d$, cada x_n es siempre la misma combinación lineal de sus d predecesores en la sucesión x . Muchas veces se presenta en la forma

$$x_n = \sum_{i=1}^d b_i x_{n-i} = 0.$$

Pero para simplificar la notación, nos quedamos con la primera forma (1). Hay infinitas sucesiones que cumplen ésta, una para cada elección de x_0, \dots, x_{d-1} . De lo que vimos de funciones generadoras esperamos que ciertos $c\lambda^n$ figuran en la solución. De hecho, ciertos λ^n son soluciones:

$$\sum_{i=0}^d a_i \lambda^{n-i} = 0$$

si y sólo si $\lambda^{n-d}(\sum_{i=0}^d a_i \lambda^{d-i}) = 0$, es decir, λ es una raíz del *polinomio característico* de la ecuación de diferencia, $\sum_{i=0}^d a_i z^{d-i}$.

Ahora, una ecuación de diferencia es un caso especial del sistema de ecuaciones de diferencia que ya hemos estudiado y resuelto. Recordemos cómo: sea F la función generadora de la solución

y consideramos la serie

$$\sum_{i=0}^d a_i F z^i.$$

Para $n \geq d$ el coeficiente de z^n es $\sum_{i=0}^d a_i F_{n-i}$, porque el efecto de multiplicar f por z^i es correrla i lugares a la derecha. Pero F_{n-i} es ni más ni menos de x_{n-i} , así que esa serie es 0 desde el lugar d en adelante, de manera que podemos decir que

$$F \sum_{i=0}^d a_i z^i = \sum_{i=0}^{d-1} c_i z^i$$

para algunos números c_0, \dots, c_{d-1} , donde $c_0 = x_0$, $c_1 = x_0 + x_1$, y en general, $c_i = \sum_{j=0}^i x_j$. Entonces F es la función racional

$$F = \frac{\sum_{i=0}^{d-1} c_i z^i}{\sum_{i=0}^d a_i z^i}.$$

Ahora, notemos que $\sum_{i=0}^d a_i z^i$ es el *polinomio recíproco* del polinomio característico:

DEFINICIÓN 6.2. El *polinomio recíproco* de un polinomio $\sum_{i=0}^d b_i z^i$ es $\sum_{i=0}^d b_i z^{d-i}$.

Las raíces del recíproco de p son los respectivos recíprocos, o inversos multiplicativos, de las de p : si $\sum_{i=0}^d b_i r^i = 0$, entonces $\frac{1}{r^d} \sum_{i=0}^d b_i r^i = 0$, así que $\sum_{i=0}^d b_i r^{i-d} = 0$, o sea $\sum_{i=0}^d a_i \left(\frac{1}{r}\right)^{d-i} = 0$. Ya hemos visto que la solución de esta función racional tiene la forma

$$F = \sum_{n \geq 0} \sum_{j=0}^{m_i-1} A_{ij} z^j \left(\sum_{i=0}^{m_i-1} a_i n^i \right) (t_i z)^n$$

Luego, notemos que si s y t son soluciones a la ecuación de diferencia, también lo es cualquier combinación lineal $bs + ct$ de las dos, porque $(bs + ct)_n = bs_n + ct_n$ para todo n , así que

$$\sum_{i=0}^d a_i (bs + ct)_{n-i} = \sum_{i=0}^d a_i (bs_{n-i} + ct_{n-i}) = \sum_{i=0}^d a_i s_{n-i} + \sum_{i=0}^d a_i t_{n-i} = 0.$$

Por lo tanto, si el polinomio característico tiene d raíces distintas r_1, \dots, r_d , para cualquier elección de los números A_1, \dots, A_d la sucesión $\sum_{i=1}^d A_i r_i^n$ es una solución a la ecuación de diferencia. Cada elección de los A_i da valores para x_0, \dots, x_{d-1} , pero una, y sólo una, da los valores correctos, la solución del siguiente sistema de d ecuaciones con las d incógnitas A_1, \dots, A_d :

$$\left\{ \sum_{i=1}^d A_i r_i^n = x_j : j = 0, \dots, d-1 \right\}.$$

Por ejemplo, sea L_n el conjunto de cadenas binarias de largo n que no contienen 00. Entonces,

$$L_n = 1L_{n-1} \cup 01L_{n-2}$$

y, siendo éstos disjuntos,

$$|L_n| = |L_{n-1}| + |L_{n-2}|,$$

la misma ecuación de diferencia que la sucesión de Fibonacci, pero con valores iniciales $|L_0| = 1$ (la cadena vacía) y $|L_1| = 2$. El polinomio característico es $z^2 - z - 1$, con raíces φ y $\widehat{\varphi}$, como hemos visto. Entonces, la solución general es $A\varphi^n + B\widehat{\varphi}^n$. $|L_0| = 0$ y $|L_1| = 2$ dan las ecuaciones

$$A + B = 0; \quad A\varphi + B\widehat{\varphi} = 2$$

con la solución $A = \frac{2}{\sqrt{5}}$, $B = -\frac{2}{\sqrt{5}}$, así que $|L_n| = \frac{2}{\sqrt{5}}\varphi^n - \frac{2}{\sqrt{5}}\widehat{\varphi}^n$.

Quedan los casos en que la ecuación característica tiene algunas raíces múltiples. En los tratamientos sin funciones generadoras, la solución de esto va por demostrar que si $(z - r)^m$ es un factor del polinomio característico, entonces $n^k r^n$ es una solución de la ecuación de diferencia

para cada $k < m$, y entonces la solución del problema se va a encontrar entre las combinaciones lineales de tales términos. Esta demostración se ve en muchos textos de matemática discreta, por ejemplo en el de Levy y Lessman, citado en la bibliografía. Aquí lo vamos a demostrar mediante lo que hemos visto de funciones generadoras.

Ya hemos visto que, dadas raíces r_1, \dots, r_k del polinomio característico, es decir los recíprocos de las raíces del denominador, con multiplicidades m_1, \dots, m_s ,

$$F = \sum_{i=1}^k \frac{\sum_{j=0}^{m_i-1} A_{ij} z^j}{(1 - r_i z)^{m_i}}$$

y que aplicando una identidad se obtiene

$$\frac{1}{(1 - r_i z)^{m_i}} = \sum_{n \geq 0} \binom{m_i + n - 1}{m_i - 1} (r_i z)^n,$$

que nos da

$$F = \sum_{i=1}^k \sum_{j=0}^{m_i-1} A_{ij} z^j \sum_{n \geq 0} \binom{m_i + n - 1}{m_i - 1} r_i^n z^n.$$

Fijemos i, j un momento y estudiemos

$$A_{ij} z^j \sum_{n \geq 0} \binom{m_i + n - 1}{m_i - 1} r_i^n z^n = \frac{A_{ij}}{r_i^j} \sum_{n \geq j} \binom{m_i + n - j - 1}{m_i - 1} r_i^n z^n.$$

Ahora notemos que para $n < j$ es $m_i + n - j - 1 < m_i - 1$, así que $\binom{m_i + n - j - 1}{m_i - 1} = 0$ (¿cuántos subconjuntos de 4 elementos tiene un conjunto de 3 elementos por ejemplo?). Entonces la suma a la derecha de la última ecuación puede ser para $n \geq 0$ y tenemos

$$F = \sum_{i=1}^k \sum_{j=0}^{m_i-1} \frac{A_{ij}}{r_i^j} \sum_{n \geq 0} \binom{m_i + n - j - 1}{m_i - 1} r_i^n z^n.$$

Recordando que $\binom{n}{m} = \frac{n(n-1)\dots(n-m+1)}{m!}$, $\binom{m_i + n - j - 1}{m_i - 1}$ es un polinomio en n de grado a lo sumo m_i , y, por tanto, para cada $i \in \{1, \dots, k\}$,

$$\sum_{j=0}^{m_i-1} A_{ij} z^j \sum_{n \geq 0} \binom{m_i + n - 1}{m_i - 1} r_i^n z^n = \sum_{j=0}^{m_i-1} \sum_{n \geq 0} a_{ij} n^j r_i^n z^n.$$

Poniendo todo junto, concluimos que

$$F = \sum_{n \geq 0} \left(\sum_{i=1}^k \sum_{j=0}^{m_i-1} a_{ij} n^j r_i^n \right) z^n$$

donde los r_i son los inversos de las raíces del denominador de F , o, equivalentemente, las raíces del polinomio característico de la ecuación de diferencia. El grado de dicho polinomio es $g = \sum_{i=1}^k m_i$, y tenemos g incógnitas a_{ij} para resolver con las g ecuaciones

$$\left\{ \sum_{i=1}^k \sum_{j=0}^{m_i-1} a_{ij} n^j r_i^n = F_n : n = 0..g-1 \right\}.$$

¿Siempre existe una solución a estas ecuaciones? Eso depende de la independencia de las soluciones $\{n^j r^n\}$. Son independientes, es decir, no existe una combinación lineal de ellas igual a 0, pero la demostración general de esto está fuera de nuestro alcance aquí. Se puede demostrar en cada caso concreto evaluando el determinante de la matriz de la $n^j r^n$ para $n = 0, \dots, d$.

Otra laguna en nuestra teoría es la validez de todas las manipulaciones con funciones racionales. También es justificable formalmente, y después vamos a ver un bosquejo de eso. Pero

si nos guía a una solución, siempre la podemos verificar por inducción usando el sistema de ecuaciones de diferencias.

En resumen, para un sistema de ecuaciones lineales de diferencia, primero plantearlo como un sistema de ecuaciones entre las funciones generadoras, reducir éste a una sola ecuación para la sucesión que interesa más, plantear la correspondiente ecuación de diferencia, y resolverla con el polinomio característico.

Volvamos al problema de las tres sucursales de una empresa de alquilar autos, con la demora de dos días entre las ciudades 1 y 3. Se nota que cada F_i que vimos allí tiene el mismo denominador: algo sorprendente porque ahora entendemos que ésto significa que cada sucesión c_i cumple la misma ecuación de diferencia. Dado que el polinomio recíproco de ese denominador es $0,38 - 26,75z + 7,67z^2 - 18,10z^3 - 63,20z^4 + 100,00z^5$, la ecuación de diferencia en cada caso es, dividiendo por el coeficiente 100 de z^5 ,

$$c_i(n) + 0,632c_i(n-1) - 0,181c_i(n-2) + 0,0767c_i(n-3) - 0,2675c_i(n-4) + 0,038c_i(n-5) = 0 \text{ para } n \geq 5.$$

El lector puede verificar eso con Maple en el ejemplo que dimos. Poniendo las raíces del polinomio característico en una lista r y diciendo a Maple `solve({seq(add(A[i] * r[i]^n, i = 1..5) = c_j(n), n = 0..4)}, {seq(A[i], i = 1..5)})`, da para cada $j \in \{1, 2, 3\}$ los A_i , $i = 1, \dots, 5$, tales que esperamos

$$\text{add}(A_i * r_i^n, i = 1..5)$$

autos en la ciudad j al principio del día n . Se puede comprobar ésto comparándolo con la solución numérica. Estos cálculos se ven en el worksheet de Maple *3cars* (ver apéndice), donde además se resuelve $F2$, cuyo numerador tiene igual grado que su denominador, requiriendo uso del cociente y resto.

EJEMPLO 6.3. ¿Cuántas cadenas binarias hay de largo n que, interpretadas en binario, son múltiplos de 3? Debe ser aproximadamente $1/3$ de las cadenas, pero no puede ser exactamente $1/3$ porque hay 2^n cadenas de largo n y 2^n no es divisible por 3. Si definimos L_n como el conjunto de cadenas de largo n en este conjunto, tenemos $L_0 = \{\epsilon\}$ (la cadena vacía, que representa 0, que es divisible por 3), $L_1 = \{0\}$, $L_2 = \{00, 11\}$, $L_3 = \{000, 011, 110\}$, $L_4 = \{0000, 0011, 0110, 1001, 1100, 1111\}$, y, cambiando la notación para ahorrar espacio, $L_5 = \{0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30\}$, $L_6 = \{0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63\}$. La tabla de $|L_n|$ para $n = 0, \dots, 8$ es

n	0	1	2	3	4	5	6	7
$ L_n $	1	1	2	3	6	11	22	43

Notamos que si n es impar, $|L_n| = 2|L_{n-1}| - 1$, y si n es par, $|L_n| = 2|L_{n-1}|$. Después de rascarnos la cabeza un poco, probamos $|L_n| = 2|L_{n-1}| - \frac{1+(-1)^{n+1}}{2}$, y se puede demostrar éso. Pero ésta es una ecuación no homogénea, pues $|L_n| - 2|L_{n-1}|$ no es 0. Ciertas ecuaciones no homogéneas admiten métodos especiales que vamos a ver después. Pero este problema es el del número de cadenas de longitud n en un conjunto regular, que hemos mencionado. El autómata tiene tres estados 0, 1, 2 con la siguiente tabla de transición:

	0	1	2
0	0	2	1
1	1	0	2

Ahora definimos $N_i(n)$ como el número de cadenas c de largo n tales que $\delta(i, c) = 0$. Esto nos da el siguiente sistema:

$$\begin{aligned} N_0(0) &= 1; & N_1(n) &= N_0(n-1) + N_1(n-1) \text{ para } n > 0 \\ N_1(0) &= 0; & N_1(n) &= N_2(n-1) + N_0(n-1) \text{ para } n > 0 \\ N_2(0) &= 0; & N_2(n) &= N_1(n-1) + N_2(n-1) \text{ para } n > 0 \end{aligned}$$

Nos interesa $N_0(n)$. Sea T_i la función generadora para N_i . El sistema de ecuaciones de diferencia equivale al sistema de ecuaciones entre las funciones generadoras

$$\begin{aligned}T_0 &= 1 + zT_0 + zT_1 \\T_1 &= zT_2 + zT_0 \\T_2 &= zT_1 + zT_2\end{aligned}$$

La resolución manual de ésto se pone engorrosa rápidamente, pero Maple lo resuelve enseguida:

`solve({T0 = 1 + z * T0 + z * T1, T1 = z * T2 + z * T0, T2 = z * T1 + z * T2}, {T0, T1, T2});`
respondiendo (redactamos un poco)

$$T_0 = \frac{1 - z - z^2}{1 - 2z - z^2 + 2z^3}, T_1 = \frac{z}{1 - z - 2z^2}, T_2 = \frac{z^2}{1 - 2z - z^2 + 2z^3}.$$

Después hay que hacer `assign(%)`. Hay que asegurar al principio, antes de invocar `solve`, que T_0 , T_1 y T_2 no tienen valores ya asignados, si es necesario con `unassign('T0', 'T1', 'T2')`. Ahora sabemos que T_0 cumple una ecuación de diferencia homogénea con polinomio característico $z^3 - 2z^2 - z + 2$, y entonces que cumple la recursión $T_0_n = 2T_0_{n-1} + T_0_{n-2} - 2T_0_{n-3}$. Ya hemos calculado los valores iniciales $T_0_0, T_0_1, T_0_2 = 1, 1, 2$ de nuestro conocimiento del problema, pero en cualquier caso podemos aplicar las relaciones indicadas antes, $F_n = p_n - q_1 F_{n-1} - \dots - q_{n-1} F_0$, que dan aquí $T_0_0 = 1$, $T_0_1 = -1 - (-2)T_0_0$, $T_0_2 = -2 - (-2)T_0_1$. Repetimos para este caso específico la matriz mostrando como $F(1 - 2z - z^2 + 2z^3) = 1 - z - z^2$:

F	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
$-2zF$	0	$-2F_0$	$-2F_1$	$-2F_2$	$-2F_3$	$-2F_4$	$-2F_5$	$-2F_6$
$-z^2F$	0	0	$-F_0$	$-F_1$	$-F_2$	$-F_3$	$-F_4$	$-F_5$
$+2z^3F$	0	0	0	$+2F_0$	$+2F_1$	$+2F_2$	$+2F_3$	$+2F_4$
p	1	-1	-1	0	0	0	0	0

En cada columna, la suma de los primeros cuatro renglones tiene que ser igual al último renglón. Entonces $F_0 = 1$, $F_1 - 2F_0 = -1$ ($F_1 = 1$), $F_2 - 2F_1 - F_0 = -1$ ($F_2 = 2$), $F_3 - 2F_2 - F_1 + 2F_0 = 0$ ($F_3 = 3$), $F_4 - 2F_3 - F_2 + 2F_1 = 0$ ($F_4 = 6$), etc.

Ahora decimos a Maple $r := [\text{solve}(\text{denom}(T_0))]$ y recibimos la respuesta $r := [-1, 1, \frac{1}{2}]$. Pero estas son las raíces del polinomio recíproco del polinomio característico, y nos interesan sus recíprocos, entonces hacemos $r := [\text{seq}(1/r[i], i = 1..3)]$ con la respuesta $r := [-1, 1, 2]$. Ahora sabemos que la solución general es $Ar_1^n + Br_2^n + Cr_3^n$ y mandamos a Maple calcular A, B, C con `inic := [1, 1, 2] : solve(({seq(A * r[i]^n = inic[n + 1], n = 0..2)}, {A, B, C});` (asegurando primero que ninguno de A, B, C tiene valores ya asignado) con respuesta $\{A = \frac{1}{6}, B = \frac{1}{2}, C = \frac{1}{3}\}$. Entonces la solución es

$$|L_n| = \frac{1}{6}(-1)^n + \frac{1}{2} + \frac{2^n}{3}.$$

De ésto, concluimos que la proporción de múltiplos de 3 entre cadenas de longitud n es $\frac{|L_n|}{2^n} = \frac{1}{3}(1 + \frac{(-1)^n}{2^{n+1}}) + \frac{1}{2^n}$

EJEMPLO 6.4. ¿Cuántas cadenas hay de longitud n en $\{0, 1\}^*$ que no tienen subcadena 010?

Requiere bastante ingenio sacar directamente una sola ecuación de diferencias. Pero los métodos que hemos visto nos permiten sacar una rutinariamente. Primero, notemos que el conjunto es regular, y un autómata finito para él tiene estados $q_\epsilon, q_0, q_{01}, q_{no}$ con tabla de transición

	q_ϵ	q_0	q_{01}	q_{no}
0	q_0	q_0	q_{no}	q_{no}
1	q_ϵ	q_{01}	q_ϵ	q_{no}

El estado inicial es q_ϵ y los estados finales son q_ϵ, q_0, q_{01} . Se ve que la única manera de llegar a q_{01} es llegar primero a q_0 y recibir entrada 1, pero la única forma de llegar a q_0 es recibir 0 estando en q_ϵ o en q_0 . Entonces la única forma de llegar a q_{no} la primera vez es recibir 010, y

cualquier 010 nos lleva a q_{no} . Construimos un sistema de ecuaciones de diferencia para $n \geq 1$, definiendo $S_q(n)$ para cada estado q como el conjunto de cadenas c de longitud n en $\{0, 1\}^*$ tales que $\delta(q, c) \in \{q_\epsilon, q_0, q_1\}$, es decir, es un estado final; y definimos $C_q(n) = |S_q(n)|$. Nos interesa C_{q_ϵ} pero para salvar la vista vamos a usar $S_\epsilon, S_0, S_{01}, S_{no}$ en lugar de $S_{q_\epsilon}, S_{q_0}, S_{q_{01}}, S_{q_{no}}$ respectivamente, y $C_\epsilon, C_0, C_{01}, C_{no} = |S_{q_\epsilon}|, |S_{q_0}|, |S_{q_{01}}|, |S_{q_{no}}|$ respectivamente.

Para los conjuntos tenemos

$$\begin{aligned} S_\epsilon(n) &= 0S_0(n-1) \cup 1S_\epsilon(n-1) \\ S_0(n) &= 0S_0 \cup 1S_{01}(n-1) \\ S_{01}(n) &= 1S_\epsilon(n-1) \end{aligned}$$

junto, por supuesto, con $S_{no} = \emptyset$. De allí es inmediato que

$$\begin{aligned} C_\epsilon(n) &= C_\epsilon(n-1) + C_0(n-1) \\ C_0(n) &= C_0(n-1) + C_{01}(n-1) \\ C_{01}(n) &= C_\epsilon(n-1) \end{aligned}$$

para todo $n > 0$. El sistema de ecuaciones es completado teniendo en cuenta que para todo estado q final, $S_q(0) = \{\epsilon\}$, así que $C_\epsilon(0) = C_0(0) = C_{01}(0) = 1$. Esa es toda la información que necesitamos para completar la solución numérica hasta cualquier n , por ejemplo:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C_ϵ	1	2	4	7	12	21	37	65	114	200	351	616	1081	1897	3329
C_0	1	2	3	5	9	16	28	49	86	151	265	465	816	1432	2513
C_{01}	1	1	2	4	7	12	21	37	65	114	200	351	616	1081	1897

Podemos comprobar que, hasta aquí, cada una de las tres sucesiones cumple con la ecuación de diferencia $x_n = 2x_{n-1} - x_{n-2} + x_{n-3}$. Requiere una chispa de inspiración para sacar esa ecuación directamente (el colaborador de este libro, Jorge Yazlle, lo hizo antes que descubrimos la aplicabilidad de funciones generadoras) pero los métodos que hemos estado viendo aquí permiten sacarlo rutinariamente, como vemos ahora.

