Metody analityczne i komputerowe
w minimalizacji funkcji boolowskich

Metoda Quine'a McCluskey'a

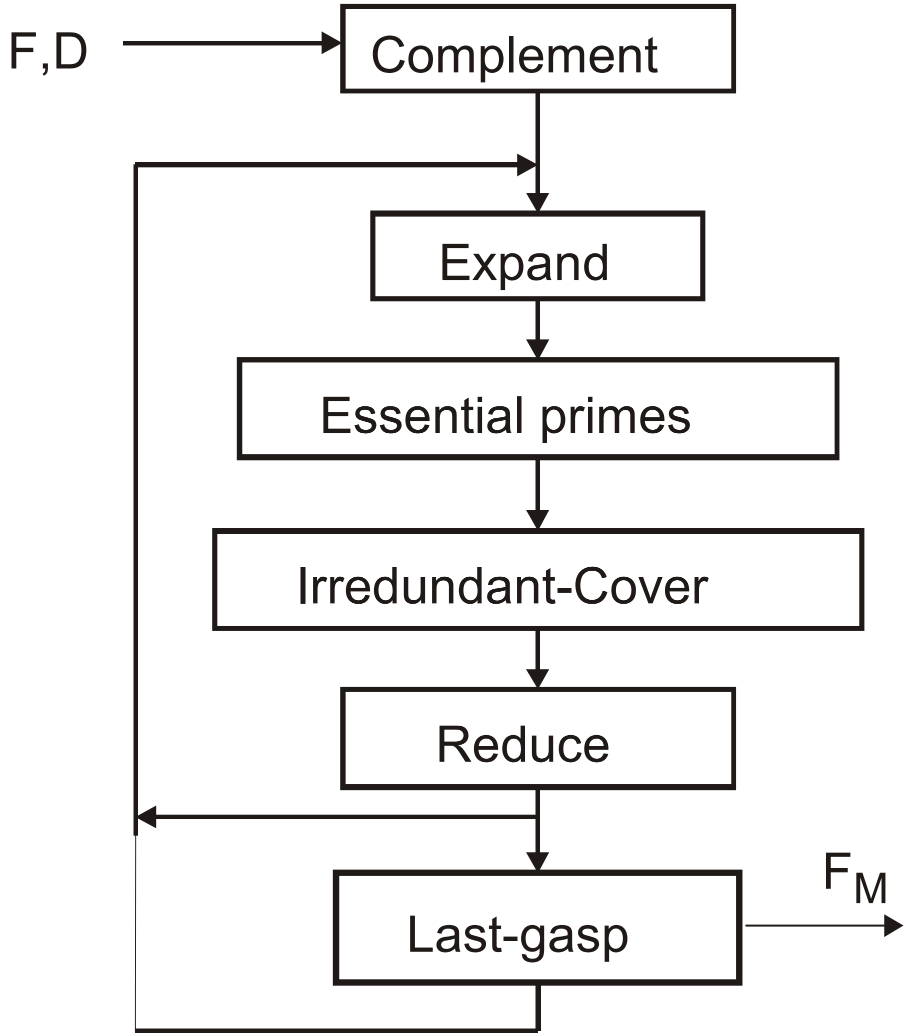
a) generacja implikantów prostych
b) selekcja implikantów (tzw. pokrycie)

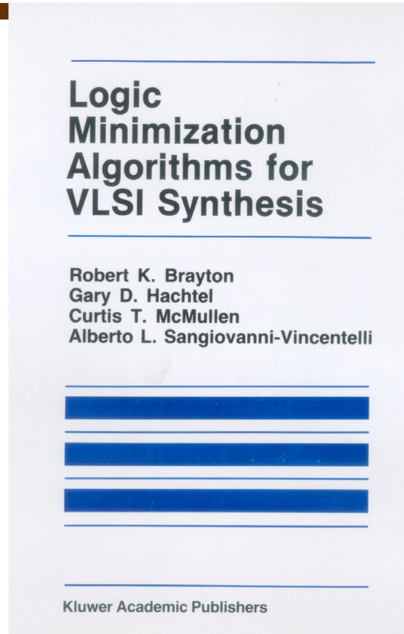
Metoda Espresso

- duża liczba różnorodnych procedur
- procedury heurystyczne
- iteracyjne poprawianie wyniku

I T P W ZPT

Procedury systemu ESPRESSO





(rozdział 6 w książce SUL)

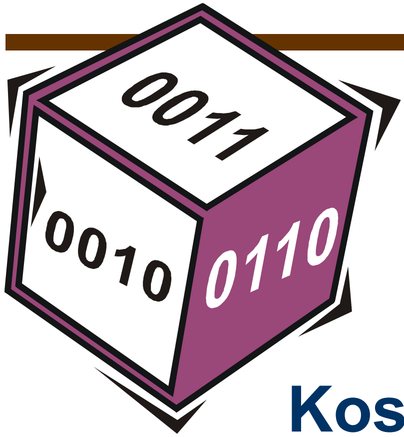
ITPW ZPT

Zmodyfikowana metoda ekspansji
(rozdział 3.3 książka SUL)

Łączy idee metody Quine'a McCluskey'a oraz metody Espresso:

a) generacja implikantów prostych (wg Espresso)
b) selekcja implikantów (wg Quine'a McCluskey'a)

Metoda ta zrealizowana w programie PANDOR jest udostępniona na www.zpt.tele.pw.edu.pl w katalogu OPROGRAMOWANIE



Pojęcia podstawowe

Kostka K to krotka o składowych 0, 1, $*$ reprezentująca zbiór wektorów zero-jedynkowych.

K = (0$*$1$*$), to zbiór wektorów:

0010
0011
0110
0111

Kostka reprezentuje **niepełny iloczyn**:

$K = 0{*}1{*} = \bar{x}_1 x_3$

I
T
P
W

ZPT

Oznaczenia

W standardzie espresso wektory (w ogólności kostki), dla których funkcja f = 1 oznacza się zbiorem ***F.***

Wektory (kostki) dla których funkcja f = 0 oznacza się zbiorem ***R.***

$$\boldsymbol{f = (F, R)}$$

I T P W

ZPT

Przykład (EXTL)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f
	1	0	0	0	1	0	1	0
	1	0	1	1	1	1	0	0
	1	1	0	1	1	1	0	0
	1	1	1	0	1	1	1	0
k_1	0	1	0	0	1	0	1	1
k_2	1	0	0	0	1	1	0	1
k_3	1	0	1	0	0	0	0	1
k_4	1	0	1	0	1	1	0	1
k_5	1	1	1	0	1	0	1	1

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Ekspansja

Ekspansja jest procesem działającym na kostkach zbiorów ***F*** i ***R***, a jej celem jest uzyskanie dla danej $k \in$ ***F*** kostki k' tak dużej, jak to tylko możliwe (tzn. z możliwie dużą liczbą pozycji o wartości $*$) i nie pokrywającej żadnego wektora zbioru ***R***.

W swoich obliczeniach Ekspansja wykorzystuje tzw. macierz blokującą ***B***.

I T P W

ZPT

Macierz blokująca

Definicja oryginalna (z książki Braytona):

Macierzą blokującą (kostkę k) nazywamy macierz $\boldsymbol{B}(k,\boldsymbol{R}) = [b_{ij}]$, w której każdy element $b_{ij} \in \{0,1\}$ jest definiowany następująco:

$b_{ij} = 1$, jeśli $k_j = 1$ oraz $r_{ij} = 0$ lub $k_j = 0$ oraz $r_{ij} = 1$;

$b_{ij} = 0$, w pozostałych przypadkach.

Macierz blokująca dla danej kostki $k \in \boldsymbol{F}$ powstaje z macierzy $\boldsymbol{R}$ przez zanegowanie tych kolumn $\boldsymbol{R}$, których pozycje są wyznaczone przez pozycje jedynek w kostce $k \in \boldsymbol{F}$.

I T P W

ZPT

Tworzenie macierzy blokującej

Wyznaczymy macierz blokującą dla kostki k_1 wiedząc, że ***F*** i ***R*** są opisane macierzami:

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Skoro k_1 = (0100101), to dla uzyskania ***B*** wystarczy w macierzy ***R*** "zanegować" kolumny drugą, piątą i siódmą.
Zatem ***B***(k_1,***R***):

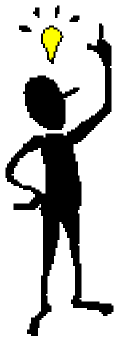
$$B = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Pokrycie kolumnowe

Pokryciem kolumnowym macierzy ***B*** jest zbiór kolumn L ($L \subseteq \{1,...,n\}$) taki, że dla każdego wiersza i istnieje kolumna $j \in L$, która w wierszu i ma jedynkę.

Zbiór *L* jest minimalnym pokryciem kolumnowym macierzy ***B,*** jeśli nie istnieje zbiór L' (tworzący pokrycie) taki, że $L \supseteq L'$.

Pokrycie kolumnowe jest pojęciem ogólnym, można go tworzyć dla każdej macierzy binarnej



Obliczanie pokrycia kolumnowego

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
B = & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}
\end{array}
\begin{array}{l} \{6,7\} \\ \{3,4\} \\ \{2,4\} \\ \{2,3,7\} \end{array}
$$

O obliczaniu *L* będziemy jeszcze mówili

L = {4,7} jest pokryciem kolumnowym.

L = {2,3,6} jest pokryciem kolumnowym.

L = {2,3} – nie, L = {2,6} – nie, L = {3,6} – nie.

Generacja (tworzenie) implikantów

Macierz blokująca $\boldsymbol{B}(k,\boldsymbol{R})$ pozwala wyznaczyć ekspansję kostki k oznaczaną $k^{+}(L,k)$ w sposób następujący: wszystkie składowe kostki k należące do L nie ulegają zmianie, natomiast składowe nie należące do L przyjmują wartość $*$.

Ekspansja kostki k jest implikantem funkcji $f = (\boldsymbol{F},\boldsymbol{R})$.

W szczególności $k^{+}(L,k)$ jest implikantem prostym, gdy L jest minimalnym pokryciem kolumnowym macierzy $\boldsymbol{B}(k,\boldsymbol{R})$.

I T P W ZPT

Generacja implikantów - przykład

$$\text{Dla } k_2 = (1000110) \text{ i macierzy } \boldsymbol{B} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

zbiór L = {4,7} jest pokryciem kolumnowym ***B***, a więc

k_2 = (1000110)

$k^+(L, k_2) = (\ast\ast\ast0\ast\ast0)$, czyli implikantem ***F*** jest $\bar{x}_4\bar{x}_7$

Natomiast dla L = {2,3,6} (inne pokrycie kolumnowe),

$k^+(L,k) = (\ast00\ast\ast1\ast) = \bar{x}_2\bar{x}_3x_6$

I T P W ZPT

Implikanty proste

Obliczając kolejno implikanty proste dla każdej $k \in \boldsymbol{F}$ uzyskuje się:

$$I_1 = \bar{x}_1$$
$$I_2 = x_2\bar{x}_6$$
$$I_3 = \bar{x}_4\bar{x}_7$$
$$I_4 = \bar{x}_2\bar{x}_3x_6$$

$$I_5 = \bar{x}_5$$
$$I_6 = \bar{x}_6\bar{x}_7$$
$$I_7 = x_3\bar{x}_6$$

Proszę zauważyć, że na tym zakończyliśmy proces generacji implikantów prostych

I T P W ZPT

… przystępujemy do procesu selekcji

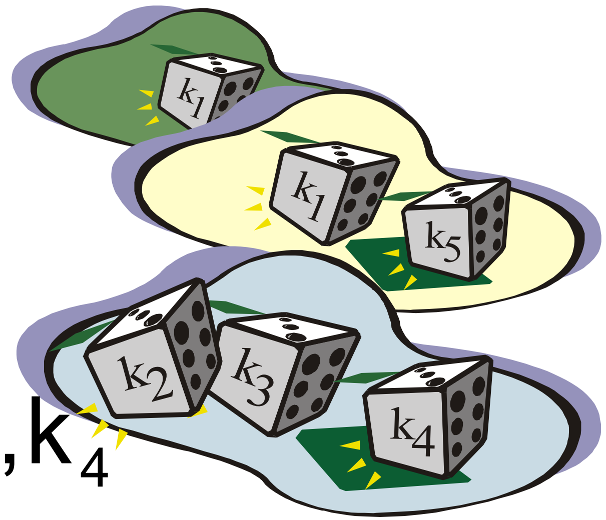
Relacja pokrycia dla kostek

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
k_1	0	1	0	0	1	0	1
k_2	1	0	0	0	1	1	0
k_3	1	0	1	0	0	0	0
k_4	1	0	1	0	1	1	0
k_5	1	1	1	0	1	0	1

$$I_1 = \bar{x}_1 \quad (0******) \supseteq k_1$$

$$I_2 = x_2\bar{x}_6 \quad (*1***0*) \supseteq k_1, k_5$$

$$I_3 = \bar{x}_4\bar{x}_7 \quad (***0**0) \supseteq k_2, k_3, k_4$$



I T P W ZPT

Tablica implikantów prostych

$$I_1 = \bar{x}_1 \qquad (0{*}{*}{*}{*}{*}{*}) \supseteq k_1$$
$$I_2 = x_2\bar{x}_6 \qquad ({*}1{*}{*}{*}0{*}) \supseteq k_1, k_5$$
$$I_3 = \bar{x}_4\bar{x}_7 \qquad ({*}{*}{*}0{*}{*}0) \supseteq k_2, k_3, k_4$$

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
k_1	1	1	0	0	0	0	0
k_2	0	0	1	1	0	0	0
k_3	0	0	1	0	1	1	1
k_4	0	0	1	0	0	0	0
k_5	0	1	0	0	0	0	1

I T P W

ZPT

Tablica implikantów prostych umożliwia wybór (selekcję) takiego minimalnego zbioru implikantów, który pokrywa wszystkie kostki funkcji pierwotnej

Selekcja implikantów prostych

	l_1	l_2	l_3	l_4	l_5	l_6	l_7
k_1	1	1	0	0	0	0	0
k_2	0	0	1	1	0	0	0
k_3	0	0	1	0	1	1	1
k_4	0	0	1	0	0	0	0
k_5	0	1	0	0	0	0	1

Inny zapis tablicy:

l_1, l_2
$\cancel{l_3, l_4}$
$\cancel{l_3, l_5, l_6, l_7}$
$\cancel{l_3}$
l_2, l_7

$\Rightarrow$

l_1, l_2
l_2, l_7

$$l_2 = x_2\bar{x}_6$$

$$l_3 = \bar{x}_4\bar{x}_7$$

Minimalne pokrycie: l_2, l_3

Minimalna formuła: $\bar{x}_4\bar{x}_7 + x_2\bar{x}_6$

I T P W ZPT

Implementacja metody – program Pandor

Fragment pliku wyjściowego Pandora:

Implicants table of function y1
10010
01000
01101
01000
00011

....

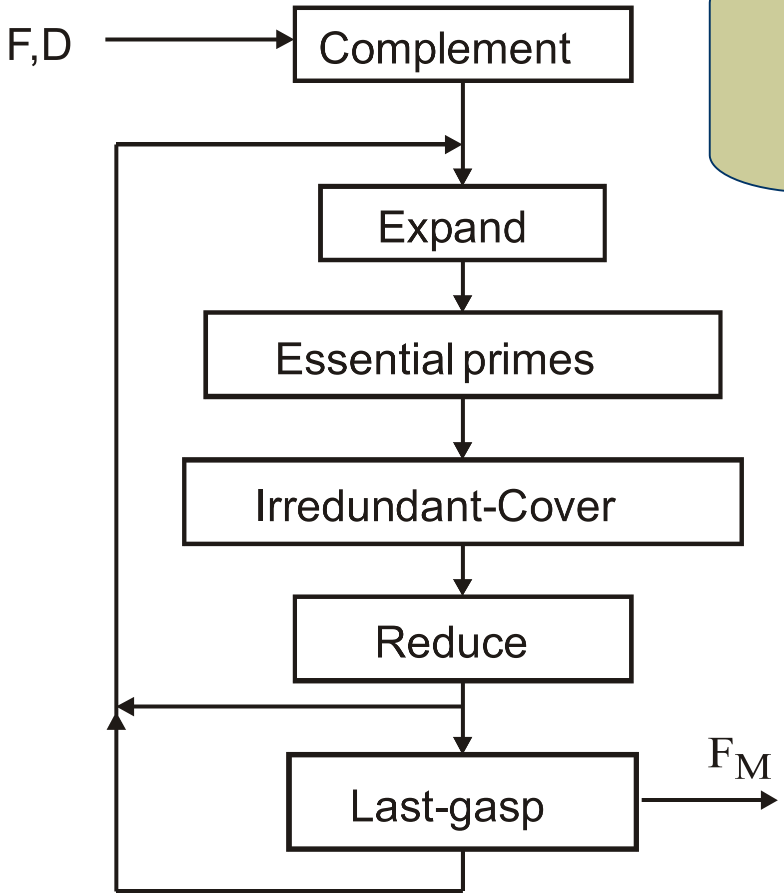
All results of function y1
y1 = !x4!x7 + x2!x6

Trochę inny zapis

Taki sam wynik generuje Espresso…

```
.type fr
.i 7
.o 1
.p 9
.ilb x1 x2 x3 x4 x5 x6 x7
.ob y
1000101 0
1011110 0
1101110 0
1110111 0
0100101 1
1000110 1
1010000 1
1010110 1
1110101 1
.e
```

…funkcja EXTL
w formacie espresso



Więcej na temat formatu espresso w książce SUL


```
.i 7
.o 1
.ilb x1 x2 x3 x4 x5 x6 x7
.ob y
.p 2
-1---0- 1
---0--0 1
.e
```

I T P W ZPT

Plik wejściowy i wyjściowy przykładu

.type fr
.i 7
.o 1
.p 9
.ilb x1 x2 x3 x4 x5 x6 x7
.ob y
1000101 0
1011110 0
1101110 0
11101
01001
10001
1010000 1
1010110 1
1110101 1
.e

.i 7
.o 1
.ilb x1 x2 x3 x4 x5 x6 x7
.ob y
.p 2
-1---0- 1
---0--0 1
.e

Skoro wyszło to samo co w obliczeniach za pomocą tylko jednej procedury ekspansji...

To jest za prosty przykład!

I
T
P
W

ZPT

Porównanie Pandora i Espresso wymaga szczegółowszych eksperymentów.

Można je przeprowadzić samodzielnie. Przykładowe pliki oraz programy Pandor i Espresso są umieszczone w katalogu *Eksperymenty do wykładów cz. 3 i cz. 4.*

Dyskusja…

Barierą ograniczającą obliczenia systematyczną metodą zastosowaną w Pandorze jest ogromna złożoność obliczeniowa zarówno w procesie generacji, jak też w procesie obliczania pokrycia kolumnowego.

Program PANDOR został stworzony po to, aby naocznie zaobserwować zjawisko wzrostu złożoności obliczeniowej wraz ze wzrostem liczby argumentów funkcji.

Funkcja EXTL ma 7 implikantów (Pandor dokonuje lepszej selekcji i oblicza ich zaledwie 5). Nie ma żadnej sprawy w obliczeniu minimalnego pokrycia kolumnowego.

Ale…

I T P W ZPT

Zagadka…

KAZ


```
.type fr
.i 21
.o 1
.p 31
100110010110011111101 1
111011111011110111100 1
001010101000111100000 1
001001101100110110001 1
100110010011011001101 1
100101100100110110011 1
001100100111010011011 1
001101100011011011001 1
110110010011001001101 1
100110110011010010011 1
110011011011010001100 1
010001010000001100111 0
100110101011111110100 0
111001111011110011000 0
101101011100010111100 0
110110000001010100000 0
110110110111100010111 0
110000100011110010001 0
001001000101111101101 0
100100011111100110110 0
100011000110011011110 0
110101000110101100001 0
110110001101101100111 0
010000111001000000001 0
001001100101111110000 0
100100111111001110010 0
000010001110001101101 0
101000010100001110000 0
101000110101010011111 0
101010000001100011001 0
011100111110111101111 0
.end
```

Ile implikantów prostych ma funkcja TL27

…a ile KAZ? Można pomylić się o 10…

I dlatego TL27 obliczy zarówno systematyczna ekspansja Pandora, jak i Espresso. Ale już funkcja KAZ jest z praktycznego punktu widzenia realna do policzenia wyłącznie programem ESPRESSO.

Ale nie może to być wynik minimalny

O innych zaletach Pandora na następnym wykładzie

TL27

```
.type fr
.i 10
.o 1
.p 25
0010111010 0
1010010100 0
0100011110 0
1011101011 0
1100010011 0
0100010110 0
1110100110 0
0100110000 0
0101000010 0
0111111011 1
0000010100 1
1101110011 1
0100100000 1
0100011111 1
0010000110 1
1111010001 1
1111101001 1
1111111111 1
0010000000 1
1101100111 1
0010001111 1
1111100010 1
1010111101 1
0110000110 1
0100111000 1
.e
```

Dalsze zalety Espresso…

Metoda Espresso jest szczególnie efektywna w minimalizacji zespołów funkcji boolowskich. Dla metod klasycznych synteza wielowyjściowych funkcji boolowskich jest procesem bardzo złożonym – trudnym do zalgorytmizowania.

Przypomnijmy przykład z poprzedniego wykładu

Układy wielowyjściowe - przykład

$y_1 = \Sigma(2,3,5,7,8,9,10,11,13,15)$

$y_2 = \Sigma(2,3,5,6,7,10,11,14,15)$

$y_3 = \Sigma(6,7,8,9,13,14,15)$

Po żmudnych obliczeniach uzyskaliśmy wynik na 5 bramkach AND

$$y_1 = \bar{b}c + \bar{a}bd + abd + a\bar{b}\bar{c}$$

$$y_2 = \bar{b}c + \bar{a}bd + bc$$

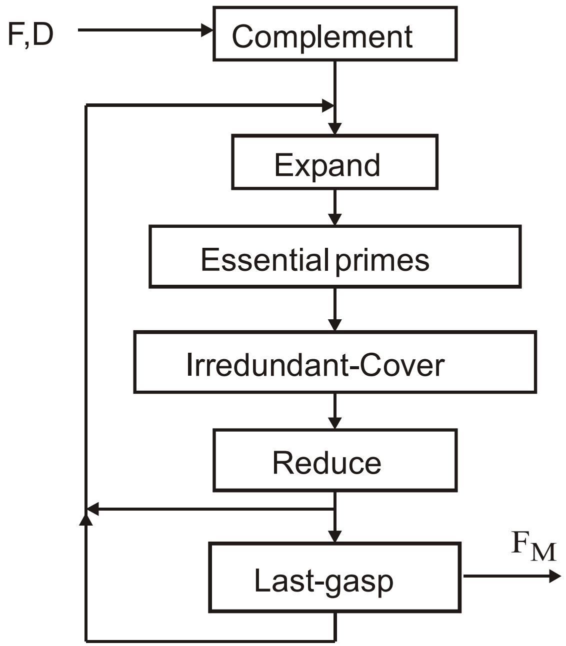
$$y_3 = abd + a\bar{b}\bar{c} + bc$$

ITPW

ZPT

Jak obliczy Espresso?

```
.type fr
.i 4
.o 3
.p 16
0000 000
0001 000
0010 110
0011 110
0100 000
0101 110
0110 011
0111 111
1000 101
1001 101
1010 110
1011 110
1100 000
1101 101
1110 011
1111 111
.e
```




```
.i 4
.o 3
.p 5
11-1 101
100- 101
01-1 110
-01-  110
-11-  011
.e
```

Łatwo sprawdzić, że jest to taki sam wynik jak na poprzedniej planszy