2 1975

P. 2229/75

prace

Instytutu Maszyn Matematycznych

rok XVII

Rok XVII

2 1975

P. 2229/75

POLITECHNIKA GDAŃSKA BIBLIOTEKA GŁÓWNA

prace
Instytutu
Maszyn
Matematycznych

Zjednoczenie Przemysłu Automatyki i Aparatury Pomiarowej "MERA"

INSTYTUT MASZYN MATEMATYCZNYCH

Copyright © 1975 - by Instytut Maszyn Matematycznych
Poland
Wszelkie prawa zastrzeżone

Komitet Redakcyjny

Bartłomiej GŁOWACKI, Andrzej KOJEMSKI, Roman KULESZA (redaktor naczelny)
Włodzimierz MARDAL (z-ca red. naczelnego), Jan RELUGA
Sekretarz Redakcji: Romana NITKOWSKA

Adres Redakcji: Instytut Maszyn Matematycznych
Branżowy Ośrodek INTE
Warszawa, ul. Krzywickiego 34
tel. 28-37-29

Druk IMM z. 80/75 n. 420 pap. piśm. kl. III g. 70 GP-II-1435/68

PW - 25/76

SPIS TREŚCI ZESZYTU 2

СОДЕРЖАНИЕ

CONTENTS

1. DAŃDA J., PIETRALA J., POZNAŃSKI Z.

Wybrane zagadnienia symulacji systemu kierowania ruchem lotniczym. Wyniki wstępne 5

Некоторые проблемы моделирования Системы Управления Воздушным Движением.Предварительные результаты (Резюме)

Some aspects of simulation of ATC system. Preliminary results (Summary)

2. PATRYN R.

Koncepcja syntezy fonematycznej (beztransientowej) 31

Общая концепция фономатического (непеременного) синтеза (Резюме)

General concept of phonematic synthesis (not-transient) (Summary)

3. ZIEMKIEWICZ A.

Wybrane procedury graficzne w systemach komputerowych 43

Избранные графические процедуры в системах цифровых вычислительных машин (Резюме)

Some graphic procedures in computer systems (Summary)

4. ROWICKI A.

Algorytm planowania obliczeń dwuprocesorowych . . 65

Алгоритм планирования двупроцессорных вычислений (Резюме)

Algorithm of scheduling for two-processors systems (Summary)

5. WITASZEK J.

Dwa algorytmy konstrukcji deterministycznego automatu skończonego na podstawie zadanej gramatyki lewostronnie lub prawostronnie liniowej 85

Два алгоритма конструкции детерминистического конечного автомата на основе заданной левосторонней или правосторонней линейной грамматики (Резюме)

Two algorithms constructing deterministic finite state automata for right or left linear grammars (Summary)

6. JARZĄBEK S., KRAWCZYK T.
LL - regular grammars (Sufficient conditions) . . 107
LL - regularne gramatyki (warunki dostateczne)
(Streszczenie)

LL - регулярные грамматики (достаточные условия)
(Резюме)

AUTORZY ARTYKUŁÓW ZAMIESZCZONYCH W ZESZYCIE 2 "PRAC IMM" 118

Prace IMM
Zeszyt 2
© 1975.04
681.322.004.14:656.7.001.57:
621.396.96

WYBRANE ZAGADNIENIA SYMULACJI SYSTEMU KIEROWANIA RUCHEM LOTNICZYM. WYNIKI WSTĘPNE

Jerzy DAŃDA
Jan PIETRALA
Zbigniew POZNAŃSKI

Pracę złożono 3.03.1975

W pracy przedstawiono uproszczony model systemu kierowania ruchem lotniczym (SKRL). Przyjęty model zdefiniowano programem w języku SIMULA-67. Omówiono wyniki badań symulacyjnych modelu przeprowadzonych na komputerze. Wykazano przydatność SIMULA-67 do opisu i modelowania dużych, złożonych systemów.

WSTĘP

Systemy kierowania ruchem lotniczym (SKRL; bądź w terminologii angielskiej Air Traffic Control Systems - ATCS) stały się w świecie stymulatorem badań w wielu dziedzinach nauki i techniki takich jak: teoria systemów, struktury algorytmów i danych, architektura maszyn cyfrowych, środki zobrazowania, organizacja systemów wielomaszynowych i wieloprocesorowych, języki problemowe itp.

Przez kierowanie ruchem lotniczym rozumie się koordynowanie ruchu lotniczego w celu zapewnienia odpowiedniego poziomu bezpieczeństwa statkom powietrznym. Innymi słowy, obiektem sterowanym jest ruch lotniczy, koordynatorem zaś zespół ludzi (kontrolerów ruchu) wyposażonych w środki niezbędne do realizacji wymienionego zadania (środki łączności, urządzenia radiolokacyjne, dokumentacja załóg lotniczych itp.).

Wzrastające nasilenie ruchu lotniczego spowodowało w konsekwencji zapotrzebowanie na automatyzację wielu funkcji SKRL, za pomocą maszyn cyfrowych. Poziom automatyzacji SKRL lub inaczej realizacja ASKRL (Automatyczne Systemy Kierowania Ruchem Lotniczym) zależy od wielu czynników takich jak nakłady na realizację, niezawodność i przepustowość systemu ASKRL, elastyczność tj. możliwość rozbudowy itp. Spowodowało to badania w wymienionych na początku kierunkach. Systemy kierowania ruchem lotniczym kwalifikują się do systemów wielkich, stąd i postać ASKRL zależy bezpośrednio od rozwoju teorii wielkich systemów. Ograniczenia czasowe nałożone na nie kwalifikują je do systemów uwarunkowanych czasowo real-time , stąd poszukiwania odpowiednich maszyn cyfrowych i środków zobrazowania danych, a które w powiązaniu z nakładami na budowę ASKRL - prowadzą do optymalnej organizacji systemów wielomaszynowych czy wieloprocesorowych. Zastosowanie metod matematycznych w fazie projektowania ASKRL i fazie eksploatacji zmusza do poszukiwania odpowiednich języków problemowych i programowania. Jedną z metod jest budowa i badanie symulacyjnych modeli systemów. W literaturze światowej pojawiło się wiele publikacji poświęconych modelowaniu systemów kierowania ruchem lotniczym [1], [2]. Modele te programowano w różnych językach np. FORTRAN-IV czy GPSS. Celem niniejszego artykułu jest przedstawienie uproszczonego modelu SKRL, opracowanego w IMM i uruchomionego w języku SIMULA-67 na komputerze CYBER-72 w ośrodku CYFRONET w Świerku pod Warszawą.

1. OPIS MODELU ASKRL

Model symuluje, z uproszczeniami, działanie w czasie SKRL-u, gdzie obiektem obsługi jest ruch lotniczy, aparat obsługi stanowi operator - kontroler ruchu lotniczego, działający w tzw. systemie proceduralnym i wspomagany przez radiolokacyjny system informacyjny (RSI).

1.1. Model ruchu lotniczego

Niech T oznacza zbiór chwil działania obiektu sterowanego, a O zbiór statków powietrznych znajdujących się w danej chwi-

li w locie nad terytorium danego państwa lub manewrujących na polach startowych lotnisk tego terytorium. W ślad za [3] ruch lotniczy interpretuje się fizycznie jako pewien kompleks sterowany, złożony ze statków $s_i \in S (i = 1, 2, \ldots, n)$ o określonych własnościach technicznych i niezawodnościowych oraz znanych charakterystykach dynamicznych w czasie t.

Formalnie do opisu ruchu lotniczego wchodzą następujące elementy: obszar powietrzny tzw. FIR (Flight Information Region) i podobszary: rejony lotnisk tzw. TMA (Terminal Area) i strefy lotnisk tzw. CTR (Control Zone) oraz odcinki dróg powietrznych tzw. AWY /Airways/ tworzących wraz z TMA i CTR sieć oznaczaną CS. Jest to opis statyczny, a do opisu dynamicznego wchodzą podobszary wyznaczone przez pewne punkty oznaczające zmierzone położenie obiektu s_i w chwili t wraz z pewnym otoczeniem μ tego położenia oraz rodziny wymienionych podobszarów opisujących aktualne położenie wszystkich czynnych w chwili t obiektów powietrznych.

Dodatkowo definiuje się pojęcie lotów mierzonych, lotów planowanych i lotów programowych.

W przypadku prezentowanego modelu, opracowanego w IMM, założono, że dla lotów planowanych i mierzonych dostępny jest każdy punkt pewnego dwuwymiarowego (x, y) wycinka przestrzeni powietrznej. Model prezentowany może generować dowolną liczbę statków powietrznych ograniczoną tylko pojemnością pamięci dostępnej w wykorzystywanym zestawie CYBER-72. W modelu realizuje się funkcję rzeczywistego SKRL, tzw. planowanie wstępne polegające (w SKRL) na wczytywaniu danych z planów lotów; (tj. numer rejsu, tor lotu w postaci punktów startu, zmiany kierunków lotów, prędkości między punktami zmiany kierunków lotów oraz punktów lądowań). Ta część w rzeczywistym SKRL może odbywać się w warunkach off-line.

Inicjacja lotów planowanych odbywa się w momencie czasu symulowanego równego czasowi planowanego startu samolotu. Model można poszerzyć o tzw. procedurę CLEARANCE realizowaną przez kontrolera ruchu i w tym przypadku lot planowany, na który nie uzyskanoby zezwolenia nie byłby realizowany mimo posiadania

planu lotu. Lot samolotu realizuje się w czasie symulowanym co pewną zadaną dyskretną wartość czasu przez obliczanie współrzędnych samolotów będących w ruchu, na odcinkach prostych wyznaczonych przez punkty zmiany kierunku (w tym punkty startu i lądowań). Symulację elementów kierowania tzw. proceduralnego, o którym wspomniano wyżej, stanowi tworzenie meldunków od samolotu do kontrolera o fakcie startu, osiągnięciu planowanego punktu zmiany kierunku (współrzędne punktu x, y, czas osiągnięcia punktu) i zakończeniu lotu (lądowanie). Należy podkreślić, że różnego rodzaju meldunki załóg stanowią cechę ruchu lotniczego w systemie kierowania proceduralnego.

1.2. Model RSI

Radiolokacyjne systemy informacyjne RSI określone w [3] jako systemy informacyjne wyposażone w kompleksy radarowe wraz z wszelkimi tego konsekwencjami stanowią obecnie nieodłączny element SKRL. Zadaniem RSI jest dostarczenie kontrolerowi ruchu lotniczego informacji o aktualnym stanie ruchu lotniczego.

Na podstawie obecności lub braku echa radiolokacyjnego w elementarnym odległościowym i azymutalnym wycinku $(\Delta R, \Delta\beta)_H$ obszaru FIR dla ustalonej wysokości H lotu statku w zależności od charakterystyk technicznych radarów, (takich jak prawdopodobieństwa fałszywego alarmu $P^{(FA)}$, prawdopodobieństwa wykrycia p^w, dokładności namiaru współrzędnych δ, rozróżnialności Δ) oraz od stopnia złożoności RSI (tj. zastosowanych układów elementarnych, agregatów i podsystemów RSI) realizujących tzw. obróbkę pierwotną, wtórną i trzeciego stopnia - kontroler otrzymuje informację bądź o położeniach wykrytych statków (obróbka pierwotna), bądź o torach statków i dodatkowych danych, bądź w postaci tras statków wraz z dodatkowymi danymi. Każda z tych informacji scharakteryzowana jest ciągami pewnych parametrów podobnych do podanych wyżej.

W prezentowanym modelu zastosowano wiele uproszczeń, w wyniku których generuje się obiekt symulujący urządzenie radarowe

o zadanej rozróżnialności Δ, dokładności namiaru współrzędnych δ, prawdopodobieństwie wykrycia równym 1 i innych jak zasięgu i czasie obrotu anteny (wielkości zmienne). Postać informacji wyjściowej jest zbliżona do postaci otrzymywanej z obróbki wtórnej, pominięto ze zrozumiałych względów mechanizm wspomnianej obróbki.

Aczkolwiek można wygenerować dowolną liczbę urządzeń radarowych (własność języka SIMULA-67) nie poszerzono modelu o syntezę informacji z wielu źródeł; jest to jednak możliwe.

1.3. Model kontrolera ruchu lotniczego

Funkcje kontrolera w SKRL zależą od organizacji służb ruchu lotniczego, np. inny zakres i charakter prac mają kontrolerzy w tzw. kontroli obszaru (ACC-Area Control Centre), inny w tzw. kontroli zbliżania (APP - Approach Control), a jeszcze inny w wieży kontrolnej (TWR - Tower). Zależą też od sposobu prowadzenia kontroli (np. kontroli czysto proceduralnej, mieszanej tj. proceduralnej z wykorzystaniem informacji nieradiolokacyjnej oraz radiolokacyjnej) oraz od zastosowanych środków i poziomu automatyzacji. W prezentowanym uproszczonym modelu przedstawiono funkcje kontrolera kontroli obszaru, który może koordynować ruch lotniczy na podstawie zgłoszeń (meldunków) od statków, czyli elementy kontroli proceduralnej, oraz zobrazowania informacji radiolokacyjnych. Kontroler ma możliwość wprowadzania zmian w planowaniu ruchu lotniczego oraz pracy urządzeń radarowych.

O g r a n i c z e n i a m o d e l u

Ze względu na wielkość i złożoność SKRL wynikają następujące problemy:

- optymalne struktury przestrzeni powietrznej FIR;
- optymalne rozlokowanie środków radionawigacyjnych i radiolokacyjnych lub odwrotnie, dobór środków przy ustalonej lokacji;

- optymalne procedury służb SKRL w warunkach automatyzacji (ASKRL);
- budowa symulatorów cyfrowych;
- optymalne konfiguracje automatycznych środków obliczeniowych itd.

Każdy z tych problemów stanowi oddzielną dziedzinę badań, z którą związane jest zaangażowanie dużego potencjału sił badaczy projektantów oraz środków i czasu.

Zadaniem zespołu pracowników Instytutu Maszyn Matematycznych było zbadanie możliwości i zalet zastosowania do problemów SKRL języka problemowego SIMULA-67, stąd badania przeprowadzono dla:

- niewielkiej liczby statków powietrznych,
- niewielkiej liczby urządzeń radiolokacyjnych,
- uproszczonej roli kontrolera ruchu lotniczego.

Nawet tak ograniczony model pozwolił na wyciągnięcie wniosków na temat możliwości i zalet języka SIMULA-67 w zastosowaniu do problematyki wielkich systemów, które prezentowane są w dalszej części pracy.

2. OPIS PROGRAMU SYMULACYJNEGO

Przy pisaniu programu w maksymalnym stopniu uwzględniano zasady programowania strukturalnego; w artykule zakładamy, że zasady te są znane czytelnikowi (patrz np. Dijkstra [4] oraz A. Mazurkiewicz [5])i że nie trzeba wykazywać zalet (np. w zakresie czytelności programu) jakie wynikają ze stosowania zasady programowania strukturalnego. W literaturze dotyczącej tematu znajdują się pozycje (patrz np. [4]), w których mówi się o łatwości stosowania programowania strukturalnego przy posługiwaniu się językiem SIMULA-67. Nasze doświadczenia potwierdzają te stwierdzenia - mamy nadzieję, że ci z czytelników, którzy zechcą wniknąć w szczegóły przytoczonego programu będą mogli wyrazić podobną opinię. Ponieważ tekst programu jest zaopatrzony w bardzo nieliczne komentarze - podany poniżej krótki opis programu ma na celu uzupełnienie tego niedostatku.

2.1. Opis struktury programu

Program składa się z części deklaracyjnej i programu głównego - deklaracje obejmują wiersze 0 - 227, program główny - 230 - 254. Widać, że zgodnie z zasadami programowania strukturalnego program główny jest stosunkowo bardzo małą częścią całego programu; nota bene - jest to również charakterystyczną cechą programów symulacyjnych w języku SIMULA.

Wiersze 0 - 8
Obejmują deklaracje zmiennych globalnych, określających warunki symulacji oraz zmiennych referencyjnych dla obiektów (procesów), które działają w programie.

Wiersze 9 - 13 link class WIADOMOSC (DL);
Jest deklaracją obiektu klasy link; w programie służy ona jako półstandardowy format wymiany wiadomości od SAMOLOT-u do SZEF-a i do RADAR-u. Obiekt WIADOMOŚĆ ma część niezmienną, złożoną z trzech integer NUMER, INNYNUMER, KOD; oraz część zmienną, złożoną z real array CONTS. Długość przesyłanej wiadomości określa aktualna wartość parametru formalnego DL. Maksymalna wartość DL wynosi 10.

Wiersze 16 - 101 process class SAMOLOT;
Jest to deklaracja obiektu klasy process i opisuje sposób zachowania się samolotu. Oczywiście, jak to już wcześniej powiedziano, jest to uproszczony model.

Wiersze 17 - 20 zawierają deklaracje lokalnych zmiennych SAMOLOT-u.

Wiersze 21 - 60 procedure PRZELOTODC;
Procedura PRZELOTODC (skrót od "przelot odcinka") zawiera opis wszystkich czynności, które wykonuje samolot lecąc po odcinku prostej, wg ustalonego wcześniej planu lotu. Opiszemy ją nieco szczegółowiej, ma ona bowiem bardziej złożoną budowę niż inne części programu. Wewnątrz PRZELOTODC mamy zadeklarowane w wierszach 22 - 23 lokalne zmienne tej procedury, a w wierszach 24 - 36 lokalną funkcję referencyjną:
ref (link) procedure TUJESTEM (KOD, CZAS, X, Y);

Zadaniem tej funkcji jest nadanie półstandardowej WIADOMOSC-i standardowej formy swego rodzaju meldunku, który jest wysyłany albo do SZEF-a, albo do RADAR-u. Wysyłanie tych meldunków wykonywane jest z różną częstotliwością w tych obu przypadkach, będzie o tym mowa dalej. Standardyzowanie formy meldunku polega na następujących czynnościach:

- generacji nowego obiektu klasy WIADOMOSC o długości zmiennej części równej 3 - wiersz 28;
- nadaniu stałym częściom WIADOMOSC-i stałym dla danego SAMOLOT-u cech MISJA i NRREJSU - wiersze 29 i 30;
- nadaniu stałej części WIADOMOSC-i KOD i zmiennym CONTS wartości aktualnych parametrów formalnych tej procedury tzn. KOD, CZAS, X, Y. Określają one odpowiednio:
  KOD - stan samolotu równy 1 - gdy samolot startuje
  2 - gdy leci
  3 - gdy ląduje
  CZAS - chwilę czasową, w której meldunek jest wysyłany
  X, Y - położenie samolotu w chwili składania meldunku.

W ten sposób ta forma meldunku umożliwia pełne zidentyfikowanie zarówno przez SZEF-a jak i przez RADAR, od którego SAMOLOT-u pochodzi dany meldunek, w której chwili meldunek został wysłany i gdzie SAMOLOT w tej chwili się znajdował.

Wyżej omówione czynności wykonywane są w wierszach 31 - 34; wiersz 35 nadaje funkcji TUJESTEM wartość zmiennej referencyjnej, określającej dostęp do wygenerowanego w trakcie działania funkcji obiektu klasy WIADOMOSC.

Wiersze 39 - 59 zawierają instrukcje procedury PRZELOTODC Omówimy teraz kolejno ich działanie. Wiersz 39 zawiera instrukcję, która generuje meldunek o położeniu samolotu na początku odcinka, wstawiając równocześnie meldunek do kolejki DLASZEFA. Dla wyjaśnienia podamy, że użycie TIME, jako drugiego parametru aktualnego procedury TUJESTEM w tym wierszu powoduje przekazanie aktualnego symulowanego czasu; TIME jest bowiem procedurą systemową języka SIMULA-67. Zmienna J określa numer kolej-

nego odcinka lotu danego samolotu, a real array X, Y, V [1..50] zawierają plan lotu w postaci położeń i szybkości samolotu dla kolejnych odcinków. Instrukcje 40 - 43 służą do obliczenia czasu przelotu samolotu SKOKT przez bieżący odcinek. Wielkość ta jest potrzebna do określenia liczby meldunków dla radaru NDELT, które mają być kolejno (w miarę postępowania symulowanego lotu) wprowadzane do kolejki DLARADARU. Długość odcinka czasu między dwoma meldunkami dla radaru określana jest przez TQU (kwant czasu dyskretyzacji) i wprowadzana jest na początku symulacji, jako jeden z parametrów programu. Instrukcje 44 - 49 przygotowują odpowiednie zmienne dla pętli (wiersze 50 - 55), która generuje meldunki i wstawia je do kolejki DLARADARU. Instrukcja 58 - 59 przesyła meldunek do szefa, gdy samolot skończy lot - pierwszy parametr aktualny procedury TUJESTEM wynosi 3, czyli jest kodem wylądowania, zgodnie z tym co wyżej powiedziano.

Wiersze 63 - 86 procedure PRZYGOTLOT;

W tych wierszach zawarta jest procedura "przygotowanie lotu" - jej działanie polega na wczytaniu (np. z czytnika kart) następujących danych stałych dla danego samolotu:

numeru rejsu (NRREJSU),
godziny odlotu (GODZODLOTU),
minuty odlotu (MINUODLOTU),
liczby manewrów (ILMAN),

czynności te wykonują instrukcje w wierszach 65 - 68. Wczytana zmienna ILMAN steruje następnie pętlą (wiersze 69 - 73), która wczytuje plan lotu (również z czytnika kart). Tak więc każdy samolot zostaje opisany zmienną liczbą kart, w których od piątej w górę zawarty jest plan lotu. Instrukcja w wierszu 76 oblicza planowany start samolotu (w sekundach od początku liczenia czasu symulowanego) PSSAM. Wartość tej zmiennej jest następnie (w wierszu 90) wykorzystywana do "uruchomienia" samolotu. Instrukcje w wierszach 78 - 84 służą do przekazania SZEFOWI danych o zaplanowanym locie. Wykorzystywana jest w tym celu WIADOMOSC, w której poszczególnym zmiennym są przypisywane inne znaczenia niż w standardowym meldunku, z tego względu właśnie WIADOMOSC została uprzednio nazwana półstandardowym forma-

tem. Rozwiązanie takie ma pewne zalety, umożliwia bowiem stosowanie jednej wspólnej procedury wydruku treści kolejek DRUKUJKOL, niezależnie od tego jakie znaczenie mają zawarte w nich informacje. Procedura DRUKUJKOL zawarta jest w wierszach 131 - 163 i nie będzie szczegółowiej opisywana.

Wiersze 89 - 100 stanowią instrukcje obiektu SAMOLOT. Jak widać, dzięki wcześniejszemu przygotowaniu odpowiednich procedur struktura tej części jest bardzo prosta, i tak:

- zostaje wykonane przygotowanie lotu przez wywołanie procedury PRZYGOTLOT - wiersz 89,
- o czasie PSSAM samolot zostaje uruchomiony - wiersz 90,
- samolot wykonuje przelot po wszystkich zaplanowanych odcinkach, pętla zawarta w wierszu 91 zapewnia bowiem (ILMAN - 1) -krotne wywołanie procedury PRZELOTODC,
- instrukcja 100 powoduje zakończenie działania samolotu.

Uwaga: instrukcje w wierszach 93 - 97 służą do zakończenia działania całego programu, chodzi o to, by SZEF i RADAR zakończyli swoje działania po wylądowaniu wszystkich samolotów; szczegóły tego rozwiązania nie będą omawiane w tym uproszczonym opisie.

Dalszy opis programu podamy jeszcze zwięźlej.

Wiersze 104 - 108 process class CZŁOWIEK;
Zawierają deklarację obiektu, który generuje obiekty SAMOLOT (wiersz 109), kolejno generowanym samolotom nadaje numery misji (kolejne liczby naturalne ze zbioru 1, ILSAM) i wygenerowane samoloty aktywizuje (uruchamia) - wiersz 111. Proces powoływania samolotów do wykonywania lotów dokonywany jest cyklicznie z okresem CYKLMISJI - wiersz 112, a sam proces zaczyna się względem początku liczenia czasu symulowanego z opóźnieniem wynoszącym ROZRUCH - wiersz 107. Po zakończeniu powoływania misji CZŁOWIEK zaczyna przeglądać kolejkę DLASZEFA, drukując jej zawartość za pomocą procedury DRUKUJKOL - wiersz 123. Proces drukowania zawartości tej kolejki wykonywany jest cyklicznie z okresem wynoszącym CYKLOBSER - wiersz 124.

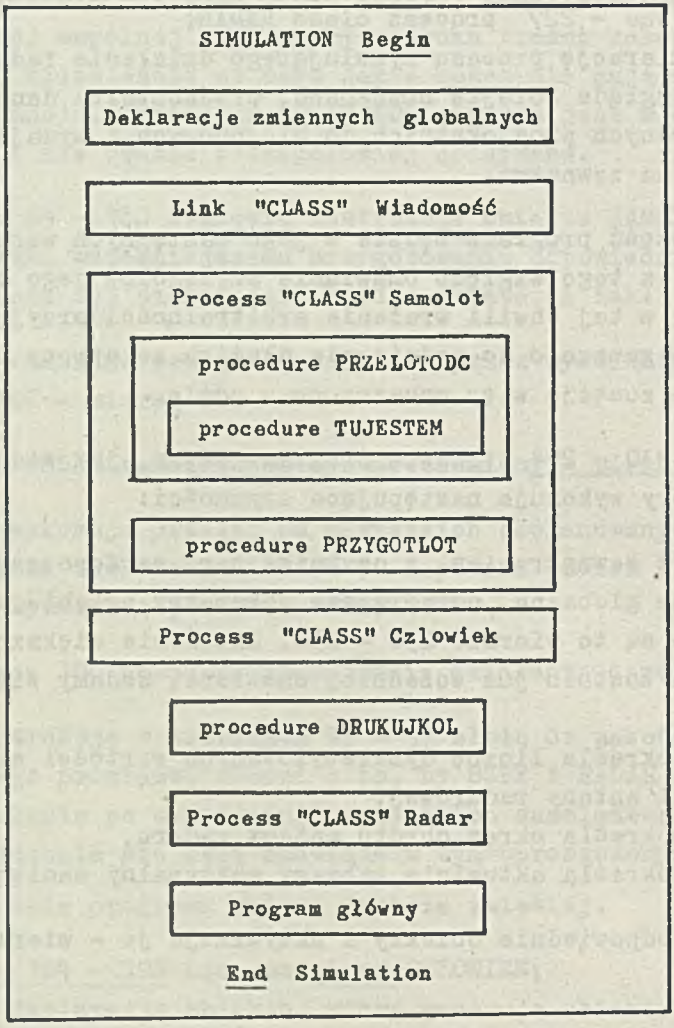
Wiersze 166 - 227 process class RADAR;
Zawiera deklarację procesu symulującego działanie radaru; proces ten przegląda kolejkę DLARADARU, przekształca dane z układu współrzędnych prostokątnych do biegunowych i wydaje wyniki obserwacji na zewnątrz.

Uwaga: ta część programu będzie w jego następnych wersjach rozbudowywana, z tego względu omawianie szczegółów jego budowy sprawiałoby w tej chwili wrażenie arbitralności przyjętych rozwiązań, lub zmuszało do nadmiernie długich ze strony autorów wyjaśnień - zostaje więc opuszczone w ogóle.

Wiersze 230 - 254 stanowią program główny, jak widać bardzo prosty, który wykonuje następujące czynności:

- wczytuje z zewnątrz (np. z czytnika kart perforowanych) pewne zmienne globalne, ustawiające parametry przebiegu symulacyjnego - są to wiersze 230 - 238. Znaczenie większości tych zmiennych zostało już wcześniej omówione, dodamy więc tylko, że:
  - ILAZYM określa liczbę dyskretyzowanych wartości azymutu obserwacji anteny radarowej,
  - TOBRAN określa okres obrotu anteny radaru,
  - RADMAX określa aktualnie wybrany maksymalny zasięg radaru,
- generuje odpowiednie obiekty i aktywizuje je - wiersze 240 - 246,
- wykonuje czynności związane z synchronicznym zakończeniem pracy odpowiednich obiektów, po wylądowaniu wszystkich samolotów - wiersze 247 - 250.

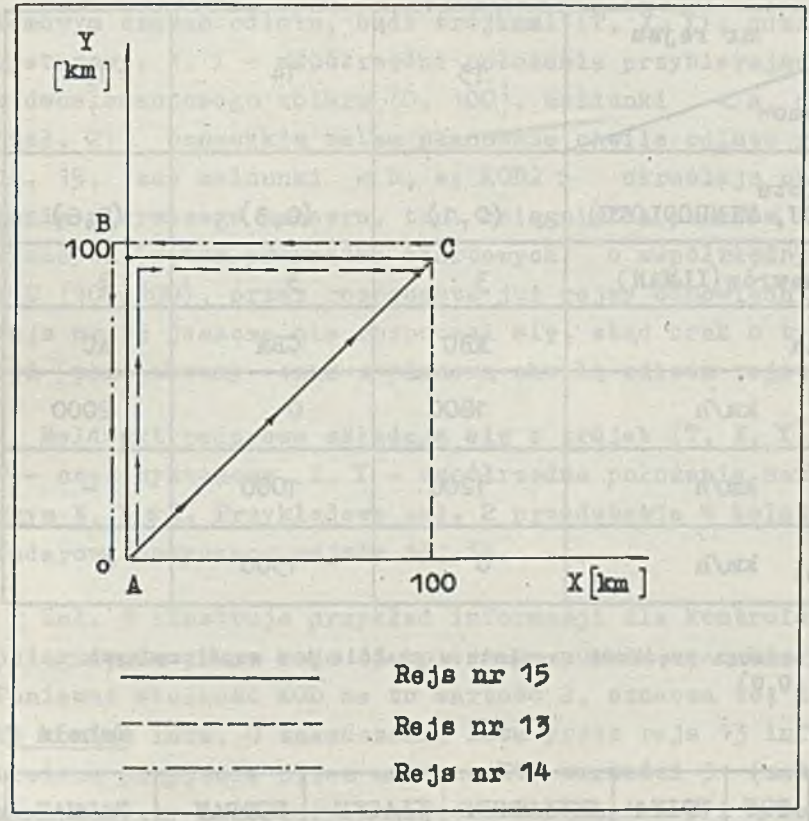
Ogólna struktura programu została przedstawiona na rys. 1, struktura ta jest również naszkicowana na wydrukach programu (załącznik 1); działanie programu dla sytuacji dynamicznej jak na rys. 2 zilustrowane jest fragmentami wyników symulacji - załącznik 2, 3, 4. Komentarz do uzyskanych wyników, jak również wartości przyjętych parametrów symulacji zawarte są w dalszej części artykułu.



Rys. 1

2.2. Parametry symulacji. Wyniki

Symulowano równoległy w czasie lot trzech samolotów (ILSAM = 3) wzdłuż boków trójkąta prostokątnego ABC (rys. 2). Każdemu samolotowi przypisano numer rejsu, planowaną chwilę odlotu, liczbę manewrów, prędkość oraz trajektorię lotu. Kolejne misje powoływane są co okres CYKLMISJI równy 64s.



Rys. 2

Uwaga: za jednostkę czasu systemowego przyjęto sekundę. Start samolotu następuje w chwili określonej dwójką (GODZODLOTU, MINUODLOTU) oznaczającą godzinę i minutę odlotu. Trajektorie lotu wszystkich obiektów latających ilustruje rys. 2. Można tu wyróżnić trzy punkty charakterystyczne: A, B, C, w których następuje skokowa zmiana prędkości samolotów. Zakłada się, że na odcinkach prostych ruch jest jednostajny, tzn. np. odcinek BC samolot pokonuje ze stałą prędkością równą tej, którą osiągnął w punkcie B. Liczba punktów charakterystycznych danej trajektorii wyznacza liczbę manewrów (ILMAN) związanych ze zmianą prędkości.

Wartości parametrów symulacji przedstawione są w Tabeli 1 oraz Tabeli 2.

Tabela 1

parametry dotyczące samolotu \ nr rejsu	13	14	15
Chwila odlotu (GODZODLOTU, MINUODLOTU)	(0,1)	(0,3)	(0,6)
Liczba manewrów(ILMAN)	3	3	2
Trajektoria	ABC	CBA	AC
$V_{A(0,0)}$ *) km/h	1800	0	2000
$V_{B(0,100)}$ km/h	1200	1000	-
$V_{C(100,100)}$ km/h	0	1500	0

*) $V_{A(0,0)}$ - oznacza prędkość samolotu w punkcie A o współrzędnych (0,0)

Tabela 2

Parametry inne	TQU	WDLY1	CYKLOBSER	ILAZYM	TOBRAN	RADMAX
Wartości	8s.	4s.	60s.	36	18s.	150 km

Fragmenty wyników symulacji ilustrują załączniki 2, 3, 4. Co okres wynoszący CYKLOBSER przeglądana jest kolejka DLASZEFA, w której znajdują się meldunki o planowanych chwilach odlotu poszczególnych samolotów oraz o wykonaniu przez nie manewrów.

Kolejka DLARADARU, w której znajdują się meldunki o chwilach osiągnięcia przez samoloty poszczególnych punktów trajektorii, sprawdzana jest z okresem TOBRAN.

Załącznik 2 ilustruje meldunki przekazywane kontrolerowi (KOLEJKA DO SZEFA O CZASIE) oraz meldunki przekazywane przez radar (KOLEJKA ROBRAD O CZASIE). Meldunki dla kontrolera są bądź

trójkami (GODZODLOTU, MINUODLOTU, PSSAM), gdzie PSSAM jest systemowym czasem odlotu, bądź trójkami (T, X, Y), gdzie T - czas systemowy, X, Y - współrzędne położenia przybierające wartości z dwuelementowego zbioru $\{0, 100\}$. Meldunki < a, c, d;KOD1 > (zał. 2) oznaczają zatem planowane chwile odlotu rejsów 13, 14, 15, zaś meldunki < b, e; KOD2 > określają chwile wykonania pierwszego manewru, tzn. osiągnięcia punktów charakterystycznych (w tym przypadku startowych) o współrzędnych A(0,0) i C (100,100), przez rozpoczęte już rejsy odpowiednio 13 i 14. Rejs nr 15 jeszcze nie rozpoczął się, stąd brak o tym informacji (por. obecny czas z planową chwilą odlotu rejsu 15).

Meldunki radarowe składają się z trójek (T, X, Y), gdzie T - czas systemowy, X, Y - współrzędne położenia samolotu, przy czym X, Y $\in$ R. Przykładowo zał. 2 przedstawia 4 kolejne meldunki radarowe dotyczące rejsów 13, 14.

Zał. 3 ilustruje przykład informacji dla kontrolera o chwili osiągnięcia przez rejs 13 kolejnego punktu charakterystycznego. Ponieważ wielkość KOD ma tu wartość 2, oznacza to, że nie jest to koniec lotu. O zakończeniu lotu przez rejs 13 informuje kontrolera przyjęcie przez zmienną KOD wartości 3 (zał. 4). To samo dotyczy pozostałych rejsów. Załączniki 3 i 4 zawierają ponadto wydruki meldunków radarowych.

W ten sposób uzyskano zobrazowanie informacji radarowej o położeniu wszystkich obiektów latających oraz informacji sygnalizujących wykonanie manewrów przez poszczególne samoloty.

3. ZAKOŃCZENIE

a) Analiza i projektowanie systemów zaliczanych do klasy systemów wielkich wydaje się być obecnie możliwe jedynie na drodze symulacji cyfrowej. Rozwiązania analityczne można otrzymać tylko przy dużych uproszczeniach, prowadzących do znacznych rozbieżności pomiędzy modelem i rzeczywistością, a zatem ich wartość praktyczna jest wątpliwa.

b) Złożoność systemów typu SKRL powoduje, że ich modele działania można tworzyć tylko w językach wyższego poziomu, wyspecjalizowanych w kierunku symulacji. Wybierając język SIMULA-67 uzyskano dodatkowo, ze względu na to, że jest on rozszerzeniem Algolu, zgodność opisu algorytmu z formą powszechnie uznaną za najwygodniejszą.

c) Język SIMULA-67 może też być użyty jako język, w którym będą pisane programy użytkowe rzeczywistego systemu KRL. Ułatwi to przechodzenie od modelu do rzeczywistości.

d) Modułowość osiągnięta w opisie SKRL za pomocą języka SIMULA-67 pozwala z jednej strony na łatwe rozszerzenie opisywanego problemu i eliminację, w miarę konieczności, wprowadzonych uproszczeń; z drugiej strony przejście od modelu do systemu rzeczywistego może być wykonywane etapami.

e) Zaproponowany model, po ewentualnej modyfikacji, może być elementem pewnego wielkiego systemu kierowania ruchem lotniczym. Stanowi on zatem przyczynek do dalszych prac projektowo-badawczych w tym zakresie.

Literatura

[1] VANDER VEER J.R., BONA L.J.: Air Traffic Control System-digital Simulation Facility, Simulation, 1971, t. 16, nr 1, s. 5-17

[2] COCHRAN J.G., PRESTON E.J.: Simulation of European Air Traffic Control System, The Radio and Electronic Engineer, 1965, t. 30, nr 5, s. 307-316

[3] JANICKI A.: Adaptacyjna synteza radiolokacyjnych systemów informacyjnych; rozprawa habilitacyjna, Informator ITWL, Warszawa 1971

[4] DIJKSTRA E.W.: Cooperating Sequential Processes, Programming Languages, Academic Press 1968

[5] MAZURKIEWICZ A.: Problemy programowania. Uwagi i refleksje w XV-lecie Instytutu Maszyn Matematycznych, materiały na sesję naukową z okazji roku nauki polskiej i XV-lecia Instytutu Maszyn Matematycznych, Warszawa 1973

ZAŁĄCZNIK 1

Program symulacyjny SKRL

```
0     SIMULATION "BEGIN"
0     "INTEGER" ILSAM,ILCYK, TOU, WDLY1,ROZRUCH, CYKLMISJI,
0     CYKLOBSER..
1     "INTEGER" ILAZYM,TOBRAN,RADMAX..
2     "INTEGER" K..
3     "BOOLEAN" TERM..
4     "COMMENT" ILAZYM..ILOSC AZYMUTOW NA OBWODZIE,BIAZYM..
4     BIEZACA WARTOSC AZYMUTU W RADIANACH,TOBRAN..OKRES
4     OBROTU ANTENY,RADMAX..AKTUALNY ZASIEG RADARU..
5     "REF"( PROCESS)SZEF ..
6     "REF"(PROCESS)RAD..
7     "REF"( HEAD) DLASZEFA, DLARADARU ..
8     "REF"(HEAD)ROBRAD,ZOBRAD..

9     LINK "CLASS" WIADOMOSC(DL).. "INTEGER" DL..

11    "BEGIN"
11    "INTEGER" NUMER, INNYNUMER,KOD..
12    "REAL" "ARRAY" CONTS(/1..10/)..
13    "END" WIADOMOSC..

16    PROCESS "CLASS" SAMOLOT ..

17    "BEGIN"
17    "INTEGER" ILMAN..
18    "INTEGER" MISJA, NRREJSU,GODZODLOTU,MINUODLOTU,J..
19    "REAL" PSSAM,RX,RY..
20    "REAL""ARRAY"X,Y,V(/1..50/)..
21    "PROCEDURE" PRZELOTODC..
22    "BEGIN"
22    "INTEGER" NDELT,KDELT..
23    "REAL" SKOKX,SKOKY,SKOKZ,SKOKT,DELTX,DELTY,DELTZ,DELTT..
24    "REF"(LINK)"PROCEDURE" TUJESTEM(KOD,CZAS,X,Y)..
25    "REAL" CZAS,X,Y.,"INTEGER" KOD..
27    "BEGIN"
27    "REF"(WIADOMOSC)WLIN..
28    WLIN.-"NEW" WIADOMOSC(3)..
29    WLIN.NUMER .= MISJA ..
30    WLIN.INNYNUMER.=NRREJSU..
31    WLIN.KOD .= KOD ..
32    WLIN.CONTS(/1/).=CZAS..
33    WLIN.CONTS(/2/).=X..
34    WLIN.CONTS(/3/).=Y..
35    TUJESTEM.-WLIN..
36    "END"..
39    TUJESTEM(2,TIME,X(/J/),Y(/J/)).INTO(DLASZEFA)..
40    SKOKX.=X(/J+1/)-X(/J/)..
41    SKOKY.=Y(/J+1/)-Y(/J/)..
42    SKOKZ.=SQRT(SKOKX"POWER"2+SKOKY"POWER"2)..
43    SKOKT .= SKOKZ*3600/V(/J/)..
```

```
44    DELTT .= TQU.;
45    NDELT .= ENTIER(SKOKT/DELTT).;
46    "IF" NDELT "EQUAL" 0 "THEN" NDELT.=1.;
48    DELTX .= SKOKX/NDELT.;
49    DELTY .= SKOKY/NDELT.;
50    "FOR" KDELT.=0 "STEP" 1 "UNTIL" NDELT "DO"
51    "BEGIN"
51    RX.=X(/J/)+DELTX*KDELT.;
52    RY.=Y(/J/)+DELTY*KDELT.;
53     HOLD (DELTT).;
54    TUJESTEM(2,TIME,RX,RY).INTO(DLARADARU).;
55    "END".;
58    "IF" J "EQUAL" ILMAN-1 "THEN"
59    TUJESTEM(3,TIME,X(/ILMAN/),Y(/ILMAN/)).INTO(DLASZEFA).;
60    "END" PRZELOTODC.;

63    "PROCEDURE" PRZYGOTLOT.;

64    "BEGIN"
64    "REF"(WIADOMOSC)WLIN.;
65    NRREJSU.= ININT.;
66    GODZODLOTU .= ININT.;
67    MINUODLOTU .= ININT.;
68    ILMAN .= ININT.;
69    "FOR" J.=1 "STEP"1 "UNTIL" ILMAN "DO"
70    "BEGIN"
70    X(/J/).=INREAL.;Y(/J/).=INREAL.;V(/J/).=INPEAL.;
73    "END".;
76    PSSAM.=GODZODLOTU*3600+MINUODLOTU*60.;
77    "COMMENT" PLANOWANY START SAMOLOTU.;
78    WLIN .- "NEW" WIADOMOSC(3).;
79    WLIN.NUMER.=MISJA.;
80    WLIN.INNYNUMER.= NRREJSU.;
81    WLIN.CONTS(/1/) .= GODZODLOTU.;
82    WLIN.CONTS(/2/) .= MINUODLOTU.;
83    WLIN.CONTS(/3/) .= PSSAM.;
84    WLIN.KOD .= 1.;
85    HOLD(ILMAN*WDLY1).;
86    WLIN.INTO(DLASZEFA)
86    "END" PRZYGOTLOT.;

89    PRZYGOTLOT.;
90    "REACTIVATE" "THIS" SAMOLOT "AT" PSSAM.;
91    "FOR"J.=1 "STEP"1 "UNTIL" ILMAN-1 "DO" PRZELOTODC.;
93    K.=K+1.;
94    "IF" K "EQUAL" ILSAM "THEN"
95    "BEGIN"
95    HOLD(80).;
96    TERM.= "TRUE".;
97    "END".;
100   PASSIVATE.;
101   "END" SAMOLOT.;
```

```
104   PROCESS "CLASS" CZLOWIEK.;
105   "BEGIN"
105   "INTEGER" I, J.;
106   "REF" (SAMOLOT) WPRO.;
107   HOLD (ROZRUCH).;
108   "FOR" I.=1 "STEP" 1 "UNTIL" ILSAM "DO"
109   "BEGIN"
109   WPRO.- "NEW" SAMOLOT.;
110   WPRO.MISJA.=I.;
111   "ACTIVATE" WPRO.;
112   HOLD (CYKLMISJI).;
113   "END".;
116   OUTTEXT(?ZAKONCZONO POWOLYWANIE MISJI,CZAS AKTUALNY = ?).;
117   OUTREAL(TIME,7,14).;OUTIMAGE.;
119   "FOR" J.=J "WHILE" "NOT" TERM "DO"
120   "BEGIN"
120   OUTTEXT(?KOLEJKA DO SZEFA O CZASIE..?).;
121   OUTREAL(TIME,7,14).;OUTIMAGE.;
123   DRUKUJKOL(DLASZEFA).;
124   HOLD(CYKLOBSER).;
125   "END".;
128   "END" CZLOWIEK.;

131   "PROCEDURE" DRUKUJKOL( KTORA).; "REF"( HEAD) KTORA.;
133   "BEGIN"
133   "INTEGER" ILOSC,J,ARRDIM,K.;
134   "REF"(WIADOMOSC)WLIN.;
135   ILOSC.= KTORA.CARDINAL.;
136   "IF" ILOSC "EQUAL" 0 "THEN"
137   "BEGIN"
137   OUTTEXT(?PUSTA?).;
138   OUTIMAGE.;
139   "GO TO" PUSTA.;
140   "END".;
143   "FOR" J.=1 "STEP" 1 "UNTIL" ILOSC "DO"
144   "BEGIN"
144   WLIN.-KTORA.FIRST.;
145   ARRDIM.=WLIN.DL.;
146   OUTTEXT(?   NUMER MISJI NUMER REJSU  KOD ?).;
147   OUTIMAGE.;
148   OUTINT(WLIN.NUMER,10).;
149   OUTINT(WLIN.INNYNUMER,10).;
150   OUTINT(WLIN.KOD,10).;
151   OUTIMAGE.;
152   "FOR" K.=1 "STEP" 1 "UNTIL" ARRDIM "DO"
153   "BEGIN"
153   OUTREAL(WLIN.CONTS(/K/),7,14).;
154   OUTIMAGE.;
155   "END".;
158   WLIN.OUT.;
159   "END".;
162   PUSTA .. .;
163   "END" DRUKUJKOL.;
```

```
166   PROCESS"CLASS"RADAR.;
167   "BEGIN"
167   "REAL" PI,FIC,FIM,AKTFI,AKTRA.;
168   "INTEGER" BIAZYM,J..
169   "REAL""PROCEDURE" ZNAKTFI(X,Y).;
170   "REAL" X,Y.;
171   "BEGIN"
171   "IF" X "GREATER" 0 "THEN" ZNAKTFI.=0.5*PI-ARCTAN(Y/X).;
173   "IF" X "LESS" 0 "THEN" ZNAKTFI.=1.5*PI-ARCTAN(Y/X).;
175   "IF" X "EQUAL" 0 "AND" Y "NOT LESS" 0 "THEN" ZNAKTFI.=0.;
177   "IF" X "EQUAL" 0 "AND" Y "LESS" 0 "THEN" ZNAKTFI.=PI.;
179   "END" ZNAKTFI.;
182   "REAL""PROCEDURE" ZNAKTRA(X,Y).;
183   "REAL" X,Y.;
184   "BEGIN"
184   ZNAKTRA.=SQRT(X*X+Y*Y).;
185   "END" ZNAKTRA.;
188   "PROCEDURE" TAKEFROQUE(FROM).;
189   "REF"(HEAD)FROM.;
190   "BEGIN"
190   "INTEGER" KOLEJNY.;
191   "INTEGER"LOS.;
192   "REF"(WIADOMOSC)WLINK.;
193   LOS.=FROM.CARDINAL.;
194   "IF"LOS"EQUAL"0"THEN""GO TO"DEYNA.;
196   "FOR"KOLEJNY.=1"STEP"1"UNTIL"LOS"DO"
197   "BEGIN"
197   WLINK.-FROM.FIRST.;
198   AKTFI.=ZNAKTFI(WLINK.CONTS(/2/),WLINK.CONTS(/3/)).;
199   AKTRA.=ZNAKTRA(WLINK.CONTS(/2/),WLINK.CONTS(/3/)).;
200   "IF" AKTFI"NOT LESS"FIM"AND"AKTFI"LESS"FIM+FIC"AND"AKTRA
200   "LESS"RADMAX
200   "THEN"
201   WLINK.INTO(ROBRAD).;
202   "END".;
205   DEYNA.. .;
206   "END" TAKEFROQUE.;
209   PI.=4*ARCTAN(1.0).;
210   FIC.=(2*PI/TOBRAN)*TQU.;
211   "COMMENT" TQU..STALA KATOWA.;
212   "FOR" J.=J "WHILE" "NOT" TERM "DO"
213   "BEGIN"
213   "FOR" BIAZYM.=0 "STEP"1 "UNTIL" ILAZYM-1 "DO"
214   "BEGIN"
214   FIM.=FIC*BIAZYM.;
215   TAKEFROQUE(OLARADARU).;
216   HOLD(TOBRAN/ILAZYM).;
217   "END".;
220   OUTTEXT(?KOLEJKA ROBRAD O CZASIE..?).;
221   OUTREAL(TIME,7,14).;
222   OUTIMAGE.;
223   DRUKUJKOL(ROBRAD).;
224   "END".;
227   "END" RADAR.;
```

```
230     ILSAM.=ININT.;
231     TOU.=ININT.;
232     WDLY1.=ININT.;
233     ROZRUCH.=ININT.;
234     CYKLMISJI.= ININT.;
235     CYKLOBSER.=ININT.;
236     ILAZYM.=ININT.;
237     TOBRAN.=ININT.;
238     RADMAX.=ININT.;
239     K.=0.;
240     DLASZEFA.- "NEW"HEAD.;
241     DLARADARU.-"NEW"HEAD.;
242     ROBRAD.-"NEW" HEAD.;
243     SZEF.-"NEW"CZLOWIEK.;
244     "ACTIVATE"SZEF.;
245     RAD.-"NEW"RADAR.;
246     "ACTIVATE"RAD.;
247     HOLD(100).;
248     BAK.. "IF" "NOT" TERM "THEN"
249     "BEGIN"
249     HOLD(100).;
250     "GOTO" BAK.;
251     "END".;
254     "END"SIMULATION.;
254             "EOP"
254             FINIS
```

ZAŁĄCZNIK 2

KOLEJKA DO SZEFA O CZASIE..224.