El sistema de ecuaciones entre las tres funciones generadoras, que llamamos F_ϵ, F_0 , y F_{01} , que corresponde al sistema de ecuaciones de diferencias, es

$$\begin{aligned} F_\epsilon &= 1 + zF_\epsilon + zF_0 \\ F_0 &= 1 + zF_0 + zF_{01} \\ F_{01} &= 1 + zF_\epsilon \end{aligned}$$

La solución (que se puede obtener de Maple mediante *solve*, como hemos visto) es

$$\begin{aligned} F_\epsilon &= \frac{1 + z^2}{1 - 2z + z^2 - z^3} \\ F_0 &= \frac{1}{1 - 2z + z^2 - z^3} \\ F_{01} &= \frac{1 - z + z^2}{1 - 2z + z^2 - z^3} \end{aligned}$$

que, teniendo en cuenta que el polinomio característico de cada una es el polinomio recíproco $z^3 - 2z^2 + z - 1$, muestra que cada una de las tres sucesiones cumple con la ecuación de diferencia que dijimos.

Para completar el ejemplo, hay que resolver el sistema de tres ecuaciones en tres incógnitas A, B, C con las tres raíces $\{r_1, r_2, r_3\} = \{1,755, 0,123 + 0,745i, 0,123 - 0,745i\}$ (redondeando a 3 decimales)

$$\{Ar_1^n + Br_2^n + Cr_3^n = C_\epsilon(n) : n = 0, 1, 2\}.$$

Así, ya que las dos raíces complejas tienen magnitud menor que 1, descubrimos que, para n grande, $C_\infty(n)$ es aproximadamente $1.267(1.755)^n$. Usando todos los dígitos que me dió Maple para A y r_1 (9 decimales), $Ar_1^{14} = 3329,007\dots$

4.1. Ecuaciones no homogéneas. Cuando una sucesión x cumple para $n \geq d$ una ecuación

$$a_0x_n + a_1x_{n-1} + \dots + a_dx_{n-d} = f_{n-d}$$

donde f_n no es 0 para todo n , se llama *ecuación no homogénea* de diferencias. Los métodos que hemos visto resuelven esto en los casos en que la función generadora de f es una función racional $\frac{p}{q}$. En tal caso, si X es la función generadora de x , entonces la siguiente tabla, en que

$a_0X :$	a_0x_0	a_0x_1	a_0x_2	a_0x_3	a_0x_4	a_0x_5	a_0x_6	\dots	a_0x_n	
$a_1zX :$	0	a_1x_0	a_1x_1	a_1x_2	a_1x_3	a_1x_4	a_1x_5	\dots	a_1x_n	
$a_2z^2X :$	0	0	a_2x_0	a_2x_1	a_2x_2	a_2x_3	a_2x_4	\dots	a_2x_n	
$a_3z^3X :$	0	0	0	a_3x_0	a_3x_1	a_3x_2	a_3x_3	\dots	a_3x_n	
$a_4z^4X :$	0	0	0	0	a_4x_0	a_4x_1	a_4x_2	\dots	a_4x_n	
suma					f_0	f_1	f_2	\dots	f_{n-d}	y se ve

que en general, ya que $\frac{p}{q}$ es la función generadora de f

$$a_0X + a_1zX + a_2z^2X + \dots + a_dz^dX = \frac{p}{q} + r,$$

donde r es, en el caso $d = 4$, el polinomio,

$$a_0x_0 + (a_0x_1z + a_1x_0)z^2 + (a_0x_2 + a_1x_1 + a_2x_0)z^3 + (a_0x_3 + a_1x_2 + a_2x_1 + a_3x_0)z^4$$

que se saca de sumar las columnas 0 a 3, y en general es

$$r = \sum_{i=0}^{d-1} \left(\sum_{j=0}^i a_j x_{d-j-1} \right) z^i.$$

Manipulando un poco, sale

$$X = \frac{p + rq}{(a_0 + a_1z + a_2z^2 + \dots + a_dz^d)q}.$$

Ya sabemos que q es el polinomio recíproco del polinomio característico de f , y $(a_0 + a_1z + a_2z^2 + \dots + a_dz^d)$ es el polinomio recíproco del polinomio característico de la ecuación homogénea $a_0x_n + a_1x_{n-1} + \dots + a_dx_{n-d} = 0$. Ahora es claro que el polinomio recíproco de un producto de polinomios es el producto de sus respectivos polinomios característicos, así que lo que hemos visto ahora es que la función generadora de la sucesión x es una función racional cuyo polinomio característico es el producto del polinomio característico de la ecuación homogénea con el de f . Esto presenta x como problema de una ecuación homogénea.

En lo que sigue a veces vamos a presentar el problema en la forma

$$a_0x_n + a_1x_{n-1} + \dots + a_dx_{n-d} = f_n$$

Tener f_{n-d} a la derecha convenía más para la justificación de las aserciones iniciales, pero no importa cual f_m aparece en la ecuación, es esencialmente el mismo problema: por reemplazo, f_n puede ser reemplazado por f'_{n-d} resultando que $f'_0 = f_d$, y no hay cambio esencial.

Por ejemplo, sea f la sucesión de Fibonacci, $f_0 = 0$, $f_1 = 1$ y $f_n = f_{n-1} + f_{n-2}$ para todo $n > 1$, y sea $x_n = \sum_{i=0}^n f_i$. Entonces $x_n - x_{n-1} = f_n$, una ecuación no homogénea. El polinomio característico de la ecuación homogénea $x_n - x_{n-1} = 0$ es $z - 1$, y el de f es $z^2 - z - 1$. Hemos visto que x es la solución de un problema de una sola ecuación de diferencias lineal cuyo polinomio característico es $(z - 1)(z^2 - z - 1)$. Las raíces son 1, φ , $\widehat{\varphi}$, y $x_0, x_1, x_2 = 0, 1, 2$. La solución va a ser $A + B\varphi^n + C\widehat{\varphi}^n$, y sale $A = -1$, $B = 1,1708$, $C = -0,1708$. x empieza

0, 1, 2, 4, 7, 12, 20, 33, 54, 88, 143, 232, 376, 609, 986, 1596.

Se nota que $x_n = x_{n-1} + x_{n-2} + 1$ para $n > 1$, o sea la ecuación no homogénea $x_n - x_{n-1} - x_{n-2} = 1$. De este punto de vista, el polinomio característico de la ecuación homogénea $x_n - x_{n-1} - x_{n-2} = 0$

es $z^2 - z - 1$, y la función $f_n = 1$ es una solución de $f_n - f_{n-1} = 0$ con polinomio característico $z - 1$. El producto es lo mismo.

Pero en una ecuación no homogénea $a_0x_n + a_1x_{n-1} + \cdots + a_dx_{n-d} = f_n$, hay una posibilidad de acortar el trabajo. Si podemos encontrar cualquier solución p a la ecuación no homogénea, es decir $a_0p_n + a_1p_{n-1} + \cdots + a_dp_{n-d} = f_n$, y g es la solución general de la ecuación homogénea $a_0x_n + a_1x_{n-1} + \cdots + a_dx_{n-d} = 0$, entonces la solución general es $g + p$. Para ver que $g + p$ es una solución: $\sum_{i=0}^d a_i(g_{n-i} + p_{n-i}) = \sum_{i=0}^d a_i g_{n-i} + \sum_{i=0}^d a_i p_{n-i} = 0 + f_n = f_n$. Entonces cualquier asignación de las d incógnitas asociadas con la solución general de la ecuación homogénea da una nueva solución a la ecuación no homogénea. Esto hace plausible que es la solución general. Por lo menos, podemos plantear d ecuaciones

$$\left\{ \sum_{i=0}^d a_i(g_{n-i} + p_{n-i}) = x_n : n = 0, \dots, d-1 \right\}$$

en las d incógnitas y si el sistema no es singular, sacar la solución al problema.

Por ejemplo, en el problema $x_n - x_{n-1} - x_{n-2} = 1$, con $x_0 = 0$ y $x_1 = 1$, una solución particular es $p_n = -1$. En este caso la ecuación homogénea es $x_n - x_{n-1} - x_{n-2} = 0$, la de la sucesión de Fibonacci. Sabemos que la solución general de ésto es $A\varphi^n + B\hat{\varphi}^n$, por tanto la solución general del problema es $A\varphi^n + B\hat{\varphi}^n - 1$. Planteamos las dos ecuaciones

$$\begin{aligned} A\varphi^0 + B\hat{\varphi}^0 - 1 &= A + B - 1 = 0 \\ A\varphi^1 + B\hat{\varphi}^1 - 1 &= 1 \end{aligned}$$

que da $A = \frac{1}{2} + \frac{3}{10}\sqrt{5}$, $B = \frac{1}{2} - \frac{3}{10}\sqrt{5}$.

En general, no es automático encontrar una solución particular. Pero en el caso de que los dos polinomios característicos no tienen raíz común, hay un algoritmo. Resulta que en ese caso alguna solución a la ecuación de diferencia que cumple f es una solución particular. Por, sea $\sum_{i,j} A_{ij}n^j s_i^n$ la solución general de esa ecuación, donde los s_i son las raíces del polinomio característico, y $\sum_{i,j} B_{ij}n^j r_i^n$ la solución general de la ecuación homogénea. Entonces la solución general de la ecuación no homogénea, que es la misma que la solución general de la ecuación homogénea que es el producto de los dos polinomios característicos mencionados, es $\sum_{i,j} A_{ij}n^j s_i^n + \sum_{i,j} B_{ij}n^j r_i^n$. Esto quiere decir que para alguna asignación a los A_{ij} y B_{ij} , digamos \hat{A}_{ij} y \hat{B}_{ij} respectivamente, si ponemos $\hat{x}_n = \sum_{i,j} \hat{A}_{ij}n^j s_i^n + \sum_{i,j} \hat{B}_{ij}n^j r_i^n$, entonces \hat{x}_n cumple la ecuación de diferencia $\hat{x}_n + a_i\hat{x}_{n-1} + \cdots + a_d\hat{x}_{n-d} = f_n$, pero $\sum_{i,j} \hat{A}_{ij}n^j s_i^n$ es la solución general de la ecuación homogénea, así que si ponemos $y_n = \sum_{i,j} \hat{A}_{ij}n^j s_i^n$ tenemos $y_n + a_1y_{n-1} + \cdots + a_dy_{n-d} = 0$, y si ponemos $z_n = \sum_{i,j} \hat{B}_{ij}n^j r_i^n$, tenemos $\hat{x}_n = y_n + z_n$ y tenemos que concluir que $a_0z_n + a_1z_{n-1} + \cdots + a_dz_{n-d} = f_n$, es decir que z_n es una solución particular.

Como ejemplo de la aplicación de ésto, volvemos al problema de $\sum_{i=0}^n f_i$ donde f es la sucesión standard de Fibonacci. La sucesión cumple $x_n - x_{n-1} = f_n$. El polinomio característico de la ecuación homogénea $x_n - x_{n-1} = 0$ es $z - 1$, y el de la sucesión de Fibonacci, $f_n - f_{n-1} - f_{n-2} = 0$ es $(z - \varphi)(z - \hat{\varphi})$, con la solución general $A\varphi^n + B\hat{\varphi}^n$. Algún miembro de esta solución general tiene que ser una solución particular de la ecuación no homogénea: es decir, para algún A, B , $A\varphi^n + B\hat{\varphi}^n - (A\varphi^{n-1} + B\hat{\varphi}^{n-1}) = f_n$. Para $n = 1, 2$, esto nos da las dos ecuaciones $A\varphi + B\hat{\varphi} - (A + B) = 1$ y $A\varphi^2 + B\hat{\varphi}^2 - (A\varphi + B\hat{\varphi}) = 1$, cuya solución es $A = \frac{5+3\sqrt{5}}{10}$, $B = \frac{5-3\sqrt{5}}{10}$. Entonces la solución general de la ecuación homogénea es $C + \frac{5+3\sqrt{5}}{10}\varphi^n + \frac{5-3\sqrt{5}}{10}\hat{\varphi}^n$. Para $n = 0$ esto da $C + \frac{5+3\sqrt{5}}{10} + \frac{5-3\sqrt{5}}{10} = 0$, o $C = -1$. Concluimos que $\sum_{i=0}^n f_i = \frac{5+3\sqrt{5}}{10}\varphi^{n+1} + \frac{5-3\sqrt{5}}{10}\hat{\varphi}^{n+1} - 1$.

Vale la pena una observación más: cuando el producto de los dos polinomios tiene por lo menos dos raíces distintas, cada problema de este tipo puede convertirse en uno en que los dos polinomios no tienen raíz común. ¿Y qué ganamos? En lugar de resolver un sistema de $d + e$ ecuaciones con $d + e$ incógnitas, resolvemos dos sistemas de e ecuaciones y e incógnitas, y uno de d ecuaciones con d incógnitas.

Como ejemplo, y uno que además es paradigma del procedimiento general, consideremos $x_n = \lfloor n/4 \rfloor$. Cumple la ecuación $x_n - x_{n-4} = 1$. La ecuación homogénea $x_n - x_{n-4} = 0$ tiene polinomio característico $z^4 - 1$, con 4 raíces distintas, $1, -1, i, -i$. En el otro lado tenemos la función $f_n = 1$ que cumple $x_n - x_{n-1} = 0$ con polinomio característico $z - 1$. Ahora ponemos $(z^4 - 1) = (z^3 + z^2 + z + 1)(z - 1)^2$, y tratamos de convertir en un problema con una ecuación no homogénea $x_n + x_{n-1} + x_{n-2} + x_{n-3} = g_n$ donde g cumple la ecuación $g_n - 2g_{n-1} + g_{n-2} = 0$. La solución general allí es $A + Bn$. Los x_n son los mismos que buscamos, $\lfloor \frac{n}{4} \rfloor$. El valor de \hat{x}_n es 0 para $n = 0, 1, 2, 3$. Correspondiente al polinomio característico $z^3 + z^2 + z + 1$ es la ecuación homogénea $x_n + x_{n-1} + x_{n-2} + x_{n-3} = 0$. Buscamos la sucesión g que cumple la ecuación $g_n - 2g_{n-1} + g_{n-2} = 0$ y tal que $\lfloor \frac{n}{4} \rfloor + \lfloor \frac{n-1}{4} \rfloor + \lfloor \frac{n-2}{4} \rfloor + \lfloor \frac{n-3}{4} \rfloor = g_n$. Sabemos que g viene de la solución general $p + qn$, por el polinomio característico $(z - 1)^2$. Poniendo $n = 3, 4$ da $0 = p + 3q$ y $1 = p + 4q$, de donde $p = -3$ y $q = 1$.

Lo hemos transformado al problema $x_n + x_{n-1} + x_{n-2} + x_{n-3} = n - 3$. Siendo que los polinomios característicos no tienen raíz en común, sabemos que algún $D + En$ es una solución particular. Poniendo $n = 3$ y 4 tenemos las dos ecuaciones

$$\begin{aligned} \sum_{m=0}^3 D + mE &= 0 \\ \sum_{m=1}^4 D + mE &= 1 \end{aligned}$$

con solución $D = -\frac{3}{8}$, $E = \frac{1}{4}$, es decir $-\frac{3}{8} + \frac{n}{4}$ es una solución particular y por lo tanto la solución general del problema es

$$A(-1)^n + Bi^n + C(-i)^n - \frac{3}{8} + \frac{n}{4}.$$

Normalmente doy un 3×3 a Maple, pero ya que el punto de esto es poder facilitar los cálculos, resolví a mano las tres ecuaciones $A + B + C = 3/8$, $-A + Bi - Ci = 1/8$, $A - B - C = -1/8$ con el resultado $\{A = 1/8, B = \frac{1-i}{8}, C = \frac{1+i}{8}\}$. Entonces

$$\left\lfloor \frac{n}{4} \right\rfloor = \frac{(-1)^n}{8} + \frac{1-i}{8}i^n + \frac{1+i}{8}(-i)^n - \frac{3}{8} + \frac{n}{4}.$$

Pruébalo.

Lo que ganamos es resolver un 3×3 y dos 2×2 en lugar de un 5×5 .

5. Sobre la aritmética sobre series infinitas

Exponer un fundamento teórico sobre las operaciones con series infinitas que hemos usado en la teoría de funciones generadoras estaría fuera de nuestro alcance, pero hay unas observaciones que pueden indicar su plausibilidad.

Nos gustaría decir que estas series forman un cuerpo respecto de sumar y multiplicar, pero falta la división. Pero sí en los casos que importan aquí, dadas dos series s y t , existe la serie s/t ; es decir, existe una serie x tal que $x \times t = s$. Para ver esto tenemos que ver la definición de multiplicación:

Sea $s = \sum_{n=0}^{\infty} s_n z^n$ y $t = \sum_{n=0}^{\infty} t_n z^n$. Entonces

$$s \times t = \sum_{n=0}^{\infty} \sum_{i+j=n} s_i t_j z^{i+j},$$

es decir, el n -ésimo término de $s \times t$ es $\sum_{i+j=n} s_i t_j$. Entonces, si buscamos una serie x tal que $x \times t = s$, sabemos inmediatamente que x_0 es necesariamente t_0/s_0 , para que podamos decir que $x_0 t_0 = s_0$, como pide la definición de \times . Entonces no puede ser que $s_0 = 0$. Hasta allí bien, no queremos dividir por la serie 0. Pero ahora preguntamos qué puede ser x_1 , y la definición

de \times exige que $x_0 t_1 + x_1 t_0 = s_1$, o sea $x_1 = \frac{s_1 - x_0 t_1}{t_0}$, y de nuevo lo único que hace falta es que $t_0 \neq 0$. De hecho, en general si hemos podido calcular x_i para todo $i \leq n$ entonces para saber x_{n+1} necesitamos hacer cumplir $\sum_{i+j=n+1} x_i t_j = s_{n+1}$ o sea $x_{n+1} t_0 + \sum_{i=1}^n x_i t_{n+1-i} = s_{n+1}$, o sea

$$x_{n+1} = \frac{s_{n+1} - \sum_{i=1}^n x_i t_{n+1-i}}{t_0}$$

y de nuevo lo único que hace falta es que $t_0 \neq 0$. Entonces división está permitida y bien definida siempre que el término 0 del denominador no es 0. Entonces habría que verificar en cada paso que pide división que se cumpla esa condición.

El resto es las definiciones de suma $((s \pm t)_i = s_i \pm t_i)$, y las leyes asociativas y distributivas, y algunas otras reglas, como que $\frac{p}{q}/r = \frac{p}{qr}$, que son demostrables directamente, aunque sea algo engorroso.

Ejercicios







- (1) Calcular el número de cadenas binarias de largo n en que no ocurre 00.
- (1) Calcular el número de cadenas binarias de largo n en que no ocurre 01.
- (2) Una empresa de alquiler de autos trabaja en Jujuy, Salta y Tucumán. Un cliente que alquila un auto en una de las tres ciudades puede devolverlo en cualquier de la tres ciudades. En la siguiente tabla el dato en el renglón de la ciudad c y la columna de la ciudad d muestra el porcentaje de los autos que van de la ciudad c a la ciudad d cada día.

	a Jujuy	a Salta	a Tucumán
desde Jujuy	60	25	15
desde Salta	15	65	20
desde Tucumán	15	15	70



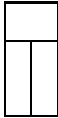
Si al empezar el día 0, 50 % de los autos de la empresa están en Salta, 40 % en Jujuy, y 10 % en Tucumán, calcule como función de n el porcentaje en Salta, Jujuy y Tucumán después de n días.

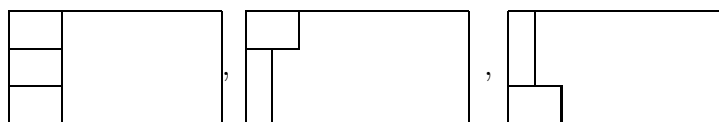
- (2) Calcule el número de cadenas le largo n que, leídos como números expresados en binario, dan
 - $4k + 1$, algún k .
 - $3k + 1$, algún k .
 - $4k + 1$ o $3k + 1$, algún k .
 - $4j + 1$ y $3k + 1$, algún j, k .

5. (1) En el ejercicio 4, el resultado de (c) es el de la cardinalidad de la unión de los conjuntos cuyos cardinalidades se calculan en (a) y (b), y el de (d) la intersección de ellos. Si sumamos (a) y (b) estamos contando dos veces las cadenas en la intersección. Entonces esa suma menos el resultado de (d) debe dar el resultado de (a). ¿Se verifica eso?

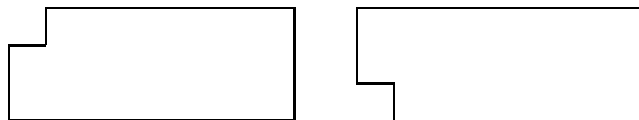
- (2) ¿Cuántas maneras hay de poner n rectángulos 2×1 en un rectángulo de $2 \times n$? Por ejemplo, para $n = 2$ son 2: , , y para $n = 3$ son 3: , , , y para $n = 4$ empezamos o con , dejando el problema de poner los otros 3 en un 2×3 , o empezamos

con , dejando el problema de poner los otros 2 en un 2×2 , un total de 5.

- (3) ¿Cuántas maneras hay de llenar un rectángulo de $3 \times n$ con rectángulos de 2×1 ? Por ejemplo, para $n = 1$ no hay ninguno. Para $n = 2$ son 3: , , . Para cualquier n , hay tres maneras de empezar:



En el primer caso, se queda con el problema de llenar un rectángulo de $3 \times n - 2$, y en los otros dos se queda con el problema de llenar un rectángulo de $3 \times n - 1$ en que se ha sacado un cuadradito de una de las esquinas de la izquierda:



Esta es otra sucesión. Se puede construir un sistema de dos ecuaciones de diferencias con las dos sucesiones.

