Metoda Quine'a – McCluskeya,
Synteza automatów skończonych

Tomasz Kapłon

18.10.2021 r.

Część I

Metoda Quine'a – McCluskeya

Przypomnienie i uzupełnienie (1/3)

Etapy projektowania układu kombinacyjnego

1. Opis działania układu.
2. Ustalenie sygnałów WE i WY (jeśli nie podano wprost w opisie).
3. Określenie funkcji przełączającej [tabela prawdy i minimalizacja].
4. Znalezienie minimalnej postaci funkcji (minimalnej zgodnie z celem minimalizacji).
5. Stworzenie schematu układu.
6. *Ocena kosztu realizacji, ewentualnie korekta układu.*

Przypomnienie i uzupełnienie (2/3)

Metody opisu układu

1. Opis słowny
2. Tablica prawdy
3. Zbiór jedynek / zer funkcji F (F^1 / F^0)
4. Funkcji boolowska – postać kanoniczna iloczynu / sumy bądź postać zminimalizowana
5. Schemat układu

Przypomnienie i uzupełnienie (3/3)

Metody minimalizacji wyrażeń logicznych

1. Przekształcenia w oparciu o prawa algebry Boole'a
2. Metoda siatek Karnaugha

   M. Karnaugh, "The map method fo synthesis of combinational logic circuits",

   Transactions of the American Institute for Electrical Engineers, Part I (vol. 72, Issue 5, Nov. 1953)
3. **Metoda Quine'a – McCluskeya**

   E. J. McCluskey Jr., "Minimization of Boolean functions",

   Bell Systems Technical Journal, **35**:6 (November 1956), 1417–1444
4. *Metoda Kazakowa*

   V. D. Kazakov, "Minimization of logical functions of great number of variables",

   Avtomat. i Telemekh., **23**:9 (1962), 1237–1242
5. *Metoda Tablic Niezgodności*
6. *Metoda ESPRESSO*

   R. Barton [et al.], "Logic minimization algorithm for VLSI synthesis",

   Kluwer Academic Publishers, ISBN 978-0-89838-164-1

Metoda Quine'a - McCluskeya

Metoda minimalizacji funkcji boolowskich

William Van Orman Quine – filozof i logik (1908 – 2000), amerykanin

Edward J. McCluskey – (1929 – 2016), amerykanin

1. Funkcja F opisujemy przy pomocy dwóch zbiorów: F^1 i F^*.

   F^1 to zbiór argumentów, dla których F przyjmuje wartość 1.

   F^* to zbiór argumentów, dla których F przyjmuje wartość nieokreśloną. Łączymy oba zbiory w zbiór F.

2. Porządkujemy (wg wartości dziesiętnej) argumenty zbioru F. Przypisujemy każdemu postać binarną.
3. Grupujemy wektory binarne o tej samej liczbie jedynek.
4. Porównujemy wektory o k-tej liczbie jedynek z wektorami o k+1 liczbie jedynek i łączymy te, które różnią się tylko na jednej pozycji. Pozycję tę oznaczamy. Wektory użyte do połączeń oznaczamy. Powtarzające się kombinacje usuwamy.

Kroki 3 i 4 powtarzamy do wyczerpania się możliwości połączeń. Każdy wektor nie podlegający dalszemu łączeniu nazywany jest implikantem prostym.

5. Wybieramy wektory nieoznaczone (implikanty proste).
6. Tworzymy macierz, w wierszach której zapisujemy implikanty proste, w kolumnach argumenty zbioru F^1. W tabeli oznaczamy miejsca przecięcia kolumny reprezentującej dany iloczyn pełny oraz wiersza odpowiadającemu implikantowi prostemu zawierającemu określony iloczyn pełny.
7. Minimalną postać funkcji otrzymujemy tworząc sumę wybranych implikantów prostych, których wybór dokonywany jest tak, aby pokrywały one wszystkie rozważane iloczyny pełne.

Metoda Quina'a – McCluskeya (2/2)

1. Funkcja *F* opisujemy przy pomocy dwóch zbiorów F^1 i F^*.

   F^1 to zbiór argumentów, dla których *F* przyjmuje wartość 1.

   F^* to zbiór argumentów, dla których F przyjmuje wartość nieokreśloną. Łączymy oba zbiory w zbiór *F*.

2. Porządkujemy (wg wartości dziesiętnej) argumenty zbioru *F*. Przypisujemy każdemu postać binarną.
3. Grupujemy wektory binarne o tej samej liczbie jedynek.
4. Porównujemy wektory o *k*-tej liczbie jedynek z wektorami o *k*+1 liczbie jedynek i łączymy te, które różnią się tylko na jednej pozycji. Pozycję tę oznaczamy. Wektory użyte do połączeń oznaczamy. Powtarzające się kombinacje usuwamy.

Kroki 3 i 4 powtarzamy do wyczerpania się możliwości połączeń. Każdy wektor nie podlegający dalszemu łączeniu nazywany jest implikantem prostym.

5. Wybieramy wektory nieoznaczone.
6. Tworzymy macierz, w wierszach której zapisujemy implikanty proste, w kolumnach argumenty zbioru F^1. W tabeli oznaczamy miejsca przecięcia kolumny reprezentującej dany iloczyn pełny oraz wiersza odpowiadającemu implikantowi prostemu zawierającemu określony iloczyn pełny.
7. Minimalną postać funkcji otrzymujemy tworząc sumę wybranych implikantów prostych, których wybór dokonywany jest tak, aby pokrywały one wszystkie rozważane iloczyny pełne.

Przykład 1 – metoda QMC (1 -7)

F = {1, 3, 4, 5, 6, 7, 8, 12}

1. Funkcja *F* opisujemy przy pomocy dwóch zbiorów F^1 i F^*.

2. Porządkujemy (wg wartości dziesiętnej) argumenty zbioru *F*. Przypisujemy każdemu postać binarną.

F	a	b	c	d
1	0	0	0	1
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
12	1	1	0	0

3. Grupujemy wektory binarne o tej samej liczbie jedynek.
4. Porównujemy wektory o *k*-tej liczbie jedynek z wektorami o *k*+1 liczbie jedynek i łączymy te, które różnią się tylko na jednej pozycji. Pozycję tę oznaczamy. Wektory użyte do połączeń oznaczamy. Powtarzające się kombinacje usuwamy.
Kroki 3 i 4 powtarzamy do wyczerpania się możliwości połączeń. Każdy wektor nie podlegający dalszemu łączeniu nazywany jest implikantem prostym.

1	0001
4	0100
8	1000
3	0011
5	0101
6	0110
12	1100
7	0111

1 - 3	00*1
1 - 5	0*01
4 - 5	010*
4 - 6	01*0
4 - 12	*100
8 - 12	1*00
3 - 7	0*11
5 - 7	01*1
6 - 7	011*

1 - 3 - 5 - 7	00*1
1 - 5 - 3 - 7	0*01
4 - 5 - 6 - 7	010*
4 - 6 - 5 - 7	01*0
4 - 12	*100
8 - 12	1*00

Powtarzające się kombinacje usuwamy.

1 - 3 - 5 - 7	0**1
~~1 - 5 - 3 - 7~~	~~0**1~~
4 - 5 - 6 - 7	01**
~~4 - 6 - 5 - 7~~	~~01**~~
4 - 12	*100
8 - 12	1*00

1 - 3 - 5 - 7	00*1
4 - 5 - 6 - 7	010*
4 - 12	*100
8 - 12	1*00

5 .Z każdej z kolumn wybieramy wektory nieoznaczone.
6. Tworzymy macierz, w wierszach której zapisujemy implikanty proste, w kolumnach argumenty zbioru F^1. W tabeli oznaczamy miejsca przecięcia kolumny reprezentującej dany iloczyn pełny oraz wiersza odpowiadającemu implikantowi prostemu zawierającemu określony iloczyn pełny.
7 .Minimalną postać funkcji otrzymujemy tworząc sumę wybranych implikantów prostych, których wybór dokonywany jest tak, aby pokrywały one wszystkie rozważane iloczyny pełne.

		1	3	4	5	6	7	8	12		
1-3-5-7	0**1	x	x				x			/a d	istotny
4-5-6-7	01**				x	x	x			/a b	
4-12	*100				x				x	b /c /d	istotny
8-12	1*00							x	x	a c /d	istotny

F = /ad + /ab + a/c/d

Przykład 1 – metoda QMC (1/6)

- 1. Funkcja *F* opisujemy przy pomocy dwóch zbiorów F^1 i F^*.

$$F = F^1\{1, 3, 4, 6, 7, 8, 12\} + F^*\{5\}$$

Przykład 1 – metoda QMC (2/6)

F = {1, 3, 4, 5, 6, 7, 8, 12}

- 2. Porządkujemy (wg. wartości dziesiętnej) argumenty zbioru *F*. Przypisujemy każdemu postać binarną.

F	a	b	c	d
1	0	0	0	1
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
12	1	1	0	0

Przykład 1 – metoda QMC (3/6)

- 3. Grupujemy wektory binarne o tej samej liczbie jedynek.

4. Porównujemy wektory o *k*-tej liczbie jedynek z wektorami o *k*+1 liczbie jedynek i łączymy te, które różnią się tylko na jednej pozycji. Pozycję tę oznaczamy. Wektory użyte do połączeń oznaczamy. Powtarzające się kombinacje usuwamy.

1	0001
4	0100
8	1000
3	0011
5	0101
6	0110
12	1100
7	0111

1 - 3	00*1
1 - 5	0*01
4 - 5	010*
4 - 6	01*0
4 - 12	*100
8 - 12	1*00
3 - 7	0*11
5 - 7	01*1
6 - 7	011*

Przykład 1 – metoda QMC (4/6)

- Kroki 3 i 4 powtarzamy do wyczerpania się możliwości połączeń. Każdy wektor nie podlegający dalszemu łączeniu nazywany jest implikantem prostym.

1	0001
4	0100
8	1000
3	0011
5	0101
6	0110
12	1100
7	0111

1 - 3	00*1
1 - 5	0*01
4 - 5	010*
4 - 6	01*0
4 - 12	*100
8 - 12	1*00
3 - 7	0*11
5 - 7	01*1
6 - 7	011*

1 - 3 - 5 - 7	00*1
1 - 5 - 3 - 7	0*01
4 - 5 - 6 - 7	010*
4 - 6 - 5 - 7	01*0
4 - 12	*100
8 - 12	1*00

Przykład 1 – metoda QMC (5/6)

Kroki 3 i 4 powtarzamy do wyczerpania się możliwości połączeń. Każdy wektor nie podlegający dalszemu łączeniu nazywany jest implikantem prostym.

- Powtarzające się kombinacje usuwamy.

1 - 3 - 5 - 7	0**1
~~1 - 5 - 3 - 7~~	~~0**1~~
4 - 5 - 6 - 7	01**
~~4 - 6 - 5 - 7~~	~~01**~~
4 - 12	*100
8 - 12	1*00

1 - 3 - 5 - 7	0**1
4 - 5 - 6 - 7	01**
4 - 12	*100
8 - 12	1*00

Przykład 1 – metoda QMC (6/6)

- 5 .Z każdej z kolumn wybieramy wektory nieoznaczone.

  6. Tworzymy macierz, w wierszach której zapisujemy implikanty proste, a w kolumnach argumenty zbioru F^1. W tabeli oznaczamy miejsca przecięcia kolumny reprezentującej dany iloczyn pełny oraz wiersza odpowiadającemu implikantowi prostemu zawierającemu określony iloczyn pełny.

  7 .Minimalną postać funkcji otrzymujemy tworząc sumę wybranych implikantów prostych, których wybór dokonywany jest tak, aby pokrywały one wszystkie rozważane iloczyny pełne.

		1	3	4	6	7	8	12		
1-3-5-7	0**1	x	x			x			/a d	istotny
4-5-6-7	01**			x	x	x			/a b	istotny
4-12	*100			x				x	b /c /d	
8-12	1*00						x	x	a c /d	istotny

F = /ad + /ab + a/c/d

Przykład 1 – metoda Karnaugha

a	b	c	d	F
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	ø
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	1	0	0	1

ab

cd	00	01	11	10
00	0	1	1	1
01	1	ø	0	0
11	1	1	0	0
01	0	1	0	0

F = /ad + /ab + a/c/d

minimalizacja zajęła < 30 sekund ☺

Metoda Karnaugha vs metoda Quine – McCluskey'a

Obie metody dają identyczne wyniki (?) i mają identyczny przebieg (?)

Metoda Karnaugha ma czytelną formę graficzną

Metoda QMC też ma czytelną formę, tyle że nie-graficzną

W obu wyszukuje się zależności relaksujących elementy wyrażenia boolowskiego

Metoda Karnaugha jest łatwa (?) do sześciu zmiennych

Metoda QMC – nie ma ograniczeń (?)

Funkcje wielu zmiennych redukuje jednak się metodami heurystycznymi, np. metodą *ESPRESSO*.

A teraz z (nieco) innej beczki (?)

Problem minimalizacji jest problemem NP-trudnym, którego czas realizacji rośnie wykładniczo.

Górne ograniczenie dla n zmiennych to $3^n/n$.

Część II

Synteza automatów skończonych

Co w temacie?

1. Synteza abstrakcyjna
2. Synteza strukturalna

Przypomnienie

Definicja 1

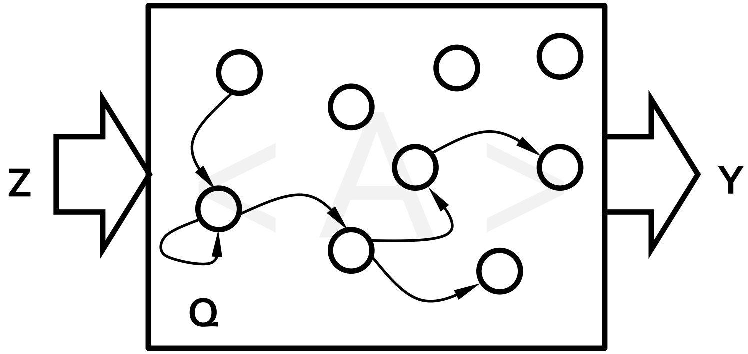
Automat skończony jest modelem matematycznym systemu dyskretnego, działającego w dyskretnych chwilach czasu. Jego działanie jest określone na skończonych zbiorach sygnałów wejściowych, stanów wewnętrznych i sygnałów wyjściowych.

Automat skończony jest przetwornikiem ciągu symboli wejściowych na ciąg symboli wyjściowych.

Wystąpienie określonych symboli na wyjściu automatu zależne jest do zestawu symboli na wejściu oraz od stanu wewnętrznego automatu.

Stan wewnętrzny związany jest z istnieniem pamięci.

Pamięć automatu jest tym większa, im więcej ma on stanów wewnętrznych.



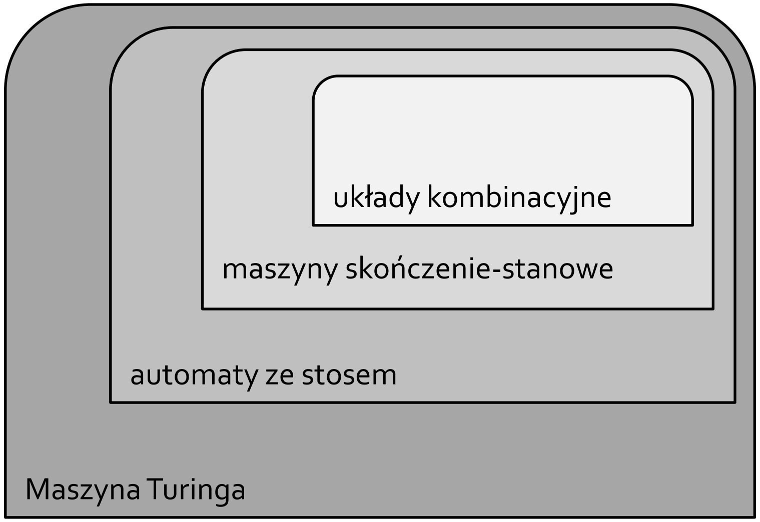
Przypomnienie i uzupełnienie

Teoria automatów (*ang. automata theory*) rozważa modele automatów i maszyn abstrakcyjnych oraz zajmuje się rozwiązywaniem problemów obliczeniowych z ich użyciem.

Teoria automatów jest ściśle związana z językami formalnymi.

Automat jest skończoną reprezentacją języka formalnego – który może być nieskończony.

Najbardziej ogólnym modelem automatu jest Maszyna Turinga.



Automaty Moore'a i Mealy'ego

Podstawowymi modelami automatów rozważanymi przez nas są: automaty Moore'a i Mealy'ego.

$A = < \Sigma, \Delta, Q, \delta, \lambda, q_o >$, gdzie:

Σ – niepusty zbiór symboli wejściowych
Δ – niepusty zbiór symboli wyjściowych
Q – niepusty zbiór symboli stanów wewnętrznych
$\delta: Q \times \Sigma \rightarrow Q$ – funkcja przejść (przejścia)
$\lambda: Q \rightarrow \Delta$ – funkcja wyjść (wyjścia)
q_o – stan początkowy, $q_o \subset Q$

$\lambda : Q \times \Sigma \rightarrow \Delta$ – funkcja wyjść (wyjścia)

Przyjmując t jako chwilę bieżącą oraz t+1 jako chwilę następną:

$$q_{t+1} = \delta(q_t, z_t), \qquad z_t = \lambda(q_t)$$

Synteza abstrakcyjna i strukturalna

Synteza abstrakcyjna – od opisu do tablicy przejść i wyjść, czyli:

- sporządzenie tablicy przejść i wyjść,
- ewentualna minimalizacja tablicy.

Synteza strukturalna – od opisu do fizycznej realizacji układu cyfrowego, czyli:

- sporządzenie tablicy przejść i wyjść,
- ewentualna minimalizacja tablicy,
- kodowanie (np. stanów),
- wybór sekwencyjnych układów logicznych (przerzutników),
- określenie funkcji wzbudzeń wejść przerzutników,
- projekt połączeń realizujących funkcje wzbudzeń i funkcje wyjść,
- budowa układu fizycznego.

Synteza abstrakcyjna

Definicja 2

Synteza abstrakcyjna automatu to określenie opisu formalnego automatu, na podstawie którego można zbudować tabele przejść i wyjść automatu. Synteza sprowadza się do przejścia od algorytmu działania do grafu przejść automatu.

Etapy syntezy:

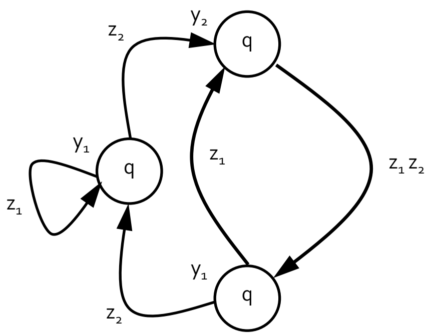
- opis działania automatu – algorytm słowny,
- przedstawienie algorytmu słownego w postaci wyrażeń regularnych bądź grafu automatu,
- określenie tablicy przejść i wyjść,
- określenie grafu przejść, jeśli znane jest wyrażenie regularne, a nie graf przejść.

Przykład 2 – synteza abstrakcyjna

Opis słowny

Automat jest w stanie początkowym. Stan początkowy sygnalizowany jest symbolem y1. Po pojawieniu się symbolu z1 pozostaje w tym stanie. Wprowadzenie symbolu z2 powoduje przejście do stanu następnego, który sygnalizowany jest symbolem y2. Automat pozostaje w tym stanie do momentu wprowadzenia symbolu z1. [...].

Graf automatu



$\Sigma = \{z_1, z_2\}$
$\Delta = \{y_1, y_2\}$
$Q = \{q_0, q_1, q_2\}$

$\delta(q_0, z_1) = q_0$
$\delta(q_0, z_2) = q_1$
$\delta(q_1, z_1) = q_2$
$\delta(q_1, z_2) = q_2$
$\delta(q_2, z_1) = q_1$
$\delta(q_2, z_2) = q_0$

$\lambda(q_0) = y_1$
$\lambda(q_1) = y_2$
$\lambda(q_2) = y_1$

Tablica przejść i wyjść

WY		y_1	y_1	y_2
stan		0	1	2
WE	z_1	0	2	1
	z_2	1	2	0

WY		y_1	y_1	y_2
stan		q_0	q_1	q_2
WE	z_1	q_0	q_2	q_1
	z_2	q_1	q_2	q_0

Przykład 2 (1/3)

- Opis słowny

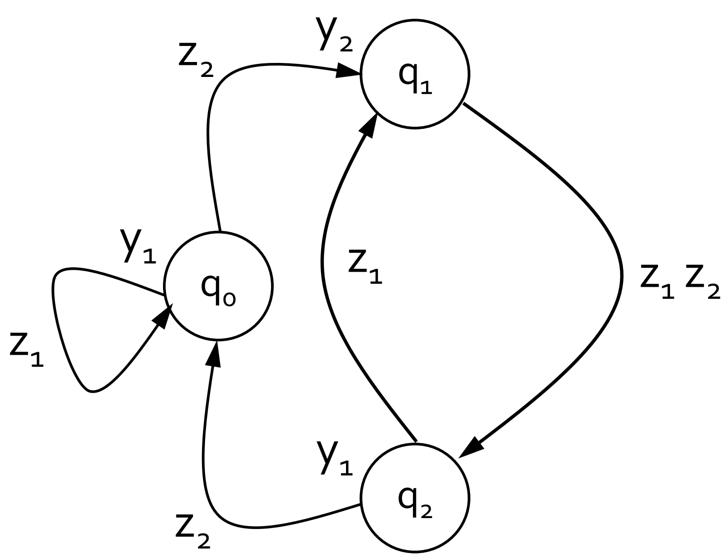
  Automat jest w stanie początkowym. Stan początkowy sygnalizowany jest symbolem y_1. Po pojawieniu się symbolu z_1 automat pozostaje w tym stanie. Wprowadzenie symbolu z_2 powoduje przejście do stanu następnego, który sygnalizowany jest symbolem y_2. Automat pozostaje w tym stanie do momentu wprowadzenia symbolu z_1. [...]

Przykład 2 (2/3)

Opis słowny

Automat jest w stanie początkowym. Stan początkowy sygnalizowany jest symbolem y1. Po pojawieniu się symbolu z1 pozostaje w tym stanie. Wprowadzenie symbolu z2 powoduje przejście do stanu następnego, który sygnalizowany jest symbolem y2. Automat pozostaje w tym stanie do momentu wprowadzenia symbolu z1. [...].

- Graf automatu



$\Sigma = \{z_1, z_2\}$
$\Delta = \{y_1, y_2\}$
$Q = \{q_0, q_1, q_2\}$

$\delta(q_0, z_1) = q_0$
$\delta(q_0, z_2) = q_1$
$\delta(q_1, z_1) = q_2$
$\delta(q_1, z_2) = q_2$
$\delta(q_2, z_1) = q_1$
$\delta(q_2, z_2) = q_0$

$\lambda(q_0) = y_1$
$\lambda(q_1) = y_2$
$\lambda(q_2) = y_1$

Przykład 2 (3/3)

$\Sigma = \{z_1, z_2\}$
$\Delta = \{y_1, y_2\}$
$Q = \{q_0, q_1, q_2\}$

$\delta(q_0, z_1) = q_0$
$\delta(q_0, z_2) = q_1$
$\delta(q_1, z_1) = q_2$
$\delta(q_1, z_2) = q_2$
$\delta(q_2, z_1) = q_1$
$\delta(q_2, z_2) = q_0$

$\lambda(q_0) = y_1$
$\lambda(q_1) = y_2$
$\lambda(q_2) = y_1$

- Tablica przejść i wyjść

WY		y_1	y_1	y_2
stan		0	1	2
WE	z_1	0	2	1
	z_2	1	2	0

WY		y_1	y_1	y_2
stan		q_0	q_1	q_2
WE	z_1	q_0	q_2	q_1
	z_2	q_1	q_2	q_0

Synteza strukturalna

Definicja 3

Synteza strukturalna automatu to stworzenie schematu układu elektronicznego odzwierciadlającego działanie automatu.

Etapy syntezy:

- kodowanie sygnałów, wejść i wyjść,
- budowa tablicy wzbudzeń przerzutników,
- określenie funkcji wzbudzeń przerzutników,
- określenie funkcji wyjścia,
- stworzenie schematu logicznego układu.

Przykład 3 - synteza strukturalna

Kodowanie sygnałów, wyjść i stanów

$\Sigma = \{z_1, z_2\}$ $\Delta = \{y_1, y_2\}$ $Q = \{q_0, q_1, q_2\}$

	Z
z_1	0
z_2	1

	Y
y_1	0
y_2	1

	Q_2	Q_1
q_0	0	0
q_1	0	1
q_2	1	0

Budowa tablicy wzbudzeń przerzutników

t			t+1									
Q_2	Q_1	Z	Q_2	Q_1	D_1	D_0	T_1	T_0	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	0	0	ø	0	ø
0	0	1	0	1	0	1	0	1	0	ø	1	ø
0	1	0	1	0	1	0	1	1	1	ø	ø	1
0	1	1	1	0	1	0	1	1	1	ø	ø	1
1	0	0	0	1	0	1	1	1	ø	1	1	ø
1	0	1	0	0	0	0	1	0	ø	1	0	ø

Minimalizacja funkcji

	T_1 / T_0	Q_2Q_1 00	01	11	10
Z	0	0/0	1/1	ø/ø	1/1
	1	0/1	1/1	ø/ø	1/0

$T_1 = Q_1 + Q_2$

$T_0 = /ZQ_1 + (Z \oplus Q_2)$

	J_1 / J_0	Q_2Q_1 00	01	11	10
Z	0	0/0	1/ø	ø/ø	ø/1
	1	0/1	1/ø	ø/ø	ø/0

$J_1 = Q_1$ $K_1 = 1$

$J_0 = (Z_1 \oplus Q_2)$ $K_0 = 1$

Przykład 3 (1/6)

Kodowanie stanów, sygnałów i wyjść

Kodowanie polega na określeniu trzech funkcji:
$f_{\Sigma} : \Sigma \rightarrow B^n$ $f_{\Delta} : \Delta \rightarrow B^m$ $f_{Q} : Q \rightarrow B^k$, gdzie B={0, 1}

Symbolom z_n, y_m i q_k, stosowanym w syntezie abstrakcyjnej, należy przypisać wektory binarne o odpowiedniej długości.

n bitów pozwala na zakodowanie 2^n symboli.
Aby zakodować *N* symboli za pomocą *n* bitów, musi zachodzić zależność $N \leq 2^n$.

Przykład 3 (2/6)

Kodowanie sygnałów, wyjść i stanów

Symbolom z_n, y_m i q_k, stosowanym w syntezie abstrakcyjnej, należy przypisać wektory binarne o odpowiedniej długości.

$\Sigma = \{z_1, z_2\}$

	Z
z_1	0
z_2	1

$\Delta = \{y_1, y_2\}$

	Y
y_1	0
y_2	1

$Q = \{q_0, q_1, q_2\}$

	Q_2	Q_1
q_0	0	0
q_1	0	1
q_2	1	0

Przykład 3 (3/6)

Budowa tablicy wzbudzeń przerzutników

Na podstawie przejść między stanami oraz tabel kodowania dla wejść i stanów można utworzyć zakodowaną tabelę przejść – przydatną przy tworzeniu tabeli wzbudzeń dla wejść przerzutników.

$\delta(q_0, z_1) = q_0$
$\delta(q_0, z_2) = q_1$
$\delta(q_1, z_1) = q_2$
$\delta(q_1, z_2) = q_2$
$\delta(q_2, z_1) = q_1$
$\delta(q_2, z_2) = q_0$

	Q_2	Q_1
q_0	0	0
q_1	0	1
q_2	1	0

t		t+1
q_i	z_i	q_i
q_0	z_1	q_0
q_0	z_2	q_1
q_1	z_1	q_2
q_1	z_2	q_2
q_2	z_1	q_1
q_2	z_2	q_0

t			t+1	
Q_2	Q_1	Z	Q_2	Q_1
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

Przykład 3 (4/6)

Budowa tablicy wzbudzeń przerzutników

Tabela wzbudzeń pokazuje jaki powinien być stan logiczny wejścia bądź wejść (przerzutnik JK) przerzutnika, aby uzyskać żądaną zmianę stanu wyjścia.

Q(t)	Q(t+1)	D(t)	T(t)	J(t)	K(t)
0	0	0	0	0	ø
0	1	1	1	1	ø
1	0	0	1	ø	1
1	1	1	0	ø	0

Przykład 3 (5/6)

Budowa tablicy wzbudzeń przerzutników

Na podstawie tabeli wzbudzeń oraz zakodowanej tabeli przejść można uzyskać tabelę wzbudzeń dla wejść przerzutników.

t			t+1									
Q_2	Q_1	Z	Q_2	Q_1	D_1	D_0	T_1	T_0	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	0	0	∅	0	∅
0	0	1	0	1	0	1	0	1	0	∅	1	∅
0	1	0	1	0	1	0	1	1	1	∅	∅	1
0	1	1	1	0	1	0	1	1	1	∅	∅	1
1	0	0	0	1	0	1	1	1	∅	1	1	∅
1	0	1	0	0	0	0	1	0	∅	1	0	∅

Przykład 3 (6/6)

Minimalizacja funkcji

t			t+1									
Q_2	Q_1	Z	Q_2	Q_1	D_1	D_0	T_1	T_0	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	0	0	ø	0	ø
0	0	1	0	1	0	1	0	1	0	ø	1	ø
0	1	0	1	0	1	0	1	1	1	ø	ø	1
0	1	1	1	0	1	0	1	1	1	ø	ø	1
1	0	0	0	1	0	1	1	1	ø	1	1	ø
1	0	1	0	0	0	0	1	0	ø	1	0	ø

Q_2Q_1

	T_1/T_0	00	01	11	10
Z	0	0/0	1/1	ø/ø	1/1
	1	0/1	1/1	ø/ø	1/0

Q_2Q_1

	J_1/J_0	00	01	11	10
Z	0	0/0	1/ø	ø/ø	ø/1
	1	0/1	1/ø	ø/ø	ø/0

$T_1 = Q_1 + Q_2$

$T_0 = /ZQ_1 + (Z \oplus Q_2)$

$J_1 = Q_1$

$J_0 = (Z_1 \oplus Q_2)$

$K_1 = 1$

$K_0 = 1$

Część III

Synteza automatów skończonych – wyrażenia regularne

Przypomnienie

Definicja 1 (przypomnienie)

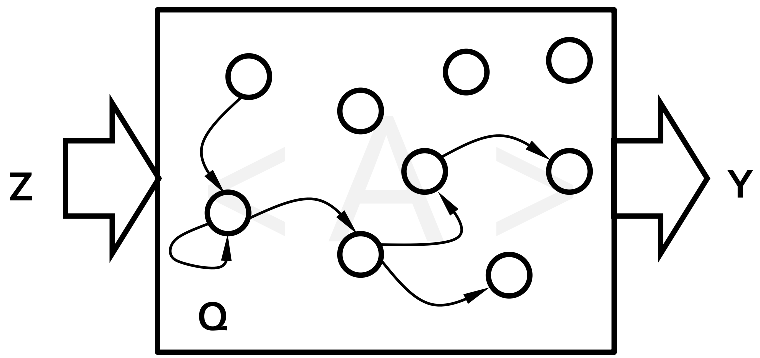
Automat skończony jest modelem matematycznym systemu dyskretnego, działającego w dyskretnych chwilach czasu. Jego działanie jest określone na skończonych zbiorach sygnałów wejściowych, stanów wewnętrznych i sygnałów wyjściowych.

Alfabet – skończony, niepusty zbiór symboli – A.

Słowo nad A to dowolny skończony ciąg elementów zbioru A

Język nad alfabetem A to dowolny podzbiór zbioru A^*. (A* - zbiór wszystkich słów nad A)

Synteza strukturalna



Z – alfabet wejściowy, Q – zbiór stanów wewnętrznych, Y – alfabet wyjściowy

Automat <A> akceptuje słowa należące do języka regularnego.

Język regularny jako zbiór słów jest reprezentowany przez wyrażenie regularne.

Synteza strukturalna

Alfabet

$$W = \{w_1, w_2, \ldots, w_i, \ldots, w_n\}$$

Słowa nad alfabetem W

$$w_1w_2w_3w_1 \quad w_4w_9w_1 \quad w_1w_3w_3w_3w_5 \quad \ldots$$

Zbiór wszystkich możliwych słów jest zbiorem nieskończonym W^*.

$$W^* = \{w_1w_2w_3w_1, w_4w_9w_1, w_1w_3w_3w_3w_5, \ldots\}$$

Na zbiorze W^* można określić rodzinę zbiorów S^*.

$$S^* = \{S_1, S_2, \ldots, S_i, \ldots, S_n\}$$

Na słowach $S_i \in S^*$ można wykonywać określone operacje: sumy, konkatenacji oraz iteracji.

Synteza strukturalna

Definicja 4

Wyrażeniem regularnym nad alfabetem Σ nazywamy ciąg znaków składający się z symboli: ø, ε, +, *, ·,), (oraz symboli a_i alfabetu Σ w następującej postaci:

1. ø, ε są wyrażeniami regularnymi,
2. wszystkie symbole $a_i \in \Sigma$ są wyrażeniami regularnymi,
3. jeśli e_1, e_2 są wyrażeniami regularnymi, to są nimi również:
   - e_1* (domknięcie *Kleene'ego*)
   - e_1e_2 (konkatenacja)
   - $e_1 + e_2$ (suma)
   - (e_1) (grupowanie)
4. wszystkie wyrażenia regularne są postaci opisanej w punktach 1 – 3.

Każde wyrażenie regularne definiuje pewien język formalny.
Każdy język definiowany przez wyrażenie regularne jest regularny.

Synteza strukturalna

Możemy wykonywać transformacje z wyrażenia regularnego:

- w zbiór słów,
- w graf przejść automatu akceptujący język reprezentowany przez to wyrażenie $r \to S(r)$,
- w gramatykę bezkontekstową regularną generującą słowa danego języka $S(r)$.

Przykład 4 – synteza strukturalna

akceptacja

$S_1 = z_1 z_1 z_2 + z_2 z_1 \quad | \; y_1$

$S_2 = z_1 z_2 + z_2 z_2 \quad | \; y_2$

$S_3 = /(S_1 + S_2) \quad | \; y_0 = \varepsilon$

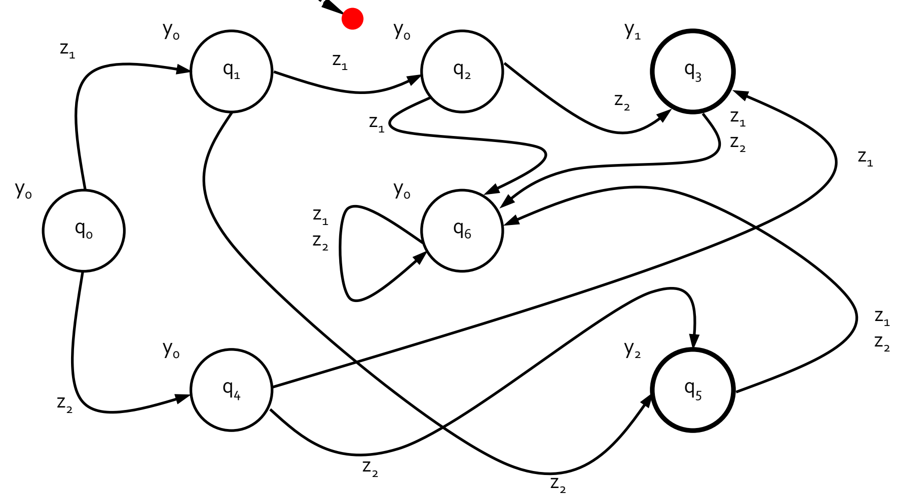
$S_2 = | \; z_1 \; | \; z_1 \; | \; + \; | \; z_2 \; | \; z_1 \; |$

0 1 6 0 4 7

$S_1 = | \; z_1 \; | \; z_1 \; | \; z_2 \; | \; + \; | \; z_2 \; | \; z_1 \; |$

0 1 2 3 0 4 5

	WY	y_0	y_0	y_0	y_1	y_0	y_2	y_0
	stan	0	1	2	3	4	5	6
WE	z_1	1	2	*	*	3	*	*
	z_2	4	5	3	*	5	*	*



Przykład 4 (1/4)

W celu określenia stanów automatu wprowadza się pojęcie „miejsca” w wyrażeniu regularnym.

Miejscem jest położenie między literami, między literą a znakiem dysjunkcji (lub / or) oraz początek i koniec wyrażenia. Miejscom przyporządkowuje się stany automatu.

- $S_1 = z_1 z_1 z_2 + z_2 z_1 \quad | \; y_1$

  $S_2 = z_1 z_2 + z_2 z_2 \quad | \; y_2$

  $S_3 = /(S_1 + S_2) \quad | \; y_0 = \varepsilon$

- $S_1 =$

|	z_1	|	z_1	|	z_2	|	+	|	z_2	|	z_1	|
0		1		2		3		0		4		5

  $S_2 =$

|	z_1	|	z_2	|	+	|	z_2	|	z_2	|
0		1		6		0		4		7

Miejsce „podstawowe” – miejsce, na lewo od którego stoi litera oraz miejsce początkowe.

Miejsce „przedpodstawowe” – miejsce, od którego na prawo stoi litera.

Przykład 4 (2/4)

$S_1 = |\ z_1\ |\ z_1\ |\ z_2\ |\ +\ |\ z_2\ |\ z_1\ |$

0 1 2 3 0 4 5

3 → y_1, 5 → y_1

$S_2 = |\ z_1\ |\ z_2\ |\ +\ |\ z_2\ |\ z_2\ |$

0 1 6 0 4 7

6 → y_2, 7 → y_2

- Tworzymy tablicę przejść i wyjść.

WY	y_0	y_0	y_0	y_1	y_0	y_1	y_2	y_2	y_0
stan	0	1	2	3	4	5	6	7	$\cancel{*}$
WE z_1	1	2	*	*	5	*	*	*	*
WE z_2	4	6	3	*	7	*	*	*	*

$\underbrace{3 \quad 4 \quad 5}_{3=5}$ $\underbrace{6 \quad 7}_{6=7}$

Przykład 4 (3/4)

WY		y_0	y_0	y_0	y_1	y_0	y_1	y_2	y_2	y_0
stan		0	1	2	3	4	5	6	7	8
WE	z_1	1	2	*	*	5	*	*	*	*
	z_2	4	6	3	*	7	*	*	*	*

$3 \equiv 5 \rightarrow 3$

$6 \equiv 7 \rightarrow 5$

- Przeprowadzamy redukcją (minimalizację) stanów równoważnych

WY		y_0	y_0	y_0	y_1	y_0	y_2	y_0
stan		q_0	q_1	q_2	q_3	q_4	q_5	q_6
WE	z_1	q_1	q_2	q_6	q_6	q_3	q_6	q_6
	z_2	q_4	q_5	q_3	q_6	q_5	q_6	q_6

Przykład 4 (4/4)

WY		y_0	y_0	y_0	$\mathbf{y_1}$	y_0	$\mathbf{y_2}$	y_0
stan		q_0	q_1	q_2	$\mathbf{q_3}$	q_4	$\mathbf{q_5}$	q_6
WE	z_1	q_1	q_2	q_6	q_6	q_3	q_6	q_6
	z_2	q_4	q_5	q_3	q_6	q_5	q_6	q_6

- Graficzna reprezentacja tabeli przejść i wyjść