A1 Apéndice al capítulo 2

Simplificación de funciones por el método de Quine-McCluskey

Sin duda, los mapas de Karnaugh constituyen un método sencillo y eficaz para la simplificación «a mano» de funciones de pocas variables; es un método gráfico que facilita la búsqueda «visual» de términos simplificables entre sí. Pero su aplicación a funciones de más de 6 variables resulta muy laboriosa y desborda la capacidad de «agrupación visual» de términos de un operador humano. En tales casos resulta conveniente disponer de métodos de simplificación automática con ayuda de computador.

Ahora bien, los métodos gráficos presentan una cierta complejidad a la hora de traducirlos en algoritmos para ser ejecutados en computador y dicha dificultad aumenta fuertemente con el número de «dimensiones» a tener en cuenta. En el caso de los mapas de Karnaugh la hoja básica de 4 variables es bidimensional y es necesario añadir una dimensión más (una nueva hoja del mapa) por cada nueva variable: para **n** variables, el número de dimensiones efectivas del mapa de Karnaugh será de **n-2**.

En cambio, existen algoritmos de simplificación de actuación «lineal» (comparación sucesiva de vectores de entrada) cuya programación sobre computador es directa. Casi todos ellos se basan en el método de simplificación desarrollado por W. V. Quine en 1955 y ampliado posteriormente por E. J. McCluskey.

Esquemáticamente este método consiste en agrupar los vectores de entrada que activan la función (dan resultado **1** para ella) en clases diferenciadas por el número de variables cuyo valor sea **1** y calcular todas las posibilidades de simplificación entre vectores de dos clases sucesivas, eligiendo aquellas que son más eficientes.

Quizás la manera más sencilla de comprender este método de simplificación sea su aplicación a un caso concreto:

*Sea la función "ser número primo" para números binarios de 5 dígitos **edcba**. Los vectores de entrada que la activan son los siguientes:*

1 = ***00001*** *2 =* ***00010*** *3 =* ***00011*** *5 =* ***00101*** *7 =* ***00111***
11 = ***01011*** *13 =* ***01101*** *17 =* ***10001*** *19 =* ***10011***
23 = ***10111*** *29 =* ***11101***
31 = ***11111***

Tomás Pollán Santamaría. Tecnología Electrónica. E.U.I.T.I.Z. Universidad de Zaragoza. tpollan@posta.unizar.es

Separados dichos vectores de entrada en clases según el número de unos que contienen y numerados sucesivamente resulta:

1	**0 0 0 0 1**
2	**0 0 0 1 0**
---	---
3	**0 0 0 1 1**
4	**0 0 1 0 1**
5	**1 0 0 0 1**
---	---
6	**0 0 1 1 1**
7	**0 1 0 1 1**
8	**0 1 1 0 1**
9	**1 0 0 1 1**
---	---
10	**1 0 1 1 1**
11	**1 1 1 0 1**
---	---
12	**1 1 1 1 1**

*Primera simplificación: cada vector se compara con todos los de la clase siguiente, seleccionando aquellos que tienen valores **1** en la misma posición que dicho vector y por tanto se diferencian en una sola variable; esta variable es eliminable (*$a + \bar{a} = 1$*) y se indica con "–" su ausencia en el término resultante:*

1;3	**0 0 0 – 1**	
1;4	**0 0 – 0 1**	
1;5	**– 0 0 0 1**	
2;3	**0 0 0 1 –**	
---	---	
3;6	**0 0 – 1 1**	
3;7	**0 – 0 1 1**	
3;9	**– 0 0 1 1**	
4;6	**0 0 1 – 1**	
4;8	**0 – 1 0 1**	
5;9	**1 0 0 – 1**	
---	---	
6;10	**– 0 1 1 1**	
7;	---	*(1) este término no es simplificable con la clase siguiente,*
8;11	**– 1 1 0 1**	*pero no es preciso mantenerlo*
9;10	**1 0 – 1 1**	*ya que ha sido recogido al simplificar la clase anterior.*
---	---	
10;12	**1 – 1 1 1**	
11;12	**1 1 1 – 1**	

Segunda simplificación: se produce en forma análoga a la primera pero solamente son simplificables aquellos vectores cuyos valores "–" coinciden, difiriendo en una de las restantes variables.