8. (4) En la ciencia de computación se define *árbol binario* así:

- \emptyset es un *árbol binario*.
- Si I y D son *árboles binarios* y x es natural entonces (x, I, D) es un *árbol binario*.

Definimos la *altura* $H(A)$ de un árbol binario así:

- $H(\emptyset) = 0$.
- $H((x, L, R)) = 1 + \max(H(L), H(R))$.

Y definimos el *tamaño* $T(A)$ de un árbol binario A así:

- $T(\emptyset) = 0$.
- $T((x, L, R)) = 1 + T(L) + T(R)$

Finalmente, definimos cuando un árbol binario es *balanceado*¹:

- \emptyset es *balanceado*.
- (x, I, D) es *balanceado* si $|H(I) - H(D)| \leq 1$ y I y D son *balanceados*.

Demuestre que existe un número k tal que la altura de un árbol binario balanceado de tamaño n es menor que $k \log n$. Sugerencia: aplique la teoría de este capítulo a la sucesión $p_h =$ el mínimo tamaño que puede tener un árbol binario balanceado de altura h . Si (x, I, D) es balanceado y tiene altura h entonces uno de I, D tiene altura $h - 1$ y el otro o $h - 1$ o $h - 2$.

¹En la ciencia de computación, esta propiedad junto con el ordenamiento de los x de izquierda a derecha, se llama *árbol AVL* (por sus inventores rusos) que presenta dos sorpresas: una exhibida en este ejercicio, que implica que la búsqueda es logarítmica, y otra que no se pierde mucho tiempo en preservar la propiedad de ser balanceado frente a inserciones y borrados (ver Knuth, *The Art of Computer Programming*, Volumen 3, *Sorting and Searching*, pp 451-458).

PROBABILIDAD DISCRETA

1. Introducción

Como los axiomas de Peano para los números naturales salen de la aritmética de los palos, los axiomas de la teoría de probabilidades salen por sentido común de lo que esperamos del comportamiento de una moneda honesta tirada como parte de un juego de azar. Y, ¿qué, exactamente, es este comportamiento?

Para empezar, decimos que “cara” (llamamos los dos lados de la moneda “cara” y “cruz”) sale igualmente a menudo que “cruz”. A ver, tengo diez monedas en el bolsillo. Están dando vuelta allí todo el día. A ver... tengo 3 cruz y 7 caras. ¿Pero no dijimos que tenía que salir cruz igualmente a menudo? Parece que salen sólo 30 % cruz. ¿Qué, no son honestas estas monedas? Aquí las sacudo bien en mis manos...ahora tengo 6 cruces y 4 caras. Otra vez... 5 de cada una, como debe ser. Lo hago otra vez, sacudiendo bien... 5 cada una otra vez. Ahora las monedas se comportan mejor. Una vez más...4 cruz, 6 cara.

¿Qué podemos decir por seguro? No que salen igual número de cruz que cara, porque a veces sí y a veces no. De hecho, pensando un poco, para que salga 5-5 tiene que pasar primero o por 5-4 o por 4-5, y de cualquier de estos dos penúltimos estados, si es cierto que la moneda es honesta, tiene que salir sólo la mitad de las veces 5-5, y el resto o 6-4 o 4-6. Pero 6-4 puede ocurrir desde 6-3, y 4-6 de 3-6, así que 6-4 a favor del uno o del otro es aún más probable que 5-5. De hecho, uno de los posibles resultados da 10-0. ¿No? Si no, entonces debe ser 9-0 en 9 tiros imposible también, porque si no, va a ocurrir alguna vez y cuando ocurra es igualmente probable que el último tiro da 10-0 o 9-1. ¿Cuál es el número máximo n tal que es posible que n tiros salen n -0? Si es posible, va a ocurrir alguna vez, porque eso es lo que quiere decir “posible”. Pero cuando ocurra, es igualmente probable que el n -ésimo tiro dé $n - 1$ o $n + 1$. Hay gente que disputa eso, que dice que la moneda tiene que dar igual números de cara y cruz, y que la moneda reacciona cuando su historia muestra que una u otra sobra, haciéndola favorecer a la que falta. Al contrario, un axioma fundamental es

El comportamiento de la moneda es independiente de su historia.

Eso implica que si se tira una moneda honesta n veces, cada una de las 2^n sucesiones posibles de resultados tiene la misma probabilidad: 10 caras tiene la misma probabilidad que XCXXXCCXCC, 2^{-10} . Ya que hay exactamente 10 de esas sucesiones con una sola cruz, la probabilidad de tener una sola cruz es $10/1024$, un poco menos que 1 en 100. Sucesiones con exactamente dos cruces en diez tiros son $\binom{10}{2}$, donde $\binom{n}{m}$ es el número de subconjuntos de m elementos de un conjunto de n , y es igual a $\frac{n \times (n-1) \times \dots \times (n-m+1)}{m!}$. Por lo tanto, la probabilidad de tener m cruces en 10 tiros es $\frac{\binom{10}{m}}{1024}$. Hagamos una tabla de estos $\binom{10}{m}$, y las probabilidades correspondientes abajo de eso:

m	0	1	2	3	4	5	6	7	8	9	10
$\binom{10}{m}$	1	10	45	120	210	252	210	120	45	10	1
prob	.00098	.0098	.044	.117	.205	.246	.205	.117	.044	.0098	.00098

Ahora hagamos 1024 pruebas de 10 tiros cada una. Para cada m , el número de pruebas en que resultó m cruces debería ser alrededor de la fracción $\frac{\binom{10}{m}}{1024}$ de las 1024 pruebas, o sea $\binom{10}{m}$. No necesariamente en forma exacta, pero cerca de allí. Pero ¿quién va a tirar la moneda 1024 veces? Respuesta: la computadora, que sabe simular los azarosos tiros de moneda. Las 1024

pruebas de 10 tiros cada una dan con sus respectivos promedios (que esperemos esté cerca de la probabilidad calculada antes):

m	0	1	2	3	4	5	6	7	8	9	10
res	0	12	58	133	205	230	221	99	49	15	2
prom	0	.012	.057	.130	.200	.225	.216	.097	.048	.146	.002

No hay gran diferencia entre los promedios y las probabilidades. Se ve que para $m_1 < m_2 \leq 5$ o $m_1 > m_2 \geq 5$, m_1 cruces ocurre menos que m_2 , tanto en las probabilidades teóricas y los resultados. En particular, 5 cruces y 5 caras es lo que ocurre más frecuentemente, lo que no sorprende, pero por el otro lado ese resultado no es muy frecuente: 252 veces en 1024, un poco menos que 25 %. Notemos que la probabilidad de tener entre 3 y 7 cruces, es decir de no variar más que 2 de igual número de cara y cruz, es 0,89. Sólo un décimo de las pruebas van a estar afuera de ese intervalo. Los resultados están bastante conformes: 0,884.

La moraleja de esto es que en la teoría de probabilidades no estamos seguros de nada, pero hay cosas de las que estamos casi seguros. Uno, por ejemplo, es que si tiro una moneda honesta 20 veces no van a salir 20 cruces, por que la probabilidad de eso es alrededor 1 en un millón. Pero eso puedo decir de cualquier resultado de este experimento: si tiro esa moneda 20 veces puedo decir que he sido testigo de un evento que tenía probabilidad 1 en un millón. ¿Porque doy significado especial a la cadena de 20 cruz, y digo que esa moneda debe ser trucha si sale así? Lo dejo para ponderar eso.

Hemos dicho que estamos casi seguros, 90 %, que de mil experimentos en que tiramos una moneda diez veces, el promedio del número de cruces va a ser entre 3 y 7, o sea dentro de 40 % del promedio teórico, o *esperanza*, de 5. Si tiro una moneda 50 veces, la probabilidad de m cruces es $\frac{\binom{50}{m}}{2^{50}}$. Acabo de calcular (con Maple)

$$\frac{\sum_{m=20}^{30} \binom{50}{m}}{2^{50}} = 0,881,$$

casi igual a la probabilidad que hay entre 3 y 7 cruces en 10 tiros, pero esta vez el intervalo es dentro de 10 % de la esperanza.

Hemos usado la palabra *esperanza*, y aún *probabilidad* en su sentido intuitivo, lo que esperamos en 10 tiros es 5 cruces y cinco cara, aunque la probabilidad de exactamente éso está cerca de 1/4.

Vamos a ver cómo es el modelo formal en que ocurren esos y otros términos.

Un *segmento inicial* de los números naturales es una de dos cosas: o todos los números menores o iguales que algún n , o todos los números. Un *espacio de probabilidad discreta* es un conjunto $\{e_i : i \in I\}$ para un segmento inicial I de los naturales, junto con una función p de ese conjunto a los reales (usualmente racionales) en $[0, 1]$ tal que $\sum_{i \in I} p(e_i) = 1$. Los e_i se llaman *eventos elementales*, p se llama *función de probabilidad*, y $p(e_i)$ es la *probabilidad* de e_i .

Un *evento* es un conjunto de eventos elementales. Extendemos p a todos los eventos definiendo para un evento A , $p(A) = \sum_{e \in A} p(e)$.

Lo que hacen los eventos es “ocurrir” con “frecuencia” $p(e)$. Si ocurren n eventos elementales, esperamos $p(e)n$ ocurrencias de un evento elemental dado e .

El espacio que corresponde a un dado es $\{e_0, e_1, e_2, e_3, e_4, e_5\}$, o simplemente $\{1, 2, 3, 4, 5, 6\}$ para darles nombres más amenos, con $p(e) = 1/6$ para cada e elemental. Un ejemplo de un evento no elemental es $\{2, 4, 6\}$, que podríamos llamar “par”. El espacio de una rueda de ruleta es $\{0, 1, 2, \dots, 36\}$.

Dada una función numérica f de los eventos elementales de un espacio E , la *esperanza* de f es $\sum_{e \in E} p(e)f(e)$. Por ejemplo, la esperanza del valor del dado es $\frac{1}{6}(1+2+3+4+5+6) = 7/2$. Esto es lo que esperamos para el promedio de un número medio grande de tiros. Si tiramos el dado 6000 veces esperamos que cada uno de los seis posibles resultados salen aproximadamente 1000 veces cada uno, y la suma de los resultados entonces aproximadamente $1000(1+2+3+4+5+6) = 21000$, para un promedio de aproximadamente $21000/6000 = 7/2$. Lo acabo de hacer tres veces

(electrónicamente, con la computadora) con estos resultados (los últimos dos renglones muestran el total de tiros para cada resultado y la proporción que representa de los 18000 tiros):

1	2	3	4	5	6	total	prom
1012	1014	1009	960	992	1013	20945	3.490
978	983	976	1013	1069	981	21155	3.525
1008	1032	969	963	1016	1012	20983	3.497
2998	3029	2934	2936	3077	3006		
.1666	.1682	.1630	.1631	.1709	.1670		

El espacio que corresponde a dos dados puede ser eventos $\{(i, j) : i = 1..6, j = 1..6\}$, o, lo que realmente ocurre: un evento es las ocurrencias simultáneas de un evento de dos espacios independientes, cada uno representando un dado. En cualquier de los dos, $p(e) = 1/36$ para cada e .

El evento “suma = 7” es $\{1, 6\}, \{2, 5\}, \{3, 4\}, \{4, 3\}, \{5, 2\}, \{6, 1\}$. La probabilidad de suma=7 es $6/36 = 1/6$.

En el juego de pase inglés de dados sólo importan los eventos suma=s. Entonces podemos considerarlo un espacio cuyos eventos son 2,3,4,5,6,7,8,9,10,11,12, con probabilidades

evento	2	3	4	5	6	7	8	9	10	11	12
prob	1/36	1/18	1/12	1/9	5/36	1/6	5/36	1/9	1/12	1/18	1/36

En el juego, un jugador tira los dados hasta que aparezca o un tiro ganador o un tiro perdedor. En el primer tiro, 2 (“ojos de víbora”), 3, y 12 (“wagones”) son perdedores y 7 y 11 son ganadores, y termina el turno. Si el primer tiro no es ni 2,3,12,7 ni 11, por el resto del juego la repetición del resultado del primer tiro es ganador, y 7 es perdedor: tiene que seguir tirando hasta salir uno de los dos.

Vamos a calcular la probabilidad de ganar para el que tira. La probabilidad de ganar en el primer tiro es $p(7) + p(11) = 2/9$. Entonces en N juegos ganamos $\frac{2}{9}N$ en el primer tiro. Para cada $s \in A = \{4, 5, 6, 8, 9, 10\}$, sea q_s la probabilidad que s se tira antes de 7. Entonces en N juegos el número ganado debe ser aproximadamente

$$\frac{2}{9}N + \sum_{s \in A} p(s)q_s N$$

porque en $Np(s)$ juegos el primer tiro es s , y en la proporción q_s de ellos, s sale antes de 7. Este número de juegos dividido por N es una estimación empírica de la probabilidad de ganar. Concluimos que la probabilidad de ganar es

$$\frac{2}{9} + \sum_{p \in A} p(s)q_s.$$

Para calcular q_s conviene establecer para dos eventos A, B la *probabilidad del evento B dado el evento A*, que se llama *probabilidad condicional* y tiene la notación compacta $p(B|A)$. Cuando digo “dado A ” estoy, en efecto, cambiando el espacio, los únicos eventos que existen mientras estamos bajo esa restricción son los de A . Para aclarar el concepto, vamos a describir un experimento para medirlo aproximadamente:

Hacemos ocurrir un número grande N de eventos elementales y registramos todos los que pertenecen a A , aproximadamente $p(A)N$. De estos, contamos cuántos también pertenecen a B . Estos van a ser aproximadamente $p(A \cap B)N$. Entonces, entre los eventos en A , la proporción de los que también están en B es, en este experimento, aproximadamente $\frac{p(A \cap B)}{p(A)}$. Concluimos que

$$p(B|A) = \frac{p(A \cap B)}{p(A)}.$$

Ahora, cuando el jugador quiere tirar s antes de 7, se decide cuando, y sólo cuando, un tiro está en el evento $\{s, 7\}$, y por tanto $q_s = p(\{s\}|\{s, 7\}) = p(\{s\} \cap \{s, 7\})/p(\{s, 7\})$ o sea

$$q_s = \frac{p(s)}{p(s) + p(7)} = \frac{p(s)}{p(s) + 1/6}$$

Entonces la probabilidad de ganar es

$$\frac{2}{9} + \frac{(1/12)^2}{1/4} + \frac{(1/9)^2}{5/18} + \frac{(1/12)^2}{1/4} + \frac{(1/12)^2}{1/4} + \frac{(1/9)^2}{5/18} + \frac{(1/12)^2}{1/4} = 0,4929.$$

Casi $1/2$.

¿Qué pasa si los dados están cargados? Digamos según este espacio, para cada dado: $p(1) = p(6) = 1/4$, $p(i) = 1/8$ para los demás. Ahora, para dos dados así, la probabilidad de tener la suma 7 es $1/16 + 1/64 + 1/64 + 1/64 + 1/64 + 1/16 = 3/16$. La cuantía completa es

2	3	4	5	6	7	8	9	10	11	12
1/16	1/16	5/64	3/32	7/64	3/16	7/64	3/32	5/64	1/16	1/16

Usando la misma fórmula que antes para la probabilidad de ganar,

$$p(7) + p(11) + \sum_{s=4,5,6,8,9,10} \frac{p(s)^2}{p(s) + p(7)}$$

la probabilidad de ganar es 0,439, menos que con los dados honestos, aunque la probabilidad de ganar en el primer tiro aumentó de $2/9$ a $1/4$.

Las leyes de la probabilidad surgen del sentido común. Se puede desarrollar axiomáticamente, pero como vimos en la teoría de números, el sentido común tiene prioridad. Pero el sentido común no es infalible. A veces el sentido común no está de acuerdo con el sentido común del otro. Y a veces la evidencia empírica no está de acuerdo con el sentido común de nadie. Por eso necesitamos un modelo formal.

Por ejemplo, en el juego de poker, tener 5 cartas del mismo palo es muy bueno. En el juego estandar, cada jugador recibe 5 cartas, se apuestan, y después cada uno puede pedir cambiar hasta tres de sus cartas por unas nuevas elegidas al azar de entre las que no se han entregado. Un jugador tiene 4 cartas de corazón y otra de diamantes. Piensa que si descarta el diamante y pide una nueva, ya que $1/4$ de las cartas son corazones, tiene 1 chance en 4 de acertar. Pero está equivocado: hay 52 cartas en total, de las cuales él tiene 5, y quedan 47. Él tiene cuatro de los corazones, y hay 13 en total, así que quedan 9 entre las 47 de donde va a venir su carta nueva. Tiene 9 chances en 47, menos que 1 en 5, de acertar. Moraleja: use sentido común, pero con cuidado.

Suponga que en un partido de tenis, el que saca tiene probabilidad p de ganar un punto dado. Por lo general $p > 1/2$, y nos interesa saber qué probabilidad de ganar un *game* resulta para el que saca. Los puntos de tenis, por algunas razones históricas olvidadas ahora, se llaman 0,15,30,40 (aún más raro, 0 se llama “love” en inglés, pero eso es una distorsión del francés para “huevo”). Si cambiamos estos por 0,1,2,3 respectivamente aquí, y si llamamos q_{ij} a la probabilidad de ganar el que saca, dada la cuenta i para el saque y j para la recepción, tenemos un sistema de ecuaciones: $q_{32} = p + (1-p)q_{33}$, $q_{23} = pq_{33}$, $p_{33} = pq_{32} + (1-p)q_{23}$ para empezar (la cuenta 3-3 se llama “iguales” y 3-2 y 2-3 se llaman ventaja saque y ventaja recepción respectivamente, porque para ganar el game hay que tener ventaja de dos puntos; además, aunque en la cancha distinguen entre “iguales” y “30 iguales”, entre “40-30” y “ventaja saque” y entre “30-40” y “ventaja recepción”, estos pares de puntos son equivalentes). De esas tres ecuaciones sacamos que la probabilidad de ganar para el saque cumple $q_{33} = p^2 + 2p(1-p)q_{33}$ o sea $q_{33} = p^2/(2p^2 - 2p + 1)$. De allí, salen q_{32} y q_{23} mediante las otras ecuaciones.

Y de allí tenemos ecuaciones $q_{ij} = pq_{(i+1)j} + (1-p)q_{i(j+1)}$ para todo par (i, j) tal que $i \neq 3$ y $j \neq 3$. Tenemos $q_{3j} = p + (1-p)q_{3(j+1)}$ con $j \in \{0, 1\}$ y $q_{i3} = pq_{(i+1)3}$ con $i \in \{0, 1\}$. Podemos continuar calculando $q_{22} = pq_{32} + (1-p)q_{23}$, $q_{31} = p + (1-p)q_{32}$ y $q_{13} = pq_{23}$. De allí podemos

calcular $q_{30} = p + (1-p)q_{31}$ y así sucesivamente, cada ronda de cálculos permitiendo otra ronda hasta tener q_{10} y q_{01} y de allí poder calcular q_{00} , la probabilidad que buscamos. Es 0,736.

Aquí mostramos los valores correspondientes, para varios valores de p :

p	.51	.52	.55	.60	.65	.70	.75	.80
q_{00}	.525	.550	.623	.736	.830	.901	.949	.978

Hay ciertos eventos de la ruleta que el jugador puede elegir con la ubicación de una sola pila de fichas en la tabla de la mesa. Por ejemplo, si la pone sobre “pares” gana otra pila igual que puso si la rueda da cualquier número par salvo 0 (o 00 en las Vegas, donde los famosos comerciantes Yanqui han puesto esa posibilidad en sus ruedas). La esperanza de su ganancia por peso apostado es $18/37 + 19/37(-1) = -1/37$ pesos, o sea una pérdida de 2,7 centavos ($-2/38$ pesos, o $-5,26$ centavos en Las Vegas). También puede jugar un pleno, es decir un solo número, que, si sale, le otorga \$35 por cada peso jugado, con una esperanza de $35(1/37) - 1(36/37) = -1/37$, igual que en el caso de los pares. Se puede jugar 1 hasta 18, la primera mitad, que gana igual que pares o nones, o la segunda mitad; también se puede jugar a la primera docena (o a la segunda, o a la tercera), ganando en cada caso 2 pesos por peso jugado, para una esperanza de $2(12/37) - 1(25/37) = -1/37$ también. Asimismo se puede jugar rojo o negro – hay 18 números rojos y 18 negros, y el 0 no es ni rojo ni negro, y paga un peso por cada peso jugado, igual que pares o nones. La regla de la casa es que por cada evento $\{e_1, \dots, e_k\}$, se paga cada pleno e_i como un pleno, si fueran todos plenos de verdad o un conjunto elegido por la tabla, así que la esperanza de la ganancia por peso del evento, si fuera un peso para cada uno de k plenos, es $(-k/37)/k = -1/37$. Si uno juega un pleno de un peso para cada número 0 hasta 36, pierde 36 de esos pesos y gana 35 por el que ganó: esperanza por peso jugado de $\$-1/37$.

El espacio de la moneda honesta es como los palos de la teoría de números, en que toda base axiomática tiene que conformarse a lo que sabemos intuitivamente de ese modelo.

Por ejemplo, preguntamos por la esperanza del número de tiros en el siguiente experimento: tirar la moneda hasta que salga cara. Para medir ésto, repetimos ese experimento N veces. Para proceder axiomáticamente, diríamos que estamos en el espacio de eventos de probabilidad $2^{-(n+1)}$ que consisten en $n \geq 0$ cruces seguidas por una cara, consistiendo de $n+1$ tiros. La esperanza es $\sum_{n=0}^{\infty} n2^{-(n+1)}$, que es una suma comparativamente fácil como van la generalidad de sumas infinitas, pero aún más fácil es escribir

$$E = \frac{1}{2} + \frac{1}{2}(1 + E)$$

que sale de considerar N repeticiones del experimento de tirar hasta que ocurra cara: La mitad terminan en el primer tiro y la otra mitad tendrían un promedio de 1 más el E que buscamos, y son $N/2$ experimentos, así que tendrían $(N/2)(1 + E)$ tiros, dando un total de

$$\frac{N}{2} + \frac{N}{2}(1 + E)$$

así que el número de tiros dividido el número de experimentos, el promedio de tiros por experimento, que esperamos que estime E , es lo que hemos escrito arriba para E .

En forma más directa, esperamos un tiro seguro, más con probabilidad de $1/2$ un número de tiros necesario para que aparezca una cara, que tiene la esperanza E : $E = 1 + \frac{1}{2}E$.

De eso, sale $E = 2$.

Si la probabilidad de cara es p , escribimos, por el mismo razonamiento, $E = 1 + (1-p)E$, y $E = 1/p$. Entonces, si se tira un dado hasta tirar 1, el número de tiros tiene esperanza 6.

El número de tiros esperado en un juego de pase inglés requiere un poco más de trabajo. Dado que ocurrió $s \in \{4, 5, 6, 8, 9, 10\}$, la esperanza del número adicional de tiros es, según el razonamiento que acabamos de hacer, $1/(p(s) + p(7))$. Entonces,

$$E = p(2) + p(3) + p(7) + p(11) + p(12) + \sum_{s \in \{4, 5, 6, 8, 9, 10\}} p(s) \left(1 + \frac{p(s)}{p(s) + p(7)}\right).$$

Con dados honestos da 3,376 hasta tres decimales.

Con dados cargados con la carga que consideramos antes, $p(1) = p(6) = 1/4$ y $p(d) = 1/8$ para $d \in \{2, 3, 4, 5\}$ con la cuantía para la suma que vimos antes, el número esperado de tiros disminuye a 2,990.

EJEMPLO 7.1. Un ejemplo mediático. En un programa de juegos de la televisión, el participante fue informado que detrás de una de tres puertas cerradas había un auto 0 kilómetros, y que detrás de cada una de las otras, una cabra. Le pedían elegir una de las tres puertas para recibir como premio lo que se encontraba atrás. Pero después de elegir su puerta, el maestro de ceremonias le abría otra puerta y se veía una cabra allí. Entonces el m.c. preguntaba al participante si quería cambiar su elección.

¿Conviene cambiar? ¿Conviene quedar con la elección original? ¿No importa; la probabilidad de ganar es la misma? Invitamos al lector a elegir, guiado por sentido común, antes de seguir. Le avisamos que este problema causó una gran controversia mediática, involucrando a expertos de la probabilidad y estadísticas. Cuando se terminó de usar ese juego, los productores del programa avisaron que convenía cambiar la elección, y fueron reprochados por un estadístico de renombre por mal informar al público.

Al elegir la puerta original, claro que hay una probabilidad de $1/3$ de haber elegido al auto. Supongamos que seguimos la estrategia de aceptar la oportunidad de cambiar. Si el auto está detrás de la puerta elegido, perdimos. Pero si el auto no está allí, ¿donde está? No está donde el m.c. abrió, porque se ve allí una cabra. Está entonces donde nos ofrecen la oportunidad de elegir. Es decir, esta estrategia gana si y sólo si el auto no está detrás de la puerta de la elección original. ¡Y la probabilidad de eso es $2/3$!

EJEMPLO 7.2. Una estafa. Un señor, con una manera quizás algo suave, ofrece a José este juego: el señor le da \$10, y José tiene que tirar una moneda hasta tener más cara que cruz, pagándole al señor 1 centavo por tiro. Entonces si le sale cara en el primer tiro José gana \$9,99. Si le sale XCC gana solo \$9,97. Si le sale XCXCXXCXXCCXCCXCC gana \$9,73. Y el señor le permite usar una moneda de su propio bolsillo si quiere.

Acabo de hacer una simulación de este juego con Maple, con el programa

```
juego := proc(p) c := 0: x := 0: while c <= x do
  if rand()/1012 < p then c := c + 1 else x:=x+1 fi od: x+c end;
```

100 ejecuciones de juego(1/2) dieron solamente dos que excedieron 1000 tiros, lo que necesita el señor para ganar. Indica solo 0,02 probabilidad de ganar él. Pero esos dos juegos ganados fueron de 437489 y 1229 tiros. El total de tiros en los 100 juegos fue 441236 que significa \$4412,36. El Señor pagó \$10 cada juego, total de \$1000; José perdió \$3412,36.

José fue a buscar una víctima para recuperar la pérdida. Encontró una, y en 100 juegos ganó 3 veces, con 18167, 23633 y 17379 tiros. El total de tiros en los 100 juegos fue 62594, así que José perdió \$374,06.

Vamos a calcular la esperanza E del número de tiros:

Tiene que cumplir $E = \frac{1}{2} + \frac{1}{2}(1 + 2E)$, o sea $E = 1 + \frac{1}{2}2E$ porque en la mitad de los juegos va a empezar con cruz, que lo obliga a ganar dos veces: una para recuperar el déficit del primer tiro, y otra para llegar a un superávit de 1, y la esperanza de eso es $2E$. Llegamos a la conclusión de que $E = E + 1$, y entonces que E es infinito.

La probabilidad de ganar el señor es muy pequeña, pero calcularla es muy engorrosa. Tres simulaciones más dan 1, 3 y 6 ganados, para un promedio de 3 en las 5 simulaciones que hemos hecho (el último tenía un juego de 2494575 tiros), entonces podemos confiar en que la probabilidad de ganar no difiere mucho de 0,03.

Aunque la esperanza del número de tiros es infinita, la probabilidad de terminar el juego es 1, porque tiene que cumplir $q = \frac{1}{2} + \frac{1}{2}q^2$, una ecuación cuadrática con una sola solución: $q = 1$.

Pero si p es la probabilidad de tirar cara con la moneda, con p no necesariamente igual a $1/2$, ese q tiene que cumplir

$$q = p + (1 - p)q^2,$$

una ecuación cuadrática con dos soluciones: $q = 1$ o $q = \frac{p}{1-p}$. Con $p > 1/2$ rechazo el segundo porque da $q > 1$ que no es una probabilidad, y me quedo con $q = 1$. ¿Qué hago con $p < 1/2$? ¿Si $p = 0,000001$ por ejemplo, ¿es razonable suponer que tarde o temprano se va a tirar más caras que cruces? En un millón de tiros espero 999999 cruces y 1 cara. No solamente es posible, sino probable que el número de cruces sea menor que 10 en un millón de tiros, ¿y puedo estar seguro de tarde o temprano recuperar ese déficit? Si no, tenemos que conformarnos con que la probabilidad de terminar el juego es $\frac{p}{1-p}$. Pero éso no es una demostración formal, que puede ser muy tediosa de hacer, aunque en el próximo capítulo veremos herramientas para manejar esta cuestión.

Ejercicios 7.1

1. (2) Si hay tres pares distintos de medias en una bolsa, y se sacan medias una por una hasta tener un par, ¿Cuál es la probabilidad de terminar sacando solo dos medias? ¿tres? ¿cuatro? Entonces, ¿cuál es la esperanza del número de medias sacadas?
2. (2) Supongamos que en el problema mediático, el participante sabe que el locutor, cuando el participante elige la puerta con el auto, elige para abrir la más izquierda de las otras dos puertas. ¿Hay una estrategia que puede aprovechar esto para tener una probabilidad mayor de $2/3$ de ganar el auto?
3. (3) Ud. está observando un juego de pase inglés. Un jugador tira 6 en su primer tiro. Un señor desconocido a su lado ofrece a apostarle a Ud. 3 pesos contra 2 que ese jugador pierda; es decir, si gana ese jugador el señor le paga 3 pesos; si pierde, Ud. le paga a él 2 pesos. ¿Es buena apuesta para Ud. si los dados son honestos? ¿Es buena apuesta para Ud. si los dados están cargados tal que la probabilidad de 1 y la de 6 es $1/4$, y la probabilidad de 2,3,4 o 5 es $1/8$ cada uno?
4. (4) En este y los siguientes cuatro ejercicios el problema central es sobre el póker de descarte, en que después de la primera ronda, en que se hacen apuestas sobre las manos de 5 cartas que tiene cada jugador, cada uno puede pedir hasta 3 cartas nuevas para reemplazar tres cartas descartadas por él antes de ver las nuevas. Un dilema típico es de una mano con cartas de número 5,6,7,8. El jugador puede descartar uno de los 8 con la esperanza de que la nueva carta sea o 4 o 9 para completar una escalera, una mano muy valiosa; o puede guardar los dos 8 esperando pescar otro 8, para tener un trio, o aún dos de 8, para un “póker”, u otro 8 y dos de 9 para tener un full, u otro par para tener dos pares. Por lo menos se queda con el par de 8, que vale algo. El cálculo de las probabilidades de todas estas posibilidades es solo parte del problema. Más problemático es determinar algún valor numérico para cada posibilidad, para poder optimizar la esperanza del valor de la mano que resulta de aplicar la decisión. Las manos de póker son
 - Escalera de color. Sucesión de 5 cartas consecutivas del mismo palo.
 - Póker. Cuatro cartas del mismo número.
 - Full (de *full house*). Tres de un número y dos de otro, por ejemplo 99933.
 - Color. Cinco cartas del mismo palo.
 - Escalera. Cinco cartas de números consecutivos sin ser una escalera de color, por ejemplo 4,5,6,7,8. Aquí el as (el 1) puede considerarse o alto o bajo; no solo as,2,3,4,5 forma una escalera, sino 10,J,Q,K,as también.
 - Trio: tres del mismo número, por ejemplo 4,4,4,8,10.
 - Dos pares. Cartas de números m,m,n,n,p para tres números distintos m,n,p.
 - Un solo par. Cartas de números m,m,n,p,q para cuatro números distintos m,n,p,q.
 Si podemos asignar un valor a cada uno de estos, podemos calcular la esperanza del valor que produce cada estrategia. Notamos que no pensamos asignar valores distintos a distintas escaleras, aunque de vez en cuando dos escaleras chocan y el más grande número gana, pero tal choque es demasiado infrecuente para tratar de incluir en el análisis. (Así que aún a la afamada escalera real no se asigna mayor valor que cualquier

otra escalera de color.) Claro es que la mano más rara va a tener más valor. Eso sugiere inmediatamente como posibilidad $1/p$ donde p es probabilidad de que salga esa mano eligiendo cinco cartas al azar del manójo de 52 cartas. El número de manos es el número de conjuntos de 5 cartas existen entre 52: sabemos que esto es $\binom{52}{5}$. Para la escalera de color, hay una escalera de 10 empezando en cada número de 1 a 10 (hay 13 cartas de cada uno de los 4 palos, y el as puede formar un 1,2,3,4,5 o un 10,J,Q,K,as.) Así que p es el número de tales escaleras de color dividido $\binom{52}{5}$, y el valor $1/p$ sería $\binom{52}{5}$ dividido por ese número.

Calcule los valores de todos las ocho manos. Sugerencia: lo que falta es saber el número de representantes de cada mano. El número de manos póker se calcula observando que con cada uno de los 13 números hay un póker que contiene las cuatro cartas de ese número acompañadas por uno de las 48 cartas restantes. Para el full cada mano es mmmnn, de los cuales hay 13×12 pares m,n y con cada uno hay que variar el palo omitido en mmm y los dos palos de nn. Para las escaleras hay 10 sucesiones numéricas y para cada uno hay que asignar una sucesión de 5 palos, y de ese número hay que remover las escaleras de color.

5. (4) Para la mano de 5,6,7,8,8 que no tiene 5,6,7,8 del mismo palo, calcule el valor esperado de las dos estrategias de guardar 5,6,7,8 y pedir una nueva, o guardar las dos 8 y pedir tres nuevas. Sugerencia: para la estrategia de pedir una sola carta nueva, hay ocho cartas en el resto del manójo que pueden completar la escalera. El otro es más complicado y es difícil evitar errores de cálculo. Pero se puede resolver con Maple si definimos las variables $esp88$, $esp888$, $esp888n$, $esp88n$, $esp88nn$, $esp88nms$ donde $esp88x..x$ significa la esperanza dado que se han recibido las cartas $x..x$ de las tres pedidas, y escribiendo las ecuaciones gobernando estas. Entonces $esp88$ va a ser la esperanza que buscamos. Una de las ecuaciones es $esp88 = (2/47)esp888 + (45/47)esp88n$, porque la primera carta es o una 8, de las cuales hay dos entre las 47 cartas distintas que el 5,6,7,8,8 dadas al jugador, o una no-8, y hay 45 de esos allí. Otra ecuación resulta suponiendo que la primera carta nueva no es una 8, y considerando que la segunda carta puede ser una 8, u otra n o una carta cuyo número no es ni ocho ni n, y recordando que la segunda carta nueva viene al azar de un conjunto de 46, así sacamos la ecuación $esp88n = (2/46)esp888n + (3/46)esp88nn + (42/46)esp88nm$. Un ejemplo más: $esp888 = (1/45)\text{valor de póker} + (44/45)esp888n$.
6. (4) Haga el mismo cálculo para la mano 5,6,7,8 de corazones y otra 8. (El cálculo de $esp88$ es el mismo que en el problema anterior.)
7. (4) Es posible objetar al valor $1/p$ porque da una razón de decenas de miles entre escalera de color y un par. Diga qué pasa con los dos ejercicios anteriores si usamos valores $\sqrt{1/p}$.
8. (2) Analice la mano 4,5,7,8,8, para los dos casos en que hay o no hay 4,5,7,8 del mismo palo, con valores $1/p$ y luego con $\sqrt{1/p}$. Note que la decisión de guardar las dos 8 y pedir tres nuevas tiene los mismos resultados que antes.

2. Autómatas estocásticos

Varios de los ejemplos que hemos visto pueden mirarse desde el punto de vista de *autómatas estocásticos*.

DEFINICIÓN 7.3. Un *autómata estocástico* es (K, π, q_0) , donde K es un conjunto a lo sumo numerable de estados, q_0 es el estado inicial y π es una función de $K \times K$ a $[0, 1]$ tal que para cada $q \in K$, $\sum_{r \in K} \pi(q, r) = 1$.

Es decir, π es una función de probabilidad en que $\pi(q, r)$ es la probabilidad de que el próximo estado sea r , dado que el estado es q . Por ejemplo, para el partido de tenis, π se define así:

- $\pi(q_{ij}, q_{i+1,j}) = p$ si $i < 3$.
- $\pi(q_{ij}, q_{i,j+1}) = 1 - p$ si $j < 3$.

- $\pi(q_{3,j}, G) = p$ si $j < 3$; $\pi(q_{i,3}, P) = 1 - p$ si $i < 3$
- $\pi(q_{3,3}, q_{3,2}) = p$; $\pi(q_{3,3}, q_{2,3}) = 1 - p$
- $\pi(G, G) = 1$, $\pi(P, P) = 1$

donde cualquier valor de π no mencionado allí es 0. G y P son los estados de “ganar” y “perder” y sus transiciones son a ellos mismos. Esta construcción hubiera sido engorrosa cuando estábamos haciendo el problema, pero sirve para ilustrar el concepto del autómata estocástico.

También admitimos un conjunto infinito, pero numerable, de estados. Eso representa el juego del señor que quería que juguemos con esperanza $-\infty$. Los estados son el conjunto de los enteros negativos junto con $\{0, 1\}$, siendo 0 el estado inicial y 1 el único estado final. Las transiciones son $\delta(i, i+1) = p$ y $\delta(i, i-1) = 1 - p$ para $i \leq 0$, y $\delta(1, 1) = 1$ y $\delta(1, 0) = 0$.

Entre las cosas que interesan de estas máquinas, están la esperanza del número de transiciones para llegar del estado p al estado q , con atención especial al caso $p = q$, la probabilidad de poder llegar a q desde p , la probabilidad de que, partiendo de p , el estado q ocurra antes del r .

La esperanza del número de transiciones desde p a q , escrito $T(p, q)$ es 0 si $p = q$ y sino

$$T(p, q) = 1 + \sum_{r \in K} \pi(p, r) T(r, q).$$

¿Y qué pasa si no hay camino de transiciones con probabilidad positiva entre p y q , o si hay entre p y r pero no entre r y q para algún r ? Ejemplo:

	p	q	r	s
p	1/3	1/3	1/3	0
q	2/3	1/3	0	0
r	0	0	1/4	3/4
s	0	0	1	0

Según la definición, $T(q, p) = 1 + \frac{1}{3}T(q, p) + \frac{2}{3}T(p, p)$, o sea $T(q, p)(1 - \frac{1}{3}) = 1$; $T(q, p) = \frac{3}{2}$. En forma semejante calculamos $T(r, s) = \frac{4}{3}$ y $T(s, r) = 1$. Pero cuando queremos calcular $T(r, q)$, encontramos con $T(r, q) = 1 + \frac{1}{4}T(r, q) + \frac{3}{4}T(s, q)$. Pero $T(s, q) = 1 + T(r, q)$ de donde tenemos que admitir $T(r, q)(1 - \frac{1}{4} - \frac{3}{4}) = \frac{7}{4}$, o $0 = \frac{7}{4}$, y este absurdo surge de suponer que esa esperanza existe cuando no hay ninguna esperanza de llegar a q desde r .

Para otro ejemplo simple, considere el autómata para aceptar múltiplos de 3 escritos en binario. Si decimos que entradas 0 y 1 ocurren con probabilidades p y q respectivamente, surge el autómata estocástico

	0	1	2
0	p	$1 - p$	0
1	$1 - p$	0	p
2	0	p	$1 - p$

Calculemos $T(1, 0)$. Tenemos $T(1, 0) = 1 + pT(0, 0) + (1 - p)T(2, 0) = 1 + (1 - p)T(2, 0)$ pues $T(0, 0) = 0$. $T(2, 0) = 1 + pT(1, 0) + (1 - p)T(2, 0)$ de donde sacamos que $(1 - (1 - p))T(2, 0) = 1 + pT(1, 0)$ y de allí que $T(2, 0) = \frac{1}{p} + T(1, 0)$. Enchufando esto en la primera ecuación da $T(1, 0) = 1 + p(\frac{1}{p} + T(1, 0))$ de donde despejando $T(1, 0)$ da $T(1, 0) = \frac{2}{1-p}$.

A veces es de interés saber la frecuencia con que ocurre un estado en una larga sucesión de transiciones. Usualmente esto ocurre con un autómata en que para cualquier estados p, q la probabilidad dado estado p de nunca entrar en estado q es 0. Esto es equivalente a decir que para cualquier estados p, q hay una sucesión $r_1 \dots r_n$ con $r_1, r_n = p, q$ y $\pi(r_i, r_{i+1}) > 0$ para $1 \leq i < n$ (ejercicio 6). En el próximo capítulo, que trata de cadenas de Markov, veremos más sobre este tema.

Ahora podemos calcular la esperanza de la longitud de la primera sucesión no nula de transiciones que, empezando con q , produce q de nuevo, y lo llamamos $T'(q)$. $1/T'(q)$ es la frecuencia de q . Lo que queremos decir con esto es, puesto que la probabilidad de alguna vez volver a q de nuevo cada vez que ocurre q , podemos mirar toda la historia como bucles que

empiezan y terminan en q , salvo un segmento inicial que va de algún estado inicial a la primera ocurrencia de q y significa siempre menos mientras pasa el tiempo y se producen más y más bucles. Después de N bucles ocurren muy cerca de $NT'(q) + d$ estados donde d es el mencionado segmento inicial que produce la primera ocurrencia de q , y la proporción de estos estados que son iguales a q es muy cerca de $(NT'(q) + d)/N$, que claramente tiene límite $T'(q)$. Cuando N es tan grande que el efecto de d practicamente desvanece, decimos que el sistema es en un *estado estacionario*.

Enfatizamos que no tenemos una definición formal del término *estado estacionario*. Refiere más bien a un estado subjetivo del observador: que está el en un estado de ignorancia sobre la historia del sistema, solamente sabe como funciona estocásticamente, y que entonces tiene que estimar el tiempo hasta la próxima ocurrencia de q en $T'(q)/2$ (la mitad de un bucle). Claro, si puede espiar el estado del sistema, ese cálculo cambiaría, por eso decimos que es cuestión del estado de ignorancia del observador.

$T'(q)$ no es $T(q, q)$. La sucesión nula no *produce* q ; no produce nada, no hubo ninguna transición. $T'(q)$ es la esperanza del largo de la sucesión no nula de transiciones que produce q de nuevo:

$$T'(q) = 1 + \sum_{p \neq q} \pi(q, p)T(p, q).$$

Aplicando ésto al estado 0 del autómata para múltiplos de 3,

$$T'(0) = 1 + pT(0, 0) + (1 - p)T(1, 0) = 3$$

independientemente de p ! (Si el autómata fuera el de módulo 4 la probabilidad del estado 0 sería la probabilidad que las últimas dos entradas sean 0, o p^2 , la probabilidad del estado 1 sería $p(1 - p)$, lo mismo para estado 2, y $(1 - p)^2$ para el estado 3. Ver ejercicios.)

Esto no sorprende si $p = 1/2$. Pero si $p = 0,99$ entonces $T(1, 0) = 200$ y $T(2, 0) = 1/0,99 + 200$, así que una vez salido del estado 0, va a pasar un rato antes de volver. Sin embargo, la frecuencia del estado 0 es $1/3$.

Si tomamos el autómata para múltiplos de 4 escritos en binarios, y consideramos 0 y 1 llegando con probabilidad p y $1 - p$, sale el siguiente autómata estocástico (un múltiplo de 4 es una cadena que termina en 00).

	00	1	0
00	p	$1 - p$	0
1	0	$1 - p$	p
0	p	$1 - p$	0

Resolviendo las ecuaciones para $T(q, r)$ y $T'(00)$ sale $T'(00) = \frac{1}{p^2}$, así que 00 tiene frecuencia p^2 . Para $p = \frac{1}{2}$ esto es $\frac{1}{4}$, de acuerdo con lo que esperamos para múltiplos de 4 sin sesgo en las entradas.

Supongamos que alguien nos propone este juego: Tiramos la moneda hasta que salga o CCX o CXC, ganamos si sale CCX primero, y perdimos si CXC sale primero. Calculamos la probabilidad de ganar con CCX mediante un autómata con estados ϵ , C, CX, CC, CCX, CXC, con las transiciones gobernadas por la probabilidad p de tirar cara:

	ϵ	C	CC	CX	CCX	CXC
ϵ	$1 - p$	p	0	0	0	0
C	0	0	p	$1 - p$	0	0
CC	0	0	p	0	$1 - p$	0
CX	$1 - p$	0	0	0	0	p
CCX	0	0	0	0	1	0
CXC	0	0	0	0	0	1

Sea P_q la probabilidad de salir CCX dado el estado q . Es claro que $P_{CC} = 1$ porque la máquina va a seguir en ese estado mientras llegan caras, y la primera cruz produce CCX. Aunque P_ϵ es lo que nos interesa, es claro que no pasa nada hasta la primera cara, así que

$P_\epsilon = P_C$. Ahora tenemos dos ecuaciones $P_C = p + (1 - p)P_{CX}$ y $P_{CX} = (1 - p)P_C$, de donde sale $P_C = \frac{1}{2-p}$.

Con $p = \frac{1}{2}$ esto es $2/3$, así que tenemos una buena apuesta. Además, sigue siéndola aunque la moneda está cargada, con cualquier $p \neq 1/2$, porque con $0 < p < 1$, $\frac{1}{2-p} > 1/2$.

2.1. Un sistema de colas. Supongamos que clientes llegan para ser atendidos y que puede haber una cola de espera. Supongamos que cuando se está atendiendo a un cliente, la probabilidad es p que llegue un nuevo cliente antes de terminar con el cliente actual. Ésta está representada por el autómata con estados los números naturales (representando el número de clientes presente, uno que se está atendiendo si no es 0, y el resto esperando en la cola de espera). Tenemos que suponer que $p < 1/2$, porque si no la cola crece sin límite. Las transiciones son $\pi(0, 1) = 1$, $\pi(i, i + 1) = p$ y $\pi(i, i - 1) = 1 - p$ si $i > 0$, y $\pi(i, j) = 0$ en cualquier otro caso.

Preguntamos cuál es el número de transiciones necesarios para ir del estado $i > 0$ al estado $i - 1$. Llamémosle T . Tiene que cumplirse $T = 1 + 2pT$, porque seguro que hay uno y la probabilidad es p que hay más, que llega un cliente para levantar el estado a $i + 1$, y en ese caso la esperanza de lo que queda es $2T$ porque hay que bajar el estado una vez para volver a i y otra vez para llegar a $i - 1$. Sale $T = 1/(1 - 2p)$.

Agregamos duración a este sistema. Digamos que los clientes llegan a λ por minuto y que esa distribución es exponencial, es decir que la probabilidad de que la próxima llegada demore más que t es $e^{-\lambda t}$. También suponemos que la atención al cliente tiene la distribución exponencial, es decir, la probabilidad de que el servicio dure más que t es $e^{-\mu t}$ (de donde es claro que no podemos admitir un $p = 1/2$ – teniendo un cliente en el sistema, la esperanza del tiempo hasta que ponga ocioso el sistema es infinito, exactamente como esperar más caras que cruz de la moneda).

La distribución exponencial es fundamental porque es la única que es sin memoria. Por ejemplo, cuando estamos esperando que llegue el próximo ómnibus, cuanto más tiempo esperamos, menos tiempo adicional esperamos, diciendo “estamos aquí desde hace diez minutos, ya tiene que llegar”. Pero si la distribución es exponencial, la perspectiva del futuro no cambia por más tiempo que pase. Probablemente no sirve para el omnibus, pero para la llegada de alguien al sistema sí porque es el resultado de muchas decisiones independientes. Y hay casos donde puede servir para el tiempo de servicio: por ejemplo la duración de una litigación o el tiempo hasta que devuelvan un libro prestado. Aquí lo usamos para el servicio porque da problemas más tratables.

El parámetro μ representa la tasa de servicio en clientes por minuto, o sea el ritmo con que trabaja el servidor.

Cuando hay un cliente en servicio, la probabilidad de que llegue un nuevo cliente antes de terminar el servicio es $p = \frac{\lambda}{\lambda + \mu}$ porque la tasa combinada de servicios y llegadas es $\lambda + \mu$, y la junta de dos distribuciones exponenciales es exponencial porque la probabilidad de ocurrir ninguno en tiempo t es el producto de las probabilidades de cada uno de no ocurrir en tiempo t o sea $e^{-\mu t} e^{-\lambda t} = e^{-(\mu + \lambda)t}$. Y $1 - p = \frac{\mu}{\lambda + \mu}$.

Preguntamos ahora cuál es la esperanza del tiempo en el sistema por cliente. Hagamos un truco. Imaginamos que atendemos a los clientes dando prioridad con interrupción al cliente recién llegado. Es decir, al llegar un cliente, si hay otro cliente en servicio se lo interrumpe y se pone en servicio el nuevo cliente, reanudando el servicio interrumpido cuando el nuevo cliente termina. Por supuesto, el nuevo cliente es sujeto al mismo tratamiento. El tiempo medio en el sistema por cliente se obtiene tomando dando prioridad al recientemente llegado, hasta el punto de interrumpir al cliente en servicio para reanudar su servicio después, cuando se va este nuevo cliente, que a su vez puede ser interrumpido. Por la falta de memoria de estas distribuciones, esto no cambia nada el sistema. Es decir, sigue siendo exponencial el tiempo hasta terminar el actual servicio, y la demora hasta la próxima llegada. Entonces sea T el tiempo esperado en el sistema por cliente. Mientras el cliente está en el sistema, la esperanza del número de llegadas es λT , y éstos tienen servicio con media $1/\mu$. Junto con el $1/\mu$ del cliente en cuestión, eso da

$T = 1/\mu(1 + \lambda T)$, con solución

$$T = \frac{1}{\mu - \lambda}$$

El tiempo de espera es el tiempo en el sistema menos el tiempo de servicio, y entonces el tiempo de espera es

$$T_E = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)}.$$

La esperanza del número de clientes en el sistema es $T_E/(1/\mu) = \mu T_E = \lambda/(\mu - \lambda)$.

2.2. La desigualdad de Chebychev. La *varianza* de una variable aleatoria X es la esperanza del cuadrado de la diferencia entre X y la esperanza $E(X)$:

$$V(X) = E((X - E(X))^2)$$

La varianza habla de la cantidad de dispersión de X . Si tengo un X de distribución desconocida, por ejemplo un dado que sospecho de ser cargado, si hay mucha varianza es más difícil de estimar la distribución partiendo de unas observaciones.

La desigualdad de Chebychev nos puede ayudar con esto:

$$\text{probabilidad } |X - E(X)| \geq \alpha \leq V(X)/\alpha^2$$

Es fácil de demostrar:

$$\begin{aligned} V(X) &= \sum_{e \in E} ((X(e) - E(X))^2 p(e)) \geq \sum_{|X(e) - E(X)| \geq \alpha} (X(e) - E(X))^2 p(e) \\ &= \sum_{(X(e) - E(X))^2 \geq \alpha^2} (X(e) - E(X))^2 p(e) \\ &\geq \sum_{(X(e) - E(X))^2 \geq \alpha^2} \alpha^2 p(e) = \alpha^2 \text{probabilidad}(|X - E(X)| \geq \alpha). \end{aligned}$$

Quisás el significado de esto se destaca más recordando que la *desviación* de X se define como la raíz cuadrada de la varianza V , en breve $\sigma^2 = V$ (recuerde que V es la esperanza de un cuadrado). Así podemos escribir la desigualdad de Chebychev en la forma

$$\text{probabilidad}((X - E(X))^2 \geq k\sigma^2) \leq \frac{1}{k}.$$

En el caso que X tiene la distribución normal, o una que se aproxima a la normal como el número de caras en n tiros, nos dice que por lo menos $3/4$ de los valores quedan entre $\pm 2\sigma$, por lo menos $8/9$ entre $\pm 3\sigma$, por lo menos $15/16$ entre $\pm 4\sigma$, etc. En la tabla de la distribución normal vemos las cotas más exigentes de 0,9544, 0,9973 y 0,9999. La ventaja de la desigualdad de Chebychev es que no requiere más conocimientos de la distribución que su media y varianza.

Tiras una moneda sospechosa 100 veces y sale 70 veces cara. ¿Que dice Chebychev de la probabilidad de eso para una moneda honesta?

Definimos $X(\text{cara}) = 1$ y $X(\text{cruz}) = 0$. Con una moneda honesta, $E(X) = 1/2$.

$$V(X) = \frac{1}{2}((1 - \frac{1}{2})^2) + \frac{1}{2}((0 - \frac{1}{2})^2) = \frac{1}{4}$$

Ahora nos interesa la suma de 100 muestras independientes de X . Lo llamamos Y . Es un teorema fácil de demostrar que cuando se trata de variables independientes, la varianza de la suma es la suma de las varianzas, así que $V(Y) = 100V(X) = 25$. Con 70 caras tenemos $Y = 70$. $|Y - E(Y)| = 20$. Según Chebychev, probabilidad $((Y - E(Y))^2 \geq 400) \leq 25/400 = 1/16$.

Mmmmm. No confiaría en esa moneda.

Ejercicios 7.2

1. (1) Dos equipos de básquet juegan una serie en que el ganador es el primero en ganar 3 partidos. Si cada uno gana con probabilidad $1/2$, ¿cuál es la probabilidad que la serie no se decide hasta el quinto partido?
2. (2) Si en la serie del ejercicio anterior, cada equipo gana con probabilidad $0,6$ jugando en su propia cancha, y los partidos alternan entre la cancha del uno y del otro, ¿cuál es la probabilidad de ganar del equipo en cuya cancha la serie empieza?
3. (3) ¿Cuál es la probabilidad exacta de que una cadena binaria de n dígitos represente un número que no es divisible ni por 2 ni por 3? Sugerencia: Construya el autómata finito, que tendrá a lo sumo 6 estados. Para cada estado q sea $N_q(n)$ el número de cadenas c de largo n tal que $\delta(q, c) \in F$. Plantee el conjunto de ecuaciones para los $N_q(n)$ en términos de los $n_p(n-1)$ y conviértalo en un sistema de ecuaciones para los correspondientes conjuntos generadoras. La probabilidad del estado final para n dígitos binarios al azar es el número de cadenas c de longitud n tal que $\delta(q_0, c)$ es final, dividido 2^n . De eso nos interesa el límite.
4. (3) ¿Cuál es la probabilidad que una cadena binaria engendrada por un proceso que produce 0 con probabilidad p (y por tanto produce 1 con probabilidad $1-p$) representa en binario un múltiplo de 6? (Sugerencia: considere un automata con estados 0, 1, 2, 3, 4, 5 que dado la representación de $n \equiv_6 i$ en el estado 0 termina en el estado i . Tenga cuidado de notar cuales estados son equivalentes a otros para el propósito de este ejercicio, así reduciendo el número de ecuaciones.)
5. (2) Un dado se tira hasta observa o 1,1,2 o 1,2,1. ¿Cuál es la probabilidad que 1,1,2 ocurre antes de 1,2,1? (Parece similar a CCX vs. CXC, pero hay diferencias importantes.)
6. (2) Demuestre, sin aplicar la teoría Markov, la aserción en el texto que la probabilidad es 0 de nunca ocurrir un estado q en una sucesión infinita de un autómata estocástica es equivalente a decir que para cualquier estado p hay una sucesión no nula de transiciones de p a q con probabilidad no nulo.
7. (2) En un sistema de cola de un servidor los clientes llegan a la tasa media de λ por minuto y el servidor trabaja a la tasa media de μ clientes por minuto cuando no está ocioso (es decir, el tiempo medio de servicio es $1/\mu$ minutos.) ¿Cuál es la fracción del tiempo que el servidor está ocioso? (No importa la distribución. En tiempo T calcule la esperanza del número de clientes que llegan, y la del tiempo de servicio que ocupan.)
8. (2) Demuestre si a un sistema de una sola cola los clientes llegan a la tasa λ y servidos a la tasa σ (el tiempo medio de servicio es $1/\sigma$), entonces siendo \bar{Q} el tamaño medio de la cola y \bar{T}_s el tiempo medio en el sistema por cliente,

$$\bar{Q} = \lambda \bar{T}_s.$$

Sugerencia: Si observamos el sistema durante un tiempo τ que empieza y termina con cola 0, y medimos $\hat{Q} = \int_0^\tau Q(t)dt$ entonces \hat{Q}/τ es la cola promedia y tiene esperanza \bar{Q} , y siendo c el número de clientes llegado al sistema, \hat{Q}/c es el promedio del tiempo en el sistema por cliente y tiene esperanza \bar{T}_s .

9. (3) Un sistema de una cola funciona así. Cada segundo hay una probabilidad p de que llegue un cliente. Si hay clientes en el sistema (se supone que uno está recibiendo servicio) y no llega un cliente, entonces sale el que recibe servicio. Calcule \bar{T}_s y \bar{Q} . Sugerencia. Para calcular \bar{T}_s , demuestre que no depende de la regla para elegir cliente para entrar en servicio, como en el ejemplo de las distribuciones exponenciales de llegadas y servicios. Entonces imponga la regla de servicio inmediatamente al entrar con interrupción del cliente en servicio si hay, a reanudar cuando este nuevo cliente sale. Para \bar{Q} aplique el problema anterior. Aquí λ es p y σ es $1-p$. \hat{Q} del experimento sería $\sum_{i=1}^\tau Q(i)$.
10. (4) Demuestre que una distribución que es sin memoria es exponencial. (Suponga que el tiempo entre llamadas por un cierto teléfono es sin memoria. Sea u la probabilidad que la próxima llamada demora más que un minuto. Si de hecho demora más que un minuto, ya que es sin memoria, la probabilidad es u que demora otro minuto. Así que

la probabilidad de demorar 2 minutos es u^2 . Si de hecho demora 2 minutos, ya que es sin memoria la probabilidad de demorar 3 minutos es u^3 . Sea $p(t)$, t real positivo, la probabilidad de demorar más que t minutos. Siguiendo las observaciones para $t=2,3$, demostrar por inducción matemática que $p(t) = u^t$ para t natural. Ahora notar que por ser sin memoria, $p(1/2)^2 = p(1)$, es decir que $p(1/2) = u^{1/2}$. Siguiendo por inducción demuestre que $p(1/n) = u^{1/n}$. Ahora atacar el $p(m/n)$ y aplicar el teorema que si dos funciones continuas son iguales en todos los racionales son iguales para todos los reales.)

11. (2) Se sospecha que un dado da demasiado veces 6. Se decida rechazar el dado si en una prueba de 100 tiros da probabilidad menos de 0,1 de salir con el número de 6 visto en esa prueba, según la desigualdad de Chebychev. Cuáles son los números máximo y mínimo de 6 que puede salir en la prueba para no ser rechazado?

LA INCOMPLETITUD DE LA ARITMÉTICA

1. Introducción

Hacia el fin del siglo 19 se sentía que la matemática estaba al borde de producir una “teoría de todo”. Esto se puede interpretar como un lenguaje formal capaz de expresar cualquier proposición en cualquier ramo de la matemática. Muchos matemáticos, notablemente el gran David Hilbert, creían percibir una necesidad de reforzar la base de la matemática en vista de las trampas planteadas contra la intuición por los conceptos nuevos, como la teoría de conjuntos de Cantor. Cantor había observado que no deberíamos hablar de un conjunto de todos los conjuntos, porque, sea B ese conjunto, y sea B' el conjunto de todos los subconjuntos de B . Cada miembro de B' es un conjunto, por tanto $B' \subset B$. Pero Cantor había demostrado que el conjunto de todos los subconjuntos de un conjunto C tiene cardinalidad mayor que el de C , por tanto B' tiene mayor cardinalidad que B , pero siendo subconjunto de B , tiene cardinalidad menor o igual que la de B , una contradicción.

Ahora, en la teoría de conjuntos, si sabemos una propiedad, o *predicado* $P(x)$ que los objetos matemáticos pueden tener o no, podemos asegurar el conjunto de todos los objetos que tengan la propiedad P : $\{x : P(x)\}$. Pero aquel B es $\{x : x \text{ es un conjunto}\}$. Entonces, por lo menos cuando estamos haciendo la teoría de conjuntos, no debemos mencionar la propiedad de ser conjunto. Esto es contraintuitivo: ningún objeto matemático que yo he percibido ofreció dificultad en saber si era conjunto o no.

Bertrand Russell, otro destacado en la búsqueda de la teoría universal matemática, descubrió una paradoja peor. Considere el predicado $x \in y$. Lo usamos continuamente. Ahora el B , si existiera, siendo un conjunto, sería miembro de sí mismo. Satisface el predicado $x \in x$. Todos los conjuntos con que hemos tratado nosotros, exceptuando B que estamos de acuerdo que no puede existir, satisface $x \notin x$. Entonces sea

$$C = \{x : x \notin x\}$$

y preguntamos ¿ $C \in C$? Pero si $C \in C$ entonces claro que no cumple el predicado $x \notin x$ que define C . Entonces C es rechazado como miembro de C : $C \notin C$. Pero si $C \notin C$ entonces sí cumple el predicado que define C , y tiene que ser aceptado como miembro de C . Hemos demostrado

$$C \in C \leftrightarrow C \notin C.$$

Entonces muchos, especialmente Hilbert, se preguntaban ¿cuáles son los axiomas de la teoría de conjuntos? ¿Como reconocemos una demostración válida? Fue Hilbert primero en enunciar explícitamente la *necesidad* de poder reconocer una demostración válida sin recurrir a nada de ingenuidad ni intuición: no usó la palabra *computable* sólo porque no se la conocía todavía, faltaba unos 30 años. Nosotros ahora diríamos *un conjunto de cadenas sobre un cierto alfabeto finito*. Y los teoremas son ciertas cadenas terminales de demostraciones.

Entonces, Hilbert dijo que primero había que definir con absoluta precisión el conjunto de cadenas admitidas como demostraciones. Una vez hecho esto, se podía poner manos a la obra de demostrar que no había una proposición P tal que había una demostración de P y otra de la negación, $\sim P$, de P . Es decir, una demostración de la *consistencia* del sistema. En 1900 este problema era el segundo de 23 problemas importantes presentados por él al Congreso Internacional de Matemáticos. En 1905 presentó un bosquejo de una posible solución, pero

tenía defectos. Su acercamiento era algebraico y no facilitaba el trabajo, y se dirigió de nuevo en hacer la matemática en lugar de justificarla.

Mientras tanto Russell y A.N. Whitehead estaban trabajando duro en el problema, que por fin publicaron en el monumental *Principia Mathematica*, tres volúmenes. No terminaron con el problema pero el nuevo acercamiento que presentaron atrajo la atención a Hilbert, y él volvió al problema. Entre 1920 y 1928 pudo formular una definición de *demostración* satisfactoria a todos en su esencia, con la excepción de reemplazar axiomas sobre cuantificadores con un axioma sobre una función ϵ que creía que iba a facilitar la demostración de consistencia. Su colaborador Ackermann publicó lo que decía era tal demostración pero von Neumann descubrió un error en eso, así que quedó sin mostrar la utilidad de esa función ϵ . La definición de demostración que damos aquí es en lo esencial lo que propuso Hilbert, exceptuando el remplazo del axioma sobre ϵ con los axiomas sobre \forall implicados por ése.

Kurt Gödel en 1931 cambió la panorama de estos intentos mostrando que con una gramática muy sencilla, que engendraba cualquier proposición de la teoría de primer orden (sin conjuntos) de números, había proposiciones que con los axiomas de Peano no se podían ni demostrar ni negar: había proposiciones *indecidibles*. Más aún, demostró que para cualquier sistema que

- es consistente (no se puede deducir una contradicción),
- es suficientemente fuerte para demostrar los axiomas de Peano,
- tiene un conjunto computable de axiomas,

hay proposiciones indecidibles.

Ya que una proposición sin variable libre tiene que ser o verdadera o falsa, eso quiere decir que hay una proposición que es verdadera pero no demostrable. Y bueno, agregamos ésa a los axiomas – los axiomas del nuevo sistema siguen siendo computables porque hemos agregado un solo axioma, y sigue consistente y suficientemente fuerte para demostrar los axiomas de Peano. Por el teorema de Gödel hay otra proposición no decidible en este nuevo sistema. No solamente eso, sino que la proposición verdadera y no demostrable es computable como función del sistema, que nos permite hablar del sistema que incluye todos los nuevos axiomas que surgen así, y por el mismo teorema de Gödel *ese* sistema tiene proposición indecidible. Hasta cuál número ordinal puede seguir éso no quiero arriesgar mi sanidad mental especulando. Pero tenemos que concluir:

No se puede capturar toda la verdad sobre números naturales en un sistema formal.

De eso le vamos a tratar de convencer.

2. Demostraciones

Primero tenemos que definir *demostración*. Para eso necesitamos definir *fórmula*, porque una demostración es una sucesión de fórmulas cumpliendo ciertos requerimientos. Lo vamos a definir mediante una gramática libre de contexto.

Vamos a hablar de un lenguaje formal cuyas cadenas se llaman *fórmulas*, que vamos a generar con la letra \mathcal{F} . Fórmulas van a ser cadenas como

- $\forall x(\forall y(\forall z\forall \sim x > 1 \ \& \ y > 1 \ \& \ z > 1 \ \& \ n > 2 \ \& \ exp(x, n) + exp(y, n) = exp(z, n))$
- $\forall n(\exists m(m > n \ \& \ prime(m)))$

Ya Ud. puede adivinar que x, y, n y m son variables y exp y $+$ son funciones. $prime$ y $>$ son predicados (a lo mejor adivinó eso también). \forall y \exists son cuantificadores. $\&$ y \sim son funciones lógicas. Probablemente Ud. diría que 1 y 2 son números, pero del punto de vista general los llamamos “constantes”. Su significado depende de qué teoría se está desarrollando: si es teoría de conjuntos, son conjuntos, si geometría Euclidiana son puntos y rectas.

Si la teoría es la de los números naturales hay un solo constante, el 0. Pero hay un *símbolo de función* S (sucesor) aplicable a cada n y que se escribe $S(n)$ (prefijo) o nS (posfijo). En el capítulo 1 usamos posfijo y S era $|$.

Queremos delinear una gramática libre de contexto para *fórmula*. Hemos mencionado varias clasificaciones de “palabras” en este lenguaje. Además de *fórmula* tenemos *función*, *predicado*, *variable*, *constante*.

También necesitamos *término*, y sucesión de términos.

Cada uno de estas clasificaciones es un conjunto de cadenas, representado por una variable sintáctica de la gramática, así:

- fórmula: \mathcal{F}
- término: \mathcal{T}
- sucesión de términos: \mathcal{S}
- función: \mathcal{O} (“operador”, estando \mathcal{F} ocupado)
- predicado: \mathcal{P}
- variable: \mathcal{V}
- constante: \mathcal{C}

A estos vamos a agregar \mathcal{G} , el conjunto de fórmulas sin \rightarrow fuera de paréntesis, que es necesario para evitar ambigüedad y asegurar que $F_1 \rightarrow F_2 \rightarrow \dots \rightarrow F_k$ se derive como si fuera $(F_1 \rightarrow \dots \rightarrow F_{k-1}) \rightarrow F_k$ (importante porque, por ejemplo, $\mathbf{F} \rightarrow (\mathbf{F} \rightarrow \mathbf{F}) = \mathbf{V}$ pero $(\mathbf{F} \rightarrow \mathbf{F}) \rightarrow \mathbf{F} = \mathbf{F}$).

Vamos a suponer que una persona, que llamamos el “usuario”, está desarrollando una teoría con agregar axiomas a nuestro sistema. El usuario define los conjuntos $\mathcal{O}, \mathcal{P}, \mathcal{V}, \mathcal{C}$ a su antojo, sujeto solo a las restricciones que todos sean regulares, que $= \in \mathcal{P}$, que \mathcal{O} y \mathcal{P} sean disjuntos, y que \mathcal{V} y \mathcal{C} sean disjuntos (variables se distinguen de predicados y funciones por no estar seguido por parentesis).

También el usuario tiene el privilegio de agregar un predicado o función nuevo, con su definición (que será alguna fórmula F , y estará siempre bajo la condición $F \rightarrow \dots$). Así hace todo matemático. Además, se le conviene agregar símbolos nuevos al alfabeto, puede hacerlo. Ya que es la primera vez que usa la función, predicado o letra, podemos hacer como estuviese ya todo el tiempo, sin incurrir una inconsistencia.

El usuario está obligado a incluir en su alfabeto los símbolos $\{\forall, \sim, \rightarrow, =\}$ ($\exists, \&, \vee$ y otras cosas útiles se pueden definir en términos de estos, como veremos) y la coma y dos parentesis.

Entonces además de las producciones necesarias para definir esas cuatro conjuntos regulares, tenemos estas producciones obligatorias:

- $\mathcal{F} \rightarrow \mathcal{G} | \mathcal{F} \rightarrow \mathcal{G}$
- $\mathcal{G} \rightarrow \mathcal{P}(\mathcal{S}) | \forall \mathcal{V} \mathcal{G} | \sim \mathcal{G} | (\mathcal{F})$
- $\mathcal{T} \rightarrow \mathcal{V} | \mathcal{C} | \mathcal{O}(\mathcal{S})$
- $\mathcal{S} \rightarrow \mathcal{T} | \mathcal{T}, \mathcal{S}$

No necesitamos \exists porque $\exists x F$ es equivalente a $\sim \forall x \sim F$ para cualquier fórmula F . A veces es conveniente escribir $\exists x F$ pero es una abreviatura para la otra fórmula.

También frecuentemente es conveniente usar $\&$ y \vee (el o inclusivo) pero $F \vee G$ va a significar $\sim F \rightarrow G$ y $F \& G$ va a significar $\sim (F \rightarrow \sim G)$.

Vamos a sacar cualquier duda que puede quedar del significado de $\sim, \rightarrow, \&, \vee$:

F	G	$\sim F$	$F \rightarrow G$	$F \& G$	$F \vee G$
F	F	V	V	F	F
F	V	V	V	F	V
V	F	F	F	F	V
V	V	F	V	V	V

Revisamos la aserción arriba que $A \& B$ equivale a $\sim (F \rightarrow \sim G)$:

F	G	$\sim G$	$F \rightarrow \sim G$	$\sim (F \rightarrow \sim G)$	$A \& B$	
F	F	V	V	F	F	
F	V	F	V	F	F	Toda función o predicado tiene sintaxis pre-
V	F	V	V	F	F	
V	V	F	F	V	V	

fija, es decir tiene la forma $f(x_1, \dots, x_k)$ donde f es función o predicado y los x_i son términos. Sin embargo, con símbolos como $+$, $-$, \times , $/$ que acostumbramos a usar con sintaxis infija, para más claridad usamos infijo porque las precedencias de los operadores son bien conocidos, y nuestro lector sabrá que $x - y + z$ escrito en prefijo es $+(-(x, y), z)$, y $x - (y + z)$ es $-(x, +(y, z))$. También usamos libremente las funciones lógicas con sintaxis infija: $A \& B$ por ejemplo en prefijo es $\&(A, B)$. En expresiones con $\&$ y \vee , además de sus equivalencias con \rightarrow, \sim dadas arriba, acuerdese que $\&$ se evalúa antes de \vee a menos que los paréntesis dirigen distinto.

Notese que se puede “sobrecargar” el nombre de una función o predicado: por ejemplo, en las fórmulas $f(x)$ y $f(x, y)$, el nombre f se refiere a dos funciones distintas que no tienen nada que ver una con la otra.

Entonces x no es una fórmula, tampoco X , pero si $x \in \mathcal{V}$ y $X \in \mathcal{P}$ entonces $X(x)$ y $\forall x X(x)$ sí son fórmulas. También si $+, s \in \mathcal{O}$ y $x, y \in \mathcal{V}$ entonces es fórmula $=(+(x, s(y)), +(s(x), y))$ y $\forall x \forall y = (+(x, s(y)), +(s(x), y))$. (Técnicamente los predicados $=, +$ tiene sintaxis prefija, pero de aquí en adelante por claridad vamos a escribirlo en infijo. Entonces las dos fórmulas anteriores, ya que la precedencia y la asociatividad de $+$ queda bien establecidos, se escribirían $x + s(y) = s(x) + y$ y $\forall z \forall y x + s(y) = s(x) + y$.)

Son los axiomas los que ponen carne y hueso en estos símbolos. Un axioma es una fórmula (derivable de la variable sintáctica \mathcal{F} , junto con el alfabeto y los conjuntos regulares $\mathcal{O}, \mathcal{P}, \mathcal{V}$ dados por el usuario). Estamos solo suponiendo que el conjunto de axiomas sea computable, y aquí hay un ejemplo. Usamos la letra s para significar “sucesor”.

1. $\forall x \sim 0 = s(x)$.
2. $\forall x \forall y (s(x) = s(y) \rightarrow x = y)$
3. $\forall x \sim x = s(x)$
4. Infinitos axiomas de inducción, ya que el lenguaje no permite hablar de conjuntos. Cada uno de estos axiomas dice para algún predicado $P(x)$ de una variable libre x ,

$$(P(0) \& \forall x P(x) \rightarrow P(s(x))) \rightarrow \forall x P(x)$$

Por ejemplo, supongamos que $P(x)$ es $x = 0$. Entonces

$$(0 = 0 \& \forall x (x = 0 \rightarrow s(x) = 0)) \rightarrow \forall x x = 0$$

es un axioma (aunque no sirve para nada).

Con los axiomas 1,2,3 y el conjunto definido en (4) tenemos el equivalente de los axiomas de Peano.

Pero antes de tal conjunto de axiomas, que varía según la teoría con que se quiere trabajar, son los axiomas lógicos y las reglas de inferencia. Para expresarlos, necesitamos un par de definiciones.

Ahora definimos cuándo una variable v ocurre libre en una fórmula o en un conjunto de fórmulas.

- Si F es una fórmula entonces v no ocurre libre en $\forall v(F)$.
- Si v ocurre libre en P entonces ocurre libre en $\sim P$ y en (P) , y para cualquier fórmula Q ocurre libre en $P \rightarrow Q$ y en $Q \rightarrow P$.
- Si $f \in \mathcal{P} \cup \mathcal{O}$ y la variable v ocurre en uno de los términos e_1, \dots, e_k entonces ocurre libre en la fórmula $f(e_1, \dots, e_k)$.
- Si Δ es un conjunto de fórmulas y v una variable entonces v ocurre libre en Δ si y solo si v ocurre libre en una de las fórmulas de Δ .

Una *proposición o sentencia* es una fórmula en que ninguna variable ocurre libre. Son las proposiciones que afirman algo. “ $x=5$ ” no afirma nada, porque no sabemos quién es x .

“ $\forall x(x = 5)$ ” afirma algo, que probablemente es falsa, aunque podemos construir un modelo en que es verdadera. Más sobre proposiciones después.

Sea F una fórmula, $v \in \mathcal{S}_F$ y t un elemento. Entonces $F(t/v)$ es la fórmula que resulta de sustituir cada ocurrencia libre de v en F por t .

Sea u, v en \mathcal{S}_F y F una fórmula. Se dice que F admite u por v si cada ocurrencia libre de v en F llega a ser una ocurrencia libre de u en $F(u/v)$.

Antes de poner los axiomas de alguna teoría, por ejemplo la teoría de números o la teoría de conjuntos, hay que poner los axiomas de la lógica. Vamos a definir lo que se llama *el cálculo de predicados de primer orden*.

Hay infinitos axiomas pero todos salen de uno de 6 esquemas que siguen por medio de reemplazar en un esquema P, Q, R con cualesquier fórmulas, v por cualquier variable, y t con cualquier elemento:

A1: $P \rightarrow (Q \rightarrow P)$

A2: $(R \rightarrow (P \rightarrow Q)) \rightarrow ((R \rightarrow P) \rightarrow (R \rightarrow Q))$

A3: $(\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q)$

A4: $(\forall v(P \rightarrow Q)) \rightarrow (\forall v(P) \rightarrow \forall v(Q))$

A5: (especificación) $\forall v P \rightarrow P(t/v)$ dado que P admite t por v

A6: $P \rightarrow \forall v(P)$ dado que v no es libre en P

y tres axiomas sobre igualdad. Sean los $x, x_i, y_i, F \in \mathcal{S}_F, G \in \mathcal{S}_P$:

E1: $x = x$

E2: $(x_1 = y_1 \& \dots \& x_n = y_n) \rightarrow (G(y_1, \dots, y_n) \rightarrow G(x_1, \dots, x_n))$

E3: $(x_1 = y_1 \& \dots \& x_n = y_n) \rightarrow F(x_1, \dots, x_n) = F(y_1, \dots, y_n)$

Además, si F es un axioma y $v \in \mathcal{S}_F$ ocurre libre en F entonces $\forall v(F)$ es un axioma.

Estos son los axiomas de la lógica. Se llaman *el cálculo de predicados*.

Un *modelo de primer orden* se construye agregando al cálculo de predicados un conjunto de fórmulas como axiomas adicionales, como hemos indicado ya con los axiomas de Peano.

Ahora la definición de *demostración* para un sistema de primer orden es muy simple: es una sucesión de fórmulas en que cada una es o un axioma o sigue por modus ponens (viene enseguida la definición de eso) de dos fórmulas anteriores en la sucesión. Conviene una definición formal más general, reflejando la forma que muchas veces toma una definición: empiezan por ejemplo con algo como “sean p, q numeros primos...”. Es decir, plantea unas fórmulas y deduce una consecuencia de ellas. La correspondiente definición formal es ésta:

DEFINICIÓN 9.1. Sea Δ un conjunto de fórmulas. Una *deducción de Δ* es una sucesión F_1, \dots, F_k de fórmulas tal que cada F_i es

- un axioma, o
- un miembro de Δ , o
- sigue por modus ponens de dos fórmulas anteriores, lo que quiere decir que hay fórmulas F_j, F_k con $j < i$ y $k < i$ tales que F_k es $F_j \rightarrow F_i$.

Decimos $\Delta \vdash F$ si F es la última fórmula de una deducción de Δ .

Decimos simplemente $\vdash F$ cuando Δ es vacío, es decir cuando $\emptyset \vdash F$.

Decimos *demostración* a una deducción del conjunto vacío.

Decimos *teorema* a una fórmula sin variable libre que es la última fórmula de una demostración.

3. El plan general

El primer objetivo es convencernos, sin argumento formal completo pero quedando bien convencidos, de que los teoremas que se consiguen con estas reglas incluyen todos que pueden encontrarse en un libro de la teoría de números, o más generalmente, que las deducciones pueden representar los métodos de demostración que usan los matemáticos, y con lo cual Hilbert

esperaba reducir la investigación matemática al estudio de estas cadenas de símbolos, una interpretación que surge del hecho que uno puede “seguir” una demostración sin tener la menor idea del significado de los símbolos que se manipulan (sin entender, por ejemplo, que $S(n)$ quiere decir “el sucesor de n ”), sino solamente entender las reglas mecánicas de la manipulación. Porque queremos ver que la matemática es más que manipulación mecánica.

El segundo objetivo es convencernos, de nuevo sin argumento formal completo pero quedando bien convencidos, de que ningún tal sistema que es lo suficiente rico para incorporar en alguna forma los axiomas de Peano y a la vez consistente, puede demostrar todo lo que es verdadero. En otras palabras, la teoría de números resiste ser capturada dentro de un sistema formal. Hilbert no esperaba eso, y parece que Bertrand Russell nunca lo captó (dijo que se alegraba de no trabajar más en las fundaciones de la matemática porque le aburrían estos “juegos” nuevos de Gödel).

La mente de Kurt Gödel, en cambio, rechazaba la idea de encerrar toda la verdad de los números en un sistema formal, y se puso a buscar cómo refutar formalmente esa posibilidad, y lo consiguió. Para entender cómo, hay que entender

- que el conjunto de cadenas que hemos definido como *demostraciones* es computable,
- cada función computable es representable en la teoría de primer orden dada arriba basada en los axiomas de Peano y
- la no computabilidad del problema de parar de las máquinas de Turing, que vimos en el capítulo de autómatas.

Vamos ahora al primer objetivo.

4. Cálculo de predicados de primer orden

Si usamos sólo los primeros tres axiomas A1, A2 y A3 el sistema se llama el *cálculo de proposiciones*. Notamos que según esto el lenguaje del cálculo de proposiciones incluye cuantificadores, pero no pueden participar activamente en una deducción. Por ejemplo, $\forall x(x=0) \rightarrow (0=0 \rightarrow \forall x(x=0))$ como sucesión de una sola fórmula, es una demostración porque es uno de los axiomas $P \rightarrow (Q \rightarrow P)$, con P como $\forall x(x=0)$ y Q como $0=0$. La expresión $P \rightarrow (Q \rightarrow P)$ es un esquema de axiomas porque está precedido por “sean P, Q fórmulas”. Podemos deducir otros esquemas de fórmulas que así llegan a ser esquemas de teoremas cuando remplazemos P, Q, R con fórmulas sin variables libres. Siguen unos que vamos a necesitar. Aquí P, Q, R siguen siendo fórmulas cualesquiera. Entonces tenemos esquemas de demostraciones, que al sustituir fórmulas para P, Q, R llegan a ser demostraciones. La segunda columna representa justificaciones para las fórmulas en la primera, por axiomas o por modus ponens (MP):

IDENTIDAD 1: $\vdash P \rightarrow P$	
1. $(P \rightarrow ((P \rightarrow P) \rightarrow P)) \rightarrow ((P \rightarrow (P \rightarrow P)) \rightarrow (P \rightarrow P))$	A2
2. $P \rightarrow ((P \rightarrow P) \rightarrow P)$	A1
3. $((P \rightarrow (P \rightarrow P)) \rightarrow (P \rightarrow P))$	MP 1,2
4. $P \rightarrow (P \rightarrow P)$	A1
5. $P \rightarrow P$	MP 3,4

Si tuviéramos en este momento disponible el teorema de la deducción, la demostración de la identidad 1 sería muy sencilla: $\{P\} \vdash P$ porque la sucesión cuya única fórmula es P constituye una deducción de P de P , entonces según el teorema de la deducción $\vdash P \rightarrow P$. Pero lamentablemente necesitamos la identidad 1 en la demostración del teorema de la deducción.

Las demostraciones de estas identidades ilustran el punto que la lógica formal consiste en la manipulación sintáctica ciega de cadenas, sin referencia al significado de estas cadenas. Tenemos confianza en los resultados de esta manipulación porque entendemos el significado de los axiomas y reglas de inferencia. El teorema de Gödel muestra que este entendimiento va más profundo que cualquier sistema de manipulación formal.

IDENTIDAD 2: $\sim P \rightarrow (P \rightarrow Q)$

- | | |
|--|--------|
| 1. $(\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q)$ | A3 |
| 2. $((\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q)) \rightarrow (\sim P \rightarrow ((\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q)))$ | A1 |
| 3. $\sim P \rightarrow ((\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q))$ | MP 1,2 |
| 4. $(\sim P \rightarrow ((\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q))) \rightarrow$
$((\sim P \rightarrow (\sim Q \rightarrow \sim P)) \rightarrow (\sim P \rightarrow (P \rightarrow Q)))$ | A2 |
| 5. $(\sim P \rightarrow (\sim Q \rightarrow \sim P)) \rightarrow (\sim P \rightarrow (P \rightarrow Q))$ | MP 4,3 |
| 6. $\sim P \rightarrow (\sim Q \rightarrow \sim P)$ | A1 |
| 7. $\sim P \rightarrow (P \rightarrow Q)$ | MP 5,6 |

Un paso importante hacia el primer objetivo es el

TEOREMA 9.2. (de la deducción). *Sean P, Q fórmulas y Δ un conjunto de fórmulas. Si $\Delta \cup \{P\} \vdash Q$ entonces $\Delta \vdash P \rightarrow Q$.*

COMENTARIO. Si Δ es finito, $\Delta = \{P_1, \dots, P_k\}$, esto va a implicar primero que si $\Delta \vdash Q$ entonces $\vdash P_1 \rightarrow (P_2 \rightarrow \dots \rightarrow (P_k \rightarrow Q) \dots)$, que, después de adquirir el teorema de las tautologías más adelante, va a dar $\vdash (P_1 \& \dots \& P_k) \rightarrow Q$.

Demostración. Sea $F_1, \dots, F_k = Q$ una deducción (definición 9.1). de Q de $\Delta \cup \{P\}$. Para cada i en $\{1, \dots, k\}$ definimos una sucesión \mathcal{S}_i tal que para cada i la concatenación de sucesiones $\mathcal{S}_1 \dots \mathcal{S}_i$ es una deducción de $P \rightarrow F_i$ de Δ . Si F_i es un axioma o un miembro de Δ entonces \mathcal{S}_i es $F_i, F_i \rightarrow (P \rightarrow F_i), P \rightarrow F_i$, la segunda fórmula por A1 y la tercero de modus ponens de las primeras dos. Así si $i = 1$ o ya tenemos los primeros $i - 1$ de los \mathcal{S}_j , $\mathcal{S}_1 \dots \mathcal{S}_i$ es una deducción de $P \rightarrow F_i$ de Δ . Si F_i es P entonces \mathcal{S}_i es la demostración de $P \rightarrow P$ que hemos visto recién.

La única otra posibilidad es que en la deducción de Q , F_i se obtiene por modus ponens, es decir para algunos $j, h < i$, F_h es $F_j \rightarrow F_i$. Entonces $i > 2$ y por la hipótesis de inducción para cada $g < i$, $\mathcal{S}_1 \dots \mathcal{S}_g$ es una deducción de $P \rightarrow F_g$ desde Δ , las fórmulas $P \rightarrow F_j$ y $P \rightarrow (F_j \rightarrow F_i)$ aparecen en $\mathcal{S}_1 \dots \mathcal{S}_{i-1}$. Entonces

$$\mathcal{S}_i \text{ es } ((P \rightarrow (F_j \rightarrow F_i)) \rightarrow ((P \rightarrow F_j) \rightarrow (P \rightarrow F_i)), (P \rightarrow F_j) \rightarrow (P \rightarrow F_i), P \rightarrow F_i$$

donde la primera fórmula es de A2, la segunda por modus ponens con la primera y $P \rightarrow (F_j \rightarrow F_i)$, y la tercera de modus ponens entre la segunda y $P \rightarrow F_j$. \square

COROLARIO 9.3. *Si $\Delta \cup \{P, Q\} \vdash R$ entonces $\Delta \vdash (P \& Q) \rightarrow R$.*

Demostración. Dos aplicaciones del teorema de deducción dan $\Delta \vdash (P \rightarrow (Q \rightarrow R))$. Pero $P \rightarrow (Q \rightarrow R)$ es equivalente a $(P \& Q) \rightarrow R$. \square

Un teorema de alta utilidad práctica en construir las demostraciones es **TEOREMA** (“gen”). Si $\Delta \vdash P$ y v no es libre en Δ entonces $\Delta \vdash \forall v(P)$.

La demostración es media larga pero de concepción sencilla. Se demuestra, en una técnica similar a la usada para teorema de deducción, por inducción que si $F_1 \dots F_k$ es una deducción desde Δ entonces existe una deducción desde Δ en que $\forall v(F_i)$ ocurre para cada i .

El cálculo de proposiciones tiene teoremas como $P \rightarrow P$, $P \vee \sim P$, $(P \rightarrow Q) \rightarrow (\sim Q \rightarrow \sim P)$, siendo P, Q fórmulas cualesquiera. Tales fórmulas se llaman *tautologías* porque tienen que ser verdaderas para cualquier asignación de valores de verdadera o falsa asignadas a P y Q . $(\forall x(x = 1) \rightarrow \forall x(x = 0)) \rightarrow (\sim \forall x(x = 0) \rightarrow \sim \forall x(x = 1))$ es una tautología a pesar de lo inútil que parece. Más generalmente,

DEFINICIÓN 9.4. Una *tautología formada por las fórmulas U_1, \dots, U_k* es una fórmula formada por las U_1, \dots, U_k , paréntesis, y operadores \sim y \rightarrow que es verdadera para cualquier asignación de verdadera o falsa a las U_1, \dots, U_k .

Por ejemplo, dado que $A \rightarrow B$ es falsa sólo si A es verdadera y B es falsa, es decir, por

A	B	$A \rightarrow B$
F	F	V
F	V	V
V	F	F
V	V	V

definición cumple la siguiente tabla:

entonces $A \rightarrow (B \rightarrow A)$ cumple la

A	B	$B \rightarrow A$	$A \rightarrow (B \rightarrow A)$
F	F	V	V
F	V	F	V
V	F	V	V
V	V	V	V

así que $A \rightarrow (B \rightarrow A)$ es una tautología.

Resulta que los teoremas del cálculo de proposiciones son exactamente las tautologías. Vamos a demostrar eso ahora en varios pasos.

LEMA 9.5. *Todo teorema demostrable con solamente los axiomas A1, A2 y A3 es una tautología.*

Demostración. Sea F_1, \dots, F_k una demostración usando sólo esos axiomas. Demostramos por inducción sobre i que cada F_i es una tautología. Notamos primero por inspección que cada axioma que surge de A1, A2 o A3 es una tautología, así que si F_i es un axioma entonces es una tautología. Eso establece la base de la inducción, porque F_1 es necesariamente un axioma. Supongamos que $1 < i \leq k$ y que F_1, \dots, F_{i-1} son tautologías. Si F_i no es axioma entonces existen $j, h < i$ tales que F_h es $F_j \rightarrow F_i$. Si hay una asignación que hace falsa F_i , ya que F_j es tautología esa asignación hace verdadera a F_j y por tanto falsa a $F_j \rightarrow F_i$ lo que contradice la hipótesis de inducción. \square

Ahora necesitamos la segunda de las dos identidades que siguen, y la segunda necesita la primera. De aquí en adelante justificamos un paso de una deducción con “deducción” cuando es por el teorema de la deducción.

IDENTIDAD 3: $\sim\sim P \rightarrow P$

- | | |
|---|---|
| 1. $\sim\sim P$ | hipótesis ($\Delta = \{\sim\sim P\}$) |
| 2. $\sim\sim P \rightarrow (\sim P \rightarrow \sim\sim\sim P)$ | Identidad 2 |
| 3. $\sim P \rightarrow \sim\sim\sim P$ | modus ponens |
| 4. $(\sim P \rightarrow \sim\sim\sim P) \rightarrow (\sim\sim P \rightarrow P)$ | A3 |
| 5. $\sim\sim P \rightarrow P$ | modus ponens 4,3 |
| 6. P | modus ponens 5,1 |
| 7. $\sim\sim P \rightarrow P$ | deducción (porque $\{\sim\sim P\} \vdash P$) |

IDENTIDAD 4. $P \rightarrow \sim\sim P$

- | | |
|---|--------------|
| 1. $\sim\sim\sim P \rightarrow \sim P$ | Identidad 3 |
| 2. $(\sim\sim\sim P \rightarrow \sim P) \rightarrow (P \rightarrow \sim\sim P)$ | A3 |
| 3. $P \rightarrow \sim\sim P$ | modus ponens |

IDENTIDAD 5. $(P \rightarrow Q) \rightarrow (\sim Q \rightarrow \sim P)$

- | | |
|--|------------------|
| 1. $P \rightarrow Q$ | hipótesis |
| 2. $\sim\sim P$ | hipótesis |
| 3. $\sim\sim P \rightarrow P$ | identidad 3 |
| 4. P | modus ponens 3,2 |
| 5. Q | modus ponens 1,4 |
| 6. $Q \rightarrow \sim\sim Q$ | identidad 4 |
| 7. $\sim\sim Q$ | modus ponens 6,5 |
| 8. $\sim\sim P \rightarrow \sim\sim Q$ | deducción, 1-7 |
| 9. $(\sim\sim P \rightarrow \sim\sim Q) \rightarrow (\sim Q \rightarrow \sim P)$ | A3 |
| 10. $\sim Q \rightarrow \sim P$ | modus ponens 9,8 |
| 11. $(P \rightarrow Q) \rightarrow (\sim Q \rightarrow \sim P)$ | deducción 1-10 |

IDENTIDAD 6. $Q \rightarrow (\sim R \rightarrow \sim (Q \rightarrow R))$

1. Q	hipótesis
2. $Q \rightarrow R$	hipótesis
3. R	modus ponens 2,1
4. $(Q \rightarrow R) \rightarrow R$	deducción: 1-3
5. $((Q \rightarrow R) \rightarrow R) \rightarrow (\sim R \rightarrow \sim (Q \rightarrow R))$	identidad 5
6. $\sim R \rightarrow \sim (Q \rightarrow R)$	modus ponens 5,4

LEMA 9.6. Sea P una fórmula formada de las fórmulas U_1, \dots, U_k . Para cada asignación de verdadera y falsa a las U_1, \dots, U_k , para cada i sea $U'_i = U_i$ si U_i es asignada verdadera, y $U'_i = \sim U_i$ si U_i es asignada falsa, y sea $P' = P$ si la asignación hace verdadera a P , y sea $P' = \sim P$ si no. Entonces

$$\{U'_1, \dots, U'_k\} \vdash P'.$$

Demostración. Procedemos por inducción sobre el número de operadores \sim y \rightarrow que se usan para formar P de las U_i . Si esto es 0 la aserción es o $P \vdash P$ o $P' \vdash P'$, obvios los dos. Suponga que la aserción se cumple para todo caso en que hay menos de i operadores, y que P tenga i operadores.

Si P es $\sim Q$ donde Q tiene $i - 1$ operadores entonces $U'_1, \dots, U'_k \vdash Q'$ por la hipótesis de inducción. Si una asignación da valor verdadero a Q entonces da falso a P , por tanto con esa asignación $Q' = Q$ y $P' = \sim P$. De la deducción de Q' agregamos $Q' \rightarrow \sim \sim Q'$, $\sim \sim Q'$ siendo el primero una instancia de la Identidad 4 y el segundo por modus ponens con Q' y $Q' \rightarrow \sim \sim Q'$: observamos que $\sim \sim Q'$ es $\sim \sim Q$ que es $\sim P$ que es P' . Si la asignación da el valor falsa a Q entonces da verdadera a P , por tanto $Q' = \sim Q$ y $P' = P$. Por la hipótesis de inducción $U'_1, \dots, U'_k \vdash \sim Q$. La deducción de $\sim Q$ es una deducción de P , y P es P' .

Supongamos que P es $Q \rightarrow R$ donde Q y R tienen menos de i operadores.

Supongamos que una asignación da el valor verdadera a R . Entonces da verdadera a P . Por lo tanto $R' = R$ y $P' = P$. Por la hipótesis de inducción $U'_1, \dots, U'_k \vdash R$. Aplicando modus ponens a R y el axioma $R \rightarrow (Q \rightarrow R)$ tenemos que $U'_1, \dots, U'_k \vdash Q \rightarrow R$, que es P , que es P' .

Ahora supongamos que la asignación da el valor falsa a Q . Entonces da verdadera a P . Entonces $Q' = \sim Q$ y $P' = P$. Por la hipótesis de inducción $U'_1, \dots, U'_k \vdash \sim Q$. Aplicando modus ponens a $\sim Q$ y $\sim Q \rightarrow (Q \rightarrow R)$ justificada por la identidad 2, tenemos $U'_1, \dots, U'_k \vdash Q \rightarrow R$, que es P , que es P' .

Finalmente, si ninguna de las asunciones anteriores se cumplen, es que la asignación da valor verdadero a Q y valor falso a R . Entonces da valor falsa a P , y concluimos que $Q' = Q$, $R' = \sim R$ y $P' = \sim P$. Por la hipótesis de inducción $U'_1, \dots, U'_k \vdash Q$ y $U'_1, \dots, U'_k \vdash \sim R$. Aplicando modus ponens a Q y al teorema $Q \rightarrow (\sim R \rightarrow \sim (Q \rightarrow R))$ tenemos $\sim R \rightarrow \sim (Q \rightarrow R)$, y aplicando modus ponens a eso y $\sim R$ nos da $\sim (Q \rightarrow R)$, que es $\sim P$, que es P' . \square

TEOREMA 9.7. (de las tautologías) *Los teoremas del cálculo de proposiciones son exactamente las tautologías.*

Demostración. Por el Lema 9.5, solamente falta demostrar que cualquier tautología es demostrable.

Sea P una tautología formada de las fórmulas U_1, \dots, U_k , y sea U'_i y P' como en Lema 9.6.

Por ese Lema $\{U'_1, \dots, U'_k\} \vdash P'$, pero siendo P una tautología formada por las U_i eso implica que $\{U'_1, \dots, U'_k\} \vdash P$ para cualquier asignación a las U_i . Ahora vamos a demostrar para todo $i \in \{1, \dots, k\}$

si $\{U'_i, \dots, U'_k\} \vdash P$ para toda asignación entonces $\{U'_i, \dots, U'_k\} - \{U'_i\} \vdash P$ para toda asignación.

Demostrada esa, el caso $i = k$ da $\vdash P$.

Suponiendo que $\{U'_i, \dots, U'_k\} \vdash P$ para toda asignación, considere una asignación a U'_{i+1}, \dots, U'_k . Si la combinamos con el asignar verdadera a U_i , haciendo que $U'_i = U_i$, tenemos $\{U_i, U'_{i+1}, \dots, U'_k\} \vdash P$ y la combinamos con el asignar falsa a U_i , haciendo que $U'_i = \sim U_i$,

teniendo $\sim U_i, U'_{i+1}, \dots, U'_k \vdash P$. Por el teorema de la deducción esas dan respectivamente $\{U'_{i+1}, \dots, U'_k\} \vdash U_i \rightarrow P$ y $\{U'_{i+1}, \dots, U'_k\} \vdash \sim U_i \rightarrow P$. Entonces podemos formar una deducción de $\{U'_{i+1}, \dots, U'_k\}$ en que ocurren las fórmulas $U_i \rightarrow P$ y $\sim U_i \rightarrow P$. Del teorema $(P \rightarrow Q) \rightarrow ((\sim P \rightarrow Q) \rightarrow Q)$ que hemos visto, podemos agregar la fórmula $(U_i \rightarrow P) \rightarrow ((\sim U_i \rightarrow P) \rightarrow P)$ (técnicamente, la demostración de ésa, cuyo esquema hemos visto). Aplicando modus ponens a esa fórmula con $U_i \rightarrow P$ agregamos $(\sim U_i \rightarrow P) \rightarrow P$, y aplicando modus ponens con esa y $\sim U_i \rightarrow P$ agregamos P , con la cual tenemos una deducción que muestra que $U'_{i+1} \dots U'_k \vdash P$, para la asignación arbitraria a las $U'_{i+1} \dots U'_k$. \square

Es claro que la manipulación de cadenas para construir demostraciones se puede hacer sin pensar en el significado de las cadenas. Es más, se debe hacer sin pensar en el significado de las cadenas. Por ejemplo, si demuestro $x = y$ e $y = z$ mi entendimiento del significado del signo $=$ me conduce a concluir que $x = z$. Pero no es axioma – tenemos que producir una demostración. Tampoco puedo sin demostración concluir de $x = y$ que $y = x$. Ni puedo, habiendo demostrado $x = y$ y $y = z$ concluir sin demostración la fórmula $x = y \& y = z$. Para desarrollar la teoría de números hay un montón de teoremas de este estilo que es necesario demostrar. Como ejemplos vamos a demostrar estos tres que acabamos de mencionar.

PROPOSICIÓN 9.8. Si $\Delta \vdash A$ y $\Delta \vdash B$ entonces $\Delta \vdash A \& B$.

Demostración. Tenemos

1. $\vdash B \rightarrow (A \rightarrow B)$ (axioma)
2. $\Delta \vdash A \rightarrow B$ (modus ponens de (1) y la hipótesis $\Delta \vdash B$).
3. $\vdash A \rightarrow (B \rightarrow (\sim (A \rightarrow \sim B)))$ (tautología – probarlo)
4. $\Delta \vdash B \rightarrow (\sim (A \rightarrow \sim B))$ (modus ponens de (3) y la hipótesis $\Delta \vdash A$.)
5. $\Delta \vdash \sim (A \rightarrow \sim B)$ (modus ponens de (4) y la hipótesis $\Delta \vdash B$.)
6. $\Delta \vdash A \& B$ por (5) y la definición de $A \& B$. \square

PROPOSICIÓN 9.9. $\vdash x = y \rightarrow y = x$.

Demostración.

- | | | |
|---|--|---|
| 1 | $(y = y \& x = y) \rightarrow (y = x \leftrightarrow y = y)$ | Dos aplicaciones de E2 y uno del teorema anterior |
| 2 | $\forall x(x = x)$ | E1 y generalización–x no libre en Δ |
| 3 | $y = y$ | A5 |
| 4 | $x = y \rightarrow y = x$ | calc de prop, 1,3 |

PROPOSICIÓN 9.10. $(x = y \& y = z) \rightarrow x = z$

Demostración.

- | | | |
|---|--|---|
| 1 | $(x = y \& z = z) \rightarrow (x = z \leftrightarrow y = z)$ | Dos aplicaciones de E2 y uno del teo anterior |
| 2 | $\forall x(x = x)$ | E1, gen. |
| 3 | $z = z$ | spec. 2 |
| 4 | $(x = y \& y = z) \rightarrow x = z$ | Tautológico dado 1,3 \square |

Hemos dado las funciones $+$ y \times como axiomas, pero si vamos a desarrollar la teoría de números vamos a definir muchos más términos, *primo*, *par*, *cociente*, *resto*, *exponenciación* para nombrar algunos y podríamos seguir la costumbre de agregar un nuevo axioma para cada definición. Pero lo que hacemos generalmente es ir dando muchas DEFINICIONES y haciendo deducciones en base de ellas, es decir, siempre vamos formando un gran Δ , que va creciendo mientras trabajamos, y vamos haciendo deducciones que muestran $\Delta \vdash F$ que hace que F sea el teorema 5.7.69 que no cuesta nada agregar a Δ porque no hay nada que diga que las fórmulas de Δ tienen que ser independientes (y si es muy axiomático el desarrollo, entonces cuando el profesor propone el ejercicio “demostrar F ” puede ser difícil determinar qué, exactamente, es el Δ vigente en ese momento).

En el capítulo 2 se demostró por inducción que $m+n' = (m+n)'$ como primer paso en demostrar que $+$ es conmutativa. Vamos a ver si podemos convertir esa demostración en una demostración formal como hemos definido aquí. Lo que quisieramos demostrar es

$$+(n, S(m)) = S(+(n, m)),$$

dados los axiomas sobre $+$ en la teoría de números. (Aunque la notación prefija para $+$ y sucesor hace más tedioso el proceso – por eso el infijo es más popular – queremos enfatizar aquí que tenemos que manipular las cadenas que se presentan con las reglas que nos dan.) La demostración consiste en demostrar $P(0)$ y $\forall n(P(n) \rightarrow P(n+1))$, donde $P(n)$ es $\forall m(+ (n, S(m)) = S(+ (n, m)))$ y aplicar axioma N3.

La manipulación mecánica necesaria para demostrar esto en el cálculo de predicados, con los axiomas que dimos arriba, se puede apreciar de este bosquejo: Sea $\nabla = \{P(n)\}$. Vamos a demostrar que $\vdash P(0)$ y que $\Delta \vdash P((S(n)))$, y por tanto $\vdash P(0) \& \forall n(P(n) \rightarrow P(S(n)))$ y consecuentemente por el axioma de inducción (N3) y modus ponens tenemos $\forall n(P(n))$, siguiendo la misma demostración como en la teoría de números, pero vamos a ver que para que sea una demostración completa en el cálculo de predicados nos faltan varios teoremas, que aunque son tan “obvios” como $P \rightarrow P$, requieren demostración. Aquí la demostración de $P(0)$:

- | | | |
|----|---|--|
| 1 | $+(m, S(0)) = +(S(m), 0)$ | ax N5 |
| 2 | $+(m, 0) = m$ | ax N4 |
| 3 | $\forall m(+ (m, 0) = m)$ | gen., m no libre en Δ |
| 4 | $+(S(m), 0) = S(m)$ | A5 |
| 5 | $+(m, S(0)) = +(S(m), 0) \& +(S(m), 0) = S(m)$ | $\vdash A$ y $\vdash B \Rightarrow \vdash A \& B$ |
| 6 | $+(m, S(0)) = +(S(m), 0) \& +(S(m), 0) = S(m)$
$\rightarrow +(m, S(0)) = S(m)$ | teo $x = y \& y = z \rightarrow x = z$
modus ponens 5,6 |
| 7 | $+(m, S(0)) = S(m)$ | N4 |
| 8 | $m = m + 0$ | E3 |
| 9 | $m = m + 0 \rightarrow S(m) = S(m + 0)$ | |
| 10 | $S(m) = S(m + 0)$ | |
| 11 | $+(m, S(0)) = S(m) \& S(m) = S(m + 0)$ | $\vdash A$ y $\vdash B \Rightarrow \vdash A \& B$ |
| 12 | $+(m, S(0)) = S(m) \& S(m) = S(m + 0)$
$\rightarrow +(m, S(0)) = S(m + 0)$ | teo $x = y \& y = z \rightarrow x = z$
modus ponens 11,12 |
| 13 | $+(m, S(0)) = S(m + 0)$ | gen. m no libre en Δ |

Y aquí, compactando con combinar varios pasos en uno, la expansión de los cuales da una demostración tan completa como la anterior, la demostración que $P(n) \vdash P(S(n))$, donde $P(n) = \forall m(+ (m, S(n)) = S(+ (m, n)))$, así completando la inducción:

- | | | |
|---|--|----------------------------|
| 1 | $\forall m(+ (m, S(n)) = S(+ (m, n)))$ | hipótesis (Δ) |
| 2 | $+(m, S(S(n))) = +(S(m), S(n))$ | N5 |
| 3 | $+(S(m), S(n)) = S(+ (S(m), n))$ | A5 aplicado a 1 |
| 4 | $+(m, (S(S(n)))) = S(+ (S(m), n))$ | 2,3 y transitividad de $=$ |
| 5 | $+(S(m), n) = S(+ (m, S(n)))$ | N5 y simetría de $=$ |
| 6 | $S(+ (m, S(n))) = S(+ (S(m), n))$ | N5 |
| 7 | $+(m, S(S(n))) = S(+ (m, S(n)))$ | 5,6 y transitividad de $=$ |

Así vemos que $P(n) \vdash P(S(n))$, así que por el teorema de la deducción $\vdash P(n) \rightarrow P(S(n))$ y ya que este Δ es vacío, $\vdash \forall n(P(n) \rightarrow P(S(n)))$. Finalmente, por el teorema de la conjunción de teoremas, $\vdash P(0) \& \forall n(P(n) \rightarrow P(S(n)))$, y por tanto por N3, $\vdash \forall n(P(n))$.

Hay mucho más trabajo que queda hasta formalizar todo, pero hemos visto cómo se hace.

5. El Teorema de Gödel

5.1. El teorema de la representación. Hemos visto a las máquinas de Turing como mecanismos para aceptar o rechazar conjuntos. Un ejemplo era $\{1^n 0 1^m 0 1^p : p = n \times m\}$ (conviene reservar el 0 para β . Así la máquina está distinguiendo una propiedad que puede tener una terna (n, m, p) de números. Decimos *propiedad* y no *predicado* porque queremos reservar *predicado* para fórmulas: un *predicado de n lugares* es una fórmula $F(x_1, \dots, x_n)$ en que cada una de las n distintas variables x_1, \dots, x_n ocurre libre. Acabamos de mencionar la propiedad expresada por la fórmula $n \times m = p$. Una propiedad de n lugares entonces es un

conjunto de n -uplas de (en el contexto actual) números naturales. Podemos definir que una propiedad computable P de n lugares es tal que $\{1^{k_1}21^{k_2}2\dots, 21^{k_n} : (k_1, \dots, k_n) \in P\}$ es aceptado por alguna máquina de Turing.

También queremos distinguir entre, por ejemplo, el predicado $F(x, y)$ y $F(6, 7)$, que es otra fórmula obtenida por sustituir 6, 7 por cada ocurrencia libre de x, y por 6, 7 respectivamente. En general, queremos distinguir entre $F(x_1, \dots, x_n)$ y $F(\mathbf{m}_1, \dots, \mathbf{m}_n)$ que es otra fórmula obtenida por sustituir los números $\mathbf{m}_1, \dots, \mathbf{m}_n$ por las ocurrencias libres de las variables x_1, \dots, x_n respectivamente. Siempre las notaciones x, y, z etc. y x_i van a representar variables, y $\mathbf{x}, \mathbf{y}, \mathbf{m}, \mathbf{n}, \mathbf{x}_i, \mathbf{y}_i$ etc. van a representar números.

El teorema que estamos por ver va a decir que cualquier propiedad computable de n -upla es expresable como una fórmula del teorema de números de primer orden.

Es claro que no perdemos generalidad restringiendo la atención para la definición de *computable* a máquinas de Turing con alfabeto $\{0, \dots, n-1\}$ para algún n , con $\beta = 0$, que consecuentemente empiezan en una configuración $0^\infty w 0^\infty$ con $w \in \{1, \dots, n-1\}^*$ y el cabezal en el primer carácter de w .

Entonces cualquier configuración puede ser expresada como una terna (q, x, y) donde q es el estado, x es el valor de la cadena hasta el cabezal interpretada en la base n (recuerda que empieza con infinitos 0), y y es el valor del revés de la cadena que empieza donde está el cabezal (ésta también empieza con infinitos 0) interpretada en la base n .

TEOREMA 9.11. (de la representación). *Para cada propiedad computable P de n lugares hay una fórmula $F(x_1, \dots, x_k)$ de n lugares en el modelo de primer orden para la teoría de números tal que si $P(\mathbf{x}_1, \dots, \mathbf{x}_n)$ entonces $\vdash F(\mathbf{x}_1, \dots, \mathbf{x}_n)$ y si $\sim P(\mathbf{x}_1, \dots, \mathbf{x}_n)$ entonces $\vdash \sim F(\mathbf{x}_1, \dots, \mathbf{x}_n)$.*

Demostración. Primero vamos a demostrar que cualquier conjunto computable C de cadenas sobre el alfabeto $\{1, \dots, n-1\}$ (sería suficiente con solo $n = 3$, pero no cuesta casi nada tomar n general) existe una fórmula $F(x)$ que lo representa en el sentido ésto: Para toda cadena $w \in \{1, \dots, n-1\}$ sea \hat{w} el número expresado por w^R leído en la base n , es decir, si $w = w_0 \dots w_k$ entonces buscamos una F tal que $\hat{w} = \sum_{0 \leq i \leq k} w_i n^i$. Entonces para todo $w \in C$, $\vdash F(\hat{w})$ y para todo $w \notin C$, $\vdash \sim F(\hat{w})$. Luego formamos con una función del primer orden $f(x_1, \dots, x_{n-1})$ que da un número m cuya representación en la base 3 es $1^{x_1}21^{x_2}2 \dots 1^{x_{n-1}}$, y calculamos $F(\hat{m})$.

Para cada estado q de la máquina definimos un predicado $q(x, y)$ cuyo significado semántico va a ser que si la máquina está en la configuración representada por (q, x, y) como descrita arriba, va a terminar aceptando. Ahora el carácter que ve el cabezal es $y \bmod n$. Por lo tanto, sea

$$\{i : \delta(q, i) \text{ está definido}\} = \{i_1, \dots, i_k\}$$

y definimos

$$q(x, y) = f_1 \& f_2 \& \dots \& f_k$$

donde cada f_j se define así: sea $\delta(q, i_j) = (r, b, d)$.

- Si r es el estado final entonces f_j es $y \bmod n = i_j$.
- Si r no es el estado final entonces
 - si $d = D$ entonces f_j es $y \bmod n = i \& r(nx + b, \lfloor y/n \rfloor)$.
 - si $d = I$ entonces f_j es $y \bmod n = i_j \& r(\lfloor x/n \rfloor, n(y - y \bmod n + b) + x \bmod n)$.

Ahora la configuración inicial cuando se pide a la máquina que haga juicio sobre $w \in \{1, \dots, n\}^*$ es $(q_0, 0, y)$ donde y es el número obtenido al leer w^R en la base n . El resto de la demostración es verificar que la configuración (q, x, y) , si no da el estado final, es la configuración (r, x', y') definido arriba con los f_j . Eso se va a ver claramente en el ejemplo que sigue a continuación.

Pero falta especificar un predicado de k lugares que representa la relación de k lugares. Para eso falta una función de k argumentos que convierta x_1, \dots, x_k en el número que se expresa en ternario como $1^{x_1}21^{x_2}2 \dots 1^{x_k}$. Esto es el ejercicio 6 que tiene una fuerte sugerencia. \square

Vamos a ver este ejemplo: sea L el conjunto de cadenas sobre $\{1, 2\}$, $L = \{1^x 2^x : x \geq 0\}$. Se puede considerarlo como representando al predicado de dos lugares $x = y$. Ahora vamos a

definir una máquina de Turing que acepta L . Recordamos que eso significa que si empieza en la configuración $\beta^\infty w \beta^\infty$ para algún $w \in \{1, 2\}^*$ con el cabezal sobre la primera letra de w entonces acepta si $w = 1^x 2^y$ para algún natural x y rechaza si no. He aquí tal máquina, con el alfabeto de trabajo $\{0, 1, 2\}$ (0 es β), y estados I (inicial), B (buscar el último 2) B_0 (cambiar el último 2 a 0), R (retornar al principio y repetir con el estado I .)

	1	2	0
I	$B, 0, \rightarrow$		OK
B	$B, 1, \rightarrow$	$B, 2, \rightarrow$	$B_0, 0, \leftarrow$
B_0		$R, 0, \leftarrow$	
R	$R, 1, \leftarrow$	$R, 2, \leftarrow$	$I, 0, \rightarrow$
OK			

Los predicados definidos son:

1. $I(x, y): \forall x \forall y (y \bmod 3 = 0 \vee (y \bmod 3 = 1 \ \& \ B(3x, \lfloor y/3 \rfloor)))$
2. $B(x, y): \forall x \forall y (y \bmod 3 = 0 \ \& \ B_0(\lfloor x/3 \rfloor, 3y + (x \bmod 3))) \vee (x \bmod 3 \neq 0 \ \& \ B(3x + (y \bmod 3), \lfloor y/3 \rfloor))$
3. $B_0(x, y): \forall x \forall y (y \bmod 3 = 2 \ \& \ R(\lfloor x/3 \rfloor, 3y + (x \bmod 3)))$
4. $R(x, y): \forall x \forall y ((y \bmod 3 = 0 \ \& \ I(3x, y/3)) \vee (y \bmod 3 \neq 0 \ \& \ R(\lfloor x/3 \rfloor, 3y + x \bmod 3)))$

La siguiente tabla muestra una computación de esta máquina en aceptar la cadena 1122, junto con los predicados asociadas a cada configuración. En paréntesis al lado de cada x, y se pone sus representaciones ternarias, que en el caso de los y son al revés de la correspondiente cadena a la derecha del cabezon en la configuración.

func	x	y	$x \bmod 3$	$y \bmod 3$	$\lfloor x/3 \rfloor$	$\lfloor y/3 \rfloor$	config
I	0	76(2211)		1		25(221)	0I11220
B	0	25((221)		1		2	00B1220
B	1(1)	8(22)		2		2	001B220
B	5(12)	2(2)		2		0	0012B20
B	17(122)	0	2	0	5		00122B0
B_0	5(12)	0	2	2	1		0012B ₀ 20
R	1(1)	2(2)	1	2	0		001R200
R	0	7(21)	0	1	0		00R1200
R	0	21(210)		0	0		0R01200
I	0	7(21)		1		7	00I1200
B	0	2(2)		2		0	000B200
B	2	0	2	0			0002B00
B_0	0	2	0	2			000B ₀ 200
R	0	0		0			00R0000
I	0	0		0			000I000
OK	0	0		0			verdadero

Ahora, debe ser claro que lo que hemos hecho para la propiedad $x = y$ podemos hacer para cualquier propiedad P computable de dos lugares: si es computable existe una máquina de Turing que acepta $\{a^x b^y : P(x, y)\}$. Y si queremos definir los predicados de dos lugares haciendo máquinas de Turing que aceptan $\{1^x 2^y : P(x, y)\}$ no cambia mucho el problema, tiene unos cuantos detalles más. Y ese formato nos permite definir predicados de cualquier k lugares, definiendo una máquina de Turing que acepta $\{1^{x_1} 2^{x_2} \dots 2^{x_k} : P(x_1, \dots, x_k)\}$, y siguiendo el método expuesto en el paradigma arriba para engendrar sentencias predicados $q(x, y)$. en particular $q_0(0, m)$ donde m es lo que vale la cadena $1^{x_1} 2^{x_2} \dots 2^{x_k}$ interpretando su revés $1^{x_k} 2^{x_{k-1}} \dots 2^{x_1}$ como una representación en la base 3 tal que $\vdash q_0(0, m)$ si $P(x_1, \dots, x_k)$ y $\vdash \sim q_0(0, m)$ si $P(x_1, \dots, x_k)$ es falso.

5.2. Computabilidad de demostraciones. Ahora es claramente computable dadas dos cadenas F y D decidir si D es una demostración de F . Para evitar la posibilidad de ambigüedad, agregamos el carácter \leftarrow al alfabeto para indicar fin de línea. Primero habrí que verificar que

las subcadenas delimitadas por los \leftrightarrow y el principio y fin de D son todas fórmulas, y tenemos una gramática para eso, bien computable. Después, que cada fórmula es o un axioma, y hemos estipulado que los axiomas son computables, o si no que para cada fórmula f en D , existen fórmulas g y $g \rightarrow f$ antes de f en D , y esto es cuestión de escanear para $\rightarrow f$, verificar que la fórmula completa es $g \rightarrow f$ para alguna fórmula g , y buscar g entre las fórmulas que preceden a f en D . Si no, buscamos otro $\rightarrow f$ y hacer lo mismo, hasta o tener éxito o terminar con D , en cuyo caso no es una demostración de nada. Si llegamos con éxito al final de D determinando que es una demostración, si su última línea es F , entonces sí D es una demostración de F , si no, es una demostración de algo pero no de F .

Entonces la propiedad que dice que D es una demostración de F es computable. Por tanto hay un predicado $d(x, y)$ de la teoría de números tal que $\vdash d(\mathbf{s}, \mathbf{t})$ si \mathbf{s} es el número de una fórmula (donde el número G_c de una cadena c es como definido anteriormente, después de incorporar \leftrightarrow al alfabeto) y \mathbf{t} es el número de una demostración de esa fórmula; y $\vdash \sim d(\mathbf{s}, \mathbf{t})$ si no.

5.3. El enunciado y primera demostración del Teorema de Gödel. Ahora, aunque hemos demostrado poco formalmente, esperamos que Ud. puede estar convencido de lo que hemos presentado hasta aquí. Podemos bosquejar la demostración del teorema de Gödel. Por lo menos, como lo enunció él, que tenía dos restricciones que en subsecuentes trabajos fueron relajadas: una con su definición de *computable*, que hoy en día sabemos que no incluía todas las funciones que consideramos computables ahora, pero de hecho incluía todas las funciones que se usan en la práctica de la matemática, y otra acerca de la noción de *consistencia*, que por supuesto dice que para ninguna fórmula F tenemos $\vdash F$ y $\vdash \sim F$. En lugar de *computable* pidió propiedad más restrictiva que ahora llamamos *recursiva primitiva* que el creía que expresaba computabilidad¹, y en lugar de *consistente* pidió una propiedad más fuerte, ésta:

DEFINICIÓN 9.12. Una teoría es ω -consistente si no existe una fórmula $F(x)$ de una sola variable libre x tal que $\vdash \sim F(\mathbf{o})$ para cada objeto \mathbf{o} (por ejemplo, en la teoría de números, cada número) pero en que sin embargo se tiene $\vdash \exists x(F(x))$.

Es fácil verificar que una teoría ω -consistente es consistente (ver ejercicio).

Vamos a ver teorías que son consistentes pero no ω -consistentes. Pero, ¿que tal si en la teoría de números tenemos para cualesquier números a, b, c, n con $n > 2$, $\vdash \sim a^n + b^n = c^n$, pero sin embargo, $\vdash \exists n, a, b, c(n > 2 \ \& \ a^n + b^n = c^n)$? Esperamos ω -consistencia en nuestras teorías.

TEOREMA (Gödel, 1931). Si una teoría del cálculo de predicados tiene axiomas computables, incluye la aritmética de Peano, y es ω -consistente entonces hay una sentencia B tal que ni $\vdash B$ ni $\vdash \sim B$.

COMENTARIO. La primera cosa que nos llama la atención en pensar en las consecuencias que resultan por negar este teorema: El conjunto de teoremas de la teoría de números sería decidible porque dado una sentencia sería que o $\vdash B$ o $\vdash \sim B$, así que pasando por todas las cadenas sobre el alfabeto de la teoría en algún momento vamos a encontrar una que es una demostración de B o de $\sim B$. Esto parecería reducir la matemática a la pura manipulación mecánica de cadenas (cosa que sospechan muchos alumnos de las escuelas) y sin duda una cosa que impulsó al trabajo de Gödel.

Demostración. Suponga que no. Va a implicar la computabilidad del problema de la parada de una máquina de Turing, en contradicción a lo que vimos en el capítulo de autómatas. Podemos saber si una cadena de caracteres es una computación de una máquina dada, y si la primera es o no una configuración inicial y la última una parada. Entonces hay un predicado $P(x)$ que representa eso, y el problema de la parada entonces es si $\exists x P(x)$ o no. Según la negación

¹Yo como joven estudiante pregunté a varios capos de la lógica cuál era la noción central de *recursivo primitivo*, y no recibí una respuesta satisfactoria. Después descubrí en un libro de Minsky que representaba aquellos programas que en lugar de la construcción “mientras *condición* repetir...” usaba solamente “repetir n veces...”

del teorema o $\vdash \exists xP(x)$ o $\vdash \sim \exists xP(x)$. Si $\vdash \sim \exists xP(x)$ pero sin embargo hay algún \mathbf{x} tal que $P(\mathbf{x})$ entonces la teoría no sería consistente, entonces deberíamos creerlo. Si $\vdash \exists xP(x)$ deberíamos creer eso también porque si de hecho todo \mathbf{x} falla en cumplir $P(\mathbf{x})$ la teoría no sería ω -consistente. Así, esta búsqueda de una demostración o la otra tiene que terminar y si la teoría es ω -consistente tiene que decirnos la verdad cuando termina. Por tanto, constituye un procedimiento efectivo para decidir si una máquina dada de Turing para o no, cosa que hemos visto no existe. \square

5.4. La Demostración de Gödel. Gödel hizo un poco más: Nuestra demostración no produce una sentencia que es indecidible en la teoría; sólo muestra que tiene que existir, pero Gödel produjo una. Vamos a bosquejar cómo.

Dada una cadena sobre el alfabeto de la teoría (incluye \leftarrow para facilitar la escritura de las demostraciones) sea, para cada cadena c sobre ese alfabeto, G_c el número de esa cadena como definimos anteriormente, y sea $G_{G_c}^{-1} = c$ para cada c .

Definimos la función $sub(n, m)$ como sigue: si $n = G_F$ para una fórmula F (es decir si $G_n^{-1} = F$) sea H la fórmula obtenida sustituyendo m para cada ocurrencia de una variable libre en F . Entonces $sub(n, m) = G_H$. En otras palabras, $sub(n, m)$ es el número de la fórmula que resulta sustituyendo el número m para cada ocurrencia de una variable libre en la fórmula G_n^{-1} .

LEMA (Diagonal). Sea $C(x)$ una fórmula con una sola variable libre x . Entonces existe una fórmula B tal que

$$\vdash B \leftrightarrow C(G_B).$$

Demostración. Sea $D(x) = C(sub(x, x))$, y $B = D(G_D)$. Tenemos

$$B = D(G_D) = C(sub(G_D, G_D))$$

por un lado, y por el otro, que $sub(G_D, G_D)$ es G_F donde F es la fórmula que resulta de meter el número G_D en la fórmula D . Pero ésa es la definición de B , así que $sub(G_D, G_D) = G_B$ y por lo tanto $C(sub(G_D, G_D)) = C(G_B)$, es decir, B y $C(G_B)$ son la misma fórmula, y la equivalencia que tanto deseamos es nada más que una tautología.

Uno puede tener la impresión que lo están engañando con este argumento, entonces vamos a verlo en acción en un ejemplo. Sea C la fórmula $x = 0$. Pero vamos a suponer, lo que es factible, que $A = \{0, x', (,), ', \forall, \sim, =, \leftarrow\}$ con códigos

$$\begin{array}{cccccccccc} 0 & x' & (&) & , & \forall & \sim & = & \leftarrow \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

Este esquema, como la mayoría de los lenguajes de programación, permite que los identificadores $x, x', x'' \dots$ pueden representar variables, funciones, o símbolos de predicados: si es sin paréntesis es variable, si es función o predicado de n lugares tiene paréntesis con n argumentos, y si es función o predicado es decidable del contexto. Entonces $D(x) = C(sub(x, x))$, si representamos sub con x' y C con x'' es $x''(x'(x, x))$ y $G_D = 122315144$. Ahora $B = D(G_D) = C(sub(G_D, G_D))$; entonces $B = C(sub(12231544, 12231544))$. Ahora para calcular $sub(12231544, 12231544)$ tomamos la fórmula $G_{12231544}^{-1}$ y reemplazamos en ella 12231544 para cada ocurrencia de su variable libre. Pero $G_{12231544}^{-1}$ es D , entonces la fórmula que resulta es $D(G_D)$.

Entonces, demostrar $B \leftrightarrow C(G_B)$ es demostrar una tautología de la forma $F \leftrightarrow F$. \square

Ahora para la sentencia indecidible, sea $C(x) = \sim \exists y(d(x, y))$. Es decir, C representa la propiedad de no ser el número G_T de un teorema T . La sentencia indecidible es la fórmula B que nos provee el Lema Diagonal con la propiedad $\vdash B \leftrightarrow C(G_B)$. Si $\vdash B$ sea \mathbf{s} el número de su demostración, es decir $\mathbf{s} = G_s$ donde s es la cadena de la demostración. Entonces $\vdash d(G_B, \mathbf{s})$, y al mismo tiempo $\vdash C(G_B)$ que dice que $\sim \exists y(d(G_B, y))$, entonces la teoría no sería consistente. Si $\vdash \sim B$ entonces $\vdash \sim C(G_B)$, por tanto $\vdash \exists y(d(x, y))$. Puesto que d representa una propiedad computable, tiene que ser $\vdash \sim d(G_B, \mathbf{s})$ para cada número \mathbf{s} , para evitar inconsistencia con la demostración de $\sim B$. Pero esto mostraría que la teoría no es ω -consistente.

Notemos una cosa interesante: Ya que la teoría es consistente y B no es decidible, podemos agregar o B o $\sim B$ como nuevo axioma sin destruir consistencia. Pero B asegura que no existe una demostración de ella, y hemos demostrado que ésa es verdadera. Si admitimos $\sim B$ como axioma estamos admitiendo $\exists y(d(G_B, y))$, entonces tenemos ahora una teoría consistente pero no ω -consistente.

Ejercicios

1. Si sacamos el axioma que dice que 0 no es sucesor de nadie, ¿qué sistema además de los números naturales cumple el resto de los axiomas?
2. Defina el predicado $x > y$ en el lenguaje de la teoría de primer orden de números. Demuestre algunas propiedades.
3. Suponga que $\exists xS(x)$ no es decidible pero $\sim S(\{\mathbf{n}\})$ es demostrable para cada número $\{\mathbf{n}\}$. (“números” son $0, 0', 0'', \dots$) Demostrar que $\forall xS(x) \rightarrow x > \{\mathbf{n}\}$ para cualquier número $\{\mathbf{n}\}$.
4. Haga una máquina de Turing para el predicado $x > y$ (por ejemplo, que acepta $\{1^x 0^y : x > y\}$). Conviértalo en predicado en la teoría de primer orden de números por el método del texto aplicado al predicado $x = y$.
5. Demuestre que ω -consistencia implica consistencia.
6. Complete la demostración del teorema de la representación, definiendo una función del primer orden de la aritmética que transforma la upla (x_1, \dots, x_k) en un número que es \hat{w} donde $w = 1^{x_1} 21^{x_2} 2 \dots 21^{x_k}$. Sugerencia: