可以设法用卡诺图来表示任意一个逻辑函数。具体的方法是:首先将逻辑函数化为最小项之和的形式,然后在卡诺图上与这些最小项对应的位置上填入**1**,在其余的位置上填入**0**,就得到了表示该逻辑函数的卡诺图。也就是说,任何一个逻辑函数都等于它的卡诺图中填入**1**的那些最小项之和。

【例 2.6.8】　用卡诺图表示逻辑函数

$$Y = A'B'C'D + A'BD' + ACD + AB'$$

解:　首先将 Y 化为最小项之和的形式

$$\begin{aligned} Y &= A'B'C'D + A'B(C + C')D' + A(B + B')CD + AB'(C + C')(D + D') \\ &= A'B'C'D + A'BCD' + A'BC'D' + ABCD + AB'CD + AB'CD' + AB'C'D \\ &\quad + AB'C'D' \\ &= m_1 + m_4 + m_6 + m_8 + m_9 + m_{10} + m_{11} + m_{15} \end{aligned}$$

画出四变量最小项的卡诺图,在对应于函数式中各最小项的位置上填入**1**,其余位置上填入**0**,就得到如图 2.6.2 所示的函数 Y 的卡诺图。

AB \ *CD*	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	0	0	1	0
10	1	1	1	1

图 2.6.2　例 2.6.8 的卡诺图

A \ *BC*	00	01	11	10
0	0	1	0	1
1	1	0	1	0

图 2.6.3　例 2.6.9 的卡诺图

【例 2.6.9】　已知逻辑函数 Y 的卡诺图如图 2.6.3 所示,试写出该函数的逻辑式。

解:　因为函数 Y 等于卡诺图中填入**1**的那些最小项之和,所以有

$$Y = AB'C' + A'B'C + ABC + A'BC'$$

二、用卡诺图化简逻辑函数

利用卡诺图化简逻辑函数的方法称为卡诺图化简法或图形化简法。化简时依据的基本原理就是具有相邻性的最小项可以合并,并消去不同的因子。由于在卡诺图上几何位置相邻与逻辑上的相邻性是一致的,因而从卡诺图上能直观地找出那些具有相邻性的最小项并将其合并化简。

1. 合并最小项的原则

若两个最小项相邻,则可合并为一项并消去一对因子。合并后的结果中只剩下公共因子。

在图 2.6.4(a)和(b)中画出了两个最小项相邻的几种可能情况。例如,图(a)中 $A'BC(m_3)$ 和 $ABC(m_7)$ 相邻,故可合并为

$$A'BC + ABC = (A' + A)BC = BC$$

合并后将 A 和 A' 一对因子消掉了,只剩下公共因子 B 和 C。

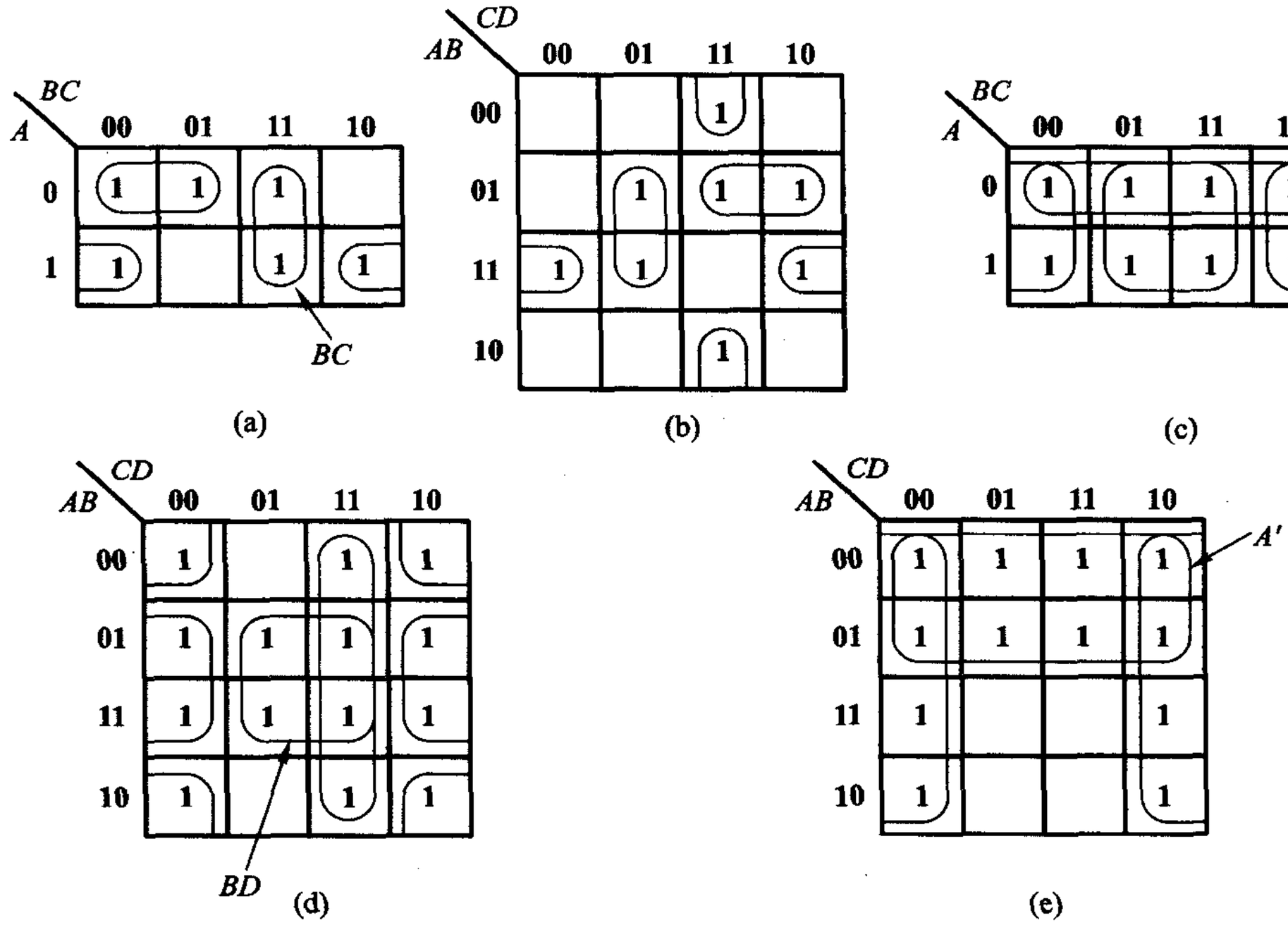

图 2.6.4 最小项相邻的几种情况

(a)、(b) 两个最小项相邻 (c)、(d) 四个最小项相邻 (e) 八个最小项相邻

若四个最小项相邻并排列成一个矩形组,则可合并为一项并消去两对因子。合并后的结果中只包含公共因子。

例如,在图 2.6.4(d)中,$A'BC'D(m_5)$、$A'BCD(m_7)$、$ABC'D(m_{13})$ 和 $ABCD(m_{15})$ 相邻,故可合并。合并后得到

$$\begin{aligned} &A'BC'D + A'BCD + ABC'D + ABCD \\ &= A'BD(C + C') + ABD(C + C') \\ &= BD(A + A') = BD \end{aligned}$$

可见,合并后消去了 A、A' 和 C、C' 两对因子,只剩下四个最小项的公共因子 B

和 D。

若八个最小项相邻并且排列成一个矩形组，则可合并为一项并消去三对因子。合并后的结果中只包含公共因子。

例如，在图 2.6.4(e)中，上边两行的八个最小项是相邻的，可将它们合并为一项 A'。其他的因子都被消去了。

至此，可以归纳出合并最小项的一般规则，这就是：如果有 2^n 个最小项相邻（$n=1,2,\cdots$）并排列成一个矩形组，则它们可以合并为一项，并消去 n 对因子。合并后的结果中仅包含这些最小项的公共因子。

2. 卡诺图化简法的步骤

用卡诺图化简逻辑函数时可按如下步骤进行：

（1）将函数化为最小项之和的形式。

（2）画出表示该逻辑函数的卡诺图。

（3）找出可以合并的最小项。

（4）选取化简后的乘积项。选取的原则是：

① 这些乘积项应包含函数式中所有的最小项（应复盖卡诺图中所有的 1）。

② 所用的乘积项数目最少。也就是可合并的最小项组成的矩形组数目最少。

③ 每个乘积项包含的因子最少。也就是每个可合并的最小项矩形组中应包含尽量多的最小项。

【例 2.6.10】 用卡诺图化简法将下式化简为最简**与或**函数式

$$Y = AC' + A'C + BC' + B'C$$

解： 首先画出表示函数 Y 的卡诺图，如图 2.6.5 所示。

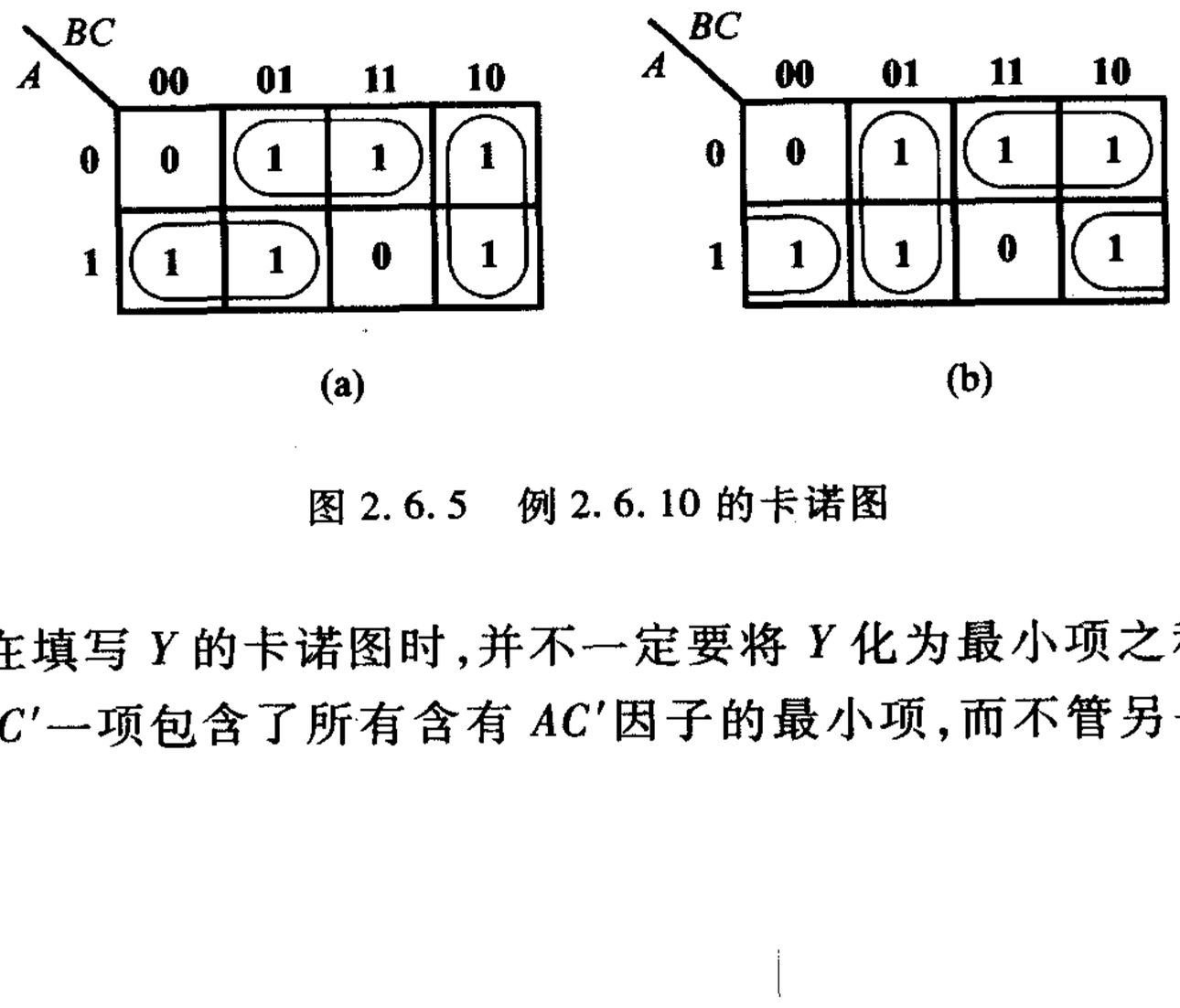

图 2.6.5　例 2.6.10 的卡诺图

事实上在填写 Y 的卡诺图时，并不一定要将 Y 化为最小项之和的形式。例如，式中的 AC' 一项包含了所有含有 AC' 因子的最小项，而不管另一个因子是 B

还是 B'。从另外一个角度讲,也可以理解为 AC' 是 ABC' 和 $AB'C'$ 两个最小项相加合并的结果。因此,在填写 Y 的卡诺图时,可以直接在卡诺图上所有对应 $A=1$、$C=0$ 的空格里填入 1。按照这种方法,就可以省去将 Y 化为最小项之和这一步骤了。

其次,需要找出可以合并的最小项。将可能合并的最小项用线圈出。由图 2.6.5(a)和(b)可见,有两种可取的合并最小项的方案。如果按图 2.6.5(a)的方案合并最小项,则得到

$$Y = AB' + A'C + BC'$$

而按图 2.6.5(b)的方案合并最小项得到

$$Y = AC' + B'C + A'B$$

两个化简结果都符合最简**与或**式的标准。

此例说明,有时一个逻辑函数的化简结果不是唯一的。

【例 2.6.11】 用卡诺图化简法将下式化为最简**与或**逻辑式

$$Y = ABC + ABD + AC'D + C'D' + AB'C + A'CD'$$

解: 首先画出 Y 的卡诺图,如图 2.6.6 所示。然后将可能合并的最小项圈出,并按照前面所述的原则选择化简后**与或**式中的乘积项。由图可见,应将图中下边两行的 8 个最小项合并,同时将左、右两列最小项合并,于是得到

$$Y = A + D'$$

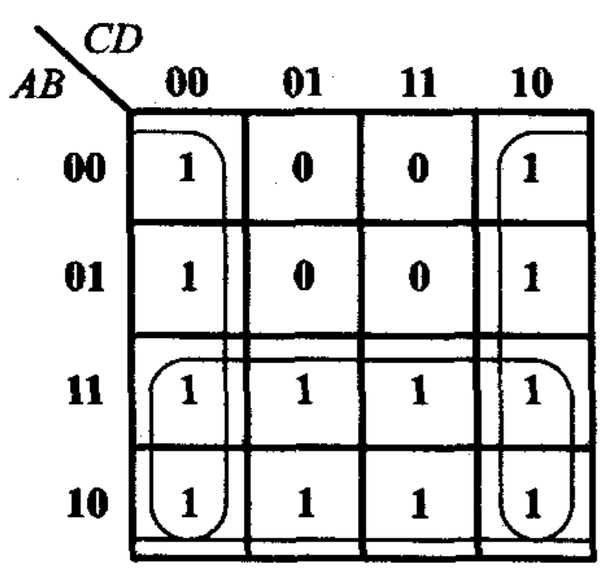

图 2.6.6 例 2.6.11 的卡诺图

从图 2.6.6 中可以看到,A 和 D' 中重复包含了 m_8、m_{10}、m_{12} 和 m_{14} 这 4 个最小项。但据 $A+A=A$ 可知,在合并最小项的过程中允许重复使用函数式中的最小项,以利于得到更简单的化简结果。

另外,还要补充说明一个问题。在以上的两个例子中,我们都是通过合并卡诺图中的 **1** 来求得化简结果的。但有时也可以通过合并卡诺图中的 **0** 先求出 Y' 的化简结果,然后再将 Y' 求反而得到 Y。

这种方法所依据的原理我们已在 2.5.4 节中做过说明。因为全部最小项之和为 **1**,所以若将全部最小项之和分成两部分,一部分(卡诺图中填入 **1** 的那些最小项)之和记作 Y,则根据 $Y+Y'=1$ 可知,其余一部分(卡诺图中填入 **0** 的那些最小项)之和必为 Y'。

在多变量逻辑函数的卡诺图中,当 **0** 的数目远小于 **1** 的数目时,采用合并 **0** 的方法有时会比合并 **1** 来得简单。例如,在图 2.6.6 所示的卡诺图中,如果将 **0**

合并，则可立即写出

$$Y' = A'D, \quad Y = ((Y'))' = (A'D)' = A + D'$$

与合并 **1** 得到的化简结果一致。

此外，在需要将函数化为最简的**与或非**式时，采用合并 **0** 的方式最为适宜，因为得到的结果正是**与或非**形式。如果要求得到 Y'的化简结果，则采用合并 **0** 的方式就更简便了。

*2.6.3　奎恩－麦克拉斯基化简法（Q－M 法）

从上一小节的内容中不难看出，虽然卡诺图化简法具有直观、简单的优点，但它同时又存在着很大的局限性。首先，在函数的输入逻辑变量较多时（例如大于 5 以后），便失掉了直观的优点。其次，在许多情况下要凭设计者的经验确定应如何合并最小项才能得到最简单的化简结果，因而不便于借助计算机完成化简工作。

公式化简法的使用虽然不受输入变量数目的影响，但由于化简的过程没有固定的、通用的步骤可循，所以同样不适用于计算机辅助化简。

由奎恩（W. V. Quine）和麦克拉斯基（E. J. McCluskey）提出的用列表方式进行化简的方法则有一定的规则和步骤可循，较好地克服了公式化简法和卡诺图化简法在这方面的局限性，因而适用于编制计算机辅助化简程序。通常将这种化简方法称为奎恩－麦克拉斯基法，简称 Q－M 法。

Q－M 法的基本原理仍然是通过合并相邻最小项并消去多余因子而求得逻辑函数的最简**与或**式。下面再结合一个具体的例子简要地介绍一下 Q－M 法的基本原理和化简的步骤。

假定需要化简的五变量逻辑函数为

$$\begin{aligned} Y(A,B,C,D,E) = {} & AB'CDE' + A'C'D'E' + A'B'C'D + A'BDE' \\ & + BCDE + ABC'(D \oplus E)' \end{aligned} \tag{2.6.3}$$

则使用 Q－M 法的化简步骤如下：

（1）将函数化为最小项之和形式，列出最小项编码表。

将式(2.6.3)化为最小项之和形式后得到

$$\begin{aligned} Y(A,B,C,D,E) = {} & A'B'C'D'E' + A'B'C'DE' + A'B'C'DE + A'BC'D'E' \\ & + A'BC'DE' + A'BCDE' + A'BCDE + AB'CDE' + ABC'D'E' \\ & + ABC'DE + ABCDE \\ = {} & \sum m(0,2,3,8,10,14,15,22,24,27,31) \end{aligned} \tag{2.6.4}$$

用 **1** 表示最小项中的原变量，用 **0** 表示最小项中的反变量，就得到了表 2.6.1 所示的最小项编码表。

表 2.6.1 式(2.6.4)最小项的编码表

最小项编号	0	2	3	8	10	14
代码	00000	00010	00011	01000	01010	01110
最小项编号	15	22	24	27	31	
代码	01111	10110	11000	11011	11111	

(2) 按包含 1 的个数将最小项分组,如表 2.6.2 中最左边一列所示。

表 2.6.2 列表合并最小项

合并前的最小项 ($\sum m_i$)							第一次合并结果 (含 $n-1$ 个变量的乘积项)							第二次合并结果 (含 $n-2$ 个变量的乘积项)						
编号	A	B	C	D	E		编号	A	B	C	D	E		编号	A	B	C	D	E	
0	0	0	0	0	0	√	0,2	0	0	0	—	0	√	0,2 8,10	0	—	0	—	0	P_8
2	0	0	0	1	0	√	0,8	0	—	0	0	0	√							
8	0	1	0	0	0	√	2,3	0	0	0	1	—	P_2	0,8 2,10	0	—	0	—	0	P_8
3	0	0	0	1	1	√	2,10	0	—	0	1	0	√							
10	0	1	0	1	0	√	8,10	0	1	0	—	0	√							
24	1	1	0	0	0	√	8,24	—	1	0	0	0	P_3							
14	0	1	1	1	0	√	10,14	0	1	—	1	0	P_4							
22	1	0	1	1	0	P_1	14,15	0	1	1	1	—	P_5							
15	0	1	1	1	1	√	15,31	—	1	1	1	1	P_6							
27	1	1	0	1	1	√	27,31	1	1	—	1	1	P_7							
31	1	1	1	1	1	√														

(3) 合并相邻的最小项。

将表 2.6.2 中最左边一列里每一组的每一个最小项与相邻组里所有的最小项逐一比较,若仅有一个因子不同,则定可合并,并消去不同的因子。消去的因子用“—”号表示,将合并后的结果列于表 2.6.2 的第二列中。同时,在第一列中可以合并的最小项右边标以“√”号。

按照同样的方法再将第二列中的乘积项合并,合并后的结果写在第三列中。

如此进行下去,直到不能再合并为止。

(4) 选择最少的乘积项。

只要将表 2.6.2 中合并过程中没有用过的那些乘积项相加，自然就包含了函数 Y 的全部最小项，故得

$$Y(A,B,C,D,E) = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 \qquad (2.6.5)$$

然而，上式并不一定是最简的**与或**表达式。为了进一步将式(2.6.5)化简，将 $P_1 \sim P_8$ 各包含的最小项列成表 2.6.3。因为表中带圆圈的最小项仅包含在一个乘积项中，所以化简结果中一定包含它们所在的这些乘积项，即 P_1、P_2、P_3、P_7 和 P_8。而且，选取了这五项之和以后，已包含了除 m_{14} 和 m_{15} 以外所有 Y 的最小项。

表 2.6.3 用列表法选择最少的乘积项

P_j \ m_i	0	2	3	8	10	14	15	22	24	27	31
P_1								①			
P_2		**1**	①								
P_3				**1**					①		
P_4					**1**	**1**					
P_5						**1**	**1**				
P_6							**1**				**1**
P_7										①	**1**
P_8	①	**1**		**1**	**1**						

剩下的问题就是要确定化简结果中是否应包含 P_4、P_5 和 P_6 了。为此，可将表 2.6.3 中有关 P_4、P_5、P_6 的部分简化成表 2.6.4 的形式。

表 2.6.4 表 2.6.3 的 P_4、P_5、P_6 部分

P_j \ m_i	14	15
P_4	**1**	
P_5	**1**	**1**
P_6		**1**

由表 2.6.4 中可以看到，P_4 行所有的 **1** 和 P_6 行所有的 **1** 皆与 P_5 中的 **1** 重叠，亦即 P_5 中的最小项包含了 P_4 和 P_6 的所有最小项，故可将 P_4 和 P_6 两行删掉。因此，可将式(2.6.5)中的 P_4 和 P_6 两项去掉，从而得到最后的化简结果

$$Y(A,B,C,D,E) = P_1 + P_2 + P_3 + P_5 + P_7 + P_8$$
$$= AB'CDE' + A'B'C'D + BC'D'E'$$
$$+ A'BCD + ABDE + A'C'E' \quad (2.6.6)$$

从上面的例子中可以看到，虽然 Q－M 法的化简过程看起来比较繁琐，但由于它有确定的流程，适用于任何复杂逻辑函数的化简，这就为编制计算机辅助化简程序提供了方便。因此，几乎很少有人用手工方法使用 Q－M 法去化简复杂的逻辑函数，而是使用基于 Q－M 法的基本原理去编制各种计算机软件，然后在计算机上完成逻辑函数的化简工作。

复习思考题

R2.6.1 卡诺图化简法所依据的基本原理是什么？

R2.6.2 卡诺图两侧变量取值的标注次序应遵守什么规则？

R2.6.3 Q－M 法所依据的基本原理是什么？

R2.6.4 公式化简法、卡诺图化简法、Q－M 化简法各有何优缺点？

2.7 具有无关项的逻辑函数及其化简

2.7.1 约束项、任意项和逻辑函数式中的无关项

在分析某些具体的逻辑函数时，经常会遇到这样一种情况，即输入变量的取值不是任意的。对输入变量取值所加的限制称为约束。同时，将这一组变量称为具有约束的一组变量。

例如，有三个逻辑变量 A、B、C，它们分别表示一台电动机的正转、反转和停止的命令，$A=1$表示正转，$B=1$表示反转，$C=1$表示停止。表示正转、反转和停止工作状态的逻辑函数可写成

$$Y_1 = AB'C' \quad (\text{正转}) \quad (2.7.1)$$
$$Y_2 = A'BC' \quad (\text{反转}) \quad (2.7.2)$$
$$Y_3 = A'B'C \quad (\text{停止}) \quad (2.7.3)$$

因为电动机任何时候只能执行其中的一个命令，所以不允许两个以上的变量同时为 1。ABC 的取值只可能是 **001**、**010**、**100** 当中的某一种，而不能是 **000**、**011**、**101**、**110**、**111** 中的任何一种。因此，A、B、C 是一组具有约束的变量。

通常用约束条件来描述约束的具体内容。显然，用上面的这样一段文字叙述约束条件是很不方便的，最好能用简单、明了的逻辑语言表述约束条件。

由于每一组输入变量的取值都使一个、而且仅有一个最小项的值为 **1**，所以当限制某些输入变量的取值不能出现时，可以用它们对应的最小项恒等于 **0** 来表示。这样，上面例子中的约束条件可以表示为

$$\begin{cases} A'B'C' = 0 \\ A'BC = 0 \\ AB'C = 0 \\ ABC' = 0 \\ ABC = 0 \end{cases}$$

或写成

$$A'B'C' + A'BC + AB'C + ABC' + ABC = 0$$

同时，将这些恒等于 **0** 的最小项称为函数 Y_1、Y_2 和 Y_3 的约束项。

在存在约束项的情况下，由于约束项的值始终等于 **0**，所以既可以将约束项写进逻辑函数式中，也可以将约束项从函数式中删掉，而不影响函数值。

有时还会遇到另外一种情况，就是在输入变量的某些取值下函数值是 **1** 还是 **0** 皆可，并不影响电路的功能。在这些变量取值下，其值等于 **1** 的那些最小项称为任意项。

我们仍以上面的电动机正转、反转和停止控制为例。如果电路设计成当 A、B、C 三个控制变量出现两个以上同时为 **1** 或者全部为 **0** 时电路能自动切断供电电源，那么这时 Y_1、Y_2 和 Y_3 等于 **1** 还是等于 **0** 已无关紧要，电动机肯定会受到保护而停止运行。例如，当出现 $A = B = C = 1$ 时，对应的最小项 $ABC(m_7) = 1$。如果把最小项 ABC 写入 Y_1 式中，则当 $A = B = C = 1$ 时 $Y_1 = 1$；如果没有把 ABC 这一项写入 Y_1 式中，则当 $A = B = C = 1$ 时 $Y_1 = 0$。因为这时 $Y_1 = 1$ 还是 $Y_1 = 0$ 都是允许的，所以既可以把 ABC 这个最小项写入 Y_1 式中，也可以不写入。因此，我们把 ABC 称为逻辑函数 Y_1 的任意项。同理，在这个例子中 $A'B'C'$、$A'BC$、$AB'C$、ABC' 也是 Y_1、Y_2 和 Y_3 的任意项。

因为使约束项的取值等于 **1** 的输入变量取值是不允许出现的，所以约束项的值始终为 **0**。而任意项则不同，在函数的运行过程中，有可能出现使任意项取值为 **1** 的输入变量取值。

我们将约束项和任意项统称为逻辑函数式中的无关项。这里所说的“无关”是指是否把这些最小项写入逻辑函数式无关紧要，可以写入也可以删除。

上一节中曾经讲到，在用卡诺图表示逻辑函数时，首先将函数化为最小项之和的形式，然后在卡诺图中这些最小项对应的位置上填入 **1**，其他位置上填入 **0**。既然可以认为无关项包含于函数式中，也可以认为不包含在函数式中，那么在卡

诺图中对应的位置上就可以填入**1**,也可以填入**0**。为此,在卡诺图中用×(或∅)表示无关项。在化简逻辑函数时既可以认为它是**1**,也可以认为它是**0**。

2.7.2 无关项在化简逻辑函数中的应用

化简具有无关项的逻辑函数时,如果能合理利用这些无关项,一般都可得到更加简单的化简结果。

为达到此目的,加入的无关项应与函数式中尽可能多的最小项(包括原有的最小项和已写入的无关项)具有逻辑相邻性。

合并最小项时,究竟把卡诺图中的×作为**1**(即认为函数式中包含了这个最小项)还是作为**0**(即认为函数式中不包含这个最小项)对待,应以得到的相邻最小项矩形组合最大、而且矩形组合数目最少为原则。

【例 2.7.1】 化简具有约束的逻辑函数

$$Y = A'B'C'D + A'BCD + AB'C'D'$$

给定约束条件为

$$A'B'CD + A'BC'D + ABC'D' + AB'C'D + ABCD + ABCD' + AB'CD' = \mathbf{0}$$

在用最小项之和形式表示上述具有约束的逻辑函数时,也可写成如下形式

$$Y(A,B,C,D) = \sum m(1,7,8) + d(3,5,9,10,12,14,15)$$

式中以 d 表示无关项,d 后面括号内的数字是无关项的最小项编号。

解: 如果不利用约束项,则 Y 已无可化简。但适当地加进一些约束项以后,可以得到

$$\begin{aligned} Y &= (A'B'C'D + \underbrace{A'B'CD}_{\text{约束项}}) + (A'BCD + \underbrace{A'BC'D}_{\text{约束项}}) \\ &\quad + (AB'C'D' + \underbrace{ABC'D'}_{\text{约束项}}) + (\underbrace{ABCD'}_{\text{约束项}} + \underbrace{AB'CD'}_{\text{约束项}}) \\ &= (A'B'D + A'BD) + (AC'D' + ACD') \\ &= A'D + AD' \end{aligned}$$

可见,利用了约束项以后,使逻辑函数得以进一步化简。但是,在确定该写入哪些约束项时尚不够直观。

如果改用卡诺图化简法,则只要将表示 Y 的卡诺图画出,就能从图上直观地判断对这些约束项应如何取舍。

图 2.7.1 是例 2.7.1 的逻辑函数的卡诺图。从图中不难看出,为了得到最大的相邻最小项的矩形组合,应取约束项 m_3、m_5 为**1**,与 m_1、m_7 组成一个矩形组。同时取约束项 m_{10}、m_{12}、m_{14} 为**1**,与 m_8 组成一个矩形组。将两组相邻的最小项合并后得到的化简结果与上面推演的结果相同。卡诺图中没有被圈进去的