Automat z parametrem wewnętrznym jako sterownik robota mobilnego

Część 2 – Implementacja w środowisku programowym Borland Delphi

prof. dr hab. inż. Jan Kazimierczak,
dr inż. Barbara Łysakowska
Instytut Cybernetyki Technicznej
Politechniki Wrocławskiej

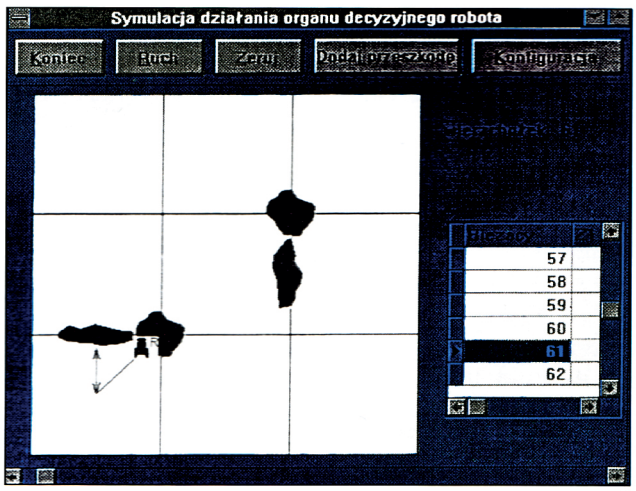
W PAR nr 4/98 zamieściliśmy 1 część artykułu, w której zaprezentowano model formalny dyskretnego sterowania robotem mobilnym poruszającym się na płaszczyźnie z losowo rozmieszczonymi przeszkodami. Synteza modelu została ukierunkowana na zastosowanie automatu z parametrem wewnętrznym, którego strukturę logiczną można przedstawić w postaci wyrażeń symbolicznych. W części 2 artykułu pokazano opis implementacji komputerowej symulacji działania sterownika robota mobilnego, będącego automatem z parametrem wewnętrznym, w środowisku programowym Delphi firmy Borland.

Synteza formalna automatu **<A'>** z parametrem wewnętrznym jako sterownika robota mobilnego, reprezentująca zarówno część operacyjną jak też część adresową, pozwoliła w części I artykułu utworzyć jego kompletną strukturę logiczną. Według niej można dokonać realizaji hardware'owej automatu, co będzie przedmiotem dalszych badań, jak również prezentowanej implementacji komputerowej w postaci programu symulującego działanie sterownika robota mobilnego poruszającego się w nieznanym terenie z przeszkodami. Interpretując proces sterowania ruchem robota jako wielochodową grę dwuosobową z naturą, można ją w sposób formalny opisać za pomocą jednostronnie zoptymalizowanego drzewa (cz. 1, rys. 2), przekształcanego w diagram stanów przejścia automatu Moore'a. Na podstawie utworzonego grafu G tego drzewa (cz. 1, rys. 3) można utworzyć wyrażenie symboliczne G'^{+} ([2], cz. 1) obrazujące strukturę automatu Moore'a, a następnie automatu **<A'>** z parametrem wewnętrzym. Synteza automatu **<A'>**, pełniącego rolę sterownika robota mobilnego, została zaprezentowana w części 1 artykułu. Do celów komputerowej symulacji działania automatu wykorzystuje się zapis struktury drzewa gry w postaci wyrażeń symbolicznych. Jedną z najważniejszych zalet wyrażeń symbolicznych jest prostota ich implementacji w pamięci komputera. Jest to istotne, ponieważ graf G można interpretować jako graf programu sterującego robotem, w którym pary (y_s, x_i) oznaczają odpowiednie procedury wykonywane pod wpływem określonych zdarzeń, symbole q_r – pozycję odpowiadających im procedur w programie, zaś z_j – rezultat wykonania procedury. W niniejszej pracy prezentowana jest programowa realizacja drzewa grafu G (cz. 1, rys. 3). Na podstawie wyrażeń symbolicznych G'^{+}, E'^{+}, F'^{+} oraz D'^{+}, reprezentujących część operacyjną (G'^{+}, E'^{+}, F'^{+}) oraz adresową (D'^{+}) struktury automatu **<A'>** z parametrem wewnętrznym, będącego sterownikiem robota mobilnego (cz. 1, rys. 4), można dokonać symulacji komputerowej jego działania. W efekcie robot będzie się poruszał w sposób bezkolizyjny w terenie z losowo rozmieszczonymi przeszkodami.

Założenia projektowe programu symulującego działanie sterownika

Celem implementacji komputerowej jest utworzenie programu zdolnego do sekwencyjnego analizowania wyrażeń symbolicznych G'^{+}, E'^{+}, F'^{+} oraz D'^{+}, tworzących strukturę sterownika robota mobilnego i generowania, na podstawie rozpoznania konfiguracji płaszczyzny roboczej, odpowiednich sygnałów wyjściowych. Aby możliwa była pełna symulacja sterowania, konieczne jest uwzględnienie różnych konfiguracji obszaru roboczego penetrowanego przez robota, a więc dowolnego rozmieszczenia na nim przeszkód [10]. Użytkownik musi mieć możliwość sytuowania przeszkód na modelu badanej powierzchni roboczej, stąd bardzo istotne stało się czytelne zobrazowanie procesu podejmowania decyzji dotyczących strategii ruchu robota. W związku z tym wszystkie elementy statyczne związane z obszarem roboczym, a także sam ruch robota, reprezentowane są w postaci graficznej na ekranie monitora (rys. 1). W procesie symulacji działania robota mobilnego na ekranie komputera widoczna jest „szachownica" 3×3 powierzchni roboczej, symbolizująca płaszczyznę, po której porusza się robot wyobrażony za pomocą ikony, tor jaki on pokonuje, znaczony strzałkami opisującymi poszczególne ruchy oraz kontury przeszkód (w formie bitmapy), które można nanosić na obraz powierzchni roboczej za pomocą specjalnej opcji.

W grafie G^{+} zostały uwzględnione tylko niektóre konfiguracje przeszkód (osiem rozmieszczeń pokazanych na rys. 1, cz. 1). W razie wystąpienia innej konfiguracji roz-



Rys. 1. Okno robocze programu z elementami graficznej reprezentacji mapy terenu i robota mobilnego

mieszczenia przeszkód na powierzchni roboczej automat <A'> znajdzie się kiedyś w stanie, w którym wygenerowany zostanie sygnał y_{18} oznaczający sytuację, w której nie są znane reguły dalszego postępowania. W przypadku realizacji programowej automatu <A'> możliwe jest proste dodawanie nowych reguł (wierzchołków grafu G^+) określających zachowanie się robota w sytuacji uprzednio nie rozpoznanej. Wykorzystując łatwość modyfikacji struktury grafu, przechowywanej w pamięci komputera w postaci wyrażeń symbolicznych, w momencie znalezienia się automatu <A'> w sytuacji opisanej wierzchołkiem y_{18}, można dodać do uprzedniej struktury nowy wierzchołek wraz z przypisanymi mu nowymi sygnałami (parametrami wewnętrznymi), czyli rozwinąć zapis odpowiedniego wyrażenia symbolicznego. Jest to swego rodzaju metoda „uczenia" automatu (z parametrem wewnętrzym) zachowania w sytuacji pojawienia się nowych przeszkód na płaszczyźnie roboczej [8, 10].

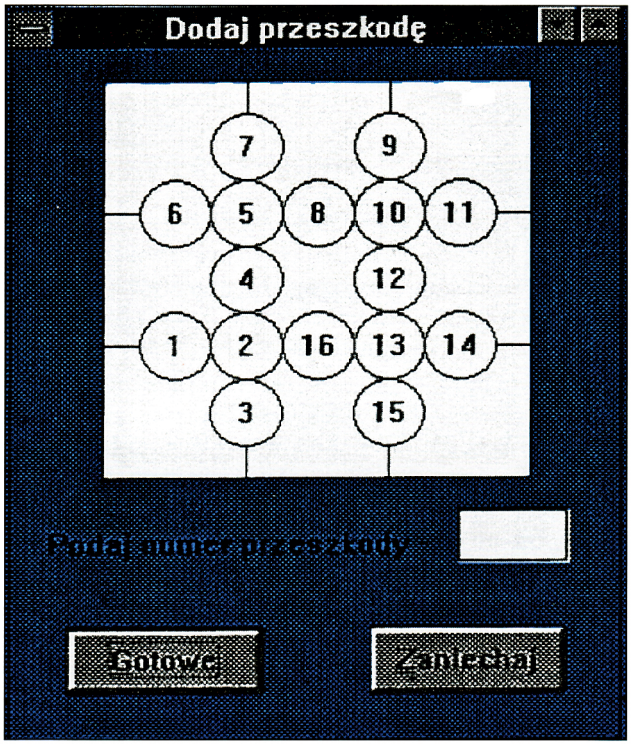
Opis działania programu symulacyjnego

Podstawowym problemem dla tego typu aplikacji jest metoda reprezentacji wyrażenia symbolicznego w pamięci komputera. Analizując fragment wyrażenia:

$$G'^{+} = {}^{0}(b_1{}^{1}(z_2e_4b_{18}, z_1e_1b_{18}, z_1e_4b_5, z_2e_1b_2 \; {}^{2}(\ldots z_3e_{13}b_{14}{}^{3}(\ldots z_1e_{12}b_9{}^{9}(\ldots)\ldots)^{0} \quad (1)$$

można zauważyć, że znajdujące się w nim termy typu $b_i^k($ symbolizują wierzchołki grafu G, zaś sekwencje $z_re_sb_j$, znajdujące się pomiędzy parami nawiasów ${}^k(\ldots{}^{k+1}($ lub ${}^k(\ldots)^k$, opisują krawędzie wychodzące z tych wierzchołków. Automat znajduje się początkowo w stanie b_1. Zgodnie z opisem w części 1, symbole typu b_i odpowiadają strategiom ruchu y_i o takich samych indeksach (tak więc w wierzchołku b_1 generowany jest sygnał y_1, w b_2 – sygnał y_2 i tak dalej). Podczas wykonywania strategii y_1 na drodze robota może wystąpić przeszkoda lub nie, a co za tym idzie – pojawi się na wejściu odpowiednio sygnał z_2 lub z_1. W programie zawarta jest tablica, w której zapisano podaną przez użytkownika konfigurację przeszkód w terenie. Na podstawie adresu obszaru przejścia, związanego z bieżącym wierzchołkiem b_i (adresy obszarów przejść ilustruje rys. 2) oraz danych pobranych z tablicy rozmieszczenia przeszkód, ustalana jest wartość sygnału parametru z. W zależności od niej, a także od aktualnej wartości parametru e następuje przejście do kolejnego wierzchołka b_k. Pomiędzy parą nawiasów ${}^k(\ldots{}^{k+1}($ lub ${}^k(\ldots)^k$ wyszukiwany jest następnie taki term, który zawiera aktualne wartości e_i i z_j, po czym program odczytuje znajdującą się w nim wartość b_z. Dodatkową strukturą wykorzystywaną w programie jest tablica grupująca dane o adresach komórek pamięci, w których zapisane są informacje dotyczące poszczególnych termów $b_i^k($. W niej odnajdowany jest adres komórki przypisanej wierzchołkowi b_z, dzięki czemu program może wykonać skok do tej komórki. Powyższy cykl będzie powtarzany aż do osiągnięcia wierzchołka b_1- lub b_{18}. Przejście do kolejnego wierzchołka odbywa się pod wpływem sygnałów z_i i e_r.

Sposób generowania sygnału został przedstawiony w części 1, wyjaśnienia wymaga jeszcze pochodzenie sygnału e_r. Zgodnie z wynikami syntezy automatu <A'> z parametrem wewnętrznym, wartość tego sygnału pochodzi z układu sekwencyjnego [E] opisanego za pomocą wyrażenia symbolicznego E'^{+}. Zatem należy równocześnie z analizą wyrażenia G'^{+} przetwarzać wyrażenie E'^{+}, w celu ustalenia bieżącej wartości parametru e. Do zapamiętywania położenia przeszkód w terenie służy specjalna jednowymiarowa tablica, w której każda komórka odpowiada jednemu obszarowi przejścia (rys. 2).



Rys. 2. Okno dialogowe do umieszczania przeszkód na mapie terenu – adresy obszarów przejść

Wśród klawiszy funkcyjnych w oknie roboczym programu znajduje się klawisz opatrzony nagłówkiem „Dodaj przeszkodę", wywołujący procedurę umożliwiającą dodawanie nowych przeszkód do konfiguracji terenu. Proces wybierania adresu obszaru, w którym ma zostać umieszczona przeszkoda, opiera się na wizualnej reprezentacji danych – program wyświetla mapę terenu podzielonego na kwadraty z naniesionymi adresami obszarów przejść, zaś osoba przeprowadzająca symulację wprowadza żądany adres w polu dialogowym. W rezultacie w odpowiednią komórkę pamięci zostanie wpisana wartość „1" (początkowo wszystkie zawierają wartość „0"), zaś na obraz terenu wyświetlony w oknie głównym naniesiony zostanie kontur przeszkody (w postaci bitmapy).

Aplikacja w zintegrowanym środowisku programowym Borland Delphi

Do praktycznej realizacji programu symulującego działanie robota mobilnego, poruszającego się po płaszczyźnie z przeszkodami, wybrane zostało środowisko Delphi

firmy Borland [10]. Daje ono możliwość tworzenia aplikacji dla systemu Windows za pomocą metod wizualnych, a także zawiera dogodne narzędzia wspomagające projektowanie interfejsu użytkownika. Spora część projektu programu sprowadza się do określenia własności narzędzi wizualizacji, wybranych spośród komponentów firmy Borland. Zasadniczą część stanowią procedury obsługujące poszczególne zdarzenia (takie jak wybór któregoś z przycisków lub otwarcie nowego okna), a te są zapisane w formie plików ASCII. Po zainstalowaniu programu w środowisku Windows i uruchomieniu go następuje wybór opcji Konfiguracja. Następnie, w celu wprowadzenia automatu w stan początkowy, wywołuje się opcję Zeruj (opcję tę można wykorzystać również do usunięcia z mapy płaszczyzny roboczej wszystkich przeszkód i rozpoczęcia cyklu pracy automatu od początku). Na szachownicy 3×3 terenu pojawia się ikona z wizerunkiem robota, określająca jego aktualne położenie na płaszczyźnie roboczej (rys. 1). Na początkowo pustą mapę terenu można, za pomocą rozkazu Dodaj Przeszkodę, nanosić przeszkody w miejscach oznaczonych adresami pokazanymi na rys. 2. Po wywołaniu rozkazu ukazuje się okno dialogowe służące do wprowadzania adresu dodawanej przeszkody. Po ustaleniu konfiguracji przeszkód na badanej powierzchni roboczej, kolejnym etapem jest sekwencyjne wykonywanie globalnej strategii ruchu robota mobilnego, złożonej ze strategii elementarnych. Wybór opcji Ruch powoduje wykonanie przez robota pojedynczej strategii y_j, co jest obrazowane na bieżąco na mapie płaszczyzny roboczej. Kolejne ruchy są powtarzane aż do osiągnięcia przez automat **<A'>** jednego ze stanów końcowych, oznaczających osiągnięcie przez robota mobilnego wszystkich punktów kontrolnych (wykonanie strategii stanu y_{17}) lub znalezienie się w sytuacji, w której nie znane są reguły postępowania (wykonanie strategii stanu y_{18}). W tym punkcie można sterownik robota „nauczyć" nowych reguł ruchu przez zaprogramowanie dalszych wierzchołków grafu G^+ w postaci nowych wyrażeń symbolicznych, dołączanych do programu.

Uwagi końcowe

Program symulujący działanie automatu z parametrem wewnętrznym jako sterownika robota mobilnego wykorzystuje opis jego struktury logicznej w postaci wyrażeń symbolicznych. Wyrażenia symboliczne są w prosty sposób implementowane w pamięci komputera, można je także rozbudowywać w trybie on line w celu powiększania powierzchni roboczej robota o nowe, losowo rozmieszczane przeszkody. Do praktycznej realizacji programu symulującego bezkolizyjny ruch robota mobilnego wybrano środowisko Delphi firmy Borland, które pozwala zastosować wizualne metody poszukiwania bezkolizyjnej ścieżki jego ruchu w otoczeniu z przeszkodami.

Rysunki oraz pełną bibliografię zamieszczono w PAR nr 4/98.

ABSTRACTS

The Automaton with an Internal Parameter as a Mobile Robot Controller (part 2)
Jan Kazimierczak, Barbara Łysakowska – p. 5
The paper (part 1) describes a formal discrete model of a mobile robot decision unit and its logical structure synthesis.
Part 2 of the paper is the program of the computer simulation being for the mobile robot controller, the automaton with an internal parameter, in the Borland Delphi environment.