Logic for Computer Science. Knowledge Representation and Reasoning.

Lecture Notes

for

Computer Science Students
Faculty EAIiIB-IEiT AGH

Antoni Ligęza

Other support material:

`http://home.agh.edu.pl/~ligeza`

`https://ai.ia.agh.edu.pl/pl:dydaktyka:logic:start`

George Boole

Figure 1: George Boole (1815-1864).

Algebra Boole'a

Algebra Boole'a jest strukturą matematyczną postaci:

$$(\{0,1\}, \vee, \wedge, \neg, 0, 1),$$

gdzie dany jest zbiór dwuelementowy, działania alternatywy (sumy), koniunkcji (iloczynu) oraz negacji (dopełnienia), a także element neutralny sumy i iloczynu. Dla uproszczenia zapisu koniunkcję $p \wedge q$ zapisujemy jako $p \cdot q$ lub pq, a negację $\neg p$ jako $\bar{p}$. Działania alegbry Boole'a spełniają następujące prawa:

- element neutralny: $p \vee 0 = p$, $p \cdot 1 = p$,
- identyczność: $p \vee 1 = 1$, $p \cdot 0 = 0$,
- idempotentność: $p \vee p = p$, $p \cdot p = p$,
- dopełnianie: $p \vee \bar{p} = 1$, $p \cdot \bar{p} = 0$,
- przemienność: $p \vee q = q \vee p$, $p \wedge q = q \wedge p$,
- łączność: $(p \vee q) \vee r = p \vee (q \vee r)$, $(p \wedge q) \wedge r = p \wedge (q \wedge r)$,
- rozdzielność: $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$, $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$,
- pochłanianie: $(p \wedge q) \vee p = p$, $(p \vee q) \wedge p = p$,
- prawa De Morgana: $\overline{p \vee q} = \bar{p} \wedge \bar{q}$, $\overline{p \wedge q} = \bar{p} \vee \bar{q}$.

Przykładami algebr Boole'a są:

- Algebra zerojedynkowa: $(\{0,1\}, \vee, \wedge, \neg, 0, 1)$,
- Zbiory: $(2^S, \cap, \cup, \bar{X}, \emptyset, S)$,
- Ciągi zerojedynkowe o długości n:
  $(\{ppp\ldots p : p \in \{0,1\}\}, \vee, \wedge, \neg, 000\ldots0, 111\ldots1)$.

Funkcje boolowskie

Niech $\mathbf{B} = \{0, 1\}$. Funkcją booleowską n-argumentową nazywamy dowolną funkcję typu:

$$f : \mathbf{B}^n \longrightarrow \mathbf{B}.$$

W ogólnym przypadku istnieje 2^{2^n} różnych funkcji boolowskich. Dla $n = 1$ mamy cztery funkcje, dla $n = 2$ jest ich 16 a dla $n = 3$ aż 256.

Przykład funkcji 3-argumentowej przedstawiono poniżej.

p	q	r	$f(p, q, r)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Funkcja ta dla $p = 0$ realizuje koniunkcję pozostałych argumentów, a dla $p = 1$ – ich alternatywę; można więc ją zapisać w postaci:

$$f(p, q, r) = \bar{p} \cdot (q \cdot r) \vee p \cdot (q \vee r).$$

Inna postać tej funkcji:

$$f(p, q, r) = \bar{p} \cdot q \cdot r \vee p \cdot \bar{q} \cdot r \vee p \cdot q \cdot \bar{r} \vee p \cdot q \cdot r.$$

Wyrażenia postaci $x \cdot y \cdot z$, lub prościej xyz, gdzie $x = p$ lub $x = \bar{p}$, $y = q$ lub $y = \bar{q}$ oraz $z = r$ lub $z = \bar{r}$ nazywamy atomami (iloczynami; mintermami) (pełnymi; maksymalnymi) w $\mathbf{B}^3$. Różnych takich atomów jest 2^n. Każdą funkcję można przedstawić jako sumę atomów.

Postać minimalna funkcji f:

$$f(p,q,r) = p \cdot q \vee p \cdot r \vee q \cdot r.$$

Dalsza redukcja z zachowaniem równoważności jest niemożliwa.

Wyrażenia boolowskie

Wyrażenia boolowskie służą do definiowania funkcji boolowskich.

Definition 1 *Niech będą dane symbole zmiennych boolowskich* $p_1, p_2, \ldots, p_n$*. Wyrażeniem boolowskim jest:*

- *symbol* 0*,* 1 *oraz każdy z symboli* $p_1, p_2, \ldots, p_n$*,*
- *jeżeli* P_1 *i* P_2 *sa wyrażeniami boolowskimi, to również* $P_1 \vee P_2$*,* $P_1 \wedge P_2$ *oraz* $\bar{P}_1$ *(*$\bar{P}_2$*) sa wyrażeniami boolowskimi.*

Każde wyrażenie boolowskie definiuje pewna funkcję booleowską. Wartości tej funkcji dla konkretnego wyrażenia wyznaczamy stosując reguły algebry Boole'a.

Dwa wyrażenia boolowskie nazywamy **równoważnymi** gdy definiują one identyczne funkcje.

Iloczynem (maksymalnym) n zmiennych boolowskich nazywamy koniunkcję (atom w $\mathbf{B}^n$) n zmiennych; żadna zmienna nie może się powtarzać.

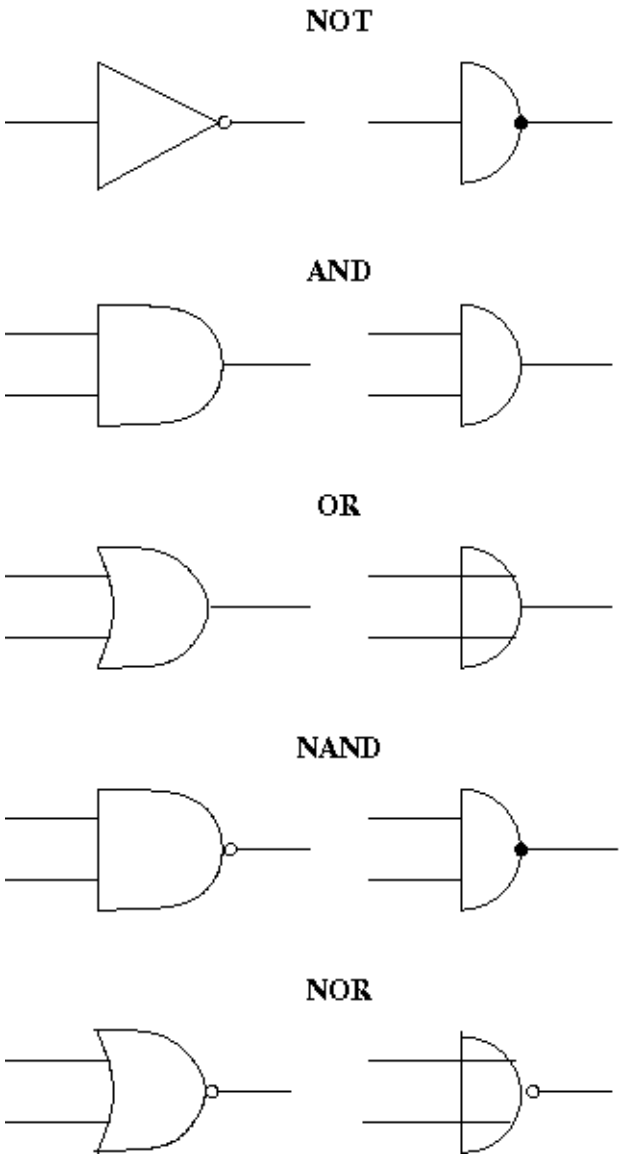
DNF. Każdą funkcję i każde wyrażenie boolowskie można przedstawić w równoważnej postaci składającej się z sumy samych iloczynów maksymalnych (mintermów) - jest to tzw. postać kanoniczna (DNF; postać normalna alternatywno-koniunkcyjna). Określenie postaci kanonicznej polega na wskazaniu które iloczyny pełne wchodzą $(+)$ do definicji funkcji, a które nie wchodzą $(-)$.

CNF. Alternatywnie, każdą funkcję i każde wyrażenie boolowskie można przedstawić w równoważnej postaci składającej się z iloczynu samych sum maksymalnych (makstermów) - jest to tzw. postać kanoniczna (CNF; postać normalna koniunkcyjno-alternatywna). Określenie postaci kanonicznej polega na wskazaniu które sumy pełne wchodzą $(-)$ do definicji funkcji, a które nie wchodzą $(+)$.

p	q	r	iloczyn	$f(p,q,r)$	czy wchodzi
0	0	0	$\bar{p}\bar{q}\bar{r}$	0	–
0	0	1	$\bar{p}\bar{q}r$	0	–
0	1	0	$\bar{p}q\bar{r}$	0	–
0	1	1	$\bar{p}qr$	1	+
1	0	0	$p\bar{q}\bar{r}$	0	–
1	0	1	$p\bar{q}r$	1	+
1	1	0	$pq\bar{r}$	1	+
1	1	1	pqr	1	+

Powyższą funkcję można zapisać również jako iloczyn sum (makstermów).

Bramki logiczne



Tabele prawdy dla bramek logicznych

Tabele wartości logicznych

Zamiast symboli prawdy i fałszu często stosujemy zapis uproszczony: 1 zamiast prawdy i 0 zamiast fałszu. Tablica prawdy dla negacji przybiera postać:

p	$\bar{p}$
0	1
1	0

Tablica prawdy dla koniunkcji przybiera postać:

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Tablica prawdy dla dysjunkcji przybiera postać:

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

Bramki NAND oraz NOR

Tablica prawdy dla funkcji NAND przybiera postać:

p	q	$\overline{pq}$
0	0	1
0	1	1
1	0	1
1	1	0

Tablica prawdy dla funkcji NOR przybiera postać:

p	q	$\overline{p \vee q}$
0	0	1
0	1	0
1	0	0
1	1	0

Układy logiczne

Z bramek logicznych można budować dowolnie złożone układy logiczne.

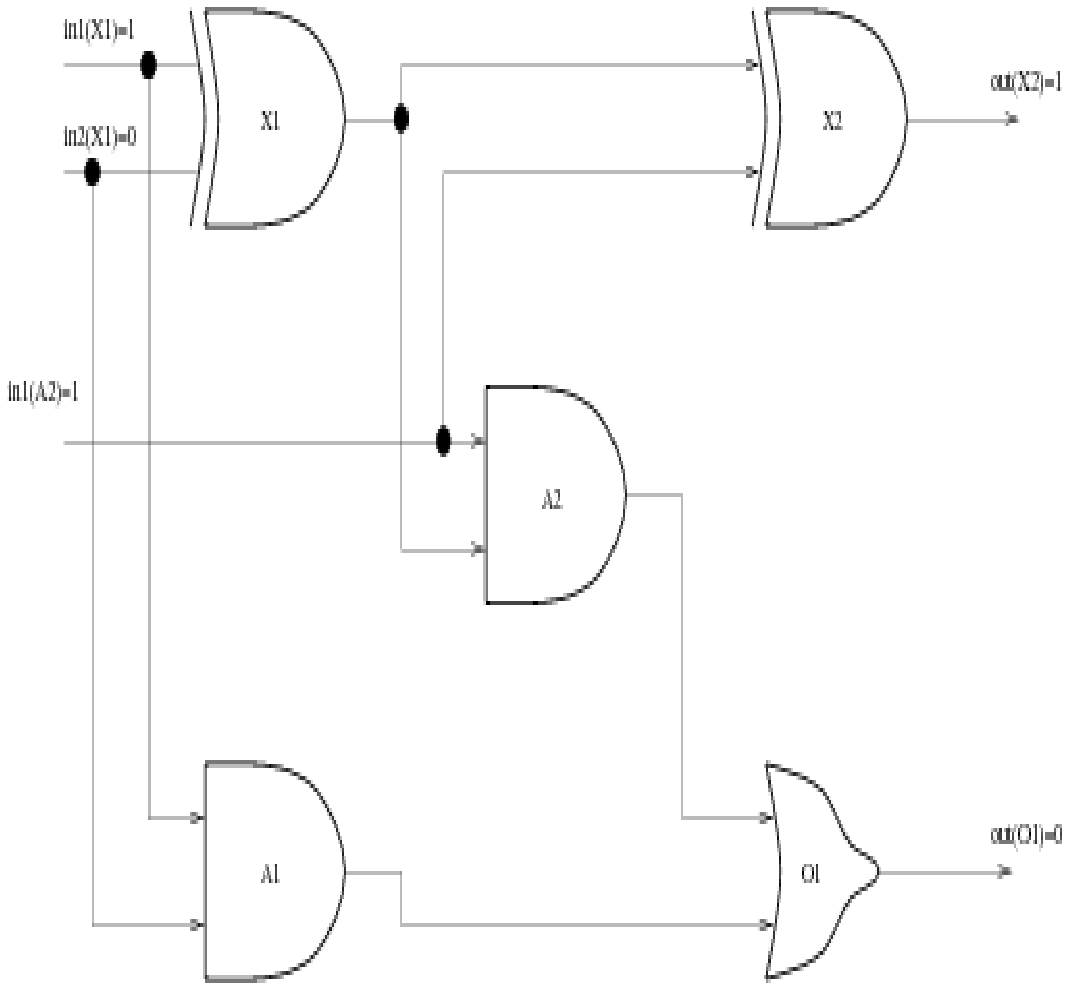


Figure 2: Przykład układu logicznego – sumator.

Analiza układów logicznych

Analiza układów logicznych polega na rekonstrukcji funkcji logicznej realizowanej przez zadany układ logiczny. Można to zrobić posługując się matrycą logiczną układu. Dla układu z Rys. 2 mamy:

inp1(X1)	inp2(X1)	inp(A2)	X1	A1	A2	out(X2)	out(O1)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	1	0	0	1	0
0	1	1	1	0	1	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	1	0	1
1	1	0	0	1	0	0	1
1	1	1	0	1	0	1	1

Na postawie powyższej tabeli można odtworzyć funkcje w postaci kanonicznej; dla uproszczenia podstawmy inp1(X1) = p, inp2(X1) = q, inp(A2) = c. Mamy:

$$out(X2) = \bar{p}\bar{q}c \vee \bar{p}q\bar{c} \vee p\bar{q}\bar{c} \vee pqc,$$

$$out(O1) = \bar{p}qc \vee p\bar{q}c \vee pq\bar{c} \vee pqc.$$

Można też określić postacie minimalne dla tych funkcji:

$$out(X2) = \bar{p}\bar{q}c \vee \bar{p}q\bar{c} \vee p\bar{q}\bar{c} \vee pqc,$$

$$out(O1) = pq \vee pc \vee qc.$$

Postać minimalna i kanoniczna funkcji out(X2) są identyczne.

Funkcje tego układu można też określić bezpośrednio w oparciu o schemat; jednak jest ona bardziej skomplikowana i trudna do dalszej analizy.

Synteza układów logicznych

Synteza układów logicznych polega na zaprojektowaniu struktury sieci logicznej realizującej określoną funkcję. Funkcja którą należy zrealizować może być zadana:

- ekstensjonalnie – w postaci tabeli wejścia-wyjście,
- intensjonalnie – w postaci wyrażenia logicznego (boolowskiego).

Etapy projektowania obejmują:

- specyfikację funkcji (np. werbalną, w postaci zestawu warunków),
- budowę tablicy opisującej relacje wejścia-wyjścia,
- generację funkcji w postaci kanonicznej (pełne mintermy/iloczyny lub makstermy/sumy),
- minimalizację i ew. faktoryzację funkcji,
- implementację z wykorzystaniem zadanych bramek logicznych.

Przykład specyfikacji funkcji

Tablica definiująca funkcję budowana jest w oparciu o werbalną specyfikację funkcji, np.:

> *Zaprojektować układ do głosowania, o trzech wejściach, który daje na wyjściu* 1 *o ile przynajmniej na dwóch wejściach pojawią się jedynki.*

p	q	r	$f(p,q,r)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Synteza układów logicznych

Dla tabeli specyfikującej działanie funkcji można łatwo zbudować postać kanoniczną funkcji. W tym celu należy:

- uwzględnić jedynie te wiersze tabeli, w których na wyjściu układu jest 1,
- dla każdego takiego wiersza zbudować iloczyn maksymalny (pełny) zawierający wszystkie zmienne wejściowe,
- jeżeli w danym wierszu zmienna wejściowa przyjmuje wartość 1 to w iloczynie minimalnym występuje bez negacji,
- jeżeli w danym wierszu zmienna wejściowa przyjmuje wartość 0 to w iloczynie maksymalnym występuje z negacją,
- wygenerowane iloczyny minimalne połączyć symbolem sumy (alternatywy).

Postać kanoniczna tej funkcji:

$$f(p,q,r) = \bar{p}\cdot q\cdot r \vee p\cdot \bar{q}\cdot r \vee p\cdot q\cdot \bar{r} \vee p\cdot q\cdot r.$$

Postać kanoniczna jest zwykle zbyt rozbudowana, aby wykorzystać ją bezpośrednio do syntezy układu. Należy zatem dokonać minimalizacji funkcji korzystając z praw algebry Boole'a.

Minimalizacja może przebiegać następująco:

$$\begin{aligned}
&\bar{p}\cdot q\cdot r \vee p\cdot \bar{q}\cdot r \vee p\cdot q\cdot \bar{r} \vee p\cdot q\cdot r = \\
&(\bar{p}\cdot q\cdot r \vee p\cdot q\cdot r) \vee (p\cdot \bar{q}\cdot r \vee p\cdot q\cdot r) \vee (p\cdot q\cdot \bar{r} \vee p\cdot q\cdot r) = \\
&q\cdot r \vee p\cdot r \vee p\cdot q.
\end{aligned}$$

Dla realizacji tej funkcji potrzeba 3 bramki AND i 2 bramki OR.

Dla realizacji funkcji z wykorzystaniem bramek NAND stosujemy przekształcenie:

$$f(p,q,r) = \overline{\overline{q \cdot r \vee p \cdot r \vee p \cdot q}} = \overline{\overline{pq} \cdot \overline{pr} \cdot \overline{qr}}$$

.

(4 bramki NAND).

Minimalizacja i przekształcanie wyrażeń

Minimalizację i przekształcanie funkcji logicznych można prowadzić z wykorzystaniem reguł algebry Boole'a; pozwalają one zachować *równoważność* wyrażeń boolowskich (funkcji). Obowiązują następujące reguły:

Reguła sklejania wyrażeń iloczynowych

Niech będą dane dwa iloczyny literałów postaci $x_1x_2\ldots x_{i-1}px_{i+1}\ldots x_n$ oraz $x_1x_2\ldots x_{i-1}\bar{p}x_{i+1}\ldots x_n$, które są identyczne poza i-tą pozycją na której występują literały komplementarne. Iloczyny te można zastąpić jednym iloczynem wg następującego schematu:

$$\frac{x_1x_2\ldots x_{i-1}p\ x_{i+1}\ldots x_n,\ x_1x_2\ldots x_{i-1}\bar{p}\ x_{i+1}\ldots x_n}{x_1x_2\ldots x_{i-1}-x_{i+1}\ldots x_n}$$

Symbol $-$ oznacza *dowolną wartość* i może zostać pominięty; otrzymamy wówczas skrócony iloczyn postaci $x_1x_2\ldots x_{i-1}x_{i+1}\ldots x_n$.

Reguła pochłaniania wyrażeń iloczynowych

W trakcie redukcji wyrażeń boolowskich może okazać się, że pewne wyrażenia przestają być niezbędne dla zapisu funkcji – stają się one nadmiarowe i jako takie mogą być usunięte. Niech będą dane dwa iloczyny literałów postaci $x_1x_2\ldots x_n$ oraz $y_1y_2\ldots y_n$. Zachodzi pochłanianie:

$$x_1x_2\ldots x_n \geq y_1y_2\ldots y_n$$

wtw. gdy dla każdego i, $x_i = y_i$ lub $x_i = -$. Inaczej, wyrażenie pochłaniające przyjmuje wartość 1 zawsze gdy wyrażenie pochłaniane przyjmuje wartość 1. Wyrażenie pochłaniane $y_1y_2\ldots y_n$ można wyeliminować (bez wpływu na równoważność; pamiętajmy, że $x_1x_2\ldots x_{i-1}x_{i+1}\ldots x_n \equiv x_1x_2\ldots x_{i-1}-x_{i+1}\ldots x_n$).

Tablice Karnaugha

Tablice Karnaugha stanowią wizualne narzędzie wspomagające minimalizację wyrażeń boolowskich i funkcji logicznych. Istota tablic Karnaugha polega na uwidocznieniu wyrażeń, które mogą podlegać sklejaniu w postaci *sąsiednich pól* odpowiednio skonstruowanej tablicy.

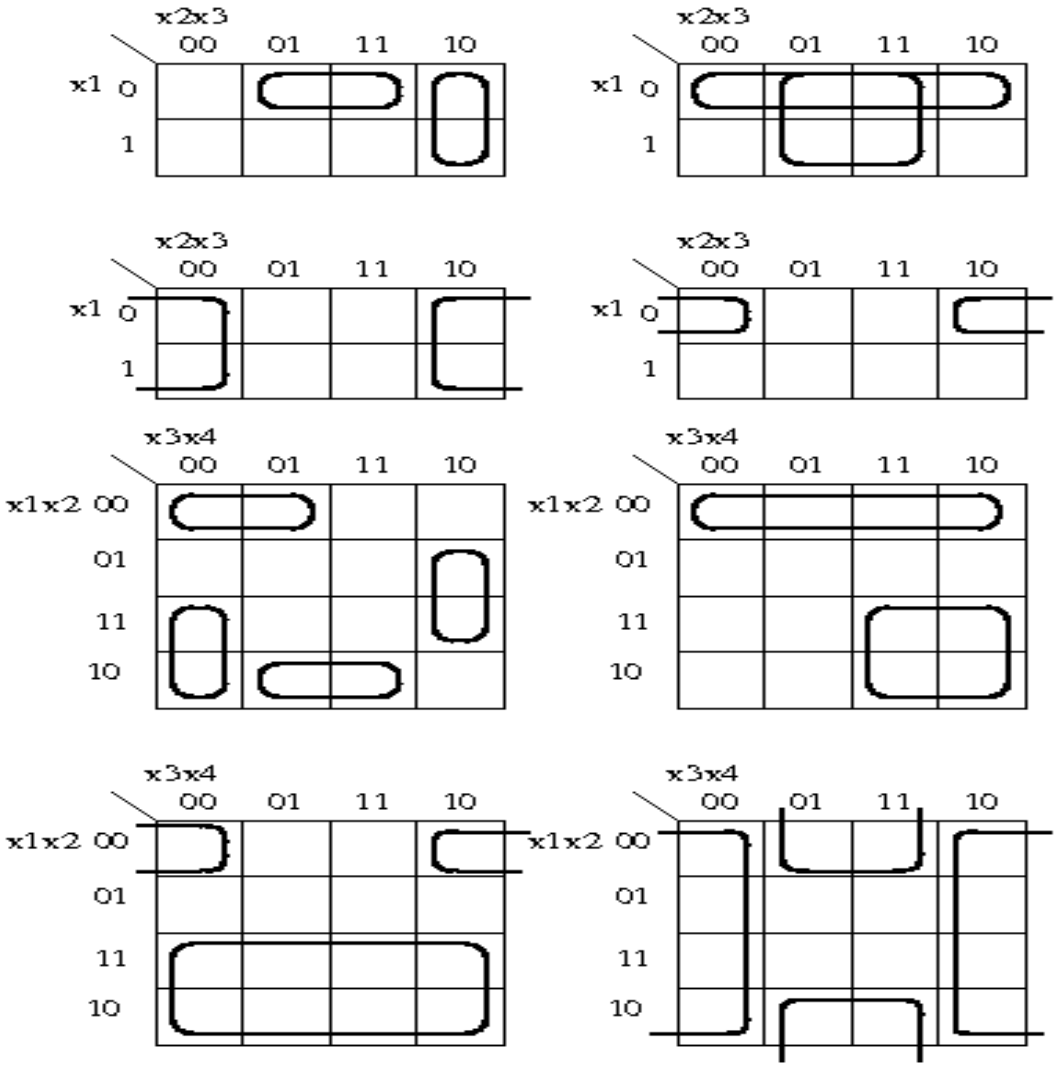


Figure 3: Przykłady tablic Karnaugha

Zasady sklejania w tablicach Karnaugha dla postaci DNF

Konstrukcja tablic Karnaugha jest taka, że każde dwie sąsiednie kratki specyfikują iloczyny logiczne różniące się dokładnie na jednej pozycji; pola takie można zatem skleić, analogicznie jak iloczyny zmiennych.

Obowiązują następujące zasady sklejania:

- oznaczenia kratek:
  - $n = 2$: 00, 01, 11, 10,
  - $n = 3$: 000, 001, 011, 010, 110, 111, 101, 100.
- sklejaniu podlegają wszystkie pola zawierające 1; pola zawierające $-$ mogą, ale nie muszą być pokryte (są uwzględniane w miarę potrzeb),
- sklejane obszary muszą mieć kształt prostokąta; sklejane mogą być również "prostokąty" skonstruowane na zasadzie sąsiedztwa pól rozmieszczonych symetrycznie na krawędziach,
- należy wykorzystać prostokąty pokrywające jak największą liczbę jedynek,
- już pokryte jedynki można wykorzystać do ponownego sklejania, o ile jest taka potrzeba,
- dla danego prostokąta budowany jest iloczyn zmiennych, przy czym:
  - jeżeli wewnątrz obszaru dana zmienna przyjmuje stale wartość 1, to jest uwzględniana bez negacji,
  - jeżeli wewnątrz obszaru dana zmienna przyjmuje stale wartość 0, to jest uwzględniana z negacją,
  - jeżeli wewnątrz obszaru dana zmienna przyjmuje wartość 1 i 0, to jest pomijana.

Zasady sklejania w tablicach Karnaugha dla postaci CNF

Konstrukcja tablic Karnaugha jest taka, że każde dwie sąsiednie kratki specyfikują sumy logiczne różniące się dokładnie na jednej pozycji; pola takie można zatem skleić, analogicznie jak sumy zmiennych.

Obowiązują następujące zasady sklejania:

- oznaczenia kratek:
  - $n = 2$: 00, 01, 11, 10,
  - $n = 3$: 000, 001, 011, 010, 110, 111, 101, 100.
- sklejaniu podlegają wszystkie pola zawierające 0; pola zawierające $-$ mogą, ale nie muszą być pokryte (są uwzględniane w miarę potrzeb),
- sklejane obszary muszą mieć kształt prostokąta; sklejane mogą być również "prostokąty" skonstruowane na zasadzie sąsiedztwa pól rozmieszczonych symetrycznie na krawędziach,
- należy wykorzystać prostokąty pokrywające jak największą liczbę jedynek,
- już pokryte zera można wykorzystać do ponownego sklejania, o ile jest taka potrzeba,
- dla danego prostokąta budowana jest suma zmiennych, przy czym:
  - jeżeli wewnątrz obszaru dana zmienna przyjmuje stale wartość 0, to jest uwzględniana bez negacji,
  - jeżeli wewnątrz obszaru dana zmienna przyjmuje stale wartość 1, to jest uwzględniana z negacją,
  - jeżeli wewnątrz obszaru dana zmienna przyjmuje wartość 1 i 0, to jest pomijana.

Przykład syntezy: sumator jednobitowy

Specyfikacja zadania

Należy skonstruować układ logiczny stanowiący sumator jednobitowy, uwzględniający również przeniesienie. Na wejściu sumatora mamy dwa sumowane bity (sygnały p i q) oraz bit przeniesienia c. Na wyjściu sumatora mamy bit wyniku s oraz bit przeniesienia t. Bit wyniku s jest równy 1 wtedy i tylko wtedy gdy suma sygnałów wejściowych jest nieparzysta (jedna albo trzy jedynki). Bit przeniesienia t jest równy 1 wtedy i tylko wtedy gdy sumowane są co najmniej dwie jedynki. Odpowiednią tablicę specyfikacji funkcji s i t podano poniżej.

p	q	c	s	t
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Postacie kanoniczne funkcji s i t mogą być odczytane bezpośrednio z tabeli; na tej podstawie można wypełnić odpowiednie tabele Karnaugha. Otrzymane postacie minimalne funkcji s i t są następujące:

$$s = \bar{p}qc \vee p\bar{q}\bar{c} \vee \bar{p}\bar{q}c \vee pqc,$$

$$t = pq \vee pc \vee qc.$$

Inna postać z wykorzystaniem EX-OR): $s = p \oplus q \oplus c$, $t = pq \vee (p \oplus q)c$.

Układy logiczne

Z bramek logicznych można budować dowolnie złożone układy logiczne.

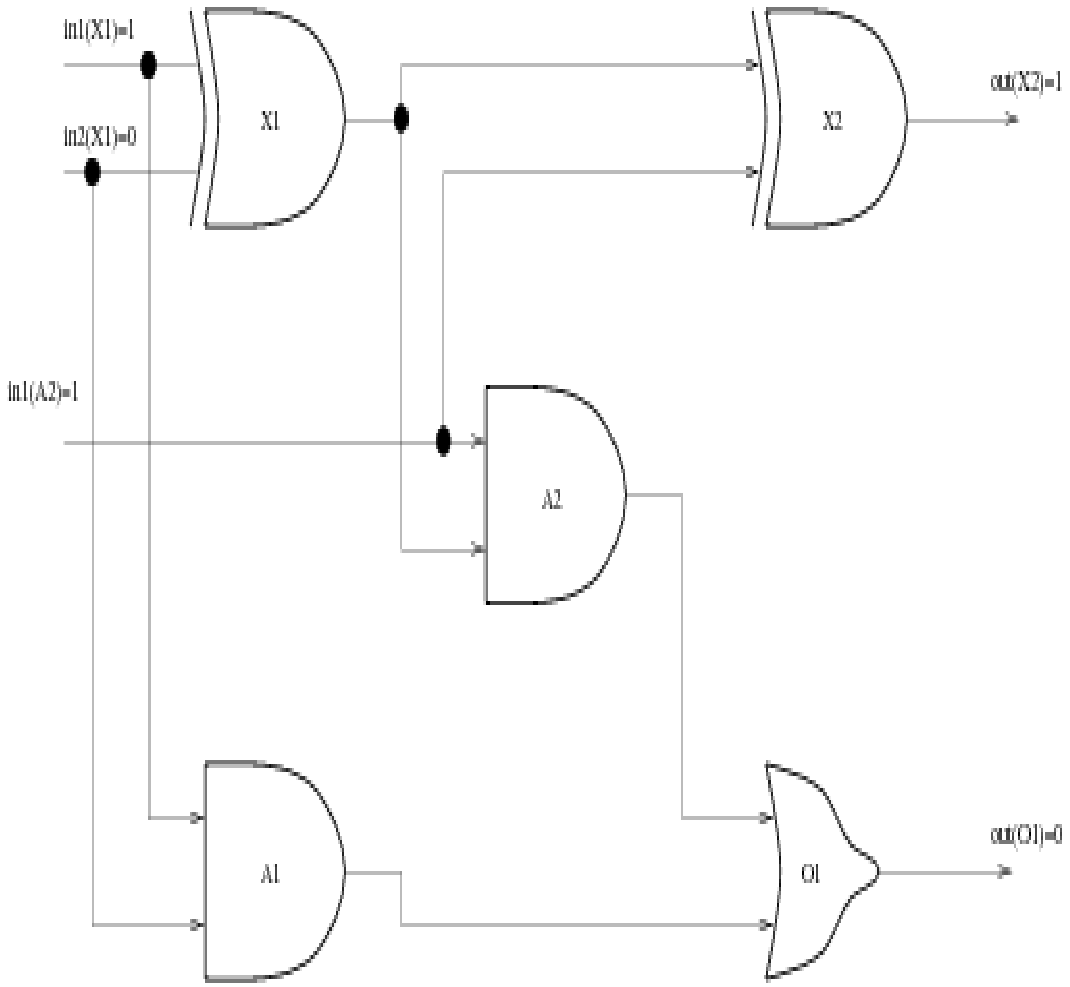


Figure 4: Przykład układu logicznego – sumator.

Example: Logical Consequence Verification (EX-LCV16)

N	p	q	r	s	$p \Rightarrow q$	$r \Rightarrow s$	$(p \Rightarrow q) \wedge (r \Rightarrow s)$	$p \vee r$	$q \vee s$	$(p \vee r) \Rightarrow (q \vee s)$
0	0	0	0	0	1	1	**1**	0	0	**1**
1	0	0	0	1	1	1	**1**	0	1	**1**
2	0	0	1	0	1	0	**0**	1	0	**0**
3	0	0	1	1	1	1	**1**	1	1	**1**
4	0	1	0	0	1	1	**1**	0	1	**1**
5	0	1	0	1	1	1	**1**	0	1	**1**
6	0	1	1	0	1	0	**0**	1	1	**1**
7	0	1	1	1	1	1	**1**	1	1	**1**
8	1	0	0	0	0	1	**0**	1	0	**0**
9	1	0	0	1	0	1	**0**	1	1	**1**
10	1	0	1	0	0	0	**0**	1	0	**0**
11	1	0	1	1	0	1	**0**	1	1	**1**
12	1	1	0	0	1	1	**1**	1	1	**1**
13	1	1	0	1	1	1	**1**	1	1	**1**
14	1	1	1	0	1	0	**0**	1	1	**1**
15	1	1	1	1	1	1	**1**	1	1	**1**

		rs			
		00	01	11	10
pq	00				
	01				
	11				
	10				

EX-LCV16: Karnaugh Tables

$$\phi = (p \Rightarrow q) \wedge (r \Rightarrow s)$$

		rs			
		00	01	11	10
pq	00	1	1	1	0
	01	1	1	1	0
	11	1	1	1	0
	10	0	0	0	0

$$\varphi = (p \vee r) \Rightarrow (q \vee s)$$

		rs			
		00	01	11	10
pq	00	1	1	1	0
	01	1	1	1	1
	11	1	1	1	1
	10	0	1	1	0

Postacie minimalne

The respective minimal forms are:

DNF:

$$\phi = \bar{p}\bar{r} \vee \bar{p}s \vee q\bar{r} \vee qs$$

$$\varphi = \bar{p}\bar{r} \vee q \vee s$$

CNF:

$$\phi = (\bar{r} \vee s) \wedge (\bar{p} \vee q)$$

$$\varphi = (\bar{p} \vee q \vee s) \wedge (\bar{r} \vee q \vee s)$$

7-Segment Display

Consider the 7-segment display as below:

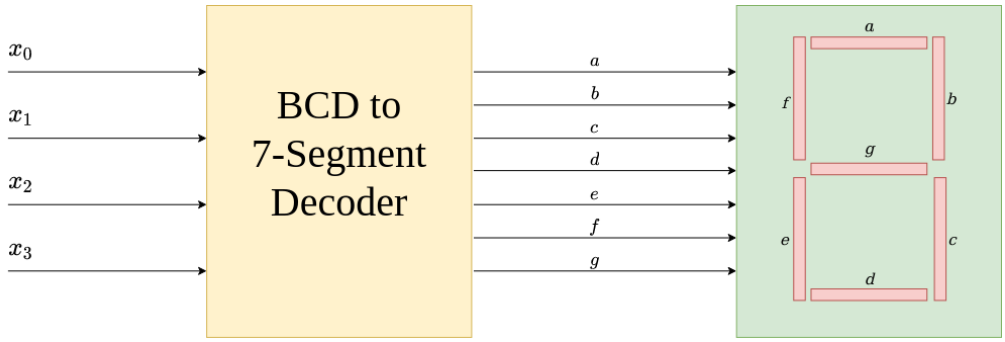


Figure 5: An example scheme presenting the input and output of the controller and connection to the 7-segment display

The function for calculating displayed digits:

$$D = x_3 * 2^3 + x_2 * 2^2 + x_1 * 2 + x_0$$

Inputs and outputs:

x: 3,2,1,0	f: a,b,c,d,e,f,g
0,0,0,0	1,1,1,1,1,1,0
0,0,0,1	0,1,1,0,0,0,0
0,0,1,0	1,1,0,1,1,0,1
0,0,1,1	1,1,1,1,0,0,1
0,1,0,0	0,1,1,0,0,1,1
0,1,0,1	1,0,1,1,0,1,1
0,1,1,0	1,0,1,1,1,1,1
0,1,1,1	1,1,1,0,0,0,0
1,0,0,0	1,1,1,1,1,1,1
1,0,0,1	1,1,1,1,0,1,1

The 7 segment activation functions

$$a = x_2x_0 + x_1 + x_3 + \bar{x}_2\bar{x}_0$$

$$b = x_1x_0 + \bar{x}_2 + \bar{x}_1\bar{x}_0.$$

$$c = x_0 + x_2 + \bar{x}_1.$$

$$d = x_2\bar{x}_1x_0 + \bar{x}_2x_1 + x_3 + x_1\bar{x}_0 + \bar{x}_2\bar{x}_0.$$

$$e = x_1\bar{x}_0 + \bar{x}_2\bar{x}_0.$$

$$f = x_3 + x_2\bar{x}_1 + x_2\bar{x}_0 + \bar{x}_1\bar{x}_0.$$

$$g = \bar{x}_2x_1 + x_3 + x_2\bar{x}_1 + x_2\bar{x}_0.$$

Algorytm Quine'a McCluskeya

Algorytm Quine'a McCluskeya

1. Zdefiniować tablicę prawdy syntetyzowanej funkcji.
2. Wyznaczyć postać kanoniczną DNF.
3. Ponumerować mintermy (implikanty pełne) definiujące jedynki.
4. Zapisać mintermy jedynkowe w postaci iloczynow zer i jedynek (reprezentacja binarna).
5. Pogrupować mintermy w grupy o identycznej liczbie jedynek.
6. Metodą kolejnego sklejania wyznaczyć wszystkie implikanty proste. Sklejaniu podlegają jedynie mintermy sąsiednich grup (różniące się na jednej pozycji).
7. Wybrać (dowolny) zbiór implikantów prostych reprezentujący (pokrywający) całą funkcję wyjściową (wszystkie mintermy).

`http://en.wikipedia.org/wiki/Quine-McCluskey`

Zastosowania techniczne i nie tylko...

Języki opisu układów cyfrowych

- Verilog

  `https://en.wikipedia.org/wiki/Verilog`;

- VHDL

  `https://en.wikipedia.org/wiki/VHDL`;

- Chisel

  `https://en.wikipedia.org/wiki/Chisel_(programming_language)`.

Programmable Logic Controllers; PLC

`https://en.wikipedia.org/wiki/Programmable_logic_controller`

Logika rozmyta

`http://zsi.tech.us.edu.pl/~nowak/si/w2_2013.pdf`

Na co komu logika?

`https://www.kul.pl/files/233/wyklady/wyklad.pdf`