NUMER MISJI	NUMER REJSU	KOD	
1	13	1	a
****	****	****	
0.000	1.000	60.000	
* * * * * * * * * *			
NUMER MISJI	NUMER REJSU	KOD	
1	13	2	b
****	****	****	
60.000	0.000	0.000	
* * * * * * * * * *			
NUMER MISJI	NUMER REJSU	KOD	
2	14	1	c
****	****	****	
0.000	3.000	180.000	
* * * * * * * * * *			
NUMER MISJI	NUMER REJSU	KOD	
3	15	1	d
****	****	****	
0.000	6.000	360.000	
* * * * * * * * * *			
NUMER MISJI	NUMER REJSU	KOD	
2	14	2	e
****	****	****	
180.000	100.000	100.000	
* * * * * * * * * *			

KOLEJKA ROBRAD O CZASIE..234.

NUMER MISJI	NUMER REJSU	KOD
2	14	2
****	****	****
204.000	93.333	100.000
* * * * * * * * * *		
NUMER MISJI	NUMER REJSU	KOD
1	13	2
****	****	****
204.000	0.000	68.000
* * * * * * * * * *		
NUMER MISJI	NUMER REJSU	KOD
2	14	2
****	****	****
212.000	90.000	100.000
* * * * * * * * * *		
NUMER MISJI	NUMER REJSU	KOD
1	13	2
****	****	****
212.000	0.000	72.000
* * * * * * * * * *		

ZAŁĄCZNIK 3

```
NUMER MISJI NUMER REJSU  KOD
     1          13        2
   ****       ****      ****
  244.000     0.000    88.000
*  *  *  *  *  *  *  *  *  *

NUMER MISJI NUMER REJSU  KOD
     2          14        2
   ****       ****      ****
  252.000    73.333   100.000
*  *  *  *  *  *  *  *  *  *

NUMER MISJI NUMER REJSU  KOD
     1          13        2
   ****       ****      ****
  252.000     0.000    92.000
*  *  *  *  *  *  *  *  *  *

KOLEJKA DO SZEFA O CZASIE..284.

NUMER MISJI NUMER REJSU  KOD
     1          13        2
   ****       ****      ****
  268.000     0.000   100.000
*  *  *  *  *  *  *  *  *  *

KOLEJKA  ROBRAD O CZASIE..288.

NUMER MISJI NUMER REJSU  KOD
     2          14        2
   ****       ****      ****
  260.000    70.000   100.000
*  *  *  *  *  *  *  *  *  *

NUMER MISJI NUMER REJSU  KOD
     1          13        2
   ****       ****      ****
  260.000     0.000    96.000
*  *  *  *  *  *  *  *  *  *

NUMER MISJI NUMER REJSU  KOD
     2          14        2
   ****       ****      ****
  268.000    66.667   100.000
*  *  *  *  *  *  *  *  *  *

NUMER MISJI NUMER REJSU  KOD
     1          13        2
   ****       ****      ****
  268.000     0.000   100.000
*  *  *  *  *  *  *  *  *  *
```

ZAŁĄCZNIK 4

KOLEJKA ROBRAD O CZASIE..576.

NUMER MISJI	NUMER REJSU	KOD
3	15	2
****	****	****
544.000	70.968	70.968
* * * * * * * * * *		

NUMER MISJI	NUMER REJSU	KOD
2	14	2
****	****	****
548.000	0.000	68.889
* * * * * * * * * *		

NUMER MISJI	NUMER REJSU	KOD
1	13	2
****	****	****
548.000	91.892	100.000
* * * * * * * * * *		

NUMER MISJI	NUMER REJSU	KOD
3	15	2
****	****	****
552.000	74.194	74.194
* * * * * * * * * *		

NUMER MISJI	NUMER REJSU	KOD
2	14	2
****	****	****
556.000	0.000	66.667
* * * * * * * * * *		

NUMER MISJI	NUMER REJSU	KOD
1	13	2
****	****	****
556.000	94.595	100.000
* * * * * * * * * *		

KOLEJKA DO SZEFA O CZASIE..584.

NUMER MISJI	NUMER REJSU	KOD
1	13	3
****	****	****
572.000	100.000	100.000
* * * * * * * * * *		

KOLEJKA ROBRAD O CZASIE..594.

NUMER MISJI	NUMER REJSU	KOD
3	15	2
****	****	****
560.000	77.419	77.419
* * * * * * * * * *		

НЕКОТОРЫЕ ПРОБЛЕМЫ МОДЕЛИРОВАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ ВОЗДУШНЫМ ДВИЖЕНИЕМ. ПРЕДВАРИТЕЛЬНЫЕ РЕЗУЛЬТАТЫ

Резюме

В статье описана модель Системы Управления Воздушным Движением. В модели указаны 3 элемента: воздушное движение, радиолокационная информационная система и диспетчер. Модель описана на языке программирования SIMULA-67. Этим элементам приписаны классы процессов (Process Class) соответственно: Samolot, Radar, Człowiek моделированные в параллельном действии. N самолета движется в двумерном пространстве по заданным планам полётов, и это представляет воздушное движение. Диспетчер получает два вида информации: от экипажа и радиолокационную по обнаруженных объектах. Обсуждены результаты моделирования и указана пригодность языка SIMULA-67 для решания сложных систем.

SOME ASPECTS OF SIMULATION OF ATC SYSTEM. PRELIMINARY RESULTS

Summary

In this paper a model of Air Traffic Control System (ATCS) is presented. The model describes: air traffic as controlled complex of aircrafts, radar information system and ATCS controller. The model is presented in SIMULA-67 programming language. Mentioned elements of the model have their equivalents in the form of Process CLASS: Samolot, Radar and Człowiek respectively. Their concurrent activities were simulated. Air Traffic is formed by a set of n aircrafts flying in 2-dimensional space according to the Flight Plans. The controller receives two kinds of information on aircrafts: crew messages and radar information on identified aircrafts. The results of simulation are discussed and advantages of SIMULA-67 for description of large scale systems are pointed out.

Prace IMM
Zeszyt 2
© 1975.04
681.327.2:534.78

OGÓLNA KONCEPCJA SYNTEZY FONEMATYCZNEJ (BEZTRANSIENTOWEJ)

Ryszard PATRYN

Pracę złożono 15.10.1974

Artykuł zawiera opis metody realizacji doświadczalnego syntezatora fonematycznego, w którym nie stosowano stanów przejściowych na granicy sąsiednich głosek. Do syntezy wykorzystano 40 głosek, których sygnały przechowywano w pamięci cyfrowej. Z tych elementów uzyskuje się podczas syntezy dowolną wypowiedź w języku polskim. Pozytywne rezultaty potwierdzają słuszność postawionej hipotezy o możliwości eliminacji stanów przejściowych w procesie syntezy fonematycznej.

1. WSTĘP

W ostatnich latach obserwuje się zainteresowanie fonicznymi urządzeniami wyjściowymi maszyn cyfrowych. Większość budowanych urządzeń zawiera syntezator, którego działanie oparte jest na zasadzie pracy wokodera, który wynaleziono z myślą wykorzystania w telekomunikacji ze względu na możliwość znacznej kompresji przesłanego sygnału. W telekomunikacji szerszego zastosowania wokodery nie znalazły prawdopodobnie z powodu niezbyt dobrej wierności odtwarzania i skomplikowanej budowy.

Doświadczenia uzyskane w pracach nad wokoderami wykorzystywane są obecnie przy budowie fonicznych urządzeń wyjściowych komputerów. W urządzeniach tego typu elementami składowymi mowy syntetycznej są zazwyczaj wyrazy lub sylaby. Sterowanie procesem syntezy jest dosyć skomplikowane ze względu na stosunkowo dużą liczbę zmiennych. Bardzo prosto przedstawia się natomiast metoda

tworzenia mowy syntetycznej z poszczególnych fonemów, których liczba w polskim języku wynosi 40, a proces syntezy jest analogiczny do tworzenia tekstu pisanego z liter.

Dotychczas panował pogląd, zwłaszcza wśród badaczy zachodnich, że realizacja syntezy fonematycznej jest niemożliwa ze względu na zniekształcenia, które powstaną na granicy dwóch fonemów w przypadku braku stanów przejściowych (transientów) [1 str. 461, 464]. Pogląd taki jest oparty na analizie naturalnego sygnału mowy, w którym występują wyraźne stany przejściowe między dwoma sąsiednimi głoskami, czyli na granicy dwóch głosek.

Wynika to z dosyć dużej stałej czasowej układów ludzkiego toru głosowego.

W sygnale mowy naturalnej zmiany częstotliwości formantów mają charakter funkcji ciągłych. Przy założeniu wytwarzania mowy syntetycznej zbliżonej do mowy naturalnej właściwości, o których wspominano wyżej, mogą się wydawać nieodzownymi do odtworzenia w procesie syntezy.

W dokonywanych dotychczas próbach syntezy fonematycznej przestrzegano zasady ciągłego przejścia sygnału jednej głoski w drugą lub wytwarzano stany przejściowe według odpowiednio opracowanych zasad [3, 4].

Tworzenie sygnału mowy syntetycznej z samych fonemów z założenia eliminuje możliwość zachowania ciągłości funkcji częstotliwości formantów w czasie w obrębie sylab lub wyrazów.

W Instytucie Maszyn Matematycznych w ramach badania możliwości wyposażenia maszyny cyfrowej w foniczne urządzenie wyjściowe wykonano pracę badawczo-doświadczalną, której tematem było sprawdzenie hipotezy o możliwości eliminacji stanów przejściowych w syntezie fonematycznej.

Taki system charakteryzuje się bardzo prostym sposobem sterowania procesem syntezy, co powinno ułatwić wprowadzenie fonicznych urządzeń wyjściowych do systemów komputerowych.

2. CEL BADAŃ

Celem podjętej pracy było wytworzenie dostatecznie zrozumiałej mowy syntetycznej, która byłaby wykorzystywana w krótkich tekstach, w wypowiedziach używanych przy komunikacji maszyna-człowiek, głównie pod kątem sprawdzenia możliwości eliminacji stanów przejściowych oraz podjęcia prób ustalenia zasad wytwarzania mowy syntetycznej z kilkudziesięciu składowych głosek.

Pozytywny rezultat badań umożliwi ustalenie założeń do dalszych prac, uzasadni wybór najdogodniejszej metody syntezy w konkretnych warunkach oraz dostarczy dodatkowych danych z zakresu badania percepcji mowy.

Rezultatem tych prac jest eksperymentalny syntezator fonematyczny, którego zasada pracy będzie omówiona w niniejszej pracy.

3. ZAŁOŻENIA DO DOKONANIA EKSPERYMENTU

Czas trwania stanów przejściowych nie przekracza 30 - 40 ms. Z badań ludzkiego układu percepcyjnego wynika, że krótkotrwające sygnały nie są dokładnie rozpoznawane, górna granica czasu niezbędnego do odczucia wysokości tonu o częstotliwości 300 Hz wynosi 35 ms, a 1000 Hz-22 ms [2 str. 255].

Niezdolność ludzkiego układu percepcyjnego do dokładnego rozpoznania krótkich sygnałów może upoważniać do podważenia aksjomatu o niezbędności stanów przejściowych.

Do podobnych wniosków można dojść również, rozpatrując model percepcji sygnału mowy polegający na porównaniu odbieranych sygnałów z zapamiętanymi wzorcami, które zawierają cechy charakterystyczne wspólne dla wszystkich sygnałów wytwarzanych przez różnych osobników. Wydaje się, że te cechy nie mogą być umiejscowione w stanach przejściowych między sąsiednimi głoskami.

Rozpatrując poza tym zagadnienie reakcji człowieka na podniety nie powinno się zapominać o niepełnej odpowiedniości podniet i odnoszonego wrażenia. Przy odbiorze sygnałów akustycznych nie

potrafimy porównywać tak precyzyjnie wrażeń różnych osobników jak można to uczynić w przypadku np. wrażeń wzrokowych przy obserwacji prostych figur geometrycznych.

Przedstawione wyżej przesłanki stanowiły podstawę do przeprowadzenia eksperymentu, uzasadniały celowość takiego badania. Dodatkowym argumentem przemawiającym na korzyść założonej metody były prace autora przeprowadzone w 1968 r. dotyczące syntezy fonematycznej, której elementami były samogłoski i wszystkie spółgłoski bezdźwięczne języka polskiego [5].

4. PODSTAWOWE ZAŁOŻENIA

Uwzględniając określony poprzednio cel eksperymentu ustalono następujące wymagania:

- do syntezy zostaną użyte głoski wyodrębnione z mowy naturalnej
- głoski dźwięczne powinny charakteryzować się jednakową wartością częstotliwości tonu krtaniowego
- głoski powinny być tak dobrane, aby mogły być stosowane w dowolnych zestawieniach (występujących w języku polskim) z innymi głoskami
- układ syntezatora powinien zapewnić łączenie głosek bez wprowadzania zakłóceń i sygnałów pasożytniczych
- sygnały głosek nie powinny być zauważalnie zniekształcone przez stosowane przetwarzania analogowo-cyfrowe i cyfrowo-analogowe
- sygnały głosek powinny być przechowywane w pamięci w postaci cyfrowej

5. PROBLEMY ZWIĄZANE Z REALIZACJĄ SYNTEZATORA

5.1. Badanie przebiegów czasowych sygnału mowy

Przyjęcie odmiennego od powszechnie stosowanego sposobu syntezy wymagało innego badania sygnału mowy. Zamiast badania wid-

ma sygnału zastosowano badanie (obserwację) przebiegów czasowych sygnału mowy.

Ponieważ nie dysponowano w momencie przystępowania do badań potrzebną aparaturą, opracowano i wykonano układ pomiarowy do obserwacji przebiegów akustycznych na oscyloskopie. (Opis układu będzie przedmiotem oddzielnej publikacji).

Metoda obserwacji przebiegu okazała się bardzo dogodna przy badaniu przebiegów głosek, określaniu początku i końca głoski, pomiarze częstotliwości tonu krtaniowego, amplitudy itp.

5.2. Wybieranie i łączenie głosek

Ze względu na łatwość operowania sygnałem przyjęto zapis cyfrowy głosek i w związku z tym pamięć cyfrową.

Z uwagi na sterowanie i mały czas dostępu wygodna byłaby pamięć rdzeniowa (pamięć typu operacyjnego), jednak zdecydowano się na zastosowanie pamięci dyskowej, która charakteryzuje się wprawdzie dużym czasem dostępu (rzędu kilkudziesięciu ms), ale tę niedogodność usunięto przez zastosowanie w układzie pamięci buforowej, natomiast uzyskano dużą pojemność, która będzie wykorzystana w dalszych badaniach, dając możliwość operowania w syntezatorze wieloma zestawami głosek.

W celu przekształcenia sygnału analogowego na cyfrowy zastosowano modulację kodowo impulsową. Przyjęto słowo maszynowe 8-bitowe, co odpowiada dynamice sygnału ca 40 dB. Częstotliwość próbkowania określono doświadczalnie badając słuchowo sygnał mowy zapisany na taśmie magnetofonowej i poddany przetworzeniu analogowo-cyfrowemu i cyfrowo-analogowemu przy różnych częstotliwościach próbkowania. Zaobserwowano przy częstotliwościach poniżej 15 kHz zniekształcenia podobne do seplenienia w fonemach S, C, Z, Dz. Przy zwiększaniu częstotliwości próbkowania powyżej 15 kHz nie zaobserwowano wyraźnej poprawy jakości odtwarzania.

5.3. Dobór głosek

W celu uzyskania materiału głosowego, z którego można wybrać odpowiadające wymaganiom syntezy głoski, ustalono zestaw sylab możliwie krótki, który zawierał wymagany repertuar głosek. Ten zestaw ulegał podczas realizacji eksperymentu modyfikacjom, które wynikały z doświadczeń uzyskiwanych w czasie prób.

Zestaw sylab, z którego wybierano głoski przedstawia się następująco: na ne no mu my i bda dba gdy wda joł łój rel że zgie niń czan dżem dze dzik są sęk zima roh pot tap szac fok siś cieć kieł.

Głoski dźwięczne umieszczono w początkowej części tekstu aby ułatwić lektorce utrzymanie stałej wysokości tonu krtaniowego, z tego samego powodu głoski m i n umieszczono bezpośrednio przy samogłoskach.

Tekst był czytany przez lektorkę (niezawodową) monotonnie bez intonacji, aby uzyskać jednakową częstotliwość tonu krtaniowego. Wymagało to nawet pewnego treningu, szczególnie w przypadku ponownego nagrywania, aby trafić na poprzednią wysokość tonu krtaniowego. Po uzyskaniu nagrania na taśmie magnetofonowej następowało wybieranie odpowiednich fragmentów oraz ustalanie podstawowych parametrów: czasu trwania, amplitudy, częstotliwości tonu krtaniowego.

W przypadku głosek, których przebieg zawierał fragment o charakterze ustalonym lub zbliżonym do ustalonego wybierano taki fragment, głoski poprzedzane przerwą zapisywano zachowując bez zmian początkową część przebiegu. Do sygnału głoski włączano również początkową przerwę, co przyczyniło się do ułatwienia sterowania w procesie syntezy. Wszystkie głoski, które nie występowały z przerwą początkową poddawano modyfikacji polegającej na kształtowaniu przebiegu w jego fazie początkowej w taki sposób, aby sygnał narastał od wartości zerowej do ustalonej w czasie około 20 ms.

W przypadku odchyłki częstotliwości tonu krtaniowego od przyjętej wartości korygowano ją przez zmianę częstotliwości próbko-

wania (przeciwną do zmierzonej) podczas przetwarzania analogowo-cyfrowego.

Po wybraniu właściwego fragmentu głoski, zapisywano go w pamięci dyskowej i przeprowadzano próby łączenia z głoskami uprzednio zapisanymi. Podczas prób okazało się, że małe różnice częstotliwości tonu krtaniowego rzędu 2-3% nie powodują zauważalnych zakłóceń, różnice rzędu 4-10% dają się skompensować przez zmianę częstotliwości próbkowania lub przez zmianę prędkości odczytu magnetofonu, ale taki zabieg udawał się tylko w przypadku niektórych głosek.

Głoski n i m musiały zostać zapisane bez jakichkolwiek modyfikacji częstotliwościowych, żadne korekcje nie dawały pożądanych rezultatów.

W tabeli 1 podano zestawienie głosek użytych w syntezatorze, w ostatniej kolumnie umieszczono sylabę, z której wyizolowano głoskę.

Tabela 1

Zestawienie głosek

Lp.	głoska	czas trwania ms	amplituda względna	sylaba
1	a	120 + 120	1,8	na
2	b	45	1	bda
3	b (końcowe)	85	1	bda
4	c	85	1,1	szac
5	cz	62	1	czan
6	d	60	1,3	dba
7	e	120 + 120	1,1	ne
8	f	145	0,3	fok
9	g	42	1	gdy
10	h	180	0,6	roh
11	i	120 + 120	0,8	i
12	j	120	1	joł
13	j (końcowe)	180	0,8	łój
14	k	60	1	sęk
15	l	105	1,3	rel

Tabela 1 o.d.

Lp.	głoska	czas trwania	amplituda względna	sylaba
16	m	120	0,9	mu
17	n	120	0,8	na
18	n (końcowe)	205	1,1	czan
19	o	120 + 120	1,8	no
20	p	40	0,5	tap
21	r	85	1,5	rel
22	s	90	1	są
23	sz	180	1,1	szac
24	t	60	0,6	pot
25	u	120 + 120	1,1	mu
26	w	105	1,1	wda
27	y	120 + 120	1,1	my
28	z	150	1,2	zgie
29	dz	110	1,2	dze
30	ż	150	1,1	że
31	dż	60	1,5	dżem
32	ą	300	1,5	są
33	ę	325	1,4	sęk
34	ć	120	1	cieć
35	ś (końcowe)	190	1,4	siś
36	ś	210	1,4	siś
37	dź	80	1,3	dzik
38	ń	120	1,2	nić
39	ń (końcowe)	172	1,2	nić
40	ł	120	1,5	łój
41	g(miękkie)	95	1	zgie

W pierwszych próbach używano oprócz wyżej wymienionych głosek d końcowe, w końcowe, m końcowe ale okazało się, że można z nich zrezygnować.

Oznaczenia głosek w tabeli wprawdzie nie odpowiadają znakom przyjętym przez lingwistów, ale nie powinny budzić wątpliwości

o jakie fonemy chodzi w danym przypadku. Uwagi "końcowe" odnoszą się do głosek, które lepiej brzmią na końcu wyrazu lub przed pauzą występującą w początkowej fazie głosek zwartych. Podane czasy trwania złożone z dwóch składników oznaczają dwa sposoby wykorzystania samogłosek. W przypadku stosowania długiej samogłoski wybiera się cały zapisany przebieg, natomiast przy krótkiej wykorzystuje się jedynie pierwszą połowę. Używanie długich samogłosek umożliwia pewnego rodzaju akcentowanie, które w dużym stopniu polepsza jakość mowy syntetycznej.

Poprzednio wspominano o modyfikacji początkowych partii przebiegów głosek, części końcowych nie poddawano jakimkolwiek zmianom, chociaż przebieg bywał kończony w sposób zupełnie przypadkowy, wynikający z określonego z góry czasu trwania podczas zapisu w pamięci dyskowej. Wydzielenie tzw. głosek końcowych polepszało nieco jakość mowy i nie było bezwzględnie konieczne z wyjątkiem j, które pobrane ze środka wyrazu nie nadawało się do stosowania na końcu i analogicznie j pobrane z końca nie nadawało się do obustronnego łączenia z innymi głoskami.

6. UKŁAD SYNTEZATORA I JEGO DZIAŁANIE

Syntezator składa się z pamięci dyskowej CDC9469 (bez jednostki sterującej), 2 pamięci buforowych o pojemności umożliwiającej zapamiętanie sygnału akustycznego o czasie trwania 120 ms każda, specjalizowanej jednostki sterującej, minikomputera MERA 302, przetwornika cyfrowo-analogowego i wzmacniacza akustycznego. Do układów opracowanych w ramach budowy syntezatora należą również zespoły umożliwiające zapisywanie sygnałów głosek w pamięci dyskowej.

Proces syntezy przebiega następująco: do pamięci operacyjnej komputera wprowadza się tekst (pisany fonetycznie). Po wprowadzeniu tekstu uruchamia się program wybierający, który powoduje wybieranie zgodnie z tekstem z pamięci dyskowej sygnałów głosek, które są wprowadzane do pamięci buforowych. Pobranie następnej głoski z pamięci dyskowej następuje po opróżnieniu jednej z pamięci buforowych.

Pobieranie sygnału z pamięci dyskowej odbywa się z prędkością ca 150 kbajtów/s, natomiast wyprowadzanie sygnału z pamięci buforowych do przetwornika cyfrowo-analogowego odbywa się z prędkością 15 kbajtów/s. Wprowadzanie tekstu do pamięci operacyjnej odbywa się za pomocą maszyny do pisania (z zestawu minikomputora) lub z czytnika taśmy dziurkowanej.

Prosty sposób redagowania wypowiedzi i natychmiastowa możliwość wysłuchania jej w dużym stopniu ułatwiały wprowadzanie zmian i uzyskanie właściwych rezultatów w krótkim czasie. Zresztą ta łatwość redagowania (składania) wypowiedzi jest charakterystyczna dla syntezatora omawianego typu.

7. WSTĘPNA OCENA WYNIKÓW

Przesłuchania przeprowadzano korzystając z głośnika na wyjściu wzmacniacza akustycznego oraz z aparatu telefonicznego.

Próbki mowy syntetycznej demonstrowano ponad 100 osobom, z których żadna nie zgłaszała zastrzeżeń co do zrozumiałości, teksty (zresztą krótkie) były odbierane prawidłowo. Część słuchaczy narzekała na monotonny charakter mowy. Te przesłuchania traktowano jako wstępną ocenę, ale ocenę upoważniającą do traktowania rezultatu pozytywnie. Badając reakcję słuchaczy na różnie przygotowane wypowiedzi zauważono, że lepiej jest odbierany tekst zbudowany analogicznie do pisanego, w którym zachowuje się przerwy między wyrazami, pomijane w naturalnym sygnale mowy, lecz z wprowadzeniem akcentu (wydłużanie samogłosek). Mowa staje się przez to wolniejsza i łatwiej zrozumiała, gdyż nie wymaga dużego natężenia uwagi słuchacza.

Zasadniczym efektem dokonanych prób jest potwierdzenie możliwości eliminacji stanów przejściowych między fonemami w sygnale mowy syntetycznej. Zmienia to pogląd nie tylko na istotę syntezy fonematycznej ale również może mieć wpływ na lepsze poznanie właściwości percepcyjnych człowieka w odniesieniu do sygnału mowy.

W odniesieniu do syntezy mowy rezultaty badań otwierają nowe możliwości nie tylko w zakresie syntezy fonematycznej, ale mogą przyczynić się do rozszerzenia badań w innych systemach, np. w syntezie parametrycznej. Ponieważ opisywane badania były jednymi z pierwszych (nie spotkano w literaturze opisu podobnych badań) nie można twierdzić, że zbadano wszystkie aspekty problemu eliminacji stanów przejściowych, tym bardziej, że próby ograniczono do języka polskiego.

Doraźne korzyści wynikające z przeprowadzonego eksperymentu polegają na stworzeniu możliwości budowy użytkowo sprawnego fonicznego urządzenia wyjściowego.

Literatura

[1] SAPOŻKOW M.A.: Sygnał mowy w telekomunikacji i cybernetyce, WNT, 1966, s. 454, 461, 465, 467

[2] ŻYSZKOWSKI Z.: Podstawy elektroakustyki, WNT, 1965, s. 325

[3] VIELIČKO V.G.: Multiplikativnyj metod foniemnogo sinteza reči s pomoščju ECBM. Seminar - Rozpoznavanije obrazov i konstruirovanije čitajuščich avtomatov - Kijów 1967

[4] SAKAI T., OHTANI K., TOMITA S.: On-line Real Time Multiple-Speech Output Systems, IFIP Congress, 1971

[5] PATRYN R.: Koncepcja układu fonicznego wyjścia maszyny cyfrowej, ETO Nowości, 1969, nr 2

ОБЩАЯ КОНЦЕПЦИЯ ФОНОМАТИЧЕСКОГО (НЕПЕРЕМЕННОГО) СИНТЕЗА

Резюме

В статье подано описание метода выполнения экспериментального фономатического синтезатора, в котором неприменено переходных состояний на стыковке фонем. Для синтеза использовано 40 фонем, сигналы которых были хранены в цифровой памяти. Во время синтеза из этих элементов получается произвольное высказания на польском языке. Положительные результаты подтверждают правильность выдвинутой гипотезы о возможности элиминации переходных состояний в процессе фономатического синтеза.

GENERAL CONCEPT OF PHONEMATIC SYNTHESIS (NOT-TRANSIENT)

Summary

The article contains a description of a method of an experimental phonometric synthesizer realization, transient states on the border of neighbouring phonems not being applied 40 phonems were used for the synthesis, their signals being stored in digital memory. Any utterance in Polish can be obtained out of the above elements during the synthesis. Positive results prove the hypothesis rightness of the possibility to eliminate transients in phonematic synthesis process.

Prace IMM
Zeszyt 2
© 1975. 04
681.322.06:681.327:
518.43:513.21

WYBRANE PROCEDURY GRAFICZNE W SYSTEMACH KOMPUTEROWYCH

Andrzej ZIEMKIEWICZ

Pracę złożono 17.09.1974

Zanalizowano stosowane metody zobrazowania okręgu za pomocą graficznych urządzeń wyjściowych. Zaproponowano proste w realizacji i dokładne algorytmy aproksymacji okręgu i odcinka oraz zbadano ich własności. Wykazano stabilność tych algorytmów i ich przydatność dla urządzeń różnych typów.

1. WSTĘP

Graficzne urządzenia wyjściowe wyposażone w generatory krzywych należą do rzadkości. Zazwyczaj okręgi i inne krzywe aproksymowane są linią łamaną lub ciągiem punktów wyznaczanych przez procesor i dopiero w tej postaci mogą być wyprowadzone przez urządzenie zewnętrzne, np. mogą być wyświetlone na ekranie grafoskopu bądź narysowane za pomocą plottera. Na sposób aproksymacji w dużej mierze wpływają ograniczenia narzucone własnościami zastosowanego urządzenia wyjściowego.

Każde urządzenie graficzne w swoim polu roboczym, stanowiącym wycinek płaszczyzny, zawiera pewną liczbę punktów adresowalnych, tworzących kwadratową siatkę.

Pierwszą grupę urządzeń stanowią urządzenia wyposażone w interpolator liniowy. W tym przypadku przez procesor zadawane są tylko wierzchołki linii łamanej, które muszą być punktami należącymi do siatki urządzenia, natomiast odcinki pomiędzy wierz-

chołkami interpolowane są przez urządzenie automatycznie. W zależności od budowy urządzenia interpolacja ta może być dokonywana w sposób analogowy lub cyfrowy.

Do drugiej grupy należą urządzenia, w których jedynym możliwym ruchem z dowolnego punktu siatki jest przesunięcie do jednego z ośmiu najbliższych punktów otoczenia.

Natomiast trzecią grupę stanowią urządzenia najprostsze, w których jedyna możliwość zobrazowania linii polega na składaniu jej z pojedynczych, indywidualnie adresowanych punktów. W wyniku niewielkich zabiegów programowych urządzenia te można zaliczyć do drugiej grupy urządzeń.

Procedury służące do zobrazowania linii za pomocą urządzeń nie zawierających interpolatorów są na ogół bardziej czasochłonne, w tym przypadku bowiem zachodzi konieczność aproksymacji nie tylko linii drugiego stopnia, ale także odcinków prostej. Z drugiej strony jednak, algorytmy opracowane dla tych urządzeń cechuje większa ogólność, ponieważ pozostają ważne także dla pozostałych typów urządzeń. Z tego względu zasługują one na szczególną uwagę.

Przedmiotem niniejszego opracowania są zagadnienia związane ze zobrazowaniem okręgu i odcinka, czyli elementów niezbędnych w graficznych systemach komputerowych, a równocześnie wystarczających dla większości zastosowań. Większa część pracy (rozdziały 1 do 4) poświęcona jest aproksymacji okręgu. W rozdziale 2 naszkicowane zostały niektóre znane metody i omówione ich podstawowe własności. W rozdziale 3 przedstawiono podstawy algorytmu opracowanego przez autora. W rozwiązaniu tym usunięte zostały wady występujące w innych metodach, a równocześnie nadaje się ono do zastosowania we wszystkich typach urządzeń. Rozdział 4 zawiera usprawnienia tego algorytmu i przystosowanie go do maszynowego przetwarzania. Rozdział 5 dotyczy aproksymacji odcinka, zawiera również uwagi na temat układowej realizacji proponowanych algorytmów aproksymacji okręgu i odcinka w jednostkach sterujących urządzeniami graficznymi.

2. STOSOWANE METODY

Najpełniejszy przegląd metod zobrazowania krzywych w graficznych systemach komputerowych wraz z obszernym wykazem literatury podany został w [1]. We wszystkich rozwiązaniach omawianych w tej pracy aproksymacja okręgu polega na wyznaczeniu pewnej liczby punktów leżących na okręgu, zwanych punktami charakterystycznymi, a następnie zbudowaniu wielokąta wpisanego o wierzchołkach w wyznaczonych punktach. Poszczególne metody różnią się sposobem wyznaczania punktów charakterystycznych.

Pierwsza metoda polega na wykorzystaniu równań różnicowych. Zgodnie ze źródłową dla tej metody pracą [2] kolejne punkty charakterystyczne wyznaczane są na podstawie zależności:

$$\left.\begin{aligned} y_{i+1} &= x_i d + y_i \\ x_{i+1} &= x_i - y_{i+1} d \end{aligned}\right|$$

gdzie: d - parametr przyrostowy.

Ponieważ nie została przeprowadzona analiza numeryczna pozwalająca stwierdzić stabilność algorytmu i określić sposób wyznaczania przyrostu d w zależności od przyjętego dopuszczalnego błędu, o jakości tej metody można wnioskować tylko na podstawie danych doświadczalnych. Z zamieszczonych w [1] fotogramów wynika (rys. 1), że przy parametrze d = 0,003 występuje wyraźnie zauważalny błąd końcowy.

Rys. 1

Wydaje się, że zastosowanie mniejszego przyrostu powinno pozwolić na dokładniejsze wyznaczanie kolejnych punktów. Okazuje się jednak, że dla d = 0,002 błąd ten wzrasta jeszcze bardziej

i końcowy promień zmniejsza się niemal do połowy wartości początkowej. Taki błąd wielkości, która w czasie przetwarzania powinna pozostawać stała, nie jest możliwy do przyjęcia i nasuwa wątpliwości co do stabilności algorytmu.

Druga z omawianych metod (parametryczna) polega na wyznaczeniu punktów charakterystycznych z równań parametrycznych okręgu:

$$\left.\begin{aligned} x &= R \cdot \cos(2\pi i/N) \\ y &= R \cdot \sin(2\pi i/N) \end{aligned}\right| \quad (i = 0, 1, 2, \ldots, N-1)$$

gdzie: N - liczba wyznaczanych punktów

Pewną niedogodnością tej metody jest przyjmowana na ogół stała wartość N niezależnie od promienia. Powoduje to przy dużych wartościach promienia powstawanie znacznego błędu w środku każdej cięciwy, lub też nadmiernie gęste rozmieszczenie punktów przy małych okręgach. Usunięcie tej usterki jest możliwe drogą pewnej komplikacji algorytmu.

Ostatnią z omawianych w [1] jest metoda parametryczna w ujęciu macierzowym, różniąca się od zasadniczej metody parametrycznej w zasadzie tylko aparatem formalnym.

Niezależnie od szczegółów różniących poszczególne rozwiązania, większość z wymienionych metod pociąga za sobą konieczność stosowania arytmetyki zmiennoprzecinkowej, przez co są one czasochłonne. Nie stanowiło to istotnego ograniczenia, gdy systemy graficzne budowane były przy użyciu dużych i szybkich maszyn. Czynnik ten nabiera jednak znaczenia w małych systemach komputerowych, a także w systemach stosujących do współpracy z urządzeniami graficznymi małe procesory satelitarne, w których działania zmiennoprzecinkowe nie są realizowane układowo. Dotyczy to zwłaszcza metody parametrycznej, wymagającej odwołań do podprogramów obliczania wartości funkcji trygonometrycznych. Ponadto wysoka precyzja obliczeń, możliwa do uzyskania w arytmetyce zmiennoprzecinkowej, jest tu mało przydatna, bo każdy wyznaczony punkt o współrzędnych rzeczywistych musi być i tak zastąpiony najbliższym punktem rastru urządzenia (najczęściej 1024 x 1024 punkty).

Zazwyczaj przyjmowane jest założenie, nie zmniejszające ogólności rozwiązań, zgodnie z którym krok siatki urządzenia przyjmowany jest za jednostkę długości. W tej sytuacji właściwą drogą wydaje się być poszukiwanie algorytmów korzystających z arytmetyki liczb całkowitych. Algorytmy takie proponowane są w [3] i [4].

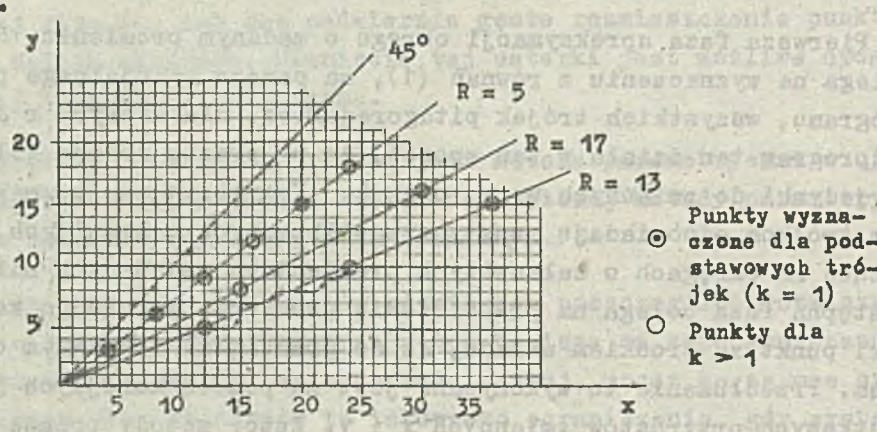
W pracy [3] przedstawiona została metoda, której celem jest wyznaczenie punktów charakterystycznych przy zastosowaniu trójek pitagorejskich. Podstawą algorytmu jest twierdzenie mówiące, że wszystkie trójki liczb (r, x, y) stanowiące rozwiązania w zbiorze liczb całkowitych równania $x^2 + y^2 = r^2$, są postaci:

$$\left.\begin{aligned} x &= (v^2 - u^2)\,k \\ y &= (2uv)\,k \\ r &= (u^2 + v^2)\,k \end{aligned}\right| \qquad (k = 1, 2, 3, \ldots) \qquad (1)$$

Pierwsza faza aproksymacji okręgu o zadanym promieniu R polega na wyznaczeniu z równań (1), za pomocą specjalnego podprogramu, wszystkich trójek pitagorejskich, dla których $r \leq R$. Podprogram ten działa w ten sposób, że po prostu dodaje kolejne jedynki do zmiennych u, v oraz k. Wszystkim tak wyznaczonym trójkom odpowiadają punkty o współrzędnych całkowitych, leżące na okręgach o całkowitych promieniach mniejszych niż R. Następna faza polega na przedłużeniu promienia łączącego każdy taki punkt ze środkiem układu, aż do przecięcia z zadanym okręgiem. Przedłużanie to wykonywane jest za pomocą kolejnych jednostkowych przyrostów zmiennych x i y. Autor metody podaje kilka strategii, których celem jest uzyskanie odpowiedniej kolejności przyrostów, zapewniającej mały błąd aproksymacji. Najlepsza z nich jest rachunkowo dosyć kosztowna. Operacja przedłużania przeprowadzana jest dla każdej z wyznaczonych uprzednio trójek (rys. 2). Po uporządkowaniu otrzymuje się zbiór punktów charakterystycznych leżących w pierwszej ćwiartce płaszczyzny. Pozostałe punkty wyznacza się korzystając z symetrii. Do niewątpliwych zalet omawianej metody należy korzystanie wyłącznie z działań na liczbach całkowitych. Poza tym, liczba wyznaczanych punktów zwiększa się automatycznie wraz ze wzro-

stem promienia. Niezależnie od oryginalności samego pomysłu algorytm ten posiada jednak kilka wad, mianowicie:

1. wyznaczone punkty rozłożone są nierównomiernie na obwodzie okręgu,
2. kolejność wyznaczonych trójek nie odpowiada uporządkowaniu punktów na obwodzie. Pociąga to za sobą konieczność pamiętania wszystkich trójek, a następnie ich sortowania, powodując nie tylko zajęcie dużego obszaru pamięci, ale także stratę czasu,
3. dla każdej trójki liczba kroków odpowiadających poruszaniu się po promieniu jest rzędu r(mod R). Ponieważ liczba wyznaczonych punktów proporcjonalna jest do promienia, zatem całkowita liczba wykonanych kroków rośnie z kwadratem promienia.



Rys. 2

Wymienione wady powodują małą sprawność algorytmu. Zgodnie z [3], implementacja maszynowa (na maszynie IBM) wykazała, że metoda ta nie była szybsza niż metoda parametryczna, wykorzystująca działania zmiennoprzecinkowe i funkcje trygonometryczne.

Niezależnie od sposobu wyznaczania punktów charakterystycznych, pewna niedoskonałość ukryta jest w samej zasadzie aproksymowania okręgu wielokątem wpisanym. Wszystkie błędy bowiem są tego samego znaku, a więc aproksymacja nie może być optymalna niezależnie od przyjętej miary błędu.

Zupełnie odmienne podejście zostało zastosowane w [4]. W pracy tej rozpatrywane jest ogólne zagadnienie aproksymacji krzywych drugiego stopnia. Koncepcję przedstawionej w niej metody można streścić następująco. Dana jest krzywa drugiego stopnia, opisana równaniem postaci:

$$F(x, y) = 0 \qquad (2)$$

gdzie F - jest wielomianem stopnia 2 zmiennych x, y.

Po zróżniczkowaniu równania (2) względem zmiennej x oraz rozwikłaniu otrzymanego wyrażenia ze względu na pochodną dy/dx otrzymujemy wyrażenie:

$$\frac{dy}{dx} = f(x, y) \qquad (3)$$

Aproksymację rozpoczyna się w dowolnym punkcie krzywej. W punkcie tym wyznaczana jest prosta:

$$y = mx + b \qquad (4)$$

której współczynnik kierunkowy obliczyć można z równania (3). Teraz wykonywany jest jeden krok algorytmu, polegający na tym, że zamiast aproksymacji krzywej (2) wykonujemy jeden krok aproksymacji prostej (4). W wyniku tej operacji otrzymywany jest nowy punkt (x, y). Przyrosty współrzędnych wykorzystywane są do obliczenia współczynnika kierunkowego prostej (4) wyprowadzonej w nowym punkcie, do czego służą rekurencyjne zależności wyprowadzone przez Autorów na podstawie równania (3). Następnym krokiem algorytmu jest krok aproksymacji nowej prostej.

Poprawność tej metody nie jest oczywista, bowiem ciąg kroków aproksymacji kolejnych stycznych do krzywej nie jest tym samym, co aproksymacja samej krzywej. Dowód stabilności metody w [4] nie został przeprowadzony, a wątpliwości nasuwają przykłady zamieszczone w [4], w których linię aproksymującą elipsę cechuje wyraźna asymetria.

Podsumowując dyskusję stosowanych rozwiązań należy stwierdzić, że żadnej ze stosowanych metod nie można uznać za w pełni zadowalającą. Najciekawsze jest ostatnie z przedstawionych

rozwiązań. Algorytm ten jest najsprawniejszym z dotychczas dyskutowanych, jest bowiem dosyć oszczędny rachunkowo i korzysta wyłącznie z działań stałoprzecinkowych. Jeżeli jednak w systemie nie zachodzi konieczność zobrazowania innych krzywych poza okręgiem, to korzystanie z algorytmu służącego do aproksymacji dowolnych krzywych byłoby używaniem narzędzia zbyt ogólnego, a więc nadmiernie złożonego. W tym przypadku korzystniejsze jest opracowanie do tego celu algorytmu specjalizowanego.

3. PROPONOWANE ROZWIĄZANIE

Podobnie jak w większości metod, możemy bez zmniejszania ogólności rozwiązań przyjąć następujące założenia upraszczające. Zagadnienie aproksymacji okręgu rozpatrywać będziemy w przestrzeni siatkowej o kwadratowej siatce, której krok przyjmujemy za jednostkę. Otoczenie dowolnego punktu $p = (x, y)$ siatki jest skończonym zbiorem punktów przestrzeni:

$$\sigma(p) = \{ p_k \}$$

gdzie

$$p_k = (x + \delta x_k, y + \delta y_k) \qquad (k = 1, 2, \ldots, K) \qquad (5)$$

Za układ odniesienia przyjmujemy lokalny układ współrzędnych o początku umieszczonym w środku zadanego okręgu. Zadany promień R jest dodatnią liczbą całkowitą.

Przy tych założeniach zobrazowanie okręgu polega na wyznaczeniu ciągu punktów siatki $\{ p_0, p_1, \ldots \}$ takich, że dla każdego $i \geqslant 0$ $p_{i+1} \in \sigma(p_i)$ oraz błąd aproksymacji każdego p_i jest mniejszy od dopuszczalnego. Dla dowolnego punktu $p_i = (x_i, y_i)$ błąd reprezentowany będzie wyrażeniem:

$$U_i = r_i^2 - R^2 \qquad (6)$$

gdzie r_i - odległość punktu p_i od środka układu.

Jak zostanie wykazane później, wyrażenie to dla przyjętych założeń odpowiada konwencjonalnym miarom błędu, a jest od nich znacznie wygodniejsze w przetwarzaniu maszynowym.

Ogólna koncepcja algorytmu jest następująca. Niech $\{p_{o,o}, p_{1,o}, \ldots, p_{i,o}\}$ będzie ciągiem już wyznaczonych punktów. Ostatnim wyznaczonym punktem jest $p_{i,o}$ a jego błąd aproksymacji wynosi $U_{i,o}$. Każdemu z punktów $p_{i,k} \in \sigma(p_{i,o})$ odpowiada błąd $U_{i,k}$.

Oznaczmy przez $m_i \in \{1, 2, \ldots, K\}$ liczbę, dla której:

$$|U_{i,m_i}| = \min_{k=1,2,\ldots,K} \{|U_{i,k}|\}$$

Wówczas jako następny wyznaczony punkt przyjmujemy:

$p_{i+1,o} = p_{i,m_i}$, a jego błędem jest $U_{i+1,o} = U_{i,m_i}$.

W ten sposób zadanie sprowadza się do znalezienia wygodnej w przetwarzaniu, rekurencyjnej zależności, służącej do obliczania wartości $U_{i,k}$ oraz znajdowania minimum ich wartości bezwzględnych.

Zachodzi związek:

$$U_{i,k} = (x_{i,o} + \delta x_k)^2 + (y_{i,o} + \delta y_k)^2 - R^2 =$$

$$= U_{i,o} + (\delta x_k)^2 + 2x_{i,o}\,\delta x_k + (\delta y_k)^2 + 2y_{i,o}\,\delta y_k$$

czyli:

$$U_{i,k} = U_{i,o} + T_{i,k} + V_{i,k} \qquad (k = 1, 2, \ldots, K) \qquad (7)$$

gdzie:

$$\left.\begin{aligned} T_{i,k} &= (\delta x_k)^2 + 2x_{i,o}\,\delta x_k \\ V_{i,k} &= (\delta y_k)^2 + 2y_{i,o}\,\delta y_k \end{aligned}\right\} \qquad (8)$$

Zmienne T oraz V podlegają następującym zależnościom:

$$T_{i+1,k} = (\delta x_k)^2 + 2x_{i+1,o}\,\delta x_k = (\delta x_k)^2 + 2\,(x_{i,o} + \delta x_{m_i})\,\delta x_k$$

Dla V zachodzi analogiczny związek, stąd:

$$\left.\begin{aligned} T_{i+1,k} &= T_{i,k} + 2\,\delta x_{m_i}\,\delta x_k \\ V_{i+1,k} &= V_{i,k} + 2\,\delta y_{m_i}\,\delta y_k \end{aligned}\right\} \qquad (9)$$

Wzory (7) - (9) przedstawić można w bardziej przejrzystej, wektorowej postaci. Krok algorytmu stanowi wówczas ciąg następujących czynności:

1. obliczenie $U_{i,k}$ z równania:

$$\overline{U}_i = \overline{U}_{i,o} + \overline{T}_i + \overline{V}_i \tag{10}$$

2. wyznaczenie m_i,

3. wykonanie ruchu m_i, polegającego na przesunięciu od punktu $p_{i,o}$ do punktu $p_{i+1,o}$, tj. na przesunięciu o wektor:

$$\delta_{m_i} = (\delta x_{m_i}, \delta y_{m_i})$$

4. modyfikacja zmiennych:

$$\left.\begin{aligned} U_{i+1,o} &= U_{i,m_i} \\ \overline{T}_{i+1} &= \overline{T}_i + \overline{\delta T}_{m_i} \\ \overline{V}_{i+1} &= \overline{V}_i + \overline{\delta V}_{m_i} \end{aligned}\right\} \tag{11}$$

gdzie: $\overline{U}_i = [U_{i,1}, U_{i,2} \cdots, U_{i,K}]$

$\overline{U}_{i,o} = [U_{i,o}, U_{i,o}, \cdots, U_{i,o}]$

$\overline{T}_i = [T_{i,1}, T_{i,2}, \cdots, T_{i,K}]$

$\overline{V}_i = [V_{i,1}, V_{i,2}, \cdots, V_{i,K}]$

natomiast $\overline{\delta T}_{m_i}$ oraz $\overline{\delta V}_{m_i}$ są wierszami m_i macierzy:

$$\left.\begin{aligned} \overline{\delta T} &= [\delta T_{j,k}]_{j,k=1,2,\ldots,K} \qquad \overline{\delta V} = [\delta V_{j,k}]_{j,k=1,2,\ldots,K} \\ \text{gdzie: } \delta T_{j,k} &= 2\,\delta x_j\,\delta x_k;\ \delta V_{j,k} = 2\,\delta y_j\,\delta y_k \end{aligned}\right\} \tag{12}$$

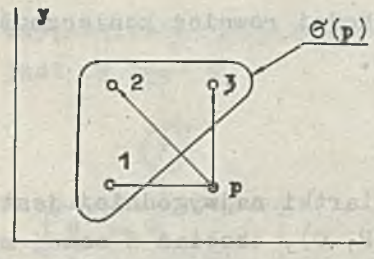
Macierze $\overline{\delta T}$ i $\overline{\delta V}$ są stałymi parametrami algorytmu i zależą wyłącznie od przyjętej definicji otoczenia $\mathcal{O}(p)$.

Bardziej szczegółowe zbadanie własności algorytmu oraz próby jego optymalizacji możliwe są po dokonaniu wyboru otoczenia. Zazwyczaj przyjmuje się, że $\mathcal{O}(p)$ zawiera od dwóch do ośmiu punktów najbliższych punktowi p.

4. USPRAWNIENIA ALGORYTMU

Znaczne uproszczenia można uzyskać przy założeniu, że zasadniczy algorytm określony jest dla pierwszej ćwiartki płaszczyzny, a jego uogólnienie na pozostałe ćwiartki dokonuje się poprzez obrót. W tym przypadku wystarczy przyjąć otoczenie trzypunktowe $\mathcal{O}(p) = \{p_1, p_2, p_3\}$, określone następującymi przyrostami:

$$\delta_1 = (-1, 0)$$
$$\delta_2 = (-1, 1)$$
$$\delta_3 = (0, 1)$$



Rys. 3

Macierze przyrostów przybierają postać:

$$\overline{\delta T} = \begin{bmatrix} 2,2,0 \\ 2,2,0 \\ 0,0,0 \end{bmatrix} \qquad \overline{\delta V} = \begin{bmatrix} 0,0,0 \\ 0,2,2 \\ 0,2,2 \end{bmatrix}$$

Po podstawieniu tych wartości do (8) otrzymujemy następujące własności zmiennych T i V:

$$T_{i,3} = V_{i,1} = 0; \quad T_{i,1} = T_{i,2}; \quad V_{i,2} = V_{i,3} \text{ dla każdego } i \geqslant 0.$$

A zatem wektory $\overline{T}_i$ oraz $\overline{V}_i$ mają postać:

$$\left.\begin{aligned} \overline{T}_i &= [T_i, T_i, 0] \\ \overline{V}_i &= [0, V_i, V_i] \end{aligned}\right| \qquad (13)$$

gdzie: $T_i = 1 - 2x_i$

$V_i = 1 + 2y_i$

Oznacza to, że zamiast przechowywania dwóch wektorów, wystarczy przechowywanie dwóch zmiennych skalarnych T_i oraz V_i. Upraszcza się także sposób modyfikacji tych zmiennych. Z (9) otrzymujemy:

$$\left.\begin{aligned} T_{i+1} &= T_i + \overline{\delta T}_{m_i} \\ V_{i+1} &= V_i + \overline{\delta V}_{m_i} \end{aligned}\right| \qquad (14)$$

gdzie: $\overline{\delta T} = [2,2,0]$
$\overline{\delta V} = [0,2,2]$

Tak więc nie zachodzi również konieczność pamiętania pełnej macierzy przyrostów.

Warunki początkowe

Dla pierwszej ćwiartki najwygodniej jest przyjąć punkt początkowy $p_{o,o} = (R, 0)$, chociaż z samej metody wynika możliwość rozpoczynania obliczeń poczynając od dowolnego punktu. Z (6) oraz (13) otrzymujemy początkowe wartości zmiennych:

$$U_{o,o} = 0; \quad T_o = 1 - 2R; \quad V_o = 1$$

Warunek zakończenia

Algorytm w pierwszej ćwiartce kontynuowany jest dopóki $x_i > 0$, co odpowiada warunkowi $T_i < 0$. Dla $x_i = 0$ następuje zakończenie działania, a ostatnim wyznaczonym punktem jest wówczas (0, R), bowiem dla tego punktu jest U = 0.

Wyznaczanie wartości m_i

Wyznaczanie wartości m_i jest znaczną częścią każdego kroku algorytmu. Pożądane jest zatem zoptymalizowanie tego fragmentu obliczeń, a w szczególności uniknięcie odwołań do funkcji ABS. Jest to możliwe, gdy oprzemy się na poniższej własności.

Tak długo, dopóki algorytm w pierwszej ćwiartce jest kontynuowany, zachodzi $x_i > 0$, $y_i \geq 0$, $(i \geq 0)$, czyli $T_i < 0$, $V_i > 0$. Stąd na podstawie (10) i (13) funkcja U posiada tę

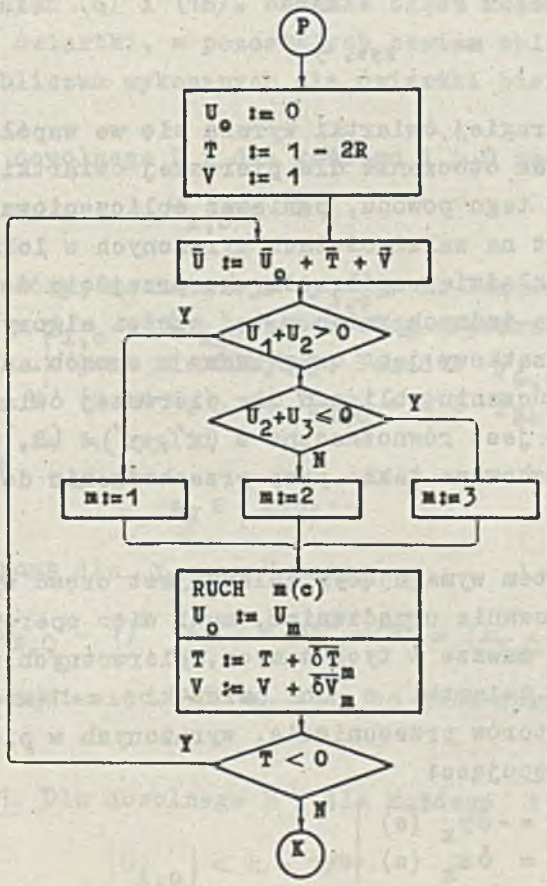
własność, iż jej wartości dla kolejnych k tworzą ciąg rosnący:

$$U_{i,1} < U_{i,2} < U_{i,3} \qquad \text{dla każdego } i \geq 0 \qquad (15)$$

Z definicji liczby m_i oraz własności (15) wynika, iż m_i jest dobrze określoną funkcją postaci:

$$m_i = \begin{cases} 1 & \text{gdy } U_{i,1} + U_{i,2} > 0 \\ 3 & \text{gdy } U_{i,2} + U_{i,3} \leq 0 \\ 2 & \text{pozost.} \end{cases} \qquad (16)$$

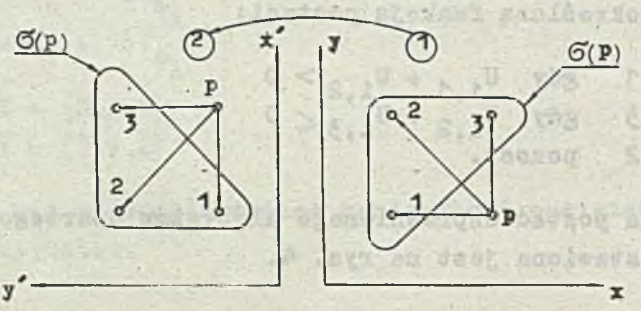
Ostateczna postać usprawnionego algorytmu opartego na (13) i (14) przedstawiona jest na rys. 4.



Rys. 4

Zmiana ćwiartki

Przejście do następnej ćwiartki dokonuje się poprzez obrót płaszczyzny o $\pi/2$. Zmianę układu współrzędnych i związanego z nim otoczenia ilustruje rys. 5.



Rys. 5

Otoczenie dla drugiej ćwiartki wyraża się we współrzędnych $(x'y')$ tak samo, jak otoczenie dla pierwszej ćwiartki we współrzędnych (x, y). Z tego powodu, ponieważ obliczeniowa część algorytmu oparta jest na zależnościach wyrażonych w lokalnym, dowolnie przyjętym układzie współrzędnych, przejście do następnej ćwiartki nie wymaga żadnych zmian w tej części algorytmu. Ponadto, warunek początkowy jest uzyskiwany w sposób automatyczny, bowiem po zakończeniu obliczeń dla pierwszej ćwiartki $(x, y) = (0, R)$, co jest równoznaczne z $(x', y') = (R, 0)$. Własność ta zostaje zachowana także przy przechodzeniu do następnych ćwiartek.

Jedynym fragmentem wymagającym zmiany jest część wykonawcza, która dotyczy sterowania urządzeniem, musi więc operować przyrostami wyrażonymi zawsze w tych samych, pierwotnych współrzędnych. Przejściu od ćwiartki c do ćwiartki $c+1 \pmod 4$ odpowiada zmiana wektorów przesunięcia, wyrażonych w pierwotnych współrzędnych następująco:

$$\left.\begin{aligned} \delta x_k(c+1) &= -\delta y_k(c) \\ \delta y_k(c+1) &= \delta x_k(c) \end{aligned}\right|$$

Błąd aproksymacji

Rozpatrzmy błąd aproksymacji przeprowadzonej według omawianego algorytmu. Wyrażenie zgodne z miarą max:

$$|r_i - R| < \varepsilon \quad (i \geqslant 0) \tag{17}$$

równoważne jest:

$$R - \varepsilon < r_i < R + \varepsilon .$$

Stąd na podstawie (6):

$$\varepsilon^2 - 2R\varepsilon < U_{i,o} < \varepsilon^2 + 2R\varepsilon \tag{18}$$

O równoważności miary max i miary U mówimy w sensie równoważności wyrażeń (6) i (18). Badanie błędu możemy ograniczyć do pierwszej ćwiartki, w pozostałych bowiem obliczenia są powtórzeniem obliczeń wykonanych dla ćwiartki pierwszej.

Lemat 1. Dla dowolnego R i dla każdego $i \geqslant 0$ zachodzi:

$$y_{i,o} \leqslant R$$

Dowód. Przypuśćmy, że istnieje $y_{j,o} > R$. Ponieważ $\delta y_k \in \{0,1\}$ oraz $y_{i+1,o} = y_{i,o} + \delta y_{m_i}$, zatem ciąg współrzędnych $\{y_{o,o}, y_{1,o}, \ldots\}$ jest niemalejący. Ponadto $y_{o,o} = 0$, musi więc istnieć takie N $(0 < N < j)$, że $y_{N,O} = R$ i $y_{N+1,O} > R$. Oznacza to, że $\delta y_{m_N} = 1$, czyli:

$$m_N \in \{2,3\} \tag{19}$$

Z drugiej strony dla $y_{N,O} = R$ mamy:

$$U_{N,1} = (x_{N,O} - 1)^2 + (R + 0)^2 - R^2 = (x_{N,O} - 1)^2 \geqslant 0$$

Na mocy (16) musi więc być $m_N = 1$, co jest sprzeczne z (19).

c.n.d.

Twierdzenie 1. Dla dowolnego R i dla każdego $i \geqslant 0$ zachodzi:

$$|U_{i,o}| < R - 1/4 \tag{20}$$

Dowód

1. Dla i = 0 teza jest oczywista, bowiem $U_{0,0} = 0$.

2. Niech nierówność (20) będzie spełniona dla pewnego $i > 0$. Jeżeli istnieje $U_{i,k} = 0$, to $U_{i+1,o} = 0$. W przeciwnym razie możliwe są następujące przypadki:

   a) wszystkie $U_{i,k}$ są tego samego znaku. Niech $U_{i,1} > 0$, wówczas $m_i = 1$, zatem:
   $0 < U_{i+1,0} = U_{i,1} = U_{i,0} + T_i < U_{i,0}$ ze względu na $T_i < 0$.
   Czyli na mocy założenia indukcyjnego:

   $$0 < U_{i+1,0} < U_{i,0} < R - 1/4$$

   W przypadku $U_{i,3} < 0$, dowód jest analogiczny.
   W szczególności przypadek ten zachodzi dla $y_{i,0} = R$, bo wówczas $U_{i,1} \geq 0$, co wynika z dowodu lematu 1.

   b) $U_{i,k}$ są różnych znaków. Jest to możliwe tylko przy $y_{i,o} < R$, czyli $V_i \leq 2R - 1$.
   Przypuśćmy, że $U_{i,1} < 0 < U_{i,2}$.
   Jeśli $|U_{i,1}| < |U_{i,2}|$, to $m_i = 1$, wówczas:

   $$|U_{i+1,0}| = |U_{i,1}| = -U_{i,1} < U_{i,1} + V_i = U_{i,2} = |U_{i,2}|$$

   Stąd:

   $$|U_{i+1,0}| < 1/2\ V_i < R - 1/2 < R - 1/4$$

   Jeśli zaś $|U_{i,1}| \geq |U_{i,2}|$, to $m_i = 2$, wtedy:

   $$|U_{i+1,0}| = |U_{i,2}| = U_{i,2} \leq -U_{i,2} + V_i = -U_{i,1} = |U_{i,1}|$$

   Stąd:

   $$|U_{i=1,0}| \leq 1/2\ V_i \leq R - 1/2 < R - 1/4$$

   W pozostałym przypadku: $U_{i,2} < 0 < U_{i,3}$, dowód jest analogiczny korzystając z nierówności $1 - 2R \leq T_i < 0$
   Zatem w każdym przypadku nierówność (20) jest spełniona dla i + 1.

c.n.d.

Wnioski

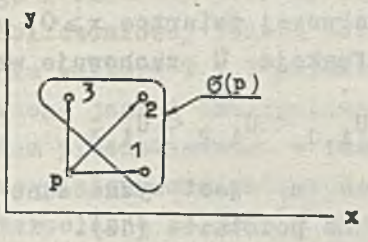
1. Ze względu na równoważność (6) i (18), twierdzenie 1 implikuje: $|r_i - R| < 1/2$, tzn. odległość każdego z wyznaczonych punktów od aproksymowanego okręgu jest mniejsza niż połowa kroku siatki.

2. Jeżeli przyjęta siatka odpowiada rastrowi zastosowanego urządzenia graficznego, to z twierdzenia 1 wynika, że dokładność aproksymacji według podanego algorytmu jest największą dokładnością możliwą do uzyskania za pomocą tego urządzenia.

5. APROKSYMACJA ODCINKA I INTERPOLACJA UKŁADOWA

Zagadnienie aproksymacji odcinka prostej rozpatrywać będziemy w lokalnym układzie współrzędnych, którego środkiem jest początek zadanego odcinka. Z zadanych parametrów odcinka zawsze można wyznaczyć ćwiartkę płaszczyzny, w której leży ten odcinek. Ograniczymy rozważania do przypadku, gdy zadany odcinek leży w pierwszej ćwiartce. Uogólnienie rozwiązania na pozostałe ćwiartki może być dokonane poprzez obrót. Przyjmujemy, że w pierwszej ćwiartce otoczenie określone jest następującymi przyrostami:

$$\delta_1 = (1,0)$$
$$\delta_2 = (1,1)$$
$$\delta_3 = (0,1)$$



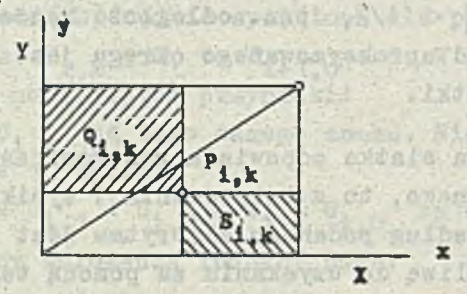
Rys. 6

Odcinek określony równaniem $Xy = Yx$ zadany jest przez podanie punktu końcowego (X, Y).

Błąd punktu $p_{i,k}$ reprezentowany będzie wyrażeniem:

$$U_{i,k} = Xy_{i,k} - Yx_{i,k} \qquad (21)$$

Wartość $U_{i,k}$ jest równa różnicy zakreskowanych na rys. 7 pól $S_{i,k}$ oraz $Q_{i,k}$ i odpowiada odległości punktu $p_{i,k}$ od zadanego odcinka.



Rys. 7

Metoda wyznaczania kolejnych punktów na podstawie wartości $U_{i,k}$ jest taka sama, jak przy aproksymacji okręgu.

Zachodzi:

$$U_{i,k} = X(y_{i,o} + \delta y_k) - Y(x_{i,o} + \delta x_k) = U_{i,o} + X\,\delta y_k - Y\delta x_k$$

Czyli:

$$\bar{U}_i = \bar{U}_{i,o} + \bar{T} + \bar{V}$$

gdzie: $\bar{T} = [-Y, -Y, 0]$
$\bar{V} = [0, X, X]$ (22)

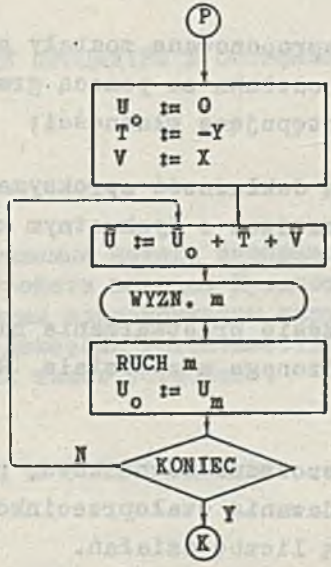
Ponieważ w pierwszej ćwiartce $x \geq 0$, $y \geq 0$, zatem przy założeniu, że $XY \neq 0$, funkcja U zachowuje własność (15):

$$U_{i,1} < U_{i,2} < U_{i,3} \qquad (i \geq 0)$$

Wynika stąd, że m_i jest wyznaczane tak samo, jak przy aproksymacji okręgu, na podstawie (16).

Punktem początkowym jest $(x_{o,o}, y_{o,o}) = (0, 0)$, a całość algorytmu naszkicowana została na rys. 8.

Adaptacja algorytmu dla pozostałych ćwiartek dokonywana jest poprzez obrót płaszczyzny, związany z modyfikacją tylko części wykonawczej i wartości początkowych.



Rys. 8

Interpolacja w urządzeniach graficznych dokonywaná jest analogowo, bądź cyfrowo. W większości grafoskopów przeważa sposób analogowy, istnieją jednak urządzenia (np. plotter inkrementalny), w których interpolacja cyfrowa stanowi jedyne możliwe rozwiązanie. W tych urządzeniach może znaleźć zastosowanie algorytm omówiony w rozdziale 4.

Łatwo można zauważyć, że przedstawiony w rozdziale 4 algorytm aproksymacji okręgu i omówiony powyżej algorytm aproksymacji odcinka, w części obliczeniowej różnią się tylko tym, że przy aproksymacji odcinka wektory T i V pozostają stałe. Jeżeli zatem urządzenie wyposażone jest w interpolator liniowy działający zgodnie z algorytmem przedstawionym w rozdz. 5, to interpolator ten może być uogólniony minimalnym kosztem w taki sposób, aby umożliwiał interpolację kołową. W tym celu wystarczy aby rejestry, w których przechowywane są wartości T_i i V_i, zrealizowane były jako liczniki, gdzie po każdym kroku do zawartości jednego z nich, lub obu odpowiednio, dodawana jest liczba 2. Przy zobrazowaniu odcinka z własności tej się nie korzysta.

6. PODSUMOWANIE

W rozdziałach 4 i 5 zaproponowane zostały algorytmy służące do zobrazowania okręgu i odcinka za pomocą graficznych urządzeń wyjściowych. Mają one następujące własności:

1. umożliwiają największą dokładność aproksymacji możliwą do uzyskania za pomocą urządzeń o dyskretnym zbiorze adresowalnych punktów,

2. błąd aproksymacji w czasie przetwarzania nie kumuluje się, co wynika z przeprowadzonego w rozdziale 4 dowodu stabilności,

3. są bardzo sprawne i oszczędne rachunkowo, ponieważ używają wyłącznie operacji dodawania stałoprzecinkowego, a każdy krok zawiera niewielką liczbę działań.

Należy podkreślić, że rozwiązania te nadają się do zastosowania zarówno w urządzeniach drugiej grupy, jak i urządzeniach wyposażonych w interpolator liniowy. W tym ostatnim przypadku jako zbiór punktów charakterystycznych może być przyjęty dowolny podzbiór wyznaczonych punktów $\{p_{i_{j1},o}, p_{i_{j2},o}, \ldots\}$. Ponadto algorytmy te mogą być zastosowane w układowych interpolatorach jednostek sterujących urządzeniami.

Przedstawione w rozdz.4 i 5 algorytmy zostały zaprogramowane i sprawdzone praktycznie w Instytucie Maszyn Matematycznych.

Literatura

[1] JUNG L.: Graficzna kompilacja obiektów geometrycznych na ekranie grafoskopu, praca doktorska, WAT 1973

[2] COHEN D.: On Linear Differences Curves, Intern. Symp. Comp. Graphics 70. Brunel Uniw., kwiecień 1970

[3] DENERT E.: A Method for Computing Points of a Circle Using only Integers, Computer Graphics and Image Processing, 1973, t. 2, nr 1

[4] PITTEWAY M.L.V.: Algorithm for Drawing Ellipses or Hyperbolae with a Digital Plotter, The Comp. Journ., 1967, t. 10, nr 3

ИЗБРАННЫЕ ГРАФИЧЕСКИЕ ПРОЦЕДУРЫ В СИСТЕМАХ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

Резюме

Анализованы применяемые методы изображения округа при помощи графических устройств вывода. Предложены просто вычисляемые и точные алгоритмы аппроксимации округа и отрезка и исследованы их свойства. Доказана устойчивость и применяемость этих алгоритмов для разных типов устройств.

SOME GRAPHIC PROCEDURES IN COMPUTER SYSTEMS

Summary

Methods utilized to image a circle using graphic output devices are analyzed. Simply computable and precise algorithms of circle and section approximation are developed. Properties of the algorithms are discussed and the proof of their stability is given. These algorithms use fixed-point arithmetic only so they are fast and useful for any graphic device.

Prace IMM
Zeszyt 2
© 1975.04
681.322.06:519.14

ALGORYTM PLANOWANIA OBLICZEŃ DWUPROCESOROWYCH

Andrzej ROWICKI

Pracę złożono 20.06.1974

W pracy podano algorytm planowania obliczeń dwuprocesorowych oraz algorytm wyznaczania czasu wykonania obliczenia. Algorytm planowania obliczeń jest optymalny w tym sensie, że zapewnia najmniejszy możliwy czas wykonania obliczenia. Jako model obliczenia przyjęto graf zorientowany nie zawierający pętli.

Praca dotyczy zagadnienia planowania obliczeń dla systemów dwuprocesorowych. Będziemy zakładali, że procesory są identyczne oraz, że obliczenia nie zawierają pętli. Ponadto zakładamy, że obliczenia składają się z zadań o jednakowym czasie wykonania.

Przy tych założeniach skonstruujemy algorytm ustalający kolejność wykonywania zadań, tak by czas wykonania obliczenia był minimalny. Skonstruowany algorytm będzie również wyznaczał czas wykonania obliczenia.

Jako model obliczenia przyjmiemy graf zorientowany nie zawierający obwodów i mający jeden wierzchołek końcowy. Model ten interpretujemy w ten sposób, że każdemu wierzchołkowi grafu jest przyporządkowane jedno zadanie. Przyporządkowanie wierzchołkom zadań jest takie, że orientacja grafu jest zgodna z następstwem wykonywania zadań obliczenia.

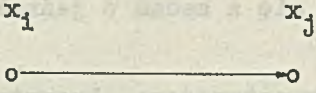
W pracy [1] podano algorytm planowania obliczeń dwuprocesorowych nie zawierających pętli. Algorytm ten nie uwzględnia za-

gadnienia wyznaczania czasu wykonania obliczenia. Przy konstrukcji tego algorytmu zakłada się, że relacja następstwa zadań obliczenia jest dana w formie jawnej, tzn. nie jest dana za pośrednictwem grafu będącego modelem obliczenia. Natomiast przy konstrukcji algorytmu planowania obliczenia, który zostanie opisany w pracy, przyjęto, że relacja następstwa zadań obliczenia jest zadana przez graf obliczenia.

Przystąpimy teraz do opisu algorytmu. Ze względu na konieczność wprowadzenia dużej liczby pojęć i oznaczeń, przyjmiemy następującą zasadę: pojęcia i oznaczenia będziemy wprowadzać stopniowo w miarę potrzeby.

Rozważmy następujący graf $G = \langle X, \Gamma, x_o \rangle$,
gdzie:
X - skończony zbiór wierzchołków
Γ - relacja zdefiniowana na zbiorze X
x_o - wierzchołek końcowy grafu G, tzn. taki wierzchołek, że zachodzi następująca relacja $\Gamma(x_o) = \emptyset$ (gdzie $\emptyset$ oznacza zbiór pusty)

Jeżeli między wierzchołkami x_i i x_j zachodzi relacja $x_j \in \Gamma(x_i)$, to sytuację tę interpretujemy w sposób następujący:



W zbiorze wierzchołków X grafu G wyróżnimy pewien podzbiór X_p spełniający następującą relację: $x \in X_p \Rightarrow \Gamma^{-1}(x) = \emptyset$. Zbiór X_p będziemy nazywali zbiorem wierzchołków początkowych grafu G.

W dalszych rozważaniach ograniczymy się do pewnej klasy grafów, które będziemy nazywali sieciami i oznaczali literą S.

Definicja. Sieć

Graf $G = \langle X, \Gamma, x_o \rangle$ nazywamy siecią, jeżeli jest spójny i nie zawiera obwodów.

Na zbiorze X określimy dwie pomocnicze funkcje ξ i η zwane etykietami. Funkcje ξ i η zdefiniujemy w sposób następujący:

$$\xi(x) = \begin{cases} 1 \text{ jeżeli } x = x_0 \\ \max\limits_{y \in P(x)} \left\{ \xi(y) + 1 \right\} \quad \text{jeżeli } y \in \Gamma(x) \end{cases}$$

Funkcja η przyporządkowuje każdemu wierzchołkowi sieci liczbę równą liczbie wierzchołków leżących na najdłuższej drodze prowadzącej z rozważanego wierzchołka do wierzchołka końcowego sieci x_0.

Liczbę p zdefiniowaną w sposób następujący:

$$p = \max_{x \in X} \left\{ \xi(x) \right\}$$

nazywamy rzędem sieci.

Funkcję η definiujemy w sposób następujący:

$$\eta(x) = p + 1 - \xi(x)$$

Łatwo zauważyć, że

$$p = \max_{x \in X} \left\{ \eta(x) \right\}$$

Funkcja η ma charakter pomocniczy, ma zapewnić normalizację rozumianą w tym sensie, by wystąpiła zgodność stopnia zaawansowania obliczenia z numeracją kroków algorytmu.

Definicja. Poziom i

Zbiór $\Omega_i \subset X$ zdefiniowany w sposób następujący:

$$\Omega_i = \left\{ x \in X : \eta(x) = i \right\}$$

nazywamy poziomem i sieci S.

Wprowadzimy jeszcze funkcję pomocniczą ζ. Funkcja ζ jest różnowartościową przekształcającą zbiór wierzchołków

sieci w podzbiór liczb naturalnych, spełniającą następującą relację:

$$x \in \Omega_i \wedge y \in \Omega_{i+1} \Longrightarrow \zeta(x) < \zeta(y)$$

Łatwo zauważyć, że zachodzi następująca relacja:

$$\max_{x \in \Omega_i} \left\{ \zeta(x) \right\} < \min_{y \in \Omega_{i+1}} \left\{ \zeta(y) \right\}$$

Przystąpimy teraz do opisu algorytmu. Celem algorytmu jest podanie rozłącznego podziału wierzchołków sieci, tak by odpowiadające im zadania można było wykonywać równocześnie oraz by czas wykonania całego obliczenia był minimalny. Ponieważ do dyspozycji mamy tylko dwa procesory, to podział będzie zawierał zbiory złożone najwyżej z dwóch wierzchołków. Natomiast by zapewnić minimalny czas wykonania obliczenia trzeba tak dokonać rozłącznego podziału zbioru wierzchołków sieci, by było w nim zawartych możliwie najwięcej zbiorów dwuelementowych.

Konstrukcja algorytmu opiera się na dwóch podstawowych funkcjach φ i ρ. Funkcja φ dla ustalonego i jest skojarzona z pewnym podzbiorem $F(i)$ zbioru wierzchołków sieci. Zadania odpowiadające elementom zbioru $F(i)$ należy wykonać w kroku i algorytmu. Wartość funkcji $\varphi(i)$ określa w ilu etapach można wykonać zadania odpowiadające elementom zbioru $F(i)$. Natomiast funkcja ρ, niezbyt precyzyjnie mówiąc, wyznacza etapy oraz ustala kolejność realizacji etapów, tzn. grupuje zadania do równoczesnego wykonania oraz podaje kolejność wykonywania grup zadań.

Definicja. Funkcja φ

Funkcję φ definiujemy w sposób następujący:

$$\varphi(i) = \left\lceil \frac{\| \Omega_i - \bigcup_{0}^{i-1} F(j) \|}{2} \right\rceil$$

gdzie $\lceil x \rceil$ oznacza najmniejszą liczbę całkowitą większą lub równą x, a $\| \omega \|$ liczność zbioru ω.

Natomiast zbiór F(j) jest pewnym zbiorem, którego definicję podamy nieco później.

Definicja. Funkcja τ

Funkcję τ definiujemy w sposób następujący:

$$\tau(S) = \sum_{1}^{P} \varphi(i)$$

Funkcja $\tau(S)$ przyporządkowuje każdej sieci pewną liczbę naturalną, którą można interpretować jako czas (liczbę kroków), potrzebną do wykonania obliczenia, którego struktura jest reprezentowana przez sieć S.

Przed podaniem definicji zbioru F (i) wprowadzimy jeszcze pewne oznaczenia i pojęcia, mianowicie:

$$\omega(i) = \Omega_i - \bigcup_{o}^{i-1} F(j)$$

$$\omega^*(i, n) = \Omega_{i+n} - \bigcup_{o}^{i-1} F(j)$$

Łatwo zauważyć, że $\omega^*(i, o) = \omega(i)$

$$\nu(i) = \begin{cases} \text{najmniejsze takie } 0 \leqslant n \leqslant p \text{ że } \| \omega^*(i, n) - \\ -\Gamma(\omega^*(i, n-1)) \| \geqslant 1 \\ 0 \text{ w przeciwnym przypadku} \end{cases}$$

$$\Delta(i) = \omega(i) \cup \omega^*(i, \nu(i)) - \Gamma(\omega^*(i, \nu(i)-1))$$

Wyróżnimy jeszcze pewne wierzchołki χ_i i χ_i^* spełniające relację: $\chi_i \in H_i^*$ i $\chi_i^* \in H_i^{**}$. Wierzchołki χ_i i χ_i^* będziemy nazywali wierzchołkami osobliwymi.

$$H_i = \left\{ x \in \omega^*(i, o) : \| \Gamma_i^*(x) \| = \min_{y \in \omega^*(i, o)} \left\{ \| \Gamma_i^*(y) \| \right\} \right.$$

$$H_i^* = \left\{ x \in H_i : \bigvee_{y \in \omega^*(i,\,1)-\Gamma(x)} \| \Gamma^*_{i+1}(y) \| = \right.$$

$$= \max_{x \in H_i} \left\{ \max_{z \in \omega^*(i,1)\ -\Gamma(x)} \left\{ \| \Gamma^*_{i+1}\ (z) \| \right\} \right\}$$

gdzie Γ^*_i jest relacją określoną w sposób następujący:

$$\Gamma^*_i(x) = \Gamma(x) \cap (\Omega_{i+1} - \bigcup_{o}^{i-1} F(j))$$

$$\Delta^*(i) = \omega^*(i,\ 1) - \Gamma(x_i)$$

$$H_i^{**} = \left\{ x \in \Delta^*(i) : \| \Gamma^*_{i+1}(x) \| = \max_{y \in \Delta^*(i)} \left\{ \| \Gamma^*_{i+1}\ (y) \| \right\} \right.$$

r(i) - reszta z dzielenia $\| \omega(i) \|$ przez 2.

Definicja. F (i)

Zbiór F(i) definiujemy indukcyjnie w sposób następujący:

1^o $F(o) = \emptyset$

2^o dla $i \neq 0$

Przypadek 1

Jeżeli r(i) = 0, to

$$F(i) = \Omega_i - \bigcup_{o}^{i-1} F(j) = \omega(i)$$

Przypadek 2

$$r\ (i) \neq\ 0$$

a) Jeżeli $\Delta^*(i) \neq \emptyset$, to

$$F(i) = \omega(i) \cup \left\{ \chi^*_i \right\}$$

b) Jeżeli $\Delta^*(i) = \emptyset \wedge \nu(i) = 0$, to

$$F(i) = \Omega_i - \bigcup_{o}^{i-1} F(j) = \omega(i)$$

c) Jeżeli $\Delta^*(i) = \emptyset \wedge \nu(i) \neq 0$ to

i) $F(i) \subset \Delta(i)$

ii) $\| F(i) \| = 2 \cdot \varphi(i)$

iii) $(x \in F(i)) \wedge (y \in \Delta(i) - F(i)) \Rightarrow \zeta(x) < \zeta(y)$

Wprowadzimy teraz na zbiorze wierzchołków sieci relację częściowego porządku $\prec$ zdefiniowaną w sposób następujący:

Definicja. Relacja $\prec$

$$x \prec y \Longleftrightarrow \rho(x) < \rho(y)$$

gdzie ρ jest pewną funkcją, którą zdefiniujemy nieco później.

Przed podaniem definicji funkcji ρ wprowadzimy pewne oznaczenia.

Niech ciąg $P(F(i)) = P(1, i), \ldots, P(n, i), \ldots, P(k, i)$ oznacza dowolny rozłączny podział zbioru $F(i)$ spełniający następujące warunki:

1^o dla $j < k$, $\| P(j, i) \| = 2$

2^o $\| P(k, i) \| = \| F(i) - \bigcup_{1}^{k-1} P(j, i) \| \leqslant 2$

3^o $\{ \chi_i, \chi_i^* \} \subset P(k, i)$

Definicja. Funkcja

Funkcję ρ definiujemy w sposób następujący:

Przypadek 1

Jeżeli $P(F(i))$ istnieje, to

$$\rho(x) = \sum_{1}^{i} \varphi(j) - \varphi(i) + n \quad \text{dla } x \in P(n, i)$$

Przypadek 2

Jeżeli $P(P(i))$ nie istnieje, to

$$\rho(x) = \sum_{1}^{i} \varphi(j) \quad \text{dla } x \in F(i)$$

Definicja. Redukt

Sieć $S' = < X', \Gamma', x_o' >$ nazywamy reduktem sieci $S = < X, \Gamma, x_o >$, jeżeli są spełnione następujące warunki:

1^o $X' \subset X \wedge \Gamma' \subset \Gamma$

2^o $x \in X' \Longrightarrow \Gamma(x) \subset X'$

Uwaga: Relacja bycia reduktem jest relacją zwrotną i przechodnią.

Własności:

Jeżeli sieć $S' = < X', \Gamma', x_o' >$ jest reduktem sieci $S = < X, \Gamma, x_o >$, to $x_o' = x_o$.

Definicja. Redukt właściwy

Redukt S nazywamy reduktem właściwym sieci S', jeżeli są spełnione warunki:

1^o $X - X' \subset X_p$

2^o $1 \leqslant \| X - X' \| \leqslant 2$

Definicja. Redukcja

Ciąg sieci $R = S_1, S_2, \ldots, S_k$ nazywamy redukcją sieci S (oznaczamy przez $R(S)$), jeżeli są spełnione następujące warunki:

1^o $S_1 = S$

2^o dla każdego $i < k$, S_{i+1} jest reduktem sieci S_i

3^o dla S_k nie istnieje redukt właściwy.

Natomiast liczność zbioru $\{ S_1, \ldots, S_n \}$ nazywamy długością redukcji sieci S (oznaczamy przez $d(R(S))$).

Definicja. Redukcja właściwa

Redukcję nazywamy właściwą, jeżeli zawiera tylko redukty właściwe.

Twierdzenie 1

Niech $\Theta(i) = \begin{cases} \emptyset & \text{dla } i = 0 \\ \{ x \in X : \rho(x) = i \} & \text{dla } i = \emptyset \end{cases}$

wówczas graf $R_i = \langle X - \bigcup_{o}^{i-1} \Theta(j), \Gamma/X - \bigcup_{o}^{i-1} \Theta(j), x_o \rangle$
jest reduktem sieci

$$S = \langle X, \Gamma, x_o \rangle$$

Dowód

Przez indukcję względem i.

1° dla i = 1 teza jest prawdziwa, bowiem $R_1 = S$, a więc jest reduktem sieci S (zwrotność relacji bycia reduktem).

2° Załóżmy że dla $i = n < \max_{x \in X} \{ \rho(x) \}$

teza jest prawdziwa, tzn. redukt

$$R_n = \langle X - \bigcup_{o}^{n-1} \Theta(j), \Gamma / X - \bigcup_{o}^{n-1} \Theta(j), x_o \rangle$$

jest reduktem sieci S.

By dowieść twierdzenia trzeba pokazać, że R_{n+1} jest reduktem sieci S. Ponieważ relacja bycia reduktem jest relacją przechodnią, to aby dowieść twierdzenia, wystarczy pokazać, że R_{n+1} jest reduktem sieci R_n.

Załóżmy, że tak nie jest tzn., że graf R_{n+1} nie jest reduktem sieci R_n. Jeżeli tak jest, to znaczy, że istnieje taki wierzchołek $x_n \in \Theta(n)$, że $\Gamma^{-1}(x_n) \subset X - \bigcup_{o}^{n} \Theta(j)$, a więc $\Gamma^{-1}(x_n) \not\subset \bigcup_{o}^{n} \Theta(j)$, co na mocy definicji zbioru $\Theta(i)$ daje

$$\min_{x \in \Gamma^{-1}(x_n)} \{ \rho(x) \} > \rho(x_n) \qquad (1)$$

Niech $x_n \in \Omega_\alpha$, wówczas na mocy definicji zbioru Ω_i uzyskujemy, że $\Gamma^{-1}(x_n) \in \Omega_p$ oraz $p < \alpha$, co na mocy definicji funkcji ρ ostatecznie daje

$$\max_{x \in \Gamma^{-1}(x_n)} \left\{ \rho(x) \right\} < \rho(x_n) \qquad (2)$$

Na mocy (1) i (2) uzyskujemy sprzeczność, a więc graf R_{n+1} jest reduktem sieci S.

Na mocy twierdzenia 1 relację częściowego porządku $\prec$ można zinterpretować w ten sposób, że ustala ona kolejność wykonywania zadań odpowiadających wierzchołkom sieci, a mianowicie, jeśli zachodzi relacja $x \prec y$, to interpretujemy ten fakt w sposób następujący: zadanie odpowiadające wierzchołkowi x należy wykonywać przed zadaniem odpowiadającym wierzchołkowi y.

Bezpośrednio z definicji redukcji oraz twierdzenia 1 wynika następujący wniosek:

Wniosek

Ciąg $R\rho(S) = R_1, \ldots, R_i, \ldots, R_k$, gdzie $k = \max_{x \in X} \left\{ \rho(x) \right\}$ jest redukcją właściwą sieci S.

Ciąg $R\rho(S)$ będziemy nazywali ρ - redukcją sieci S.

Zajmiemy się teraz pokazaniem, że algorytm wykonywania zadań oparty na relacji częściowego porządku $\prec$ daje optymalne rozwiązanie, tzn. że funkcja $\tau(S)$ wyznacza najmniejszą możliwą liczbę kroków potrzebną do wykonania obliczenia reprezentowanego przez sieć S.

Twierdzenie 2

Niech $d(R\rho(S))$ oznacza długość ρ-redukcji sieci S, wówczas

$$d(R\rho(S)) = \tau(S)$$

Dowód

Na mocy definicji długości redukcji uzyskujemy

$$d(R\rho(S)) = \max_{x \in X} \left\{ \rho(x) \right\} \qquad (1)$$

Natomiast z (1) na mocy definicji funkcji ρ uzyskujemy

$$d\,(R_\rho\,(S)) \;=\; \sum_{1}^{p} \varphi(j) \qquad (2)$$

Co na mocy definicji funkcji $\tau(S)$ daje

$$d\,(R_\rho(S)) \;=\; \tau(S) \qquad (3)$$

Twierdzenie 3

Dla sieci $S = \langle X, \Gamma, x_o \rangle$ nie istnieje taka redukcja właściwa $R\,(S)$, że

$$d(R\,(S)) \;<\; d\,(R\;\;(S))$$

Dowód

Niech ciąg

$$R_\rho(S) \;=\; R_1, \ldots, R_n, \ldots, R_k \qquad (1)$$

będzie ρ-redukcją właściwą sieci S. Rozważmy redukt właściwy R_{n-1} spełniający następujący warunek

$$\theta(n) \subsetneq \Omega_i \quad \text{oraz} \quad \|\,\theta\,(n)\,\| \;=\; 1 \qquad (2)$$

Warunek (2) jest spełniony jeżeli zachodzi następujący przypadek

$$r(i) \neq 0, \quad \Delta^*(i) = \emptyset, \quad \nu(i) = 0 \qquad (3)$$

Załóżmy, że redukt R_{n-1} jest pierwszym (w sensie przyjętej numeracji reduktów w ρ-redukcji) z reduktów spełniających warunek (2) . Załóżmy, że dla poziomu i-1 zachodzi następujący przypadek

$$\varphi(i-1) > 1, \quad r(i-1) \neq 0, \quad \nu(i-1) = 0, \Delta^*(i-1) \neq 0 \qquad (4)$$

Na mocy (4) istnieje taki wierzchołek x_i spełniający następującą relację

$$x_i \in F(i-1) \quad \text{oraz} \quad F(i-1) \cap \Omega_i = \{x_i\} \tag{5}$$

co na mocy definicji zbioru $F(i-1)$ oznacza, że $x_i = \chi^*_{i-1}$.

Ponieważ $\Delta^*(i) = \emptyset$ (patrz (3)), to

$$\Gamma^*_i(\theta(i)) = \omega^*(i, 1) \tag{6}$$

Ponieważ $\theta(i) = \{\chi_i\}$ oraz $x_i = \chi^*_{i-1}$, to na mocy (6) uzyskujemy ostatecznie, że dla każdego $x \in \omega^*(i, o) \cup \{x_i\}$, $\Gamma^*_i(x) = \omega^*(i, 1)$, co oznacza, że nie można utworzyć żadnego reduktu właściwego zawierającego równocześnie wierzchołki ze zbiorów $\omega^*(i, o) \cup \{x_i\}$ i $\omega^*(i, 1)$.

Powtarzając powyższe rozumowanie do kolejnych reduktów z ciągu (1) dochodzimy do wniosku, że nie istnieje redukcja właściwa sieci S o długości mniejszej od długości ρ-redukcji sieci S.

Z twierdzeń 1, 2 i 3 wynikają następujące wnioski:

1° Relacja porządkująca $\prec$ wyznacza optymalną kolejność wykonywania operacji tzn. taką, dla której liczba kroków (czas wykonywania obliczenia) jest najmniejszy.

2° Dla ustalonego i, zadania odpowiadające elementom zbioru $\theta(i)$ można wykonywać równocześnie.

3° Minimalny czas (liczbę kroków) wykonania obliczenia wyznacza funkcja $\tau(S)$.

Przykładowo rozpatrzymy trzy sieci G_1, G_2, G_3 przedstawione na rysunku 1. Sieci te posiadają jednakową liczbę wierzchołków oraz jednakowy rozkład wierzchołków na poziomy.

Dla sieci G_1, G_2, G_3 uzyskujemy następujące wartości funkcji ξ i η przedstawione w poniższej tabeli:

Tabela 1

	a	b	c	d	e	f	g	h	k
$\xi_{1,2,3}$	5	4	3	3	3	2	2	2	1
$\eta_{1,2,3}$	1	2	3	3	3	4	4	4	5

Na mocy tabeli 1 uzyskujemy:

$$p = 5$$

oraz

$$\Omega_1 = \{a\}, \quad \Omega_2 = \{b\}, \quad \Omega_3 = \{c, d, e\}$$
$$\Omega_4 = \{f, g, h\}, \quad \Omega_5 = \{k\}$$

Na podstawie uzyskanych wartości poziomów funkcję ζ określamy w sposób następujący:

Tabela 2

	a	b	c	d	e	f	g	h	k
ζ	1	2	3	4	5	6	7	8	9

Przystąpimy teraz do wyznaczania funkcji $\varphi(i)$ i zbioru $F(i)$ dla sieci G_3.

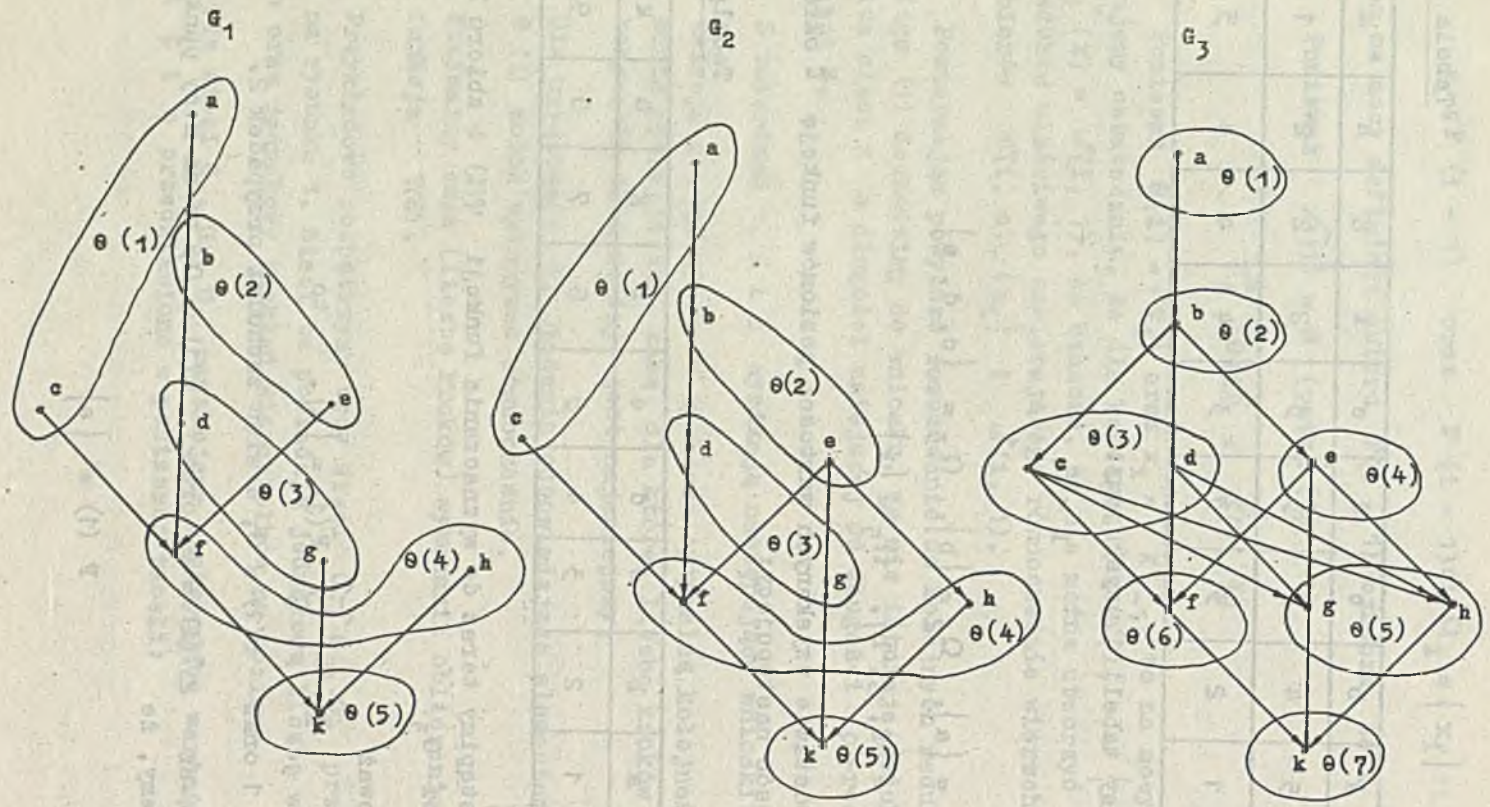
Ponieważ

$$\Omega_1 - \bigcup_{0}^{1-1} F(j) = \{a\}, \quad \text{to}$$

$\varphi(i) = 1$ oraz $r(1) \neq 0$, a więc zachodzi przypadek 2.

Ponieważ $\Delta^*(i) = \emptyset$ oraz $\nu(i) = 0$, to na mocy punktu b uzyskujemy, że

$$F(1) = \{a\}$$


G1
a
θ (1)
b
θ(2)
c
d
e
θ(3)
f
g
θ(4)
h
k
θ(5)
G2
a
θ (1)
b
θ(2)
c
d
e
θ(3)
f
g
h
θ(4)
k
θ(5)
G3
a
θ (1)
b
θ (2)
θ(3)
c
d
e
θ(4)
f
g
h
θ(6)
θ(5)
k
θ(7)

Rys. 1

Postępując podobnie uzyskujemy, że $\varphi(2) = 1$ oraz

$$F(2) = \{b\}$$

Ponieważ

$$\Omega_3 - \bigcup_0^2 F(j) = \{c, d, e\}$$

to $\varphi(3) = 2$ oraz $r(3) \neq 0$, a więc zachodzi przypadek 2.

W związku z tym przystąpimy do wyznaczenia zbioru $\Delta^*(3)$. By wyznaczyć $\Delta^*(3)$ musimy wyznaczyć wierzchołek osobliwy x_3. Okazuje się, że każdy element zbioru $\{c, d, e\}$ jest wierzchołkiem osobliwym. W związku z tym, dla ustalenia uwagi, przyjmiemy, że $x_3 = e$, a więc

$$\Delta^*(3) = \omega^*(3,1) - \Gamma(e) = \{f, g, h\} - \{f, g, h\} = \emptyset$$

Ponieważ $\Delta^*(3) = \emptyset$ oraz $\nu(3) = 0$, to zachodzi przypadek 2b, a więc

$$F(3) = \{c, d, e\}$$

Ponieważ

$$\Omega_4 - \bigcup_0^3 F(j) = \{f, g, h\}$$

to $\varphi(4) = 2$ oraz $r(4) \neq 0$

Podobnie jak w przypadku poprzednim każdy element zbioru $\{f, g, h\}$ jest wierzchołkiem osobliwym.

Niech $x_4 = f$ to wówczas

$$\Delta^*(4) = \emptyset \quad \text{oraz} \quad \nu(4) = 0$$

a więc na mocy przypadku 2b uzyskujemy

$$F(4) = \{f, g, h\}$$

Ponieważ

$$\Omega_8 - \bigcup_0^4 F(j) = \{k\}$$

to $\varphi(5) = 1$ oraz $r(5) \neq 0$, to zachodzi przypadek 2.

Ponieważ $\Delta^*(5) = \emptyset$ oraz $\nu(5) = 0$, to na mocy punktu b uzyskujemy

$$F(5) = k$$

Uzyskane wyniki dla sieci G_3 zestawiono w poniższej tabeli.

Tabela 3

i	1	2	3	4	5
$\varphi(i)$	1	1	2	2	1
F(i)	a	b	c, d, e	f, g, h	k

Na mocy tabeli 3 uzyskujemy

$$\tau(G_3) = \sum_{1}^{5} \varphi(j) = 7$$

Przystąpimy teraz do wyznaczenia wartości funkcji ρ dla wierzchołków sieci G_3.

Ponieważ $\varphi(1) = \varphi(2) = 1$, to $P(F(i))$ dla $i = 1, 2$ nie istnieje, a więc zachodzi przypadek 2. Na mocy przypadku 2 uzyskujemy:

$$\rho(a) = 1 \quad \text{i} \quad \rho(b) = 2$$

Ponieważ $\varphi(3) = 2$, to $P(F(3))$ istnieje, a więc zachodzi przypadek 1. Ponieważ $F(3) = \{c, d, e\}$ oraz $\chi_3 = e$, to

$$\rho\{c, d\} = 3 \quad \text{i} \quad \rho(e) = 4$$

Ponieważ $\varphi(4) = 2$, to $P(F(4))$ istnieje, a więc zachodzi przypadek 1. Ponieważ $F(4) = \{f, g, h\}$ oraz $\chi_4 = f$, to

$$\rho\{g, h\} = 5 \quad \text{i} \quad \rho(f) = 6$$

Ponieważ $\varphi(5) = 1$, to $P(F(5))$ nie istnieje, a więc zachodzi przypadek 2, co oznacza, że

$$(k) = 7$$

Na mocy uzyskanych wartości funkcji ρ uzyskujemy następujące zbiory Θ dla sieci G_3.

Tabela 4

i	1	2	3	4	5	6	7
$\Theta(i)$	a	b	c, d	e	g, h	f	k

Postępując w podobny sposób jak dla sieci G_3 uzyskujemy dla sieci G_1 następujące wyniki

Tabela 5

i	1	2	3	4	5
$\varphi(i)$	1	1	1	1	1

Tabela 6

i	1	2	3	4	5
Θ (i)	a, c	b, e	d, g	f, h	k

Natomiast dla sieci G_2 następujące wyniki:

Tabela 7

i	1	2	3	4	5
$\varphi(i)$	1	1	1	1	1

Tabela 8

i	1	2	3	4	5
$\Theta(i)$	a, c	b, e	d, g	f, h	k

Na mocy tabel 5 i 7 uzyskujemy

$$\tau(G_1) = \tau(G_2) = 5$$

Zbiory Θ dla sieci G_1, G_2, G_3 przedstawiono na rys. 1. Na podstawie rysunku 1 łatwo się przekonać, że obliczeń reprezentowanych przez sieci G_1, G_2, G_3 nie można wykonać w mniejszej liczbie kroków od liczby kroków wyznaczonych na podstawie obliczenia wartości funkcji τ. W pracy [2] rozpatrywano identyczne sieci, i na podstawie wprowadzonego tam algorytmu oszacowano minimalne liczby kroków dla sieci G_1, G_2, G_3. Uzyskane wyniki pokrywają się z wynikami otrzymanymi w niniejszej pracy.

Literatura

[1] COFFMAN E.G., GRAHAM R.L.: Optimal Scheduling for Two-Processor Systems, Acta Informatica, 1972, t. 1, nr 3, s. 200-213

[2] ROWICKI A.: Algorytmy określające liczbę procesorów oraz czas trwania obliczenia. Algorytmy, zeszyt specjalny, 1974

АЛГОРИТМ ПЛАНИРОВАНИЯ ДВУПРОЦЕССОРНЫХ ВЫЧИСЛЕНИЙ

Резюме

В работе указан алгоритм планирования двупроцессорных вычислений, а также алгоритм определения времени для выполнения вычисления. Алгоритм планирования вычислений - оптимальный в том смысле что он гарантирует самое короткое возможное время для выполнения вычисления. В качестве модели вычисления принято граф ориентированный без петлей.

ALGORITHM OF SCHEDULING FOR TWO-PROCESSORS SYSTEMS

Summary

Given the algorithm of scheduling for two-processors systems and the algorithm of determination of computation time. The algorithm of scheduling for two-processors systems is optimal in the sense it ensures the shortest possible time of computation. An oriented acyclic graph is taken as a modd of computation.

Prace IMM
Zeszyt 2
© 1975.04
681.322.001.1

DWA ALGORYTMY KONSTRUKCJI DETERMINISTYCZNEGO AUTOMATU SKOŃCZONEGO NA PODSTAWIE ZADANEJ GRAMATYKI LEWOSTRONNIE LUB PRAWOSTRONNIE LINIOWEJ

Jacek WITASZEK

Pracę złożono 10.01.1975

W pracy przedstawiono dwa algorytmy konstrukcji deterministycznych automatów skończonych akceptujących języki regularne generowane przez zadane gramatyki odpowiednio prawostronnie względnie lewostronnie liniowe. Wykazano poprawność prezentowanych algorytmów oraz przedstawiono niektóre własności otrzymanych automatów.

1. WSTĘP.

Jednym z problemów, pojawiających się w trakcie konstrukcji translatora, a dokładniej analizatora leksykalnego, jest problem skonstruowania deterministycznego automatu skończonego akceptującego język definiowany przez zadaną gramatykę prawowzględnie lewostronnie liniową.

Dotychczas znane są dwie zasadnicze metody rozwiązywania tego zagadnienia. Pierwsza z nich [1], [5] polega na skonstruowaniu automatu skończonego według prostego algorytmu podanego przez Chomsky'ego [3]. Tak otrzymany automat jest jednak na ogół niedeterministyczny. W metodzie tej dochodzi zatem dodatkowy etap determinizacji oraz ewentualnie jeszcze optymalizacji automatu skończonego.

W drugiej metodzie (patrz np. [6]) stosuje się algorytm Brzozowskiego [2], który od razu daje deterministyczny automat skończony i to o minimalnej liczbie stanów. Jednak punktem wyjścia dla algorytmu Brzozowskiego nie jest gramatyka, lecz wyrażenie regularne. W metodzie dochodzi zatem znowu dodatkowy etap polegający na znalezieniu dla zadanej gramatyki, prawostronnie lub lewostronnie liniowej, odpowiedniego wyrażenia regularnego.

W niniejszej pracy przedstawiono dwa algorytmy konstrukcji deterministycznego automatu skończonego na podstawie zadanej gramatyki. Pierwszy z nich, algorytm 1, znajduje zastosowanie w przypadku gramatyk prawostronnie liniowych, drugi, algorytm 2, w przypadku gramatyk lewostronnie liniowych. Zaletą prezentowanej tutaj metody jest fakt, że nie wymaga ona dodatkowych etapów. Otrzymany w wyniku automat skończony nie jest wprawdzie minimalny, posiada natomiast inną właściwość, istotną dla pewnych zastosowań. Jak pokazano w p. 3 i p. 4 znając ciąg stanów, przez które przechodził ten automat w trakcie analizy słowa danego języka L(G) można określić ciąg produkcji użytych w wywodzie tego słowa w gramatyce G.

2. POJĘCIA I OZNACZENIA

Na początku naszych rozważań podamy definicje pojęć występujących w dalszej treści oraz wprowadzimy oznaczenia. I tak:

Alfabetem nazywamy skończony i niepusty zbiór symboli. Alfabety oznaczać będziemy literą V.

Napisem nad danym alfabetem nazywamy skończony (również pusty) ciąg symboli tego alfabetu. Liczbę elementów tego ciągu nazywamy długością napisu. Długość napisu x oznaczamy przez $|x|$.

Napis utworzony z pustego ciągu symboli - tzw. napis pusty - oznaczamy przez ε. Oczywiście $|\varepsilon| = 0$.

Zbiór wszystkich napisów nad alfabetem V oznaczać będziemy przez V^*.

Gramatyką nazywamy system $< V_T, V_N, S, P >$, w którym V_T i V_N są dwoma rozłącznymi alfabetami, zwanymi odpowiednio alfabetem terminalnym i alfabetem nieterminalnym, S jest wyróżnionym symbolem alfabetu nieterminalnego, zwanym symbolem startowym, zaś P jest skończonym zbiorem tzw. produkcji (patrz niżej).

Gramatyki oznaczamy literą G.

Symbole alfabetu terminalnego nazywamy symbolami terminalnymi; oznaczamy je małymi literami z początku alfabetu łacińskiego, a więc np. a, b, c, itp.

Napisy nad alfabetem terminalnym tj. elementy zbioru V_T^* oznaczamy małymi literami z końca alfabetu łacińskiego, a więc np. x, y, z, itp.

Symbole alfabetu nieterminalnego nazywamy symbolami nieterminalnymi; oznaczamy je dużymi literami z początku alfabetu łacińskiego, np. A, B, C itp.

Sumę alfabetów $V_T \cup V_N$ nazywamy alfabetem gramatyki i oznaczamy literą V. Symbole ze zbioru V, jeżeli nie orzekamy o nich czy należą do V_T, czy też do V_N, oznaczamy dużymi literami z końca alfabetu łacińskiego, a więc np. X, Y, Z itp. Napisy nad alfabetem V oznaczamy małymi literami z początku alfabetu greckiego np. α, β, γ itp.

Produkcją nazywamy parę $< \alpha, \beta >$, taką że w α występuje co najmniej jeden symbol nieterminalny. Zgodnie z ogólnie przyjętą konwencją produkcję $< \alpha, \beta >$ zapisujemy w postaci: $\alpha \longrightarrow \beta$, przy czym α nazywamy lewą, zaś β - prawą stroną tej produkcji. Produkcje oznaczamy literą p, a zapis $n(p)$ oznacza długość prawej strony produkcji p.

Weźmy pod uwagę gramatykę $G = < V_T, V_N, S, P >$. Jeżeli dla pewnych γ i δ istnieją takie $\gamma_1, \gamma_2, \alpha$ i β , że $\gamma = \gamma_1 \alpha \gamma_2$, $\delta = \gamma_1 \beta \gamma_2$ i $\alpha \longrightarrow \beta$ należy do P, to mówimy, że γ wywodzi bezpośrednio δ w gramatyce G i piszemy $\gamma \underset{G}{\Longrightarrow} \delta$ lub $\gamma \Longrightarrow \delta$ - gdy nie ma wątpliwości o jaką gramatykę chodzi.

Jeżeli dla pewnych γ i δ istnieje ciąg $\delta_0, \delta_1 \ldots \delta_n$, $n > 0$, taki, że $\gamma = \delta_0$, $\delta = \delta_n$ oraz $\delta_{i-1} \Longrightarrow \delta_i$, dla $i = 1 \ldots n$, to mówimy, że γ wywodzi δ w gramatyce G i piszemy $\gamma \overset{+}{\underset{G}{\Longrightarrow}} \delta$ lub krócej $\gamma \overset{+}{\Longrightarrow} \delta$, gdy wiadomo o jaką gramatykę chodzi.

Ciąg $\delta_0, \delta_1, \ldots \delta_n$ nazywamy wywodem δ z γ w gramatyce G. Czynność polegającą na przekształceniu δ_{i-1} w δ_i nazywamy krokiem wywodu.

Jeżeli dla pewnych γ i δ bądź $\gamma = \delta$, bądź $\gamma \overset{+}{\underset{G}{\Longrightarrow}} \delta$, to piszemy $\gamma \overset{*}{\underset{G}{\Longrightarrow}} \delta$ lub krócej $\gamma \overset{*}{\Longrightarrow} \delta$.

Językiem definiowanym przez gramatykę G nazywamy zbiór wszystkich takich x, że $S \overset{+}{\Longrightarrow} x$. Język definiowany przez G oznaczamy przez L (G).

Gramatykę G nazywamy prawostronnie liniową jeżeli wszystkie jej produkcje są bądź postaci $A \longrightarrow x\,B$, bądź postaci $A \longrightarrow x$.

Gramatykę G nazywamy lewostronnie liniową jeżeli wszystkie jej produkcje są bądź postaci $A \longrightarrow B\,x$ bądź postaci $A \longrightarrow x$.

Konfiguracją produkcji p w gramatyce G lub krócej konfiguracją nazywamy parę $<p, j>$, taką że j jest liczbą całkowitą z przedziału: 1, n(p) + 1; j nazywamy wskaźnikiem konfiguracji.

Symbolem wyznaczonym przez konfigurację $<p, j>$ nazywamy symbol j prawej strony p, gdy $j \leqslant n(p)$. Gdy $j = n(p) + 1$ wówczas mówimy, że konfiguracja $<p, j>$ wyznacza koniec produkcji p.

Jeżeli G jest gramatyką prawostronnie liniową i $<p, j>$ jest konfiguracją w tej gramatyce, to prawym uzupełnieniem $<p, j>$ jest zbiór konfiguracji, który definiujemy następująco:

a) gdy $<p, j>$ wyznacza symbol terminalny lub koniec produkcji, wówczas prawe uzupełnienie tej konfiguracji składa się z $<p, j>$;

b) gdy $<p, j>$ wyznacza symbol nieterminalny, powiedzmy A, wówczas prawe uzupełnienie składa się z $<p, j>$ oraz wszystkich konfiguracji $<B \longrightarrow \alpha, 1>$, takich że $B \longrightarrow \alpha$ jest produkcją w G i $A \overset{*}{\Longrightarrow} B$.

Jeżeli G jest gramatyką lewostronnie liniową zaś $<p, j>$ jest konfiguracją w tej gramatyce, to lewym uzupełnieniem p, j jest zbiór konfiguracji, który definiujemy następująco:

a) gdy $<p, j>$ nie wyznacza końca produkcji, wówczas lewe uzupełnienie tej konfiguracji składa się z $<p, j>$;

b) gdy $<p, j>$ wyznacza koniec produkcji, wówczas lewe uzupełnienie tej konfiguracji składa się z $<p, j>$ oraz z wszystkich konfiguracji postaci $<C \longrightarrow B\,x, 2>$, takich, że $C \longrightarrow B\,x$ jest produkcją w G oraz $B \overset{*}{\Longrightarrow} A$, gdzie A oznacza lewą stronę p.

Weźmy pod uwagę dowolny zbiór konfiguracji Q. Prawym (lewym) uzupełnieniem Q nazywamy sumę prawych (lewych) uzupełnień poszczególnych jego elementów, Prawe uzupełnienie Q oznaczać będziemy przez $\check{Q}$ natomiast lewe przez $\hat{Q}$.

Automatem skończonym nazywamy system $<K, V_I, \delta, q_o, F>$, gdzie K jest skończonym i niepustym zbiorem stanów automatu. Stany oznaczać będziemy przez q. V_I jest alfabetem wejściowym, δ jest funkcją częściową: $K \times V_I \longrightarrow 2^K$ zwaną funkcją przejścia, q_o jest wyróżnionym stanem, zwanym stanem początkowym, F jest podzbiorem zbioru stanów i zwane jest zbiorem stanów końcowych. Automaty skończone oznaczać będziemy przez M.

Symbole alfabetu wejściowego nazywamy symbolami wejściowymi i oznaczamy małymi literami z początku alfabetu wejściowego np. a, b, c; napisy nad V_I nazywamy napisami wejściowymi i oznaczamy małymi literami z końca alfabetu łacińskiego, a więc x, y, z itp.

Funkcję δ umówimy się rozszerzać na dziedzinę $K \times V_I^*$ przyjmując:

$$\delta(q, \varepsilon) = q$$

$$\delta(q, x\,a) = \bigcup_{q' \in \delta(q, x)} \delta(q', a)$$

Językiem akceptowanym przez automat skończony $M = < K, V_I, \delta, q_o, K >$ nazywamy zbiór wszystkich takich x, że $\delta(q_o, x)$ jest określone i przecięcie $\delta(q_o, x)$ z F jest niepuste. Język ten oznaczamy przez L(M).

Automat skończony nazywamy deterministycznym jeżeli $\delta(q, a)$ tam gdzie jest określone, jest zbiorem jednoelementowym.

3. ALGORYTM 1

Niech dana będzie gramatyka prawostronnie liniowa $G = < V_T, V_N, S, P >$. Dla gramatyki tej konstruujemy automat skończony $M(G) = < K, V_I, \delta, q_o, F >$ przyjmując $V_I = V_T$ oraz tworząc K (którego elementy są zbiorami konfiguracji), q_o, δ oraz F według następującego algorytmu:

1. utwórz zbiór Q_o wszystkich konfiguracji $< p, 1 >$, takich że lewa strona p jest symbolem startowym G. Przyjmij $q_o = \check{Q}_o$ oraz $K = \{ q_o \}$

2. niech $q \in K$ i niech $a_1, a_2, \ldots, a_n$ jest zbiorem symboli terminalnych wyznaczonych przez konfiguracje należące do q.

   Dla każdego a_i wykonaj następujące czynności:

2.1. utwórz zbiór Q_{ai} złożony z wszystkich takich konfiguracji, które należą do q i wyznaczają a_i;

2.2. we wszystkich konfiguracjach należących do Q_{ai} zwiększ wartość wskaźnika o 1;

2.3. przyjmij $q' = \check{Q}_{ai}$, $K = K \cup \{ q' \}$ oraz $\delta(q, a_i) = q'$

Powtarzaj p. 2 aż do wyczerpania wszystkich $q \in K$.

3. utwórz zbiór stanów końcowych F. F składa się z wszystkich takich stanów, które zawierają co najmniej jedną konfigurację wyznaczającą koniec produkcji.

Jak łatwo zauważyć proces opisany powyższym algorytmem zawsze się zakończy, skończona jest bowiem liczba wszystkich konfiguracji w danej gramatyce.

Przykład 1

Niech dla gramatyki G jest:

$V_T = \{ c, d \}$
$V_N = \{ A, S \}$
$P = \{ S \longrightarrow c\,d\,S,\ S \longrightarrow A,\ A \longrightarrow c\,A,\ A \longrightarrow \varepsilon \}$

Zgodnie z algorytmem 1 dla M(G) przyjmujemy:
$V_I = V_T = \{ c, d \}$ po czym przechodzimy do wykonania p. 1.
Daje on kolejno:

$$Q_o = \{ \langle S \longrightarrow c\,d\,S, 1 \rangle,\ \langle S \longrightarrow A, 1 \rangle \},$$

$$q_o = \check{Q}_o = \{ \langle S \longrightarrow c\,d\,S, 1 \rangle,\ \langle S \longrightarrow A, 1 \rangle,\ \langle A \longrightarrow c\,A, 1 \rangle,\ A \longrightarrow \varepsilon, 1 \rangle \}$$ oraz

$K = \{ q_o \}$.

Na razie jedynym elementem K jest q_o więc dla q_o wykonujemy p. 2.

Jedynym symbolem terminalnym wyznaczonym przez konfiguracje należące do q_o jest symbol c.

W wyniku wykonania dla symbolu c czynności podanych w p. 2.1, 2.2 i 2.3 otrzymujemy kolejno:

$$Q_{oc} = \{ \langle S \longrightarrow c\,d\,S, 1 \rangle,\ \langle A \longrightarrow c\,A, 1 \rangle \}$$

a następnie

$$Q_{oo} = \{ \langle S \longrightarrow c\ d\ S, 2\rangle \quad \langle A \longrightarrow c\ A, 2\rangle \},$$

$$q_1 = \overset{\times}{Q}_{oo} = \{ \langle S \longrightarrow c\ d\ S, 2\rangle \quad \langle A \longrightarrow c\ A, 2\rangle \quad \langle A \longrightarrow c\ A, 1\rangle \quad \langle A \longrightarrow \varepsilon, 1\rangle \},$$

$$K = K \cup \{q_1\} = \{q_0, q_1\} \quad \text{oraz}$$
$$\delta(q_0, c) = q_1$$

Teraz do K należą już dwa elementy. Ponieważ dla q_0 czynności p. 2 zostały już wykonane należy wykonać p. 2 dla q_1. Wyniki wykonania p. 1 i p. 2 algorytmu 1 zamieszczono w tabeli 1.

Tabela 1

i	q_i		a	$\delta(q_i, a)$	rodzaj stanu
0	$\langle S \longrightarrow c\ d\ S, 1\rangle$ $\langle S \longrightarrow A, 1\rangle$ $\langle A \longrightarrow c\ A, 1\rangle$ $\langle A \longrightarrow \varepsilon, 1\rangle$	Q_0	c	q_1	końcowy
1	$\langle S \longrightarrow c\ d\ S, 2\rangle$ $\langle A \longrightarrow c\ A, 2\rangle$ $\langle A \longrightarrow c\ A, 1\rangle$ $\langle A \longrightarrow \varepsilon, 1\rangle$	$Q_{oo} = Q_{3o}$	c d	q_2 q_3	końcowy
2	$\langle A \longrightarrow c\ A, 2\rangle$ $\langle A \longrightarrow c\ A, 1\rangle$ $\langle A \longrightarrow \varepsilon, 1\rangle$	$Q_{1o} = Q_{2o}$	c	q_2	końcowy
3	$\langle S \longrightarrow c\ d\ S, 3\rangle$ $\langle S \longrightarrow c\ d\ S, 1\rangle$ $\langle S \longrightarrow A, 1\rangle$ $\langle A \longrightarrow c\ A, 1\rangle$ $\langle A \longrightarrow \varepsilon, 1\rangle$	Q_{1d}	c	q_1	końcowy

Ostatnią czynnością jest wyznaczenie zbioru stanów końcowych - p. 3. Ponieważ, jak to widać z tabeli 1, w każdym ze stanów występuje konfiguracja wyznaczająca koniec produkcji, mianowi-

cie konfiguracja $<A \longrightarrow \varepsilon, 1>$, więc wszystkie stany M(G) są również jego stanami końcowymi czyli F = K.

Koniec przykładu 1

Zajmiemy się teraz omówieniem własności automatu M(G) skonstruowanego według podanego wyżej algorytmu 1. Przede wszystkim należy zauważyć, że M(G) jest automatem deterministycznym. Wynika to wprost z konstrukcji funkcji δ (p. 23) . Dla M(G) zachodzą również dwa następujące lematy:

Lemat 1

Jeżeli dla pewnych $a_1 \ldots a_r$ oraz q mamy: $\delta(q_0, a_1 \ldots a_r) = q$, to każda konfiguracja należąca do q jest postaci:

$$<A \longrightarrow a_{s+1} \ldots a_r \alpha, r - s + 1>$$

dla pewnych $s \leqslant r$, A oraz α , takich że

$$S \overset{*}{\Longrightarrow} a_1 \ldots a_s A \Longrightarrow a_1 \ldots a_r \alpha$$

Dowód lematu przeprowadzimy przez indukcję ze względu na r.

Dla r = 0, jak widać, teza lematu zachodzi mamy bowiem $q = q_0$, a zgodnie z algorytmem 1 każda konfiguracja należąca do q_0 jest postaci $<A \longrightarrow \alpha , 1>$ dla A takiego, że $S \overset{*}{\Longrightarrow} A$. Ponieważ $A \longrightarrow \alpha$ jest produkcją w G, więc dla A i α mamy: $S \overset{*}{\Longrightarrow} A \Longrightarrow \alpha$.

Przypuśćmy teraz, że teza lematu zachodzi dla r mniejszych od pewnego n i przyjmijmy, że r = n i że

$$\delta(q_0, a_1 \ldots a_n) = q \qquad (1)$$

Zgodnie z definicją rozszerzenia δ istnieje takie q', że:

$$\delta(q_0, a_1 \ldots a_{n-1}) = q' \quad \text{i} \quad \delta(q', a_n) = q \qquad (2)$$

Weźmy dowolną konfigurację $<p, j>$ należącą do q. Ponieważ $\delta(q', a_n) = q$, więc zgodnie z algorytmem 1 do q' musi należeć konfiguracja $<p', i>$ wyznaczająca a_n i taka, że $<p, j>$ należy do prawego uzupełnienia $<p', i + 1>$. Z za-

łożenia indukcyjnego wynika, że $<p', i>$ musi być postaci

$$<B \longrightarrow a_{t+1} \dots a_{n-1} \beta , \quad n - t> \tag{3}$$

dla pewnych B, $t \leqslant n - 1$ oraz β, takich że

$$S \overset{*}{\Longrightarrow} a_1 \dots a_t B \Longrightarrow a_1 \dots a_{n-1} \beta. \tag{4}$$

Ponadto zgodnie z tym co powiedziano wyżej musi być

$$\beta = a \beta' \tag{5}$$

W celu stwierdzenia czy $<p, j>$ spełnia tezę lematu rozważymy dwa następujące przypadki:

a) $<p, i + 1>$ wyznacza symbol terminalny lub koniec produkcji. W tym wypadku $<p, j> = <p', i + 1>$, skąd wobec (3), (4) i (5) mamy:

$$<p, j> = <B \longrightarrow a_{t+1} \dots a_{n-1} a_n \beta' , n - t + 1>$$

oraz

$$S \overset{*}{\Longrightarrow} a_1 \dots a_t B \Longrightarrow a_1 \dots a_n \beta' ,$$

czyli $<p, j>$ spełnia tezę lematu.

b) $<p', i + 1>$ wyznacza symbol nieterminalny, tj. istnieje takie C, że

$$\beta' = C. \tag{6}$$

Zgodnie z definicją prawego uzupełnienia może być bądź

$<p, j> = <p', i + 1>$ i wówczas teza lematu zachodzi z tych samych co w a) względów, bądź

$<p, j> = <A \longrightarrow \alpha , 1>$ dla pewnych A i α takich, że $A \longrightarrow \alpha$ jest produkcją w G oraz

$$C \overset{*}{\Longrightarrow} A \tag{7}$$

Ale na podstawie (4), (5), (6) i (7) mamy:

$$S \overset{*}{\Longrightarrow} a_1 \dots a_t B \Longrightarrow a_1 \dots a_n C \overset{*}{\Longrightarrow} a_1 \dots a_n A \Longrightarrow a_1 \dots a_n \alpha$$

a to oznacza, że teza lematu zachodzi również i w tym wypadku.

Koniec dowodu.

Lemat 2

Jeżeli w G zachodzi

$S \overset{*}{\Longrightarrow} a_1 \dots a_s A \Longrightarrow a_1 \dots a_r \alpha$ dla pewnych $a_1, a_2 \dots a_r$, $s \leqslant r$, A oraz α, to dla M(G) $\delta(q_0, a_1 \dots a_r)$ jest określone i jeżeli wartość $\delta(q_0, a_1 \dots a_r)$ oznaczyć przez q, to do q należy konfiguracja:

$$< A \longrightarrow a_{s+1} \dots a_r \alpha \ , \ r - s + 1 >$$

Dowód lematu przeprowadzimy przez indukcję ze względu na r.

Dla r = 0 mamy oczywiście $\delta(q_0, \varepsilon) = q_0$, a ponieważ z założenia jest

$$S \overset{*}{\Longrightarrow} A \Longrightarrow \alpha \ ,$$

więc zgodnie z algorytmem 1 do q_0 musi należeć konfiguracja $< A \longrightarrow \alpha \ , \ 1 >$.

Załóżmy teraz, że teza lematu zachodzi dla r mniejszych od n i że

$$S \overset{*}{\Longrightarrow} a_1 \dots a_s A \Longrightarrow a_1 \dots a_n \alpha \qquad (8)$$

Wykażemy, że teza lematu zachodzi również dla r = n. Zauważmy przede wszystkim, że zgodnie z założeniem indukcyjnym istnieje takie q', że

$$\delta(q_0, a_1 \dots a_{n-1}) = q'$$

Rozważymy dwa następujące przypadki

a) $s \leqslant n - 1$. W tym wypadku z założenia indukcyjnego i z (8) wynika, że do q' należy konfiguracja

$$< A \longrightarrow a_{s+1} \ldots a_n \alpha,\ n - s > .$$

Jak widać wyznacza ona symbol a_n, a zatem zgodnie z algorytmem 1 wartość $\delta(q', a_n)$ jest określona i jeżeli oznaczyć ją przez q, to do q należy konfiguracja

$$< A \longrightarrow a_{s+1} \ldots a_n \alpha,\ n - s + 1 >$$

b) $s = n$. W tym wypadku (8) musi mieć postać

$$S \overset{*}{\Longrightarrow} a_1 \ldots a_t\, C \Longrightarrow a_1 \ldots a_n\, B \overset{*}{\Longrightarrow} a_1 \ldots a_n\, A \Longrightarrow a_1 \ldots a_n \alpha \quad (9)$$

dla pewnych B, C oraz pewnego $t \leqslant s - 1 = n - 1$.

Jak wynika z założenia indukcyjnego, do q' należy konfiguracja:

$$< C \longrightarrow a_{t+1} \ldots a_n\, B,\ n - t > .$$

Wyznacza ona a_n, a więc jak to wynika z algorytmu 1 wartość $\delta(q', a_n)$ jest określona. Oznaczmy tę wartość przez q. Na podstawie algorytmu 1 do q należą wszystkie konfiguracje składające się na uzupełnienie konfiguracji:

$< C \longrightarrow a_{t+1} \ldots a_n\, B,\ n - t + 1 >$, a więc wobec (9) również i konfiguracja

$$< A \longrightarrow \alpha ,\ 1 > .$$

Koniec dowodu.

Na podstawie podanych wyżej lematów udowodnimy następujące twierdzenie:

Twierdzenie 1

Automat skończony M(G) akceptuje dokładnie język definiowany przez gramatykę G.

Dowód:

Weźmy pod uwagę pewien napis $x = a_1 \dots a_n$ i załóżmy, że M(G) akceptuje x, tzn. $\delta(q_0, x) = q$ dla pewnego q należącego do F.

Na podstawie lematu 1 każda konfiguracja należąca do q jest postaci:

$< B \longrightarrow a_{t+1} \dots a_n \alpha , n - t + 1 >$ dla pewnych B, α oraz $t \leqslant n$, takich że

$$S \overset{*}{\Longrightarrow} a_1 \dots a_t B \Longrightarrow a_1 \dots a_n \alpha .$$

Ponieważ q jest stanem końcowym, więc co najmniej jedna z tych konfiguracji musi wyznaczać koniec produkcji.

Istnieją zatem takie $s \leqslant n$ oraz A, że

$$< A \longrightarrow a_{s+1} \dots a_n, n - s + 1 >$$

należy do q i że

$$S \overset{*}{\Longrightarrow} a_1 \dots a_s A \Longrightarrow a_1 \dots a_n . \qquad (10)$$

Ale (10) oznacza że x należy do L(G).

Załóżmy teraz, że $x = a_1 \dots a_n$ jest słowem języka L(G). Mamy zatem

$S \overset{*}{\Longrightarrow} a_1 \dots a_t A \Longrightarrow a_1 \dots a_n$ dla pewnego A oraz $t \leqslant n$. Zgodnie z lematem 2 oznacza to, że wartość $\delta(q_0, x)$ jest określona i jeżeli oznaczyć ją przez q, to do q należy między innymi konfiguracja

$$< A \longrightarrow a_{t+1} \dots a_n, n - t + 1 >$$

Ponieważ konfiguracja ta wyznacza koniec produkcji, więc q jest stanem końcowym, a to oznacza, że M(G) akceptuje x.

Koniec dowodu.

Na wstępie wspomnieliśmy o jeszcze jednej własności M(G): na podstawie ciągu stanów, przez które przechodził M(G) anali-

zując słowo języka L(G), można określić ciąg produkcji użytych w kolejnych krokach wywodu tego słowa.

Pokażemy teraz jak efektywnie znaleźć wspomniany ciąg produkcji.

Przypuśćmy, że M(G) - analizując zadany napis $x \in L(G)$ - znajdował się kolejno w stanach:

$$q^0, q^1, \ldots q^n.$$

W celu znalezienia ciągu produkcji użytych w kolejnych krokach wywodu słowa x w G należy wykonać czynności według następującego algorytmu:

Algorytm 11

1. Znajdź w q^n konfigurację wyznaczającą koniec produkcji. Ponieważ $x \in L(G)$ więc wobec twierdzenia 1 $q^n \in F$ i konfiguracja taka musi należeć do q^n. Produkcja wchodząca w skład tej konfiguracji jest ostatnim elementem poszukiwanego ciągu.
   Oznacz przez A lewą stronę tej produkcji, a przez l - długość jej prawej strony.
   Przyjmij r = n.

2. Przyjmij r = r - l.

3. Sprawdź czy r = 0 i A = S;
   Jeżeli tak, to zakończ proces poszukiwania produkcji stosowanych w wywodzie.

4. W stanie q^r znajdź konfigurację, która wyznacza A. Produkcja wchodząca w skład tej konfiguracji jest kolejnym - licząc od końca - elementem poszukiwanego ciągu.

5. Oznacz przez A lewą stronę znalezionej w p. 4 produkcji, a przez l - liczbę symboli terminalnych w jej prawej stronie. Przejdź do p. 2.

Poprawność powyższego algorytmu jest prostym wnioskiem z lematów 1 i 2. Z lematów tych wynika również, że jeżeli gramatyka

jest niejednoznaczna, to wynik opisanych wyżej czynności może również być niejednoznaczny.

Przykład 2

Weźmy pod uwagę gramatykę G i automat M(G) z przykładu 1 oraz napis: c d c c.

M(G) analizując ten napis przechodzi kolejno przez stany:

$$q^0 = q_0,\ q^1 = q_1,\ q^2 = q_3,\ q^3 = q_1,\ q^4 = q_2.$$

W stanie $q^4 = q_2$ występuje jedna konfiguracja wyznaczająca koniec produkcji, a mianowicie $< A \longrightarrow \varepsilon\ ,\ 1 >$. W związku z tym $A \longrightarrow \varepsilon$ jest ostatnią produkcją poszukiwanego ciągu. Ponieważ długość prawej strony tej produkcji jest równa zeru, więc w tym samym stanie poszukujemy konfiguracji, które wyznaczałaby A. Jest nią jak widać z tabeli 1 konfiguracja $< A \longrightarrow c\ A,\ 2 >$.

Przedostatnią produkcją poszukiwanego ciągu jest więc $A \longrightarrow c\ A$. Dla produkcji tej jest $l = 1$, przechodzimy więc do stanu $q^3 = q_1$ i poszukujemy w nim konfiguracji wyznaczającej A. Jest nią $< S \longrightarrow c\ A,\ 2 >$ i w związku z tym trzecią - licząc od końca - produkcją w poszukiwanym ciągu jest produkcja $A \longrightarrow c\ A$. Dla produkcji tej znowu $l = 1$. Rozważamy więc stan $q^2 = q_3$.

W stanie tym konfiguracją wyznaczającą A jest $S \longrightarrow A,\ 1$. Otrzymujemy zatem produkcję $S \longrightarrow A$. Ponieważ dla produkcji tej jest $l = 0$, więc rozważamy znowu stan q_3, tym razem poszukując konfiguracji wyznaczającej S. Jest nią $< S \longrightarrow c\ d\ S,\ 3 >$. Kolejną, licząc od końca, produkcją, jest zatem $S \longrightarrow c\ d\ S$. Dla produkcji tej $l = 2$. W wyniku wykonania czynności opisanych w p. 2 przechodzimy teraz do stanu $q^0 = q_0$, a ponieważ spełniony jest warunek badany w p-cie 3 - kończymy poszukiwania. Otrzymany ciąg produkcji

$$S \longrightarrow c\ d\ S,\quad S \longrightarrow A,\quad A \longrightarrow c\ A,\quad A \longrightarrow c\ A,\quad A \longrightarrow \varepsilon$$

jest, jak łatwo zauważyć, ciągiem produkcji zastosowanych w kolejnych krokach wywodu słowa c d c c w gramatyce G.

4. ALGORYTM 2

Niech $G = \langle V_T, V_N, S, P \rangle$ jest gramatyką lewostronnie liniową. Dla gramatyki tej tworzymy automat skończony $M(G) = \langle K, V_I, \delta, q_o, F \rangle$, przyjmując $V_I = V_T$ oraz konstruując K (którego elementy są zbiorami konfiguracji), δ, q_o i F według następującego algorytmu:

1. Utwórz zbiór Q_o wszystkich takich konfiguracji $\langle p, 1 \rangle$, że prawa strona p nie zawiera symbolu nieterminalnego. Przyjmij $q_o = \hat{Q}_o$ oraz $K = \{q_o\}$.

2. Niech $q \in K$ i niech $a_1 \ldots a_n$ jest zbiorem symboli wyznaczonych przez konfiguracje należące do q. Dla każdego a_i wykonaj następujące czynności:
   2.1. Utwórz zbiór Q_{ai} złożony z wszystkich takich konfiguracji, które należą do q i wyznaczają a_i.
   2.2. We wszystkich konfiguracjach w Q_{ai} zwiększ wartość wskaźnika o 1.
   2.3. Przyjmij $q' = \hat{Q}_{ai}$, $K = K \cup \{q'\}$ oraz $\delta(q, a_i) = q'$.

Powtarzaj p. 2 aż do wyczerpania wszystkich $q \in K$.

3. Utwórz zbiór stanów końcowych F. F składa się z tych stanów, które zawierają co najmniej jedną konfigurację postaci $\langle S \longrightarrow \alpha, |\alpha| + 1 \rangle$ tj. wyznaczającą koniec takiej produkcji, której lewa strona jest symbolem startowym.

Skonstruowany według powyższego algorytmu automat M(G) jest oczywiście automatem deterministycznym. Ponadto dla M(G) zachodzą:

Lemat 3

Jeżeli dla pewnych $a_1 \ldots a_n$ oraz q mamy: $\delta(q_o, a_1 \ldots a_n) = q$, to każda konfiguracja należąca do q jest bądź postaci:

$\langle A \longrightarrow a_1 \ldots a_n x, n + 1 \rangle$ dla pewnych A oraz x,

bądź postaci:

$<A \longrightarrow B\ a_{t+1} \ldots a_n\ x,\ n - t + 2>$ dla pewnych A, B, x oraz $t \leqslant n$, takich, że

$$A \Longrightarrow B\ a_{t+1} \ldots a_n\ x \overset{*}{\Longrightarrow} a_1 \ldots a_n\ x.$$

Lemat 4

Jeżeli dla pewnych $a_1 \ldots a_n$, x, A, B oraz $t \leqslant n$ bądź

$$A \Longrightarrow B\ a_{t+1} \ldots a_n\ x \overset{*}{\Longrightarrow} a_1 \ldots a_n\ x,$$

bądź

$$A \Longrightarrow a_1 \ldots a_n\ x,$$

to istnieje takie q, że $q = \delta(q_0, a_1 \ldots a_n)$ i do q należy odpowiednio bądź konfiguracja

$$<A \longrightarrow B\ a_{t+1} \ldots a_n\ x,\ n - t + 2>,$$

bądź konfiguracja

$$<A \longrightarrow a_1 \ldots a_n\ x,\ n + 1>$$

Twierdzenie 2

Językiem akceptowanym przez M(G) jest język definiowany przez G.

Własności M(G) podane powyżej w formie lematów oraz twierdzenia dowodzi się w podobny sposób jak odpowiednie własności automatu skonstruowanego według algorytmu 1. W związku z tym dowodów lematów 3 i 4 oraz twierdzenia 2 nie będziemy już tutaj zamieszczali.

Podobnie jak w przypadku automatu skonstruowanego według algorytmu 1, tak i tutaj, znając ciąg stanów, przez które przechodził M(G) analizując zadane $x \in L(G)$, można określić ciąg produkcji użytych w kolejnych krokach wywodu x w gramatyce G. Jeżeli przez q^0, $q^1 \ldots q^n$ oznaczyć kolejne stany, w których

znajdował się automat M(G) w trakcie analizy słowa x, to odpowiedni algorytm jest następujący:

Algorytm 21

1. W stanie q^n znajdź konfigurację $\langle p, n(p) + 1\rangle$, taką, że lewa strona p jest symbolem startowym. