

Ćwiczenia laboratoryjne z Logiki Układów Cyfrowych

ćwiczenie 208

Temat: Komputerowa realizacja automatów skończonych

1. Cel ćwiczenia

Celem ćwiczenia jest praktyczne zapoznanie się ze sposobem programowej realizacji na komputerze automatów skończonych (na przykładzie automatów typu Moore'a).

2. Program ćwiczenia

1. Zapoznanie się z metodą transformacji formalnego modelu automatu skończonego na model odpowiedni do realizacji programowej.
2. Zapoznanie się z programem działającym na elementach modelu formalnego automatu.
3. Wprowadzenie danych reprezentujących w pamięci komputera model formalny automatu.
4. Testowanie programowego wariantu automatu skończonego.
5. Opracowanie wyników testowania automatu.

3. Wiadomości podstawowe

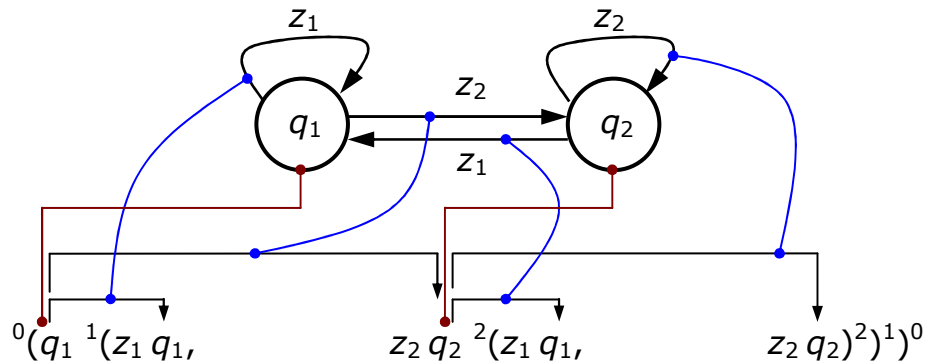
Automat skończony jest modelem formalnym (matematycznym) systemu lub procesu przebiegającego w dyskretnych chwilach czasu [2]. Jest on więc pewną abstrakcją, którą można jednak urealnić budując odpowiedni układ funkcjonalny o zachowaniu zgodnym z reprezentującym go modelem matematycznym. Automat skończony można zrealizować jako układ elektroniczny (hardware) lub jako program komputerowy (software). Na przykład każdy układ sekwencyjny w strukturze fizycznej komputera jak też każdy program składowy systemu operacyjnego komputera można rozpatrywać jako pewien automat skończony. Odzwierciedla to dualizm między strukturą fizyczną komputera (hardware) i jego oprogramowaniem (software).

Działanie automatu skończonego jest jednoznacznie określone przez funkcję przejść, która może być reprezentowana w postaci grafu (ang. *transition diagram*). Dla prostych problemów graf może być stworzony intuicyjnie, na podstawie słownego opisu zadania, które powinno być realizowane przez automat. Zastosowanie języka wyrażeń regularnych umożliwia przeprowadzenie tego procesu w sposób formalny. Przy programowej realizacji automatu jego graf musi być reprezentowany w pamięci komputera, powinien więc zostać przekształcony na odpowiednią postać symboliczną.

3.1. Reprezentacja grafów za pomocą wyrażeń symbolicznych

Jednym z wielu sposobów opisu grafów są wyrażenia symboliczne. Elementami składowymi wyrażenia symbolicznego są oznaczenia wierzchołków i łuków grafu oraz specjalne symbole organizujące wyrażenie, czyli indeksowane pary nawiasów w postaci $^{n}(\dots)^n$ oraz przecinek jako symbol rozdzielający. Idea zapisu grafu w postaci wyrażenia

symbolicznego zostanie zilustrowana na przykładzie prostego grafu. Powiązanie elementów grafu i wyrażenia symbolicznego pokazane jest na rys. 1.



Rys. 1. Transformacja grafu na wyrażenie symboliczne

Efektom końcowym transformacji grafu jest następujące wyrażenie:

$$G^+ = {}^0(q_1^1(z_1 q_1, z_2 q_2^2(z_1 q_1, z_2 q_2)^2)^1)^0 \quad (1)$$

Podstawowe zasady tworzenia wyrażenia symbolicznego są następujące:

- całe wyrażenie zamknięte jest w parze nawiasów ${}^0(\dots)^0$,
- wszystkie elementy wyrażenia zawarte są w parach nawiasów typu ${}^n(\dots)^n$, mogą być one zagnieżdżone, z rosnącymi indeksami ${}^n(\dots^{n+1}(\dots)^{n+1}\dots)^n$, dopuszczalna jest postać typu ${}^n(\dots^{n+1}(\dots)^{n+1}, \dots^{n+1}(\dots)^{n+1}, \dots)^n$
- przed każdym nawiasem otwierającym, z wyjątkiem ${}^0($, stoi symbol q_i będący oznaczeniem wierzchołka grafu,
- term postaci $q_i {}^n(z_m q_j$ reprezentuje łuk z_m , prowadzący od wierzchołka początkowego q_i do wierzchołka końcowego q_j , łuki prowadzące z q_i do innych wierzchołków zapisywane są po przecinku, na przykład $q_i {}^n(z_m q_j, z_n q_k$ dla łuku z_n prowadzącego do q_k ,
- wszystkie łuki wychodzące z wierzchołka q_i muszą być opisane wewnątrz jednej pary nawiasów $q_i {}^n(\dots)^n$, oznacza to, że każdy symbol q_i może pojawić się przed nawiasem otwierającym tylko raz,
- wierzchołek końcowy w termie postaci $z_m q_j$ może otwierać kolejny poziom zagnieżdżenia nawiasów czyli $q_i {}^n(\dots z_m q_j {}^{n+1}(z_n q_k$ pod warunkiem, że symbol q_j nie wystąpił już wcześniej przed nawiasem otwierającym.

Wyrażenie symboliczne dla danego grafu może być zapisane na wiele sposobów. Poniżej pokazana jest alternatywna postać wyrażenia dla przykładowego grafu. Ma ona mniejszy stopień zagnieżdżenia ale jest mniej optymalna, jeśli chodzi o liczbę użytych symboli:

$$G^+ = {}^0(q_1^1(z_1 q_1, z_2 q_2)^1, q_2^1(z_1 q_1, z_2 q_2)^1)^0 \quad (2)$$

Zwykle przyjmuje się zasadę, że przejścia z wierzchołków pojawiających się w kolejnych termach $z_m q_j$ są rozwijane w głąb dopóki jest to możliwe, to znaczy symbol q_j nie pojawił się już wcześniej przed nawiasem otwierającym.

Warto zwrócić uwagę, że aby odnaleźć wszystkie łuki wychodzące z wierzchołka q_i wystarczy ustalić miejsce wystąpienia symbolu q_i przed nawiasem otwierającym ${}^n($ a następnie wypisać wszystkie termy $z_m q_j$ zawarte wewnątrz pary nawiasów ${}^n(\dots)^n$, pomijając przy tym zawartość wszystkich ewentualnych nawiasów zagnieżdżonych.

3.2. Zapis symboliczny grafu automatu

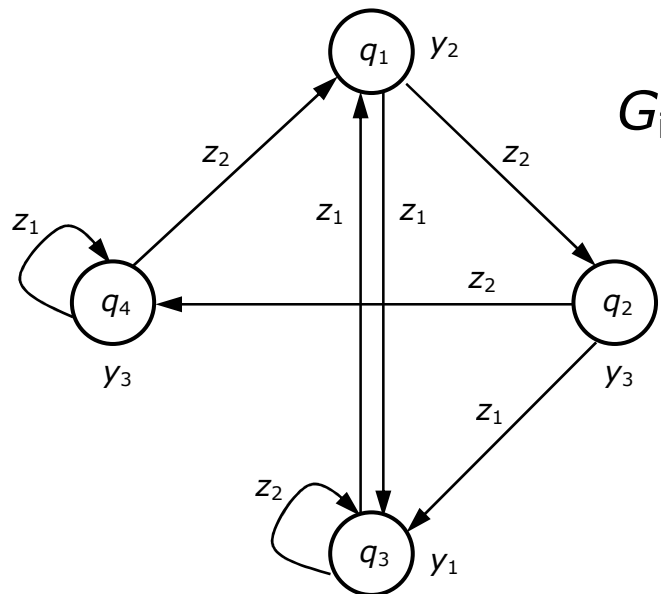
Punktem wyjścia do programowej realizacji automatu Moore'a jest transformacja grafu automatu na wyrażenie symboliczne. Załóżmy, że zadany jest automat Moore'a reprezentowany "szóstką":

$$\langle Z, Q, Y, \Phi, \Psi, q_s \rangle$$

gdzie:

Z - alfabet wejściowy	$Z = \{ z_1, z_2 \}$
Y - alfabet wyjściowy	$Y = \{ y_1, y_2, y_3 \}$
Q - zbiór stanów wewnętrznych	$Q = \{ q_1, q_2, q_3, q_4 \}$
Φ - funkcja przejść	$q(t+1) = \Phi(q(t), z(t))$
Ψ - funkcja wyjść	$y(t) = \Psi(q(t))$
q_s - stan początkowy automatu.	$q_s = q_2$

Przyjmujemy, że działanie tego automatu określone jest przez graf przedstawiony na rys. 2. Graf ten, oznaczony symbolem G_i , jest transformowany na wyrażenie symboliczne G_i^+ (według zasad opisanych w punkcie 3.1).



Rys. 2. Przykładowy graf automatu Moore'a

Wyrażenie G_i^+ opisujące graf G_i z rys. 2 ma postać:

$$G_i^+ = {}^0(q_2^1(z_1q_3^2(z_1q_1^3(z_2q_2, z_1q_3)^3, z_2q_3)^2, z_2q_4^2(z_2q_1, z_1q_4)^2)^1)^0 \quad (3)$$

W przedstawionym wyrażeniu symbol q_r stojący przed nawiasem otwierającym k (oznacza wierzchołek grafu, natomiast term typu $q_r^k(z_j q_s$ reprezentuje krawędź wychodzącą z wierzchołka q_r i prowadzącą do wierzchołka q_s).

Wyrażenie G_i^+ (1) nie reprezentuje w pełni grafu G_i , ponieważ nie uwzględnia przyporządkowania wierzchołkom q_r sygnałów wyjściowych y_i . Stąd też, aby uzyskać

kompletną reprezentację grafu G_i , wyrażeniu G_i^+ należy przyporządkować tabelę \bar{Q}_i ($q_r \leftrightarrow y_j$) o następującej postaci:

Tabela \bar{Q}_i

q_r	y_j
q_1	y_2
q_2	y_3
q_3	y_1
q_4	y_3

W celu zapisania symbolicznej reprezentacji grafu G_i w pamięci komputera, wyrażenie G_i^+ jest przekształcane tak, aby za dowolnym termem typu q_r^k (w wyrażeniu G_i^+ znalazły się termy typu $z_j q_s$ reprezentujące wszystkie krawędzie wychodzące z danego wierzchołka q_r). Produktem tego przekształcenia jest wyrażenie symboliczne G_i^{++} o postaci:

$$G_i^{++} = {}^0(q_2^1(z_2 q_4, z_1 q_3^2(z_2 q_3, z_1 q_1^3(z_2 q_2, z_1 q_3)^3)^2)^1, q_4^1(z_2 q_1, z_1 q_4)^1)^0 \quad (4)$$

Aby zoptymalizować operacje na wyrażeniu G_i^{++} w tabeli \bar{Q}_i dodana zostanie kolumna zawierająca adresy termów q_r^k (w wyrażeniu G_i^{++}). Interpretacja tych adresów zależy od szczegółów implementacji. Tabela \bar{Q}_i po rozszerzeniu otrzymuje nazwę \tilde{Q}_i i jest przedstawiona poniżej:

Tabela \tilde{Q}_i

q_r	adres q_i^k	y_j
q_1	x_3	y_2
q_2	x_1	y_3
q_3	x_2	y_1
q_4	x_4	y_3

Wyrażenie symboliczne G_i^{++} (4) i tabela \tilde{Q}_i są głównymi elementami programowego wariantu automatu w pamięci komputera.

3.3. Programowy wariant automatu Moore'a

Automat Moore'a reprezentowany jest w pamięci przez wyrażenie symboliczne G_i^{++} (4) i tabelę \tilde{Q}_i . Są to elementy statyczne, na których operuje odpowiedni program. Do pamiętania bieżących wartości sygnałów wejściowych, wyjściowych oraz stanu automatu służą następujące elementy:

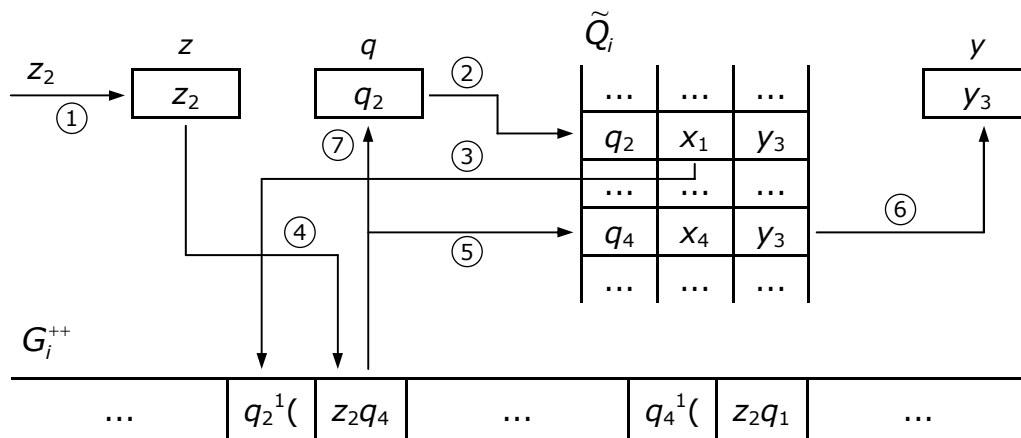
- słowo "z" zawierające aktualny sygnał wejściowy $z_j \in Z$,
- słowo "q" zawierające aktualny stan wewnętrzny $q_r \in Q$,
- słowo "y" zawierające aktualny sygnał wyjściowy $y_i \in Y$.

Konfiguracja elementów programowego wariantu automatu Moore'a i podstawowe operacje wykonywane na tych elementach przedstawione są na rys. 3.

Załóżmy, że w słowie "q" ustawiony jest stan początkowy automatu q_2 . Podany na

wejście symbol z_2 zapamiętywany jest w słowie "z". Operacja ta oznaczona jest na rys. 3 symbolem ①. W kolejnej operacji, oznaczonej jako ②, symbol q_2 jest szukany w tabeli \tilde{Q}_i . W wierszu q_2 tabeli \tilde{Q}_i znajduje się adres x_1 wierzchołka q_2 w wyrażeniu G_i^{++} . Na podstawie adresu x_1 , w operacji ③, w wyrażeniu G_i^{++} odszukany zostanie wierzchołek q_2 , stojący przed nawiasem otwierającym q_2^1 . W operacji ④ przeszukiwana jest zawartość pary nawiasów $^1(\dots)^1$ aby odnaleźć term zawierający symbol z_2 . Wynikiem operacji ④ jest term $z_2 q_4$, w którym symbol q_4 oznacza nowy stan automatu, dla którego powinien być wygenerowany odpowiedni sygnał wyjściowy. Stąd też w operacji ⑤ w tabeli \tilde{Q}_i szukany jest wiersz oznaczony symbolem q_4 . Znaleziony wiersz zawiera symbol y_3 , który w operacji ⑥ zapisywany jest w słowie "y". W operacji ⑦ uaktualniany jest stan bieżący automatu przez zapisanie q_4 w słowie "q".

Po wykonaniu opisanych powyżej operacji dla jednego przejścia następuje aktualizacja danych przekazywanych przez interfejs wyjściowy programu. Operacje dla kolejnych przejść wykonywane są w analogiczny sposób.



Rys. 3. Elementy programowego wariantu automatu Moore'a

4. Opis programu umożliwiającego realizację automatu skończonego

4.1. Ogólna koncepcja programu

W poprzednim punkcie przedstawiono ogólne zasady programowej realizacji automatów skończonych. Obecnie zostanie omówiony sposób implementacji programowego wariantu automatu typu Moore'a na komputerze PC.

Z wcześniejszych rozważań wynika, że program reprezentujący swoim działaniem automat skończony powinien realizować trzy podstawowe funkcje:

- wprowadzanie danych określających strukturę automatu i wstępne sprawdzenie ich poprawności,
- symulację działania automatu (według zasad opisanych w punkcie 3.3),
- graficzną prezentację działania automatu na ekranie komputera.

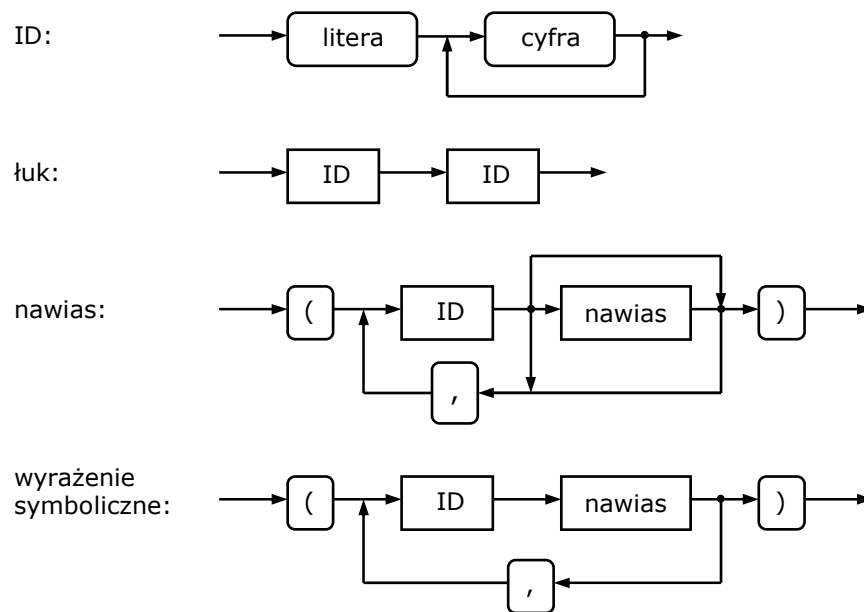
Poniżej zostanie przedstawiony sposób realizacji tych funkcji z zaznaczeniem różnic w stosunku do wcześniej omówionych założeń.

4.1.1. Wprowadzanie struktury automatu do pamięci komputera

Struktura automatu skończonego reprezentowana jest w pamięci komputera w postaci wyrażenia symbolicznego. W omawianej implementacji wyrażenie symboliczne może być wprowadzane zarówno w wersji G_i^+ (3) lub wersji G_i^{++} (4). Nie podaje się natomiast indeksów nawiasów (są one automatycznie dołączane przez procedurę czytającą dane wejściowe). Oprócz wyrażenia symbolicznego podaje się także zbiór symboli wejściowych Z , zbiór stanów wewnętrznych Q , zbiór symboli wyjściowych Y oraz funkcję wyjść reprezentowaną przez tabelę \bar{Q}_i ($q_r \leftrightarrow y_j$).

Po wprowadzeniu wyrażenia symbolicznego poddawane jest ono analizie (sprawdzana jest poprawność jego składni). W tym celu wykorzystywane są diagramy składni wyrażenia symbolicznego przedstawione na rys. 4. Zastosowany algorytm analizy składniowej charakteryzują dwie cechy:

- określenie kolejnego kroku analizy zależy tylko od obecnego stanu oraz od pojedynczego, aktualnie wczytywanego symbolu,
- żadnego z kolejnych kroków analizy nie można cofnąć. Jest to tzw. analiza bez powrotów z wyprzedzeniem o jeden symbol (ang. *one-symbol-lookahead without backtracking*) [5].



Rys. 4. Diagramy składni wyrażenia symbolicznego

W przypadku stwierdzenia poprawności wyrażenia symbolicznego umieszczane jest ono w pamięci, w przeciwnym przypadku sygnalizowany jest błąd. Ostatecznie w pamięci zapisane jest wyrażenie symboliczne G_i^{++} oraz tabela \tilde{Q}_i .

Wraz z analizą składni wyrażenia symbolicznego tworzona jest lista łuków grafu automatu. Jest ona wykorzystywana do graficznej prezentacji grafu automatu na ekranie komputera i dokładniej zostanie opisana później.

4.1.2. Symulacja działania automatu

Procedura symulująca działanie automatu skończonego oparta jest na algorytmie przedstawionym w punkcie 3.3. Została ona rozszerzona o podprogram umożliwiający

wprowadzanie z klawiatury sekwencji symboli wejściowych. Po wczytaniu symbolu wejściowego wykonywane są kolejne operacje zgodnie ze schematem przedstawionym na rys. 3. W ich wyniku otrzymujemy nowy stan automatu q_r oraz odpowiadający mu symbol wyjściowy y_j . W trakcie symulacji działania automatu na ekranie wyświetlany jest graf automatu. Wierzchołek grafu odpowiadający aktualnemu stanowi automatu jest odpowiednio oznaczony. Wraz z analizą kolejnych symboli wejściowych następuje uaktualnienie obrazu grafu na ekranie komputera. Sposób uzyskania grafu automatu na ekranie komputera zostanie przedstawiony w następnym punkcie.

4.1.3. Graficzne przedstawienie grafu automatu na ekranie komputera

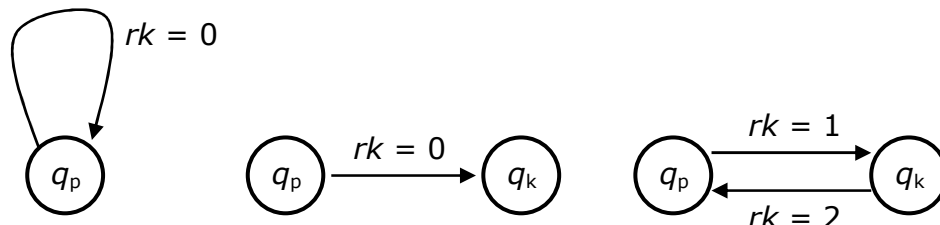
Jednym ze sposobów opisu działania automatu skończonego jest podanie jego grafu przejść. Graf automatu reprezentowany jest wówczas przez zbiór wierzchołków V odpowiadający stanom wewnętrznym automatu oraz przez zbiór krawędzi E określający możliwe przejścia między stanami automatu. W celu uzyskania na ekranie odpowiadającego badanemu automatu grafu przejść należy określić, na podstawie wyrażenia symbolicznego, zbiór krawędzi tego grafu. Krawędź opisywana jest następującymi parametrami:

$$k = (q_p, q_k, z_j, rk)$$

gdzie:

- q_p - stan początkowy (aktualny),
- q_k - stan końcowy (następny),
- z_j - sygnał wejściowy powodujący przejście automatu ze q_p do stanu q_k ,
- rk - rodzaj krawędzi.

Lista krawędzi tworzona jest równoległe podczas analizy wyrażenia symbolicznego (patrz punkt 4.1.1). Otrzymujemy wówczas zbiór krawędzi, dla których określone są q_p , q_k oraz z_j . Następnie zostaje określony rodzaj każdej krawędzi zgodnie z oznaczeniami przedstawionymi na rys. 5.



Rys. 5. Rodzaje krawędzi występujące w grafie automatu

Dysponując opisanym zbiorem krawędzi można narysować graf automatu. Odpowiednia procedura składa się z trzech części:

- rysowanie wierzchołków (stan aktualny jest wyróżniony),
- rysowanie łuków (krawędzi) odpowiadających przejściom między stanami,
- opisanie krawędzi grafu odpowiednimi symbolami wejściowymi z_j .

4.2. Opis działania programu

Menu główne programu umożliwia wybranie jednej z następujących opcji:

- DANE
- EDYCJA
- SYMULACJA

Wybór żądanej opcji następuje przez wprowadzenie z klawiatury wyróżnionego znaku dla odpowiedniego pola menu lub za pomocą klawiszy kierunkowych (strzałek) i klawisza <ENTER>. Rezygnacja z wybranej opcji następuje w wyniku naciśnięcia klawisza <ESC>.

4.2.1. Opcja DANE

Wybór tej opcji umożliwia realizację jednej z następujących funkcji:

- wczytanie danych z dysku,
- zapisanie danych na dysk,
- inicjowanie wprowadzania danych,
- przeglądanie katalogu,
- zmiana katalogu,
- wyjście z programu.

Do danych określających strukturę automatu należą:

- zbiór symboli wejściowych Z ,
- zbiór stanów wewnętrznych Q ,
- zbiór symboli wyjściowych Y ,
- funkcja przejść (w postaci wyrażenia symbolicznego),
- funkcja wyjść (w postaci tabeli).

Po wczytaniu danych są one wyświetlane na ekranie komputera wraz z odpowiadającym im grafem automatu.

4.2.2. Opcja EDYCJA

Opcja ta umożliwia wprowadzanie lub modyfikację danych określających strukturę automatu. Przy wprowadzaniu danych należy pamiętać, że:

- identyfikator określający symbol wejściowy, stan wewnętrzny lub symbol wyjściowy może składać się z pojedynczej litery i odpowiedniego indeksu np. $z1, q5, y3$,
- poszczególne elementy wprowadzanych danych oddzielane są przecinkiem np. $Z = \{ z1, z2, z3 \}$,
- przy wprowadzaniu wyrażenia symbolicznego nie podaje się indeksów nawiasów (są one dodawane automatycznie przez program).

W przypadku wprowadzenia błędnych danych sygnalizowany jest rodzaj błędu oraz miejsce jego wystąpienia. Przy poprawnym zakończeniu edycji na ekranie zostają wyświetlone aktualne dane oraz odpowiadający im graf automatu.

4.2.3. Opcja SYMULACJA

Opcja SYMULACJA umożliwia testowanie automatu, którego struktura została wczytana ze zbioru dyskowego lub wprowadzona za pomocą opcji EDYCJA. Po jej wyborze istnieje możliwość realizacji jednej z następujących funkcji:

- zmiana aktualnego stanu wewnętrznego automatu,
- wprowadzenie słowa wejściowego (ciągu symboli wejściowych),
- praca ciągła automatu,
- praca krokowa automatu.

Po określeniu słowa wejściowego można testować automat na dwa sposoby:

- automat generuje na podstawie słowa wejściowego słowo wyjściowe (pracy automatu nie można wówczas przerwać),

- praca automatu śledzona jest krok po kroku (po każdym symbolu wejściowym praca automatu jest wstrzymywana).

Na ekranie można obserwować kolejne zmiany stanu automatu, a ponadto wyświetlana jest dokumentacja jego działania zawierająca listę kolejnych stanów wewnętrznych oraz generowanych symboli wyjściowych.

W trakcie testowania automatu może powstać sytuacja, że dane przejście w automacie jest nieokreślone (w przypadku automatów niezupełnych). Następuje wówczas sygnalizacja błędu oraz wstrzymywana jest dalsza praca automatu.

Podczas testowania automatu istnieje również możliwość zmiany jego aktualnego stanu bez wprowadzania symbolu wejściowego.

5. Przebieg ćwiczenia

Dla problemów zadanych przez prowadzącego, w postaci opisu słownego, przygotować i przetestować automaty Moore'a:

- określić graf automatu Moore'a odzwierciedlający zadany problem (sprawdzić czy nie zawiera on stanów równoważnych),
- zapisać graf automatu w postaci wyrażenia symbolicznego G_i^{++} i tabeli wyjść \bar{Q}_i ,
- wprowadzić dane do programu i przetestować komputerowy model automatu podając odpowiednie ciągi symboli wejściowych,
- zaprojektować ciąg testowy o jak najmniejszej długości, który umożliwiłby pełne przetestowanie automatu (sprawdzenie wszystkich przejść). W przypadku niektórych automatów może być potrzebny więcej niż jeden taki ciąg.

6. Sprawozdanie z ćwiczenia

- umieścić temat i cel ćwiczenia,
- umieścić grafy automatów dla problemów podanych przez prowadzącego ćwiczenie,
- przedstawić wyrażenia symboliczne reprezentujące grafy automatów,
- umieścić wyniki testowania programowych wariantów automatów,
- podać krótkie wnioski z ćwiczenia.

7. Literatura

- [1]. Bromirski J.: Teoria automatów, WNT, Warszawa 1972
- [2]. Hopcroft J. E., Ullman J.D.: Introduction to Automata Theory, Languages and Computation, Addison-Wesley Publishing Company, Reading, Massachusetts 1979
- [3]. Kazimierczak J.: System Cybernetyczny, Wyd. Wiedza Powszechna (seria "Omega"), Warszawa 1978
- [4]. Kazimierczak J., Kluska J., Kaczmarek A., Podstawy teorii automatów - Laboratorium, Wyd. Politechniki Rzeszowskiej, Rzeszów 1984
- [5]. Wirth N.: Algorytmy + struktury danych = programy, WNT. Warszawa 1989