1,3; 4,6	**0 0 – – 1**	
1,3; 5,9	**– 0 0 – 1**	
1,4; 3,6	**0 0 – – 1**	
1,5; 3,9	**– 0 0 – 1**	
2,3;	**0 0 0 1 –**	*(2) término no simplificable con la clase siguiente; es preciso mantenerlo, ya que no ha sido recogido en los términos anteriores.*
---	---	
3,6; 9,10	**– 0 – 1 1**	
3,7;	**0 – 0 1 1**	*(2)*
3,9; 6,10	**– 0 – 1 1**	
4,6;	----------	*(1) término simplificado con la clase anterior*
4,8;	**0 – 1 0 1**	*(2)*
5,9;	----------	*(1)*
---	---	
6,10;	----------	*(1)*
8,11;	**– 1 1 0 1**	*(2)*
9,10;	----------	*(1)*
---	---	
10,12;	**1 – 1 1 1**	*(2)*
11,12;	**1 1 1 – 1**	*(2)*

Agrupando términos iguales en cada una de las clases:

1,3,4,6	**0 0 – – 1**	**A**
1,3,5,9	**– 0 0 – 1**	**B**
2,3	**0 0 0 1 –**	**C**
---	---	
3,6,9,10	**– 0 – 1 1**	**D**
3,7	**0 – 0 1 1**	**E**
4,8	**0 – 1 0 1**	**F**
---	---	
8,11;	**– 1 1 0 1**	**G**
---	---	
10,12;	**1 – 1 1 1**	**H**
11,12;	**1 1 1 – 1**	**I**

No es posible una nueva simplificación de estos términos, pues en ningún caso coinciden las "–" de vectores de dos clases sucesivas.

Tabla de cobertura: finalizado el proceso de simplificación se construye la tabla de cobertura de los vectores iniciales por los términos resultantes de la simplificación.

	Términos								
vectores iniciales	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
1	**x**	**x**							
2			**x**						
3	**x**	**x**	**x**	**x**	**x**				
4	**x**					**x**			
5		**x**							
6	**x**			**x**					
7					**x**				
8						**x**	**x**		
9		**x**		**x**					
10				**x**				**x**	
11							**x**		**x**
12								**x**	**x**

Se trata de elegir en la tabla de cobertura los términos de forma que con el mínimo número de ellos se cubran todos los vectores iniciales; para hacer esta selección se comienza por los casos en que el vector inicial solamente es cubierto por uno de los términos resultantes y, por tanto, dicho término ha de ser necesariamente recogido (como por ejemplo, para los vectores 2, 5 y 7, que requieren los términos **C**, **B** *y* **E**, *respectivamente).*

En la tabla anterior el subconjunto: **A**, **B**, **C**, **E**, **G** *y* **H** *cubre todos los vectores iniciales:* y = **A** + **B** + **C** + **E** + **G** + **H**

$$y = \bar{e}.\bar{d}.a + \bar{d}.\bar{c}.a + \bar{e}.\bar{d}.\bar{c}.b + \bar{e}.\bar{c}.b.a + d.c.\bar{b}.a + e.c.b.a$$

La aplicación del método de Quine-McCluskey, hecha «a mano», resulta bastante pesada y la selección de términos en el mapa de cobertura puede ser compleja, pero, en cambio, este método resulta muy apropiado para su programación en computador; dicha programación viene facilitada por el hecho de que la diferencia aritmética de los valores decimales correspondientes a dos vectores simplificables es una potencia entera de 2.

En el caso de funciones con vectores de entrada para los que «no importa» (*don´t care*) el valor booleano de salida que se les asigne, dichos vectores son tomados también inicialmente, pero se prescinde de ellos en la tabla de cobertura.

Por ejemplo, en el caso de aplicar la función "ser número primo" a las cifras decimales, el proceso de simplificación sería el siguiente:

cifras que son números primos:

1 = ***0001*** *2 =* ***0010*** *3 =* ***0011*** *5 =* ***0101*** *7 =* ***0111***

vectores de entrada que no se van a presentar (pues no corresponden a cifras decimales):

10 = ***1010*** *11 =* ***1011*** *12 =* ***1100*** *13 =* ***1101*** *14 =* ***1110*** *15 =* ***1111***

Estableciendo las clases según el número de unos que tales vectores contienen y numerados sucesivamente resulta:

1	**0 0 0 1**
2	**0 0 1 0**
---	---
3	**0 0 1 1**
4	**0 1 0 1**
(5)	**1 0 1 0**
(6)	**1 1 0 0**
---	---
7	**0 1 1 1**
(8)	**1 0 1 1**
(9)	**1 1 0 1**
(10)	**1 1 1 0**
---	---
(11)	**1 1 1 1**

Se han indicado entre paréntesis aquellos vectores de entrada para los cuales es indiferente (no importa) la salida que se asigne; dichos vectores se toman sólo a efectos de simplificación.

Primera simplificación:

1;3	**0 0 - *1***	*3;7*	**0 - 1 1**	*7;(11)*	**- 1 1 1**
1;4	**0 – 0 1**	*3;(8)*	**- 0 1 1**	*(8);(11)*	**1 - 1 1**
2;3	**0 0 1 –**	*4;7*	**0 1 - 1**	*(9);(11)*	**1 1 - 1**
2;(5)	**– 0 1 0**	*4;(9)*	**- 1 0 1**	*(10);(11)*	**1 1 1 -**
---		*(5);(8)*	**1 0 1 -**		
		(5);(10)	**1 - 1 0**		
		(6);(9)	**1 1 0 -**		
		(6);(10)	**1 1 - 0**		

Segunda simplificación:

1,3; 4,7	**0 – – 1**	*3,7; (8),(11)*	**– – 1 1**
1,4; 3,7	**0 – – 1**	*3,(8); 7,(11)*	**– – 1 1**
2,3; (5),(8)	**– 0 1 –**	*4,7 ;(9),(11)*	**– 1 – 1**
2,(5); 3,(8)	**– 0 1 –**	*4,(9); 7,(11)*	**– 1 – 1**
---		*(5),(8); (10),(11)*	**1 – 1 –**
		(5),(10); (8),(11)	**1 – 1 –**
		(6),(9); (10),(11)	**1 1 – –**
		(6),(10); (9),(11)	**1 1 – –**

Agrupando términos iguales en cada una de las clases:

1,3,4,7	**0 – – 1**	**A**
2,3,(5),(8)	**– 0 1 –**	**B**
---	---	
3,7,(8),(11)	**– – 1 1**	**C**
4,7,(9),(11)	**– 1 – 1**	**D**
(5),(8),(10),(11)	**1 – 1 –**	
(6),(10),(9),(11)	**1 1 – –**	

No es posible una nueva simplificación de estos términos, pues en ningún caso coinciden las "–" de vectores de dos clases sucesivas. Tampoco es necesario recoger los últimos dos términos porque solamente cubren a vectores de entrada para los que es indiferente la salida que se produzca.

La tabla de cobertura para los vectores que interesan (1, 2, 3, 4, 7) es la siguiente:

Términos

Vectores iniciales	**A**	**B**	**C**	**D**
1	**x**			
2		**x**		
3	**x**	**x**	**x**	
4	**x**			**x**
7	**x**		**x**	**x**

Necesariamente ha de tomarse el término **B** *para cubrir el vector 2 y junto con el término* **A** *cubren todos los vectores:* $y = \bar{d}.a + \bar{c}.b$

El método de Quine-McCluskey ha sido extendido de forma que no sea necesario conocer los términos mínimos o vectores de entrada que activan la función, sino que pueda aplicarse a partir de cualquier expresión de la función como suma de términos producto, es decir, a partir de los subconjuntos de valores de las entradas que hacen la función **1**.

Denominemos subvector de entrada a cualquier subconjunto de valores de las entradas que no incluya a todas ellas (por ejemplo, para 4 variables **dcba**: **00-1**, **-101, 1-0-**, **-1-1**, **-0--**,…). A cada subvector de entrada le corresponde un término producto, que, obviamente, no contiene todas las variables de entrada (en el ejemplo anterior: $\bar{d}.\bar{c}.a$, $c.\bar{b}.a$, $d.\bar{b}$, $c.a$, $\bar{c}$,…).

En general, en funciones cuyo número de variables no sea pequeño resulta mucho más sencillo conocer y anotar un conjunto completo de los subvectores de entrada que la activan que detallar todos los vectores de entrada que dan resultado **1**, ya que el número de los segundos puede ser muy superior al de los primeros.

*Por ejemplo, un codificador de prioridad de 7 líneas de entrada (numeradas de 1 a 7), tal que en su salida señale el número de la línea superior que se encuentre activada, estará configurado por 3 funciones C B A de 7 variables **g f e d c b a**, con 128 términos mínimos posibles. Pero los únicos subvectores de entrada que interesan son los siguientes (se expresa al lado de cada uno el vector de salida que debe producir):*

	C B A
1 **- - - - - -**	***1 1 1***
0 1 **- - - - -**	***1 1 0***
0 0 1 **- - - -**	***1 0 1***
0 0 0 1 **- - -**	***1 0 0***
0 0 0 0 1 **- -**	***0 1 1***
0 0 0 0 0 1 **-**	***0 1 0***
0 0 0 0 0 0 1	***0 0 1***
0 0 0 0 0 0 0	***0 0 0***

El procedimiento de aplicación del método de Quine-McCluskey a partir de un conjunto completo de subvectores de entrada que activan la función es diferente del indicado anteriormente. En este caso es preciso comparar cada subvector con todos los demás, para detectar aquellos casos en que un subvector es simplificable con otro o está contenido en él (resulta cubierto por el otro subvector); además, es preciso aplicar un teorema de consenso entre los subvectores, cuando ello sea posible, para generar nuevos subvectores que también activan la función y que pudieran facilitar la simplificación.

El teorema de consenso entre subvectores puede ser enunciado en la siguiente forma: si $a.\alpha$ y $\bar{a}.\beta$ son términos producto que corresponden a subvectores de entrada que activan la función **y**, también activa dicha función el subvector correspondiente al término $\alpha.\beta$ (si lo hubiere, pues $\alpha.\beta$ puede ser nulo como resultado de incluir una variable y su negada).

*En el caso del codificador de prioridad de 7 líneas de entrada la primera de sus tres funciones booleanas C da valor **1** para los cuatro primeros subvectores, que son los únicos que activan dicha función:*

1 - - - - - - **0 1 - - - - -** **0 0 1 - - - -** **0 0 0 1 - - -**

Por simplificaciones sucesivas se obtienen los siguientes subvectores que cubren a los anteriores:

1 - - - - - - **- 1 - - - - -** **- - 1 - - - -** **- - - 1 - - -**

$$C = g + f + d + c$$

*La segunda de las funciones B da valor **1** para el siguiente conjunto de subvectores, el cual resulta completo, es decir, cubre todas las posibilidades de activar dicha función:*

1 - - - - - - **0 1 - - - - -** **0 0 0 0 1 - -** **0 0 0 0 0 1 -**

Por simplificaciones sucesivas se llega a los siguientes subvectores:

1 - - - - - - **- 1 - - - - -** **- - 0 0 1 - -** **- - 0 0 - 1 -**

$$B = g + f + \bar{e}.\bar{d}.c + \bar{e}.\bar{d}.b = g + f + \bar{e}.\bar{d}.(c + b)$$

*La tercera función A da valor **1** para el siguiente conjunto de subvectores:*

1 - - - - - - **0 0 1 - - - -** **0 0 0 0 1 - -** **0 0 0 0 0 0 1**

Simplificando se obtienen los siguientes subvectores:

1 - - - - - - **- 0 1 - - - -** **- 0 - 0 1 - -** **- 0 - 0 - 0 1**

$$A = g + \bar{f}.e + \bar{f}.\bar{d}.c + \bar{f}.\bar{d}.\bar{b}.a = g + \bar{f}.[e + \bar{d}.(c + \bar{b}.a)]$$

Una excelente presentación del método de Quine-McCluskey generalizado, de su formulación algorítmica y de su programación informática se encuentra en el libro de J.F. Wakerley, *Digital Design. Principles and Practices* (Prentice-Hall International Editions. 2000. 3ª edición. Páginas 236 - 243).

Por otra parte, el método de Quine-McCluskey también permite abordar la simplificación conjunta de múltiples funciones de las mismas variables, de forma que se minimice el número total de términos producto necesarios (y no el número de términos producto para cada una de ellas): *simplificación multifunción*.