2. Funkcje logiczne

2.1. Dwuelementowa algebra Boole'a

Nazwa algebra Boole'a pochodzi od nazwiska dziewiętnastowiecznego angielskiego filozofa i matematyka, który zaproponował formalny sposób opisu praw logiki Arystotelesa. *Dwuelementowa algebra Boole'a* jest sformalizowanym uogólnieniem rachunku zdań [12]. Formalnie jest definiowana jako system algebraiczny postaci

$$\langle B, +, \cdot, \bar{}, 0, 1 \rangle, \tag{2.1}$$

gdzie $B = \{0, 1\}$ to zbiór elementów algebry, $+$, $\cdot$ są symbolami dwuargumentowych operatorów alternatywy i koniunkcji, $\bar{}$ jest symbolem jednoargumentowej operacji negacji, a 0 i 1 to wyróżnione stałe algebry.

Zależnie od dziedziny zastosowań operator alternatywy jest również oznaczany jako $\cup$, $\vee$, operator koniunkcji jako $\cap$, $\wedge$, a operator negacji symbolami $\sim$, $\neg$. W informatyce powszechnie stosuje są zapis OR, AND, NOT. Definicje operatorów w algebrze Boole'a przedstawia tab. 2.1.

Tab. 2.1. Definicje operatorów algebry Boole'a

a	b	*alternatywa* $a + b$	*koniunkcja* $a \cdot b$	*negacja* $\bar{a}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Uwaga: podobnie jak symbol mnożenia, operator koniunkcji jest zwyczajowo pomijany w zapisywanych wyrażeniach, stąd: $a \cdot b = ab$.

Dla dowolnych $a, b, c \in B$ spełnione są poniższe pary aksjomatów, które w pełni charakteryzują własności algebry Boole'a:

- domknięcie
  $a + b \in B$ $\qquad$ $a \cdot b \in B,$ (2.2)
- przemienność
  $a + b = b + a$ $\qquad$ $a \cdot b = b \cdot a,$ (2.3)
- rozdzielność
  $a + b \cdot c = (a + b) \cdot (a + c)$ $\qquad$ $a \cdot (b + c) = a \cdot b + a \cdot c,$ (2.4)
- istnienie elementu neutralnego
  $a + 0 = a$ $\qquad$ $a \cdot 1 = a,$ (2.5)
- istnienie dopełnienia
  $a + \bar{a} = 1$ $\qquad$ $a \cdot \bar{a} = 0.$ (2.6)

Używając powyższych aksjomatów, można wyprowadzić dodatkowe zależności nazywane prawami algebry Boole'a:

- prawo łączności
  $a + (b + c) = (a + b) + c$ $a \cdot (b \cdot c) = (a \cdot b) \cdot c,$ (2.7)
- prawo pochłaniania
  $a + a \cdot b = a$ $a \cdot (a + b) = a,$ (2.8)
- prawa sklejania
  $(a + b) \cdot (a + \bar{b}) = a$ $a \cdot b + a \cdot \bar{b} = a,$ (2.9)
- prawa idempotentności
  $a + a = a$ $a \cdot a = a,$ (2.10)
- prawa de Morgana
  $\overline{a + b} = \bar{a} \cdot \bar{b}$ $\overline{a \cdot b} = \bar{a} + \bar{b},$ (2.11)
- prawa konsensusu
  $(a + b) \cdot (\bar{a} + c) \cdot (b + c) = (a + b) \cdot (\bar{a} + c),$
  $a \cdot b + \bar{a} \cdot c + b \cdot c = a \cdot b + \bar{a} \cdot c,$ (2.12)
- prawa elementów neutralnych
  $a + 1 = 1$ $a \cdot 0 = 0,$ (2.13)
  $\bar{0} = 1$ $\bar{1} = 0,$ (2.14)
- prawo podwójnej negacji
  $\bar{\bar{a}} = a.$ (2.15)

2.2. Definicja funkcji logicznej

Funkcją logiczną (*boolowską*, *przełączającą*) *n* zmiennych $f(x_1, x_2, \cdots, x_n)$ nazywane jest odwzorowanie postaci:

$$f: X \to B,$$

gdzie: $B = \{0, 1\}$, $X \subseteq B^n = B \times B \times \cdots \times B$.

Zgodnie z powyższą definicją, funkcja logiczna przyporządkowuje każdemu elementowi z dziedziny wartość ze zbioru B, tzn. 0 lub 1. Dziedzinę funkcji logicznej *n* zmiennych tworzą wartości należące do *n* – elementowego iloczynu kartezjańskiego zbioru *B*, tzn. *n* – elementowe ciągi będące kombinacjami zerojedynkowymi.

Funkcję logiczną nazywamy *zupełną*, jeżeli $X = B^n = B \times B \times \cdots \times B$, tzn. dziedziną funkcji jest cały iloczyn kartezjański, czyli jej wartości są określone dla wszystkich możliwych *n*-elementowych ciągów tworzących dziedzinę. W przypadku funkcji *n* zmiennych liczba możliwych ciągów zerojedynkowych wynosi 2^n. Jeżeli $X \subset B^n$, tzn. dziedziną funkcji jest podzbiór iloczynu kartezjańskiego, czyli jej wartości są określone tylko dla niektórych ze wszystkich możliwych ciągów *n*-elementowych, to funkcję nazywamy *niezupełną* lub *nie w pełni określoną*.

2.3. Reprezentacja funkcji logicznej

Jak pokazano w punkcie 1.2, sygnały binarne, które są szczególnie istotne we współczesnych układach automatyki przemysłowej, mogą być opisane przy pomocy funkcji logicznej. Ważnym etapem projektowania tego typu układów jest odpowiedni opis funkcji reprezentujących wszystkie przetwarzane sygnały. Można w tym celu wykorzystać kilka sposobów reprezentacji funkcji logicznej. Poniżej scharakteryzowano najczęściej stosowane metody: *opis słowny*, *tabela prawdy*, *wyrażenie logiczne* oraz *schemat logiczny*. Zapis za pomocą wyrażeń logicznych jest przedstawiony ogólnie, ten sposób reprezentacji funkcji zostanie omówiony szczegółowo w kolejnych dwóch punktach.

Opis słowny

Metoda ta przedstawia działanie funkcji w formie opisu określającego jakie wartości są przyporządkowywane poszczególnym elementom z dziedziny. Jest najmniej precyzyjną formą reprezentacji funkcji logicznej, jednak w przypadku projektowania układu automatyki od podstaw sporządzenie opisu słownego stanowi zazwyczaj pierwszy etap prac. Ta forma reprezentacji nie narzuca żadnych formalnych reguł, należy jednak postarać się o zachowanie precyzji wypowiedzi i uwzględnić wszystkie aspekty funkcjonowania układu, uwzględniając wszystkie sygnały, które będą w nim przetwarzane.

Tabela prawdy

W tej metodzie działanie funkcji logicznej n zmiennych jest przedstawiane w postaci tabeli, która określa wartość funkcji dla każdego możliwego n-elementowego ciągu z dziedziny. Tabela składa się z $(n+1)$ kolumn oraz (2^n+1) wierszy. Pierwsze n kolumn zawiera wartości zmiennych, a kolumna ostatnia wartości funkcji. Pierwszy wiersz jest nagłówkiem tabeli i opisuje zmienne, kolejne wiersze zawierają ciągi zerojedynkowe z dziedziny oraz wartości funkcji, które są przyporządkowane do danych ciągów. W podręczniku przyjęto zasadę opisu wierszy zgodnie z naturalnym kodem dwójkowym. W odróżnieniu od opisu słownego, tabela prawdy dostarcza jednoznaczną reprezentację funkcji logicznej. W przypadku projektowania układów automatyki przygotowanie tabeli stanowi zazwyczaj drugi etap prac, w którym ewentualne niejednoznaczności zawarte w opisie słownym są eliminowane.

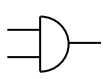
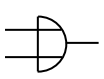
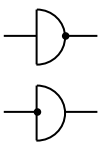
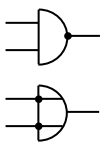
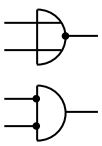
Wyrażenie logiczne

Do zwartego opisu funkcji logicznej wykorzystywane są wyrażenia zapisane za pomocą operatorów algebry Boole'a (zob. p. 2.1). Ta forma reprezentacji funkcji jest szczególnie istotna w przypadku projektowania układów automatyki, ponieważ stanowi wstęp do ich praktycznej realizacji. Dla prostych funkcji, opisanych na niewielkiej liczbie zmiennych, wyrażenie logiczne może być zapisane po analizie opisu tekstowego lub zawartości tabeli prawdy. W ogólnym przypadku taka operacja wymaga zastosowania systematycznego podejścia, którego szczegóły zostaną przedstawione w kolejnych punktach.

Schematy logiczne

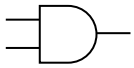
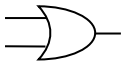
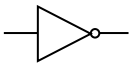
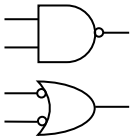
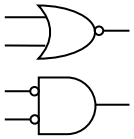
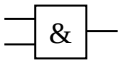
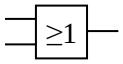
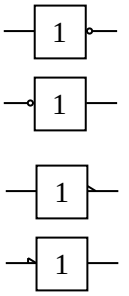
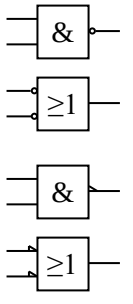
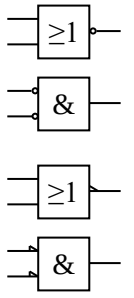
Przypisując poszczególnym operacjom logicznym pewne symbole graficzne, można przedstawić funkcję w postaci schematu, który stanowi alternatywę dla wyrażenia logicznego. Symbole wprowadzane przez normy: DIN 40700 oraz IEEE Std 91/91a-1991 zostały przedstawione odpowiednio w tab. 2.2 i 2.3.

Tab. 2.2. Symbole normy DIN 40700

koniunkcja	*alternatywa*	*negacja*	*negacja koniunkcji*	*negacja alternatywy*
				

Uwaga: Zgodnie z normą DIN 40700 wejścia i wyjścia bloków realizujących operacje logiczne mogą być negowane poprzez wprowadzanie symbolu „•".

Tab. 2.3. Symbole normy IEEE Std 91/91a-1991

koniunkcja	*alternatywa*	*negacja*	*negacja koniunkcji*	*negacja alternatywy*
				
				

Uwaga: Zgodnie z normą IEEE Std 91/91a-1991 wejścia i wyjścia bloków mogą być negowane poprzez wprowadzanie symboli: „○” i „▷”.

Przykład

Funkcja logiczna trzech zmiennych $f(a,b,c)$ przyjmuje wartość 1, gdy zmienna a ma wartość 1 i co najmniej jedna z dwóch pozostałych zmiennych ma wartość 0. W pozostałych przypadkach funkcja przyjmuje wartość 0.

Z analizy opisu słownego wynika, że f jest funkcją zupełną (jej wartości są określone dla wszystkich ciągów z dziedziny) i przyjmuje wartość 1 dla trzech kombinacji zmiennych a, b, c: 100, 101 i 110. Taką funkcję można zilustrować poniższą tabelą prawdy.

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	**1**	0
0	**1**	0	0
0	**1**	**1**	0
1	0	0	**1**
1	0	**1**	**1**
1	**1**	0	**1**
1	**1**	**1**	0

Z przedstawionej wcześniej analizy oraz zawartości tabeli prawdy wynika, że funkcja przyjmuje wartość 1 dla trzech możliwych przypadków:

- $a = 1$ (prawda) i $b, c = 0$ (fałsz),
- $a, c = 1$ (prawda) i $b = 0$ (fałsz),
- $a, b = 1$ (prawda) i $c = 0$ (fałsz),

co prowadzi do wyrażenia logicznego w postaci alternatywy koniunkcji:

$$f_1(a,b,c) = a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}.$$

Analizując działanie funkcji f, można dodatkowo zauważyć, że przypadki, w których przyjmuje ona wartość 1 mogą być opisane krótko w jednej z dwóch form:

- $a = 1$ i b= 0 lub $a = 1$ i $c = 0$,
- $a = 1$ i jednocześnie $b = 0$ lub $c = 0$.

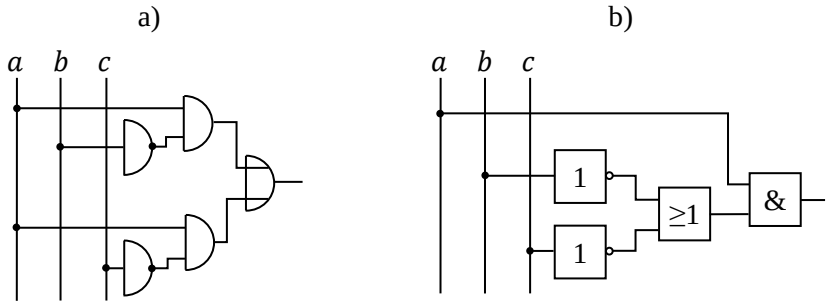
Takie podejście prowadzi do dwóch równoważnych wyrażeń logicznych opisujących funkcję f:

$$f_2(a,b,c) = a\bar{b} + a\bar{c},$$

$$f_3(a,b,c) = a(\bar{b} + \bar{c}).$$

Wszystkie przedstawione wyrażenia poprawnie opisują działanie funkcji f. W tym przypadku mogły być one zapisane po analizie opisu słownego i tabeli prawdy, ponieważ funkcja użyta w przykładzie jest stosunkowo prosta. W przypadku ogólnym wyprowadzenie takich wyrażeń może być znacznie trudniejsze i zazwyczaj wymaga zastosowania odpowiedniego podejścia, które zostanie omówione w kolejnych punktach.

Korzystając z uzyskanych wyrażeń, funkcję f można przedstawić w postaci schematów logicznych zilustrowanych na rys. 2.1.



Rys. 2.1. Schematy logiczne przykładowej funkcji:
a) f_2 w notacji DIN 40700, b) f_3 w notacji IEEE Std 91/91a-1991

2.4. Postać kanoniczna funkcji logicznej

2.4.1. Rozwinięcie funkcji logicznej

Twierdzenie o *rozwinięciu funkcji logicznej* (nazywane również twierdzeniem o *rozkładzie* lub *rozwinięciem Shannona*) pozwala na zapis wyrażeń reprezentujących funkcję logiczną na podstawie przygotowanej tabeli prawdy. Istnieją dwie formy tego twierdzenia, które prowadzą do zapisu funkcji odpowiednio w postaci alternatywy koniunkcji lub koniunkcji alternatyw poszczególnych zmiennych. Praktyczne wykorzystanie twierdzenia o rozwinięciu zostanie pokazane w kolejnym punkcie.

Twierdzenie 2.1.

Każdą funkcję logiczną *n* zmiennych $x_1, \dots, x_n$ można rozwinąć względem dowolnej zmiennej x_i do alternatywy dwóch składników postaci:

$$f(x_1, \dots, x_i, \dots, x_n) = f(x_1, \dots, 1, \dots, x_n)x_i + f(x_1, \dots, 0, \dots, x_n)\bar{x}_i, \tag{2.16}$$

gdzie współczynniki rozwinięcia są *dopełnieniami algebraicznymi* (*kofaktorami*) funkcji f, utworzonymi przez funkcje $n-1$ zmiennych powstałe z funkcji f przez zastąpienie zmiennej x_i odpowiednio wartościami 1 i 0.

Dowód

1° Niech $x_i = 1$:

$$\begin{aligned} f(x_1, \dots, 1, \dots, x_n) &= f(x_1, \dots, 1, \dots, x_n) \cdot 1 + f(x_1, \dots, 0, \dots, x_n) \cdot \bar{1} = \\ &= f(x_1, \dots, 1, \dots, x_n) \cdot 1 + f(x_1, \dots, 0, \dots, x_n) \cdot 0. \end{aligned}$$

Z (2.5) i (2.13) wynika, że dla dowolnego a: $a \cdot 1 = a$ i $a \cdot 0 = 0$, więc:

$$f(x_1, \dots, 1, \dots, x_n) \cdot 1 = f(x_1, \dots, 1, \dots, x_n),$$
$$f(x_1, \dots, 0, \dots, x_n) \cdot 0 = 0$$

Z (2.5) wynika, że dla dowolnego a: $a + 0 = a$, więc:

$$f(x_1, \dots, 1, \dots, x_n) \cdot 1 + f(x_1, \dots, 0, \dots, x_n) \cdot \bar{1} = f(x_1, \dots, 1, \dots, x_n).$$

2° Niech $x_i = 0$:

$$\begin{aligned} f(x_1, \dots, 0, \dots, x_n) &= f(x_1, \dots, 1, \dots, x_n) \cdot 0 + f(x_1, \dots, 0, \dots, x_n) \cdot \bar{0} = \\ &= f(x_1, \dots, 1, \dots, x_n) \cdot 0 + f(x_1, \dots, 0, \dots, x_n) \cdot 1. \end{aligned}$$

Przez analogię do 1°:

$$f(x_1, \dots, 1, \dots, x_n) \cdot 0 + f(x_1, \dots, 0, \dots, x_n) \cdot 1 = f(x_1, \dots, 0, \dots, x_n).$$

❑

Twierdzenie 2.2.

Każdą funkcję logiczną *n* zmiennych $x_1, \dots, x_n$ można rozwinąć względem dowolnej zmiennej x_i do koniunkcji dwóch składników postaci:

$$\begin{aligned} f(x_1, \dots, x_i, \dots, x_n) = \; & (f(x_1, \dots, 0, \dots, x_n) + x_i) \cdot \\ & \cdot (f(x_1, \dots, 1, \dots, x_n) + \bar{x}_i), \end{aligned} \tag{2.17}$$

gdzie współczynniki rozwinięcia są podobnie jak w twierdzeniu 2.1 dopełnieniami algebraicznymi funkcji f.

Dowód

1° Niech $x_i = 1$:

$$\begin{aligned} f(x_1, \dots, 1, \dots, x_n) &= (f(x_1, \dots, 0, \dots, x_n) + 1) \cdot (f(x_1, \dots, 1, \dots, x_n) + \bar{1}) = \\ &= (f(x_1, \dots, 0, \dots, x_n) + 1) \cdot (f(x_1, \dots, 1, \dots, x_n) + 0). \end{aligned}$$

Z (2.5) i (2.13) wynika, że dla dowolnego a: $a + 1 = 1$ i $a + 0 = a$, więc:

$$f(x_1, \dots, 0, \dots, x_n) + 1 = 1,$$
$$f(x_1, \dots, 1, \dots, x_n) + 0 = f(x_1, \dots, 1, \dots, x_n)$$

Z (2.5) wynika, że dla dowolnego a: $a \cdot 1 = a$, więc:

$$(f(x_1, \dots, 0, \dots, x_n) + 1) \cdot (f(x_1, \dots, 1, \dots, x_n) + \bar{1}) = f(x_1, \dots, 1, \dots, x_n).$$

2° Niech $x_i = 0$:

$$\begin{aligned} f(x_1, \dots, 0, \dots, x_n) &= (f(x_1, \dots, 0, \dots, x_n) + 0) \cdot (f(x_1, \dots, 1, \dots, x_n) + \bar{0}) = \\ &= (f(x_1, \dots, 0, \dots, x_n) + 0) \cdot (f(x_1, \dots, 1, \dots, x_n) + 1). \end{aligned}$$

Przez analogię do 1° można pokazać, że:

$$(f(x_1, \dots, 0, \dots, x_n) + 0) \cdot (f(x_1, \dots, 1, \dots, x_n) + \bar{0}) = f(x_1, \dots, 0, \dots, x_n).$$

❑

2.4.2. Kanoniczna postać dysjunkcyjna

Korzystając z twierdzenia 2.1, dowolną funkcję logiczną n zmiennych $x_1, \dots, x_n$ można rozwinąć względem zmiennej x_1 do postaci:

$$f(x_1, x_2, x_3, \dots, x_n) = f(1, x_2, x_3, \dots, x_n)x_1 + f(0, x_2, x_3, \dots, x_n)\bar{x}_1.$$

Rozwijając otrzymane dopełnienia algebraiczne względem zmiennej x_2, funkcję f można przedstawić w postaci:

$$\begin{aligned} f(x_1, x_2, x_3, \dots, x_n) = {} & f(1,1, x_3, \dots, x_n)x_1x_2 + f(1,0, x_3, \dots, x_n)x_1\bar{x}_2 + \\ + {} & f(0,1, x_3, \dots, x_n)\bar{x}_1x_2 + f(0,0, x_3, \dots, x_n)\bar{x}_1\bar{x}_2. \end{aligned}$$

Powtórzenie operacji rozwijania funkcji dla wszystkich zmiennych prowadzi do alternatywy koniunkcji nazwanej *kanoniczną postacią dysjunkcyjną funkcji*:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) = {} & f(1,1, \dots,1)x_1x_2 \dots x_n + f(0,1, \dots,1)\bar{x}_1x_2 \dots x_n + \\ + {} & f(1,0, \dots,1)x_1\bar{x}_2 \dots x_n + f(1,1, \dots,0)x_1x_2 \dots \bar{x}_n + \\ + {} & \dots + f(0,0, \dots,0)\bar{x}_1\bar{x}_2 \dots \bar{x}_n, \end{aligned} \tag{2.18}$$

gdzie $f(1,1, \dots,1)$ to wartość funkcji dla $x_1 = 1,\ x_2 = 1, \dots, x_n = 1$, $f(0,1, \dots,1)$ to wartość funkcji dla $x_1 = 0,\ x_2 = 1, \dots, x_n = 1$ itd., koniunkcje wszystkich zmiennych postaci $x_1x_2 \dots x_n$, $\bar{x}_1x_2 \dots x_n$ itd. nazywane są *iloczynami pełnymi* lub *minitermami*.

Poszczególne elementy funkcji zapisanej w kanonicznej postaci dysjunkcyjnej stanowią koniunkcję wartości funkcji o stałych argumentach oraz iloczynów pełnych. Liczba takich koniunkcji odpowiada liczbie elementów dziedziny, dla których wartość funkcji jest określona (dla funkcji zupełnej 2^n elementów). Warto zauważyć, że w tej postaci funkcji zmienna o wartości 0 w iloczynie pełnym występuje z negacją (zob. (2.16)).

Znając wartości funkcji dla wszystkich elementów dziedziny (np. dysponując funkcją opisaną za pomocą tabeli prawdy), można wykorzystać zależność (2.18) do zapisu funkcji w postaci wyrażenia logicznego. Z prawa (2.13) wynika, że $a \cdot 0 = 0$, więc wynikiem każdej koniunkcji składowej, w której występuje zerowa wartość funkcji jest zero (niezależnie od wartości zmiennych w iloczynie pełnym). Ponieważ zero nie zmienia wyniku alternatywy (z (2.5): $a + 0 = a$), to zapisując dysjunkcyjną postać funkcji, można pominąć wszystkie elementy, w których jest ona zerem jako nieistotne dla ostatecznego wyniku. Takie podejście do zapisu wyrażenia logicznego zostanie zilustrowane na przykładzie przedstawionym w punkcie 2.4.4.

2.4.3. Kanoniczna postać koniunkcyjna

Podobnie jak w punkcie poprzednim, ale korzystając z twierdzenia 2.2, dowolną funkcję logiczną n zmiennych $x_1, \dots, x_n$ można rozwinąć względem zmiennej x_1 do postaci koniunkcji alternatyw:

$$f(x_1, x_2, x_3, \dots, x_n) = (f(0, x_2, x_3, \dots, x_n) + x_1) \cdot (f(1, x_2, x_3, \dots, x_n) + \bar{x}_1).$$

Rozwijając otrzymane dopełnienia algebraiczne względem zmiennej x_2, funkcję f można przedstawić w postaci:

$$\begin{aligned} f(x_1, x_2, x_3, \dots, x_n) = {} & (f(0,0, x_3, \dots, x_n) + x_1 + x_2) \cdot \\ & \cdot (f(0,1, x_3, \dots, x_n) + x_1 + \bar{x}_2) \cdot \\ & \cdot (f(1,0, x_3, \dots, x_n) + \bar{x}_1 + x_2) \cdot \\ & \cdot (f(1,1, x_3, \dots, x_n) + \bar{x}_1 + \bar{x}_2). \end{aligned}$$

Powtórzenie operacji rozwijania funkcji dla wszystkich zmiennych prowadzi do koniunkcji alternatyw nazwanej *kanoniczną postacią koniunkcyjną funkcji*:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) = \quad & (f(0,0,\dots,0) + x_1 + x_2 + \dots + x_n) \cdot \\ & \cdot (f(1,0,\dots,0) + \bar{x}_1 + x_2 + \dots + x_n) \cdot \\ & \cdot (f(0,1,\dots,0) + x_1 + \bar{x}_2 + \dots + x_n) \cdot \\ & \cdot (f(0,0,\dots,1) + x_1 + x_2 + \dots + \bar{x}_n) \cdot \\ & \cdot \dots \cdot (f(1,1,\dots,1) + \bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n), \end{aligned} \tag{2.19}$$

gdzie $f(0,0,\dots,0)$ to wartość funkcji dla $x_1 = 0,\ x_2 = 0, \dots, x_n = 0$, $f(1,0,\dots,0)$ to wartość funkcji dla $x_1 = 1,\ x_2 = 0, \dots, x_n = 0$ itd., alternatywy wszystkich zmiennych postaci $x_1 + x_2 + \dots + x_n$, $\bar{x}_1 + x_2 + \dots + x_n$ itd. nazywane są *sumami pełnymi* lub *maxtermami*.

Poszczególne elementy funkcji zapisanej w kanonicznej postaci koniunkcyjnej są utworzone przez alternatywę wartości funkcji o stałych argumentach oraz sum pełnych. Liczba takich alternatyw jest równa liczbie elementów dziedziny, dla których funkcja jest określona. Warto zauważyć, że w przypadku tej postaci funkcji zmienna o wartości 1 w sumie pełnej występuje z negacją (zob. (2.17)).

Podobnie jak w przypadku postaci dysjunkcyjnej, znając wartości funkcji, można wykorzystać zależność (2.19) do zapisu wyrażenia logicznego (w tym przypadku w równoważnej postaci koniunkcyjnej). Z prawa (2.13) wynika, że $a + 1 = 1$, a więc gdy wartość funkcji wynosi 1, wynikiem alternatywy składowej jest jeden (niezależnie od wartości zmiennych w sumie pełnej). Jedynka nie zmienia wyniku koniunkcji (z (2.5): $a \cdot 1 = a$), dlatego zapisując postać koniunkcyjną, można pominąć wszystkie elementy, w których funkcja jest jedynką jako nieistotne dla ostatecznego wyniku. Takie podejście do zapisu wyrażenia logicznego zostanie zilustrowane na przykładzie przedstawionym w kolejnym punkcie.

2.4.4. Przykład

W punkcie 2.3 przedstawiono funkcję trzech zmiennych $f(a,b,c)$, która przyjmuje wartość 1, gdy zmienna a ma wartość 1 i co najmniej jedna z dwóch pozostałych zmiennych ma wartość 0. W pozostałych przypadkach funkcja przyjmuje wartość 0. Tabela prawdy reprezentująca tak opisaną funkcję została przedstawiona obok.

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	**1**	0
0	**1**	0	0
0	**1**	**1**	0
1	0	0	**1**
1	0	**1**	**1**
1	**1**	0	**1**
1	**1**	**1**	0

Na powyższym przykładzie zostanie zilustrowane wykorzystanie zależności (2.18) i (2.19) do zapisu równoważnych wyrażeń logicznych opisujących działanie funkcji.

Jak pokazano w punkcie 2.4.2, korzystając z twierdzenia 2.1, każdą funkcję logiczną można przedstawić w kanonicznej postaci dysjunkcyjnej danej zależnością (2.18). Zgodnie z przeprowadzoną analizą w takim przypadku można pominąć wszystkie elementy, dla których wartość funkcji jest zerem. W przypadku przykładowej funkcji, której tabela prawdy została przedstawiona powyżej, istotne są jedynie trzy wiersze, w których funkcja przyjmuje wartość 1. Należy utworzyć iloczyny pełne (*minitermy*) odpowiadające tym wierszom, pamiętając, że zmienna jest negowana, gdy przyjmuje wartość zero:

- $a = 1, b = 0, c = 0 \rightarrow a\bar{b}\bar{c}$,
- $a = 1, b = 0, c = 1 \rightarrow a\bar{b}c$,
- $a = 1, b = 1, c = 0 \rightarrow ab\bar{c}$.

Alternatywa otrzymanych iloczynów pełnych tworzy kanoniczną postać dysjunkcyjną funkcji f (wartość funkcji w każdym przypadku wynosi 1, więc nie zmienia wyniku koniunkcji i może być pominięta):

$$f(a,b,c) = a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}.$$

Analogicznie, korzystając z analizy przedstawionej w punkcie 2.4.3 oraz twierdzenia 2.2, każdą funkcję logiczną można przedstawić w kanonicznej postaci koniunkcyjnej (2.19). W tym przypadku należy pominąć wszystkie elementy, dla których wartość funkcji wynosi jeden. W przykładowej funkcji istotnych jest pięć wierszy, w których funkcja jest zerem. Należy utworzyć sumy pełne (*maxtermy*) odpowiadające tym wierszom, pamiętając, że zmienna jest negowana, gdy przyjmuje wartość jeden:

- $a = 0, b = 0, c = 0 \rightarrow a + b + c$,
- $a = 0, b = 0, c = 1 \rightarrow a + b + \bar{c}$,
- $a = 0, b = 1, c = 0 \rightarrow a + \bar{b} + c$,
- $a = 0, b = 1, c = 1 \rightarrow a + \bar{b} + \bar{c}$,
- $a = 1, b = 1, c = 1 \rightarrow \bar{a} + \bar{b} + \bar{c}$.

Koniunkcja otrzymanych w ten sposób sum pełnych tworzy kanoniczną postać koniunkcyjną funkcji f (wartość funkcji w każdym przypadku wynosi 0, więc nie zmienia wyniku alternatywy i może być pominięta):

$$f(a, b, c) = (a + b + c)(a + b + \bar{c})(a + \bar{b} + c)(a + \bar{b} + \bar{c})(\bar{a} + \bar{b} + \bar{c}).$$

Podsumowanie

- *W celu zapisania formy dysjunkcyjnej należy*: wybrać wiersze tabeli, w których funkcja ma wartość 1, utworzyć iloczyny pełne negując zmienne o wartościach 0 i zapisać alternatywę iloczynów pełnych.
- *W celu zapisania formy koniunkcyjnej należy*: wybrać wiersze tabeli, w których funkcja ma wartość 0, utworzyć sumy pełne negując zmienne o wartościach 1 i zapisać koniunkcję sum pełnych.

2.5. Postać minimalna funkcji logicznej

Zapis wyrażenia logicznego w postaci kanonicznej na ogół prowadzi do funkcji, której postać jest nieoptymalna (zawiera więcej operacji niż jest to niezbędne). Projektując układy automatyki, dąży się do uzyskania wyrażeń możliwie najmniejszych, ponieważ przekłada się to na znacznie prostsze, a w konsekwencji bardziej efektywne programy realizowane przez urządzenia sterujące. Minimalizację wyrażeń logicznych można wykonać poprzez odpowiednie przekształcenie postaci kanonicznych z zastosowaniem praw algebry Boole'a (zob. p. 2.1). Takie podejście zazwyczaj wymaga przeprowadzenia wielu przekształceń i indywidualnego potraktowania każdej funkcji, a w przypadku rozbudowanych wyrażeń logicznych jest czasochłonne.

Alternatywą jest wykorzystanie jednej z metod optymalizacji: Karnaugha lub Quine'a-McCluskeya (QMC). Obydwa podejścia opierają się na podobnej idei i prowadzą do identycznych rezultatów, ale różnią się zastosowaniem. Metoda Karnaugha jest metodą graficzną i może być używana do minimalizacji funkcji o niewielkiej liczbie zmiennych (do sześciu). Metoda Quine'a-McCluskeya może być zastosowana do funkcji o większej liczbie zmiennych i jest prostsza do implementacji w formie programu komputerowego. W kolejnych punktach zostaną przedstawione podstawy minimalizacji z wykorzystaniem praw algebry Boole'a oraz metoda Karnaugha.

2.5.1. Prawa algebry Boole'a

Podstawowymi prawami wykorzystywanymi podczas minimalizacji funkcji logicznych są prawa sklejania (2.9) oraz idempotentności (2.10):

- $(a + b) \cdot (a + \bar{b}) = a \qquad a \cdot b + a \cdot \bar{b} = a$,
- $a + a = a \qquad a \cdot a = a$.

Prawa sklejania pozwalają zredukować zmienną w wyrażeniach *sąsiednich logicznie*, czyli takich, które są zbudowane z tych samych zmiennych i różnią się tylko pojedynczą negacją. Prawo idempotentności ma charakter pomocniczy i pozwala na tworzenie duplikatów elementów minimalizowanego wyrażenia w celu uzyskania odpowiedniej liczby par wyrażeń sąsiednich logicznie.

Przykład 1

Rozważmy kanoniczną postać dysjunkcyjną funkcji z punktu 2.4.4:

$$f(a, b, c) = a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}.$$

Można zauważyć, że element $a\bar{b}\bar{c}$ jest sąsiedni logicznie zarówno z $a\bar{b}c$ (różna postać zmiennej c), jak i $ab\bar{c}$ (różna postać zmiennej b). W celu minimalizacji po pierwsze korzysta się z prawa idempotentności:

$$a\bar{b}\bar{c} = a\bar{b}\bar{c} + a\bar{b}\bar{c},$$

co pozwala zapisać funkcję w postaci:

$$f(a,b,c) = a\bar{b}\bar{c} + a\bar{b}c + a\bar{b}\bar{c} + ab\bar{c}.$$

W tak przekształconej funkcji istnieją dwie pary wyrażeń sąsiednich logicznie, do których można zastosować prawa sklejania:

$$a\bar{b}\bar{c} + a\bar{b}c = a\bar{b},$$
$$a\bar{b}\bar{c} + ab\bar{c} = a\bar{c},$$

co prowadzi do nowej postać funkcji po redukcji:

$$f(a,b,c) = a\bar{b} + a\bar{c}.$$

W tym przypadku można dodatkowo skorzystać z własności rozdzielności (2.4) do wyłączenia zmiennej a przed nawias, co prowadzi do minimalnej postaci:

$$f(a,b,c) = a(\bar{b} + \bar{c}).$$

Przykład 2

Rozważmy kanoniczną postać koniunkcyjną funkcji z punktu 2.4.4:

$$f(a,b,c) = (a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+\bar{c}).$$

W tym przypadku można wyróżnić kilka kombinacji wyrażeń sąsiednich logicznie. W celu zapewnienia najmniejszej liczby przekształceń najwygodniej jest skorzystać z prawa idempotentności do powielenia wyrażenia $(a+\bar{b}+\bar{c})$, co prowadzi do następujących redukcji na mocy prawa sklejania:

$$(a+b+c)(a+b+\bar{c}) = (a+b),$$
$$(a+\bar{b}+\bar{c})(a+\bar{b}+c) = (a+\bar{b}),$$
$$(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+\bar{c}) = (\bar{b}+\bar{c}),$$

co prowadzi do nowej postaci funkcji po redukcji:

$$f(a,b,c) = (a+b)(a+\bar{b})(\bar{b}+\bar{c}).$$

W powyższej funkcji można wyróżnić dwa wyrażenia sąsiednie logicznie i zastosować do nich prawo sklejania:

$$(a+b)(a+\bar{b}) = a.$$

Ostatecznie otrzymuje się minimalne wyrażenie opisujące funkcję f w takiej samej formie jak uzyskana w poprzednim przykładzie:

$$f(a,b,c) = a(\bar{b} + \bar{c}).$$

2.5.2. Tablice Karnaugha

Do opisu funkcji boolowskiej *n*-zmiennych oprócz klasycznych tabel prawdy o 2^n wierszach można wykorzystać tablice Karnaugha. Są one konstruowane jako kwadratowe lub prostokątne siatki zbudowane z 2^n kratek. Dla parzystej liczby argumentów tablica ma kształt kwadratu o wymiarach $2^{0,5n} \times 2^{0,5n}$, dla nieparzystej prostokąta o wymiarach $2^{0,5(n-1)} \times 2^{0,5(n+1)}$. Poszczególne wiersze i kolumny reprezentują jedną lub kilka zmiennych i są opisane *kodem Graya*. Przykłady tablic Karnaugha dla funkcji 2, 3 i 4 zmiennych zostały przedstawione na rys. 2.2.

a)

a \ b	0	1
0		
1		

b)

a \ bc	00	01	11	10
0				
1				

c)

a \ bc	00	01	11	10
00				
01				
11				
10				

Rys. 2.2. Tablice Karnaugha funkcji o: a) n = 2, b) n = 3 i c) n = 4 zmiennych

Kod Graya należy do grupy kodów refleksyjnych. Kod *n*-pozycyjny powstaje z kodu *n*–1 pozycyjnego zgodnie z następującym algorytmem:

- zapisz układ symboli kodu *n*–1 pozycyjnego w kolumnie,
- w kolejnych wierszach do układu symboli kodu *n*–1 pozycyjnego dopisz te same symbole, ale w odwrotnej kolejności (w odbiciu lustrzanym),
- do pierwotnych symboli dodaj poprzedzające „0”, do symboli dopisanych poprzedzającą „1”.

Wielokrotne powtórzenie powyższego algorytmu pozwala na utworzenie kodu Graya o dowolnej liczbie pozycji. Kolejne etapy tworzenie kodu 3-pozycyjnego zostały przedstawione w tab. 2.4.

Tab. 2.4. Tworzenie 3-pozycyjnego kodu Graya

Kod 1-poz.	*Po odbiciu*	*Kod 2-poz.*	*Po odbiciu*	*Kod 3-poz.*
0	0	00	00	000
1	1	01	01	001
	1	11	11	011
	0	10	10	010
			10	110
			11	111
			01	101
			00	100

Istotną z punktu widzenia wykorzystania tablic Karnaugha cechą kodu Graya jest postać sąsiednich wartości – dwa kolejne kody różną sią dokładnie jedną pozycją. Na przykład dla kodu 2-pozycyjnego: 00 → zmiana drugiej pozycji → 01 → zmiana pierwszej pozycji → 11 → zmiana drugiej pozycji → 10. Warto też zauważyć, że taka własność dotyczy pierwszego i ostatniego kodu w szeregu. Na przykład dla kodu 2-pozycyjnego: 10 → zmiana pierwszej pozycji → 00. Jak zostanie to pokazane poniżej, taki sposób opisu wierszy i kolumn tablic Karnaugha decyduje o możliwości ich wykorzystania jako efektywnego narzędzia do minimalizacji wyrażeń logicznych. *Uwaga*: opisane metody działają tylko w przypadku tablic, których wiersze i kolumny są opisane kodem Graya, zmiana sposobu kodowania prowadzi do nieprawidłowych wyników.

Każda kratka w tablicy Karnaugha może być jednoznacznie określona przez podanie jej adresu utworzonego przez połączenie kodów opisujących wiersz i kolumnę. Adres wyznacza wartości zmiennych, na których funkcja jest opisana, a wartość wprowadzona we wnętrzu kratki określa wartość funkcji odpowiadającą tym zmiennym. Warto zauważyć, że ze względu na sposób kodowania wierszy i kolumn kolejność wartości w tablicy Karnaugha różni się od kolejności w tabeli prawdy dla przyjętej w podręczniku konwencji zapisu. Na rys. 2.3 pokazano odpowiedniki poszczególnych kratek z tablicy Karnaugha w tabeli prawdy dla pewnej funkcji trzech zmiennych $f(x_1, x_2, x_3)$. Należy zauważyć różnicę w ułożeniu wartości z dwóch ostatnich kolumn: trzeci element w tablicy Karnaugha (adres 011) znajduje się w czwartym wierszu tabeli prawdy, czwarty (adres 010) w wierszu trzecim. Podobne przestawienie występuje w przypadku elementów 7 i 8 (adresy 111, 110), które znajdują się odpowiednio w wierszu ósmym i siódmym. W przypadku funkcji większej liczby zmiennych różnice dotyczą całych wierszy, np. w tablicach opisujących funkcje czterech zmiennych przestawieniu ulegają wszystkie elementy z wiersza trzeciego i czwartego (zob. rys. 2.11).

$f(x_1, x_2, x_3)$

x_1 \ x_2x_3	**00**	**01**	**11**	**10**
0	①	②	③	④
1	⑤	⑥	⑦	⑧

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	①
0	0	**1**	②
0	**1**	0	④
0	**1**	**1**	③
1	0	0	⑤
1	0	**1**	⑥
1	**1**	0	⑧
1	**1**	**1**	⑦

Rys. 2.3. Układ wartości w tablicy Karnaugha i tabeli prawdy dla funkcji trzech zmiennych

2.5.3. Tablice Karnaugha i prawa sklejania

Podobnie jak jeden wiersz w tabeli prawdy, pojedyncza kratka tablicy Karnaugha odpowiada jednej kombinacji zmiennych, czyli jednemu *iloczynowi pełnemu* w kanoniczej postaci dysjunkcyjnej lub jednej *sumie pełnej* w kanonicznej postaci koniunkcyjnej. Reguły zapisu poszczególnych wyrażeń wynikają z twierdzenia o rozwinięciu funkcji logicznej (zob. p. 2.4.1) i są zgodne z zasadami przedstawionymi w punktach 2.4.2 i 2.4.3:

- iloczyn pełny jest budowany w postaci koniunkcji wszystkich zmiennych, negowane są te zmienne, których wartości wynoszą 0,
- suma pełna jest budowana w postaci alternatywy wszystkich zmiennych, negowane są te zmienne, których wartości wynoszą 1.

Iloczyny oraz sumy pełne odpowiadające kolejnym kratkom tablicy Karnaugha przedstawionej na rys. 2.3 zostały zapisane w tab. 2.5.

Tab. 2.5. Sumy i iloczyny pełne funkcji trzech zmiennych

nr kratki	*adres*	*iloczyn pełny*	*suma pełna*
①	000	$\bar{x}_1\bar{x}_2\bar{x}_3$	$x_1 + x_2 + x_3$
②	001	$\bar{x}_1\bar{x}_2x_3$	$x_1 + x_2 + \bar{x}_3$
③	011	$\bar{x}_1x_2x_3$	$x_1 + \bar{x}_2 + \bar{x}_3$
④	010	$\bar{x}_1x_2\bar{x}_3$	$x_1 + \bar{x}_2 + x_3$
⑤	100	$x_1\bar{x}_2\bar{x}_3$	$\bar{x}_1 + x_2 + x_3$
⑥	101	$x_1\bar{x}_2x_3$	$\bar{x}_1 + x_2 + \bar{x}_3$
⑦	111	$x_1x_2x_3$	$\bar{x}_1 + \bar{x}_2 + \bar{x}_3$
⑧	110	$x_1x_2\bar{x}_3$	$\bar{x}_1 + \bar{x}_2 + x_3$

Z tablicy Karnaugha, tak jak z tabeli prawdy, można odczytać kanoniczną postać dysjunkcyjną (alternatywa iloczynów pełnych z kratek, dla których wartość funkcji wynosi 1) i kanoniczną postać koniunkcyjną (koniunkcja sum pełnych z kratek, dla których wartość funkcji wynosi 0). Ze względu na własności kodu użytego do opisu wierszy i kolumn (refleksyjny kod Graya) tablica Karnaugha pozwala dodatkowo na zapis funkcji w *postaci normalnej*:

- *Normalna postać dysjunkcyjna* jest alternatywą koniunkcji dowolnej liczby zmiennych z negacjami lub bez. Takie wyrażenie powstaje z kanonicznej postaci dysjunkcyjnej po zastosowaniu praw sklejania dla wybranych iloczynów pełnych.
- *Normalna postać koniunkcyjna* jest koniunkcją alternatyw dowolnej liczby argumentów funkcji z negacjami lub bez. Takie wyrażenie powstaje z kanonicznej postaci koniunkcyjnej po zastosowaniu praw sklejania dla wybranych sum pełnych.

Kratki tablicy Karnaugha położone obok siebie w tym samym wierszu lub tej samej kolumnie (kratki o wspólnej krawędzi) są określane jako sąsiednie. Analizując wyrażenia reprezentujące iloczyny i sumy

pełne, łatwo zauważyć, że takie kratki odpowiadają wyrażeniom *sąsiednim logicznie*, więc jak pokazano w punkcie 2.5.1 podlegają prawom sklejania opisanym zależnościami (2.9).

Przykład 1

Rozpatrując alternatywę iloczynów pełnych oraz koniunkcję sum pełnych wyrażeń odpowiadających kratkom 2 i 3 przedstawionym w tab. 2.5, łatwo zauważyć, że podlegają one sklejeniu:

$$\bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 = \bar{x}_1x_3,$$
$$(x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + \bar{x}_3) = x_1 + \bar{x}_3.$$

Analogiczne zależności można pokazać dla pozostałych par kratek sąsiednich, w tym również dla kratek leżących w pierwszej i ostatniej kolumnie. Analiza wyrażeń odpowiadających kratce 1 i 4 oraz 5 i 8 pokazuje, że różnią się tylko jedną negacją, więc można potraktować je jako kratki sąsiednie i zastosować do nich prawa sklejania.

Porównując adresy kratek sąsiednich, można wykazać, że zawsze różnią się na jednej pozycji odpowiadającej zmiennej, która ulega redukcji (jest zanegowana w jednym wyrażeniu i występuje bez negacji w drugim). Na tej podstawie można sformułować ogólną regułę, która pozwala na odczytanie zredukowanego wyrażenia (wyniku sklejania) wprost z tablicy Karnaugha:

Zredukowane wyrażenie odpowiadające alternatywie iloczynów pełnych lub koniunkcji sum pełnych zawiera zmienne, których wartości są takie same (odpowiadający im składnik adresu nie zmienia się), redukcji podlega zmienna, której wartości są różne (odpowiadający jej składnik adresu jest różny w obydwu kratkach).

Zasady konstruowania takich wyrażeń nie zmieniają się w stosunku do zasad sformułowanych dla postaci kanonicznych: przy zapisie koniunkcji (elementów postaci dysjunkcyjnej) negowane są zmienne o wartości 0, a przy zapisie alternatyw (elementów postaci koniunkcyjnej) zmienne o wartości 1.

Przykład 2

Kratki o numerach 2 i 3 rozpatrywane w przykładzie 1 mają adresy: 001 i 011. Zmianie ulega drugi składnik ($x_2 = 0$ w kratce 2 i $x_2 = 1$ w kratce 3), więc zmienna x_2 jest redukowana. Pozostałe składniki adresów są identyczne i odpowiadają zmiennym $x_1 = 0$ oraz $x_3 = 1$. Stąd x_1 należy zanegować przy zapisie koniunkcji, a x_3 przy zapisie alternatywy, więc zredukowane formy wyrażeń odpowiadające koniunkcji i alternatywie kratek wynoszą odpowiednio:

$$\bar{x}_1x_3, x_1 + \bar{x}_3.$$

Sklejaniu podlegają również większe grupy sąsiadujących kratek, muszą one jednak spełniać odpowiednie warunki. Jak pokazano wcześniej, skleić można grupę dwóch sąsiadujących kratek, co prowadzi do wyrażenia, z którego zostanie wyeliminowana jedna zmienna. Jeżeli obok tej grupy znajduje się grupa dwóch kratek, z której po sklejeniu eliminowana jest ta sama zmienna, to sklejone grupy będzie można skleić ponownie. Wynika stąd, że należy wybierać grupy, które pozwolą na łączenie w pary kolejnych wyrażeń i wielokrotne powtarzanie operacji sklejania. Prowadzi to do ogólnej zasady:

Sklejaniu podlegają prostokątne grupy kratek składające się z 2^k elementów, gdzie $k = 0,1,2,3, \ldots$

Oznacza to, że prawidłowe grupy mogą składać się z jednej (przypadek szczególny, daje iloczyn lub sumę pełną), dwóch, czterech, ośmiu itd. kratek. Wyznaczenie grupy o niedozwolonej liczbie elementów nie pozwoli na prawidłowe zastosowanie praw sklejania i prowadzi do błędnych wyników. Przykłady prawidłowych grup dla funkcji trzech i czterech zmiennych zostały pokazane na końcu tego punktu, odpowiednio na rys. 2.4 i 2.5.

Przykład 3

Czteroelementowej grupie zawierającej kratki 2, 3, 6, 7 z rys. 2.3 odpowiadają wyrażenia w postaci dysjunkcyjnej (alternatywa koniunkcji) oraz koniunkcyjnej (koniunkcji alternatyw) postaci:

$$\bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2x_3,$$
$$(x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3).$$

Sklejając kratki 2 i 3 oraz 6 i 7, otrzymuje się:

$$\bar{x}_1 x_3 + x_1 x_3, (x_1 + \bar{x}_3)(\bar{x}_1 + \bar{x}_3).$$

Jak łatwo zauważyć, otrzymane wyrażenia są sąsiednie logicznie, więc podlegają prawom sklejania, co prowadzi do ostatecznej formy:

$$x_3, \qquad \bar{x}_3.$$

Identyczne rezultaty można uzyskać, analizując adresy kratek: 001, 011, 101, 111. Tylko składnik odpowiadający zmiennej x_3 jest taki sam we wszystkich adresach, więc zgodnie z regułą sformułowaną na stronie 19 zmienna x_3 pozostaje, natomiast zmienne x_1 oraz x_2 podlegają redukcji. Ponieważ $x_3 = 1$ dla wszystkich kratek, wystąpi bez negacji w postaci dysjunkcyjnej i zostanie zanegowane w postaci koniunkcyjnej. Prowadzi to do wyrażeń postaci:

$$x_3, \qquad \bar{x}_3,$$

zgodnych z wynikiem uzyskanym powyżej.

Przykład 4

Jak pokazano wcześniej sąsiednimi są również kratki znajdujące się w skrajnych kolumnach, więc kratki 1, 4, 5, 8 z tablicy przedstawionej na rys. 2.3 tworzą prawidłową grupę czteroelementową. Ich adresy wynoszą odpowiednio: 000, 010, 100, 110. Jak można zauważyć, we wszystkich adresach taką samą wartość ma jedynie składnik, który odpowiada zmiennej x_3. Oznacza to, że zmienne x_1 oraz x_2 podlegają redukcji, a zmienna x_3 będzie zanegowana w postaci dysjunkcyjnej i zapisana bez negacji w postaci koniunkcyjnej ($x_3 = 0$), co prowadzi do wyrażeń postaci:

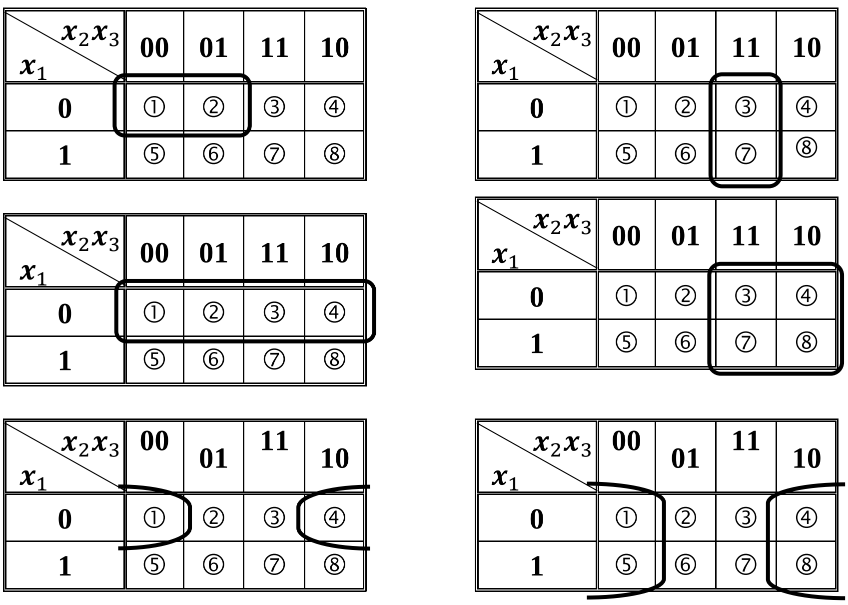
$$\bar{x}_3, \qquad x_3.$$

Przykład 5

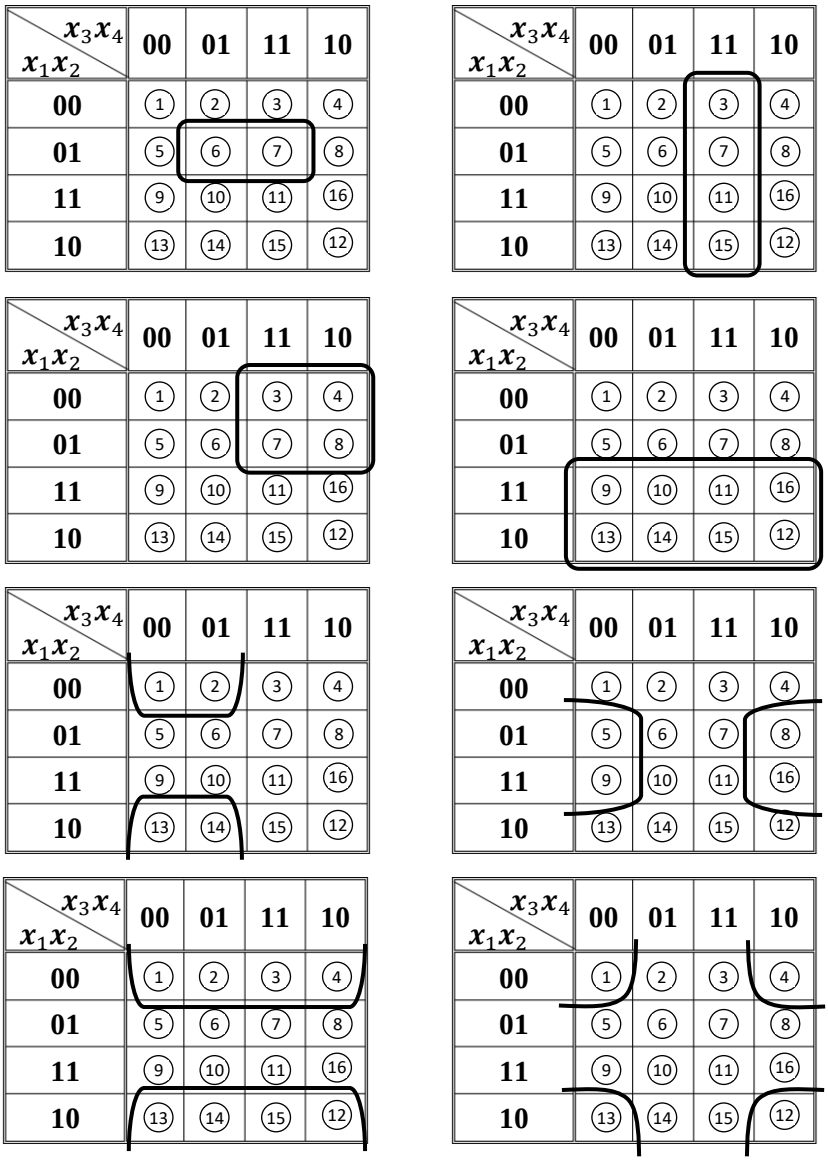
W tablicy przedstawionej na rys. 2.3 można utworzyć grupę złożoną ze wszystkich elementów jednego wiersza. Przykładowo kratki 5, 6, 7, 8 tworzą prawidłową grupę czteroelementową. Ich adresy to: 100, 101, 111, 110. Tylko składnik odpowiadający zmiennej x_1 ma taką samą wartość we wszystkich adresach ($x_1 = 1$), więc zastosowanie praw sklejania prowadzi do wyrażeń:

$$x_1, \qquad \bar{x}_1,$$

Na rys. 2.4 i 2.5 pokazano przykłady grup w tablicy Karnaugha dla trzech i czterech zmiennych.



Rys. 2.4. Przykładowe grupy kratek dla funkcji trzech zmiennych



Rys. 2.5. Przykładowe grupy kratek dla funkcji czterech zmiennych

W przykładach przedstawionych na rys. 2.5 warto zwrócić uwagę na grupy zawierające kratki pierwszego i ostatniego wiersza (adresy wierszy 00 i 10 różną się na jednej pozycji) oraz szczególny przypadek grupy złożonej z czterech narożników. Tego typu grupa jest dozwolona, ponieważ kratka 1 sąsiaduje z 4 i 13, 4 z 1 i 12 itd. Prawidłowa nie będzie para złożona tylko z 1 i 12 lub 4 i 13.

2.5.4. Minimalizacja funkcji logicznych z wykorzystaniem tablic Karnaugha

Tworząc grupy zawierające odpowiednio wszystkie jedynki lub wszystkie zera i stosując techniki sklejania kratek przedstawione w poprzednim punkcie, można wykorzystać tablicę Karnaugha do zapisania normalnej postaci dysjunkcyjnej lub normalnej postaci koniunkcyjnej funkcji logicznej (zob. s. 18). Dodatkowo jeżeli wybór grup jedynek lub zer będzie optymalny, możliwe jest uzyskanie odpowiednio *minimalnej normalnej postaci dysjunkcyjnej* oraz *minimalnej normalnej postaci koniunkcyjnej*. Wyrażenia takie zawierają możliwie najmniejszą liczbę składowych (tzn. minimalną liczbę koniunkcji lub alternatyw), a wyrażenia składowe zawierają najmniejszą możliwą liczbę argumentów. Ostatecznie można w ten sposób uzyskać najprostsze wyrażenie opisujące funkcję logiczną w postaci normalnej, czyli jej zapis w *normalnej postaci minimalnej*.

W celu wyznaczenia minimalnej normalnej postaci koniunkcyjnej za pomocą tablicy Karnaugha, należy zapisać funkcję w tablicy i zastosować następujący algorytm:

- wyszukaj i zaznacz wśród niezaznaczonych jeszcze kratek tablicy możliwie największe, samodzielne grupy zer obejmujące 2^k kratek (k = …, 3, 2, 1, 0),
- jeżeli po wyodrębnieniu wszystkich samodzielnych grup w tablicy pozostają niezaznaczone zera, utwórz z nich grupy, łącząc je (jeżeli to możliwe) z kratkami zaznaczonymi tak, aby uzyskać największą grupę obejmującą 2^k kratek (grupy mogą mieć część wspólną),
- dla każdej wyznaczonej grupy zer znajdź argumenty funkcji, których wartości są takie same dla wszystkich elementów w grupie, pozostałe odrzuć (wykorzystaj zasady łączenia kratek opisane w p. 2.5.3),
- dla tak powstałych grup argumentów zapisz alternatywę, negując składniki, dla których funkcja ma wartość jeden,
- koniunkcja alternatyw zapisanych dla wszystkich grup stanowi minimalną normalną postać koniunkcyjną funkcji.

W celu wyznaczenia minimalnej normalnej postaci dysjunkcyjnej, należy postąpić analogicznie, jednak w tym przypadku poszukiwane są największe możliwe grupy jedynek złożone z 2^k kratek. Dla każdej grupy wyznacza się argumenty o takich samych wartościach i zapisuje ich koniunkcje, negując składniki, dla których funkcja ma wartość jeden. Alternatywa wszystkich koniunkcji stanowi poszukiwaną postać dysjunkcyjną.

Uwaga: warunkiem uzyskania normalnej postaci minimalnej jest wyznaczenie największych możliwych grup zer (postać koniunkcyjna) lub jedynek (postać dysjunkcyjna). Pozostawienie sąsiadujących grup, które mogły być połączone oznacza pominięcie możliwych do wykonana operacji sklejania, które doprowadzą do mniejszej postaci wyrażenia.

Przykład 1

Należy wyznaczyć minimalną normalną koniunkcyjną i dysjunkcyjną postać funkcji trzech zmiennych $f(a, b, c)$ opisanej w punkcie 2.4.4.

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	**1**	0
0	**1**	0	0
0	**1**	**1**	0
1	0	0	**1**
1	0	**1**	**1**
1	**1**	0	**1**
1	**1**	**1**	0

a \ bc	00	01	11	10
0	0	0	0	0
1	**1**	**1**	0	**1**

Rys. 2.6. Tabela prawdy i odpowiadająca jej tablica Karnaugha funkcji z punktu 2.4.4

Minimalna normalna postać koniunkcyjna

W celu uzyskania postaci koniunkcyjnej, należy wyznaczyć największe grupy zer. W przypadku funkcji opisanej tablicą z rys. 2.6, największą możliwą jest grupa złożona z 2^2 kratek obejmująca pierwszy wiersz (rys. 2.7a). Po jej zaznaczeniu pozostaje jedno zero, które można połączyć z wartością znajdującą się w kratce powyżej, tworząc grupę złożoną z 2^1 kratek (rys. 2.7b).

a \ bc	00	01	11	10
0	0	0	0	0
1	**1**	**1**	0	**1**

a) największa grupa zer

a \ bc	00	01	11	10
0	0	0	0	0
1	**1**	**1**	0	**1**

b) dołączenie niezaznaczonego zera

Rys. 2.7. Grupy zer w tablicy Karnaugha

Ostatecznie w tablicy wyznaczone zostały dwie grupy złożone z czterech i dwóch kratek. Adresy elementów (wartości a, b, c odpowiadające kratkom tablicy) wchodzących w skład tych grup to odpowiednio:

- grupa czteroelementowa: 000, 001, 011, 010,
- grupa dwuelementowa: 011, 111.

Zgodnie z algorytmem przedstawionym powyżej dla każdej z grup, należy wyznaczyć argumenty funkcji (wartości a, b, c), które są takie same dla wszystkich elementów w grupie – te składniki utworzą wyrażenie będące częścią postaci koniunkcyjnej. Argumenty, których wartości są różne ulegną redukcji zgodnie z prawem sklejania (zob. p. 2.5.3). W tym celu można zapisać adresy kratek tworzących grupę w tabeli i wykreślić kolumny, które zawierają różne wartości jak pokazano na rys. 2.8.

a	b	c
0	0	0
0	0	1
0	1	1
0	1	0

a	b	c
0	1	1
1	1	1

Rys. 2.8. Redukcja argumentów funkcji

Po redukcji w pierwszej grupie pozostaje argument $a = 0$, natomiast w drugiej grupie argumenty $b = 1$ oraz $c = 1$. Odpowiadają im poniższe alternatywy (w postaci koniunkcyjnej negowane są argumenty o wartości 1):

$$a \qquad \text{oraz} \qquad \bar{b} + \bar{c}.$$

Koniunkcja wyznaczonych w ten sposób alternatyw tworzy poszukiwaną minimalną normalną postać koniunkcyjną funkcji:

$$f(a, b, c) = a(\bar{b} + \bar{c}).$$

Minimalna normalna postać dysjunkcyjna

W celu wyznaczenia postaci dysjunkcyjnej, należy wyznaczyć największe grupy jedynek. Uwzględniając, że kratki położone w kolumnach skrajnych traktuje się jako sąsiadujące, łatwo zauważyć, że w przypadku omawianej funkcji możliwe jest wskazanie dwóch grup złożonych z dwóch kratek (pierwsze dwa elementy lub pierwszy i ostatni element drugiego wiersza). Dla wyniku końcowego nie jest istotne, która z nich zostanie wybrana jako pierwsza – w tym przykładzie wybrano grupę pokazaną na rys. 2.9a. Po jej zaznaczeniu pozostaje jedna niezaznaczona jedynka, którą można połączyć z wartością z pierwszej kratki drugiego wiersza (rys. 2.9b).

Uwaga: nie można stworzyć grupy złożonej z trzech sąsiadujących jedynek, ponieważ liczba elementów nie jest potęgą dwójki.

a \ bc	**00**	**01**	**11**	**10**
0	0	0	0	0
1	**1**	**1**	0	**1**

a) największa grupa jedynek

a \ bc	**00**	**01**	**11**	**10**
0	0	0	0	0
1	**1**	**1**	0	**1**

b) dołączenie niezaznaczonej jedynki

Rys. 2.9. Grupy jedynek w tablicy Karnaugha

Ostatecznie w tablicy wyznaczone zostały dwie grupy złożone z dwóch kratek. Adresy elementów (wartości a, b, c odpowiadające kratkom tablicy) wchodzących w skład tych grup to odpowiednio:

- pierwsza grupa: 100, 101,
- druga grupa: 100, 110.

Zgodnie z algorytmem przedstawionym powyżej, dla każdej z grup należy wyznaczyć argumenty funkcji (wartości a, b, c), które są takie same dla wszystkich elementów w grupie – te składniki utworzą wyrażenie będące częścią postaci dysjunkcyjnej. Argumenty, których wartości są różne ulegną redukcji zgodnie z prawem sklejania (zob. p. 2.5.3). Tak jak w przypadku postaci koniunkcyjnej, adresy kratek można zapisać w tabeli i wykreślić kolumny, które zawierają różne wartości (rys. 2.10).

a	b	c
1	0	0
1	0	1

a	b	c
1	0	0
1	1	0

Rys. 2.10. Redukcja argumentów funkcji

Po redukcji w pierwszej grupie pozostają argumenty $a = 1$ i $b = 0$, natomiast w drugiej argumenty $a = 1$ i $c = 0$. Odpowiadają im poniższe koniunkcje (w postaci dysjunkcyjnej negowane są argumenty o wartości 0):

$$a\,\bar{b} \qquad \text{oraz} \qquad a\,\bar{c}.$$

Alternatywa (dysjunkcja) wyznaczonych w ten sposób koniunkcji tworzy poszukiwaną minimalną normalną postać dysjunkcyjną funkcji:

$$f(a,b,c) = a\bar{b} + a\bar{c}.$$

Przykład 2

Należy wyznaczyć minimalną normalną koniunkcyjną i dysjunkcyjną postać funkcji czterech zmiennych $f(a,b,c,d)$ opisanej tabelą prawdy oraz odpowiadającą jej tablicą Karnaugha przedstawioną na rys. 2.11.

Minimalna normalna postać koniunkcyjna

W celu uzyskania postaci koniunkcyjnej, należy wyznaczyć możliwie największe grupy zer. Funkcja czterech zmiennych, opisana tablicą 4×4, daje większe możliwości wyznaczenia grup: należy pamiętać o sąsiedztwie kratek położonych w skrajnych wierszach i kolumnach oraz szczególnym przypadku sąsiedztwa kratek z czterech narożników (zob. rys. 2.5). W przypadku analizowanej funkcji największa możliwa grupa składa się z 2^3 kratek (rys. 2.12a). Po jej zaznaczeniu pozostaje jedno zero, które można połączyć z kratką poniżej, tworząc drugą grupę złożoną z 2^1 elementów (rys. 2.12b).

a	b	c	d	$f(a,b,c,d)$
0	0	0	0	**1**
0	0	0	**1**	**1**
0	0	**1**	0	0
0	0	**1**	**1**	**1**
0	**1**	0	0	0
0	**1**	0	**1**	0
0	**1**	**1**	0	0
0	**1**	**1**	**1**	0
1	0	0	0	**1**
1	0	0	**1**	**1**
1	0	**1**	0	**1**
1	0	**1**	**1**	**1**
1	**1**	0	0	0
1	**1**	0	**1**	0
1	**1**	**1**	0	0
1	**1**	**1**	**1**	0

ab \ cd	**00**	**01**	**11**	**10**
00	**1**	**1**	**1**	0
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

Rys. 2.11. Tabela prawdy i odpowiadająca jej tablica Karnaugha funkcji czterech zmiennych z przykładu 2

ab \ cd	00	01	11	10
00	**1**	**1**	**1**	0
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

a) największa grupa zer

ab \ cd	00	01	11	10
00	**1**	**1**	**1**	0
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

b) dołączenie niezaznaczonego zera

Rys. 2.12. Grupy zer w tablicy Karnaugha

Ostatecznie w tablicy wyznaczone zostały dwie grupy złożone z ośmiu i dwóch kratek. Adresy elementów (wartości a, b, c, d odpowiadające kratkom tablicy) wchodzących w skład tych grup to odpowiednio:

- ośmioelementowa: 0100, 0101, 0111, 0110, 1100, 1101, 1111, 1110,
- dwuelementowa: 0010, 0110.

Zgodnie z algorytmem dla każdej z grup należy wyznaczyć argumenty funkcji (wartości a, b, c, d), które są takie same dla wszystkich elementów w grupie – utworzą one wyrażenie będące częścią postaci koniunkcyjnej. Argumenty, których wartości są różne ulegną redukcji zgodnie z prawem sklejania (zob. p. 2.5.3). W tym celu można wykorzystać technikę pokazaną w przykładzie 1, zapisać adresy kratek tworzących grupę w tabeli i wykreślić kolumny, które zawierają różne wartości (rys. 2.13).

a	*b*	*c*	*d*
0	1	0	0
0	1	0	1
0	1	1	1
0	1	1	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0

a	*b*	*c*	*d*
0	0	1	0
0	1	1	0

Rys. 2.13. Redukcja argumentów funkcji

Po redukcji w pierwszej grupie pozostaje argument $b = 1$, natomiast w drugiej argumenty $a = 0$, $c = 1$, $d = 0$. Odpowiadają im poniższe alternatywy (w postaci koniunkcyjnej negowane są argumenty o wartości 1):

$$\bar{b} \qquad \text{oraz} \qquad a + \bar{c} + d.$$

Koniunkcja wyznaczonych w ten sposób alternatyw tworzy poszukiwaną minimalną normalną postać koniunkcyjną funkcji:

$$f(a, b, c, d) = \bar{b}(a + \bar{c} + d).$$

Minimalna normalna postać dysjunkcyjna

W celu wyznaczenia postaci dysjunkcyjnej, należy wyznaczyć największe grupy jedynek. Uwzględniając fakt, że kratki położone w skrajnych wierszach sąsiadują ze sobą, łatwo zauważyć, że w przypadku omawianej funkcji możliwe jest wskazanie trzech różnych grup złożonych z 2^2 kratek (cały ostatni wiersz, pierwsze dwa elementy pierwszego i ostatniego wiersza oraz drugi i trzeci element pierwszego i ostatniego wiersza). Dla wyniku końcowego nie jest istotne, która z nich zostanie wybrana jak pierwsza – w tym przykładzie wybrano grupę obejmującą ostatni wiersz (rys. 2.14a). Po jej zaznaczeniu pozostają trzy jedynki, które można połączyć z elementami w ostatnim wierszu, tworząc dwie grupy czteroelementowe, tak jak zostało to pokazane kolejno na rys. 2.14b i 2.14c.

Uwaga: często popełnianym w takim przypadku błędem jest próba utworzenia grupy sześcioelementowej, złożonej z jedynek z pierwszego i ostatniego wiersza. Taka grupa nie jest prawidłowa, ponieważ liczba elementów nie jest potęgą dwójki, więc prowadzi do błędnej postaci minimalnej.

ab \ cd	00	01	11	10
00	**1**	**1**	**1**	0
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

a) największa grupa jedynek

ab \ cd	00	01	11	10
00	**1**	**1**	**1**	0
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

b) dołączenie dwóch jedynek

ab \ cd	00	01	11	10
00	**1**	**1**	**1**	0
01	0	0	0	0
11	0	0	0	0
10	**1**	**1**	**1**	**1**

c) dołączenie pozostałej jedynki

Rys. 2.14. Grupy jedynek w tablicy Karnaugha

Ostatecznie w tablicy wyznaczone zostały trzy grupy czteroelementowe. Adresy elementów (wartości argumentów a, b, c, d odpowiadające kratkom tablicy) wchodzących w skład tych grup to odpowiednio:

- pierwsza grupa: 1000, 1001, 1011, 1010,
- druga grupa: 0000, 0001, 1000, 1001,
- trzecia grupa: 0001, 0011, 1001, 1011.

Zgodnie z algorytmem dla każdej z grup należy wyznaczyć argumenty funkcji (wartości a, b, c, d), które są takie same dla wszystkich elementów w grupie – utworzą one wyrażenia będące częścią postaci dysjunkcyjnej. Argumenty, których wartości są różne ulegną redukcji zgodnie z prawem sklejania (zob. p. 2.5.3). W tym celu można ponownie wykorzystać technikę pokazaną w przykładzie 1, zapisać adresy kratek tworzących grupę w tabeli i wykreślić kolumny, które zawierają różne wartości (rys. 2.15).

a	*b*	*c*	*d*
1	0	0	0
1	0	0	1
1	0	1	1
1	0	1	0

a	*b*	*c*	*d*
0	0	0	0
0	0	0	1
1	0	0	0
1	0	0	1

a	*b*	*c*	*d*
0	0	0	1
0	0	1	1
1	0	0	1
1	0	1	1

Rys. 2.15. Redukcja argumentów funkcji

Po redukcji w pierwszej grupie pozostają argumenty $a = 1, b = 0$ w drugiej $b = 0, c = 0$, natomiast w trzeciej $b = 0, d = 1$. Odpowiadają im poniższe koniunkcje (w postaci dysjunkcyjnej negowane są argumenty o wartości 0):

$$a\,\bar{b}, \qquad \bar{b}\,\bar{c}, \qquad \bar{b}\,d.$$

Alternatywa (dysjunkcja) wyznaczonych w ten sposób koniunkcji tworzy poszukiwaną minimalną normalną postać dysjunkcyjną funkcji:

$$f(a, b, c, d) = a\bar{b} + \bar{b}\bar{c} + \bar{b}d.$$

2.5.5. Wykorzystanie tablic Karnaugha do analizy wyrażeń logicznych

Przedstawione powyżej metody zapisu wyrażeń logicznych z wykorzystaniem tablic Karnaugha mogą być wykorzystane do przeprowadzenia ich analizy. Uwzględniając, że postać dysjunkcyjna powstaje z grup jedynek, a postać koniunkcyjna z grup zer i mając na uwadze zasady negowania argumentów

(0 w postaci dysjunkcyjnej lub 1 w postaci koniunkcyjnej), można ustalić układ grup argumentów, co pozwala na szybkie zapisanie tablicy Karnaugha dowolnego wyrażenia logicznego.

Przykład 1

Należy utworzyć tablicę Karnaugha funkcji logicznej zapisanej wyrażeniem:

$$f(a,b,c) = a\bar{b} + a\bar{c}.$$

Po pierwsze należy zauważyć, że jest to wyrażenie zapisane w postaci dysjunkcyjnej i zawiera dwa składniki, dlatego zostało zapisane z dwóch grup jedynek. Analizując rozkład negacji, można wnioskować, że:

- grupa $a\bar{b}$ powstała z kratek tablicy, w których $a = 1$ i $b = 0$ (w postaci dysjunkcyjnej negowane są 0), a c jest dowolne, więc składa się z elementów o adresach: 100 oraz 101,
- grupa $a\bar{c}$ powstała z kratek, w których $a = 1$ i $c = 0$, a b jest dowolne, więc składa się z elementów o adresach 100 oraz 110.

Pozwala to na ustalenie kratek, w których znajdują się 1 (pozostałe kratki zawierają 0), co prowadzi do tablicy Karnaugha w postaci pokazanej na rys. 2.16.

a \ *bc*	**00**	**01**	**11**	**10**
0	0	0	0	0
1	**1**	**1**	0	**1**

Rys. 2.16. Grupy jedynek w tablicy Karnaugha

Przykład 2

Należy utworzyć tablicę Karnaugha funkcji logicznej zapisanej wyrażeniem:

$$f(a,b,c) = a(\bar{b} + \bar{c}).$$

Po pierwsze należy zauważyć, że jest to wyrażenie zapisane w postaci koniunkcyjnej i zawiera dwa czynniki, więc zostało zapisane z dwóch grup zer. Analizując rozkład negacji można wnioskować, że:

- grupa a powstała z kratek tablicy, w których $a = 0$ (w postaci koniunkcyjnej negowane są 1), a b i c są dowolne, a więc składa się z elementów o adresach: 000, 001, 010, 011,
- grupa $(\bar{b} + \bar{c})$ powstała z kratek, w których $b = 1$ i $c = 1$, a a jest dowolne, a więc składa się z elementów o adresach 011 oraz 111.

Pozwala to na ustalenie kratek, w których znajdują się 0 (pozostałe kratki zawierają 1), co prowadzi do tablicy Karnaugha w postaci pokazanej na rys. 2.17.

a \ *bc*	**00**	**01**	**11**	**10**
0	0	0	0	0
1	**1**	**1**	0	**1**

Rys. 2.17. Grupy zer w tablicy Karnaugha

Podejście przedstawione na powyższych przykładach może być wykorzystane do analizowania funkcji logicznych pod kątem ich optymalności. Analiza rozkładu grup w tabeli Karnaugha pozwala eliminować składniki zbędne (takie, które można usunąć bez wpływu na uzyskiwane wartości) i sprawdzić czy wykonane zostały wszystkie możliwe operacje sklejania. W dalszych rozważaniach znajdzie to również zastosowanie przy eliminacji hazardu (zob. p. 3.5).

Przykład 3

Należy sprawdzić, czy funkcja logiczna

$$f(a,b,c) = ab + \bar{b} + \bar{a}\bar{b}$$

jest zapisana w postaci minimalnej.

Jest to wyrażenie w postaci dysjunkcyjnej, które składa się z trzech grup:

- grupa ab z kratek, w których $a = b = 1$ (adresy 111, 110),
- grupa $\bar{b}$ z kratek, w których $b = 0$ (adresy 000, 001, 100, 101),
- grupa $\bar{a}\bar{b}$ z kratek, w których $a = b = 0$ (adresy 000, 001),

co prowadzi do tabeli Karnaugcha pokazanej na rys. 2.18a.

a \ bc	00	01	11	10
0	1	1	0	0
1	1	1	1	1

a) przed optymalizacją

a \ bc	00	01	11	10
0	1	1	0	0
1	1	1	1	1

b) po optymalizacji

Rys. 2.18. Grupy jedynek w tablicy Karnaugha

Analizując układ grup, łatwo zauważyć, że dwuelementowa grupa złożona z kratek 000 i 001 (odpowiednik składnika $\bar{a}\bar{b}$) w całości zawiera się w czteroelementowej grupie złożonej z kratek 000, 001, 100 i 101 (odpowiednik składnika $\bar{b}$). Oznacza to, że składnik $\bar{a}\bar{b}$ jest zbędny i może być wyeliminowany bez wpływu na wartość wyrażenia. Dodatkowo dwuelementowa grupa złożona z kratek 111 i 110 (odpowiednik składnika ab) może być powiększona do czteroelementowej, co prowadzi do zredukowania wyrażenia. Z analizy wynika, że funkcja nie jest zapisana w postaci minimalnej, jednak optymalizując układ grup (rys. 2.18b), można ją zapisać w minimalnej postaci dysjunkcyjnej

$$f(a,b,c) = a + \bar{b}.$$

2.6. Zadania

2.1. Dana jest funkcja logiczna czterech zmiennych $f(x,y,z,q)$. Funkcja przyjmuje wartość 1, gdy $y = q = 1$ lub $y = q = 0$, w pozostałych przypadkach przyjmuje wartość 0. Przygotuj tablicę prawdy i zapisz funkcję f w dysjunkcyjnej i koniunkcyjnej postaci kanonicznej.

2.2. Dana jest funkcja logiczna czterech zmiennych $g(x,y,z,q)$. Funkcja przyjmuje wartość 0, gdy $x = y = 0$ lub $y = 0$ i $z = 1$, w pozostałych przypadkach przyjmuje wartość 1. Przygotuj tablicę prawdy i zapisz funkcję g w dysjunkcyjnej i koniunkcyjnej postaci kanonicznej.

2.3. Korzystając z metody Karnaugha, zapisz minimalną dysjunkcyjną i koniunkcyjną postać funkcji f z zadania 2.1 oraz funkcji g z zadania 2.2.

2.4. Korzystając z minimalnych form zapisanych w zadaniu 2.3, narysuj schematy logiczne funkcji f i g zgodnie z normą DIN 40700 oraz IEEE Std 91/91a-1991.

2.5. Przeprowadź analizę poniższych funkcji i zapisz je w postaci minimalnej:

a) $f(x,y,z,q) = \bar{x}\bar{z} + \bar{x}\bar{z}\bar{q} + yz\bar{q} + x\bar{z}$,

b) $g(x,y,z,q) = (x + y + z)(\bar{x} + y)(x + \bar{y} + z + q)(x + y + \bar{z})$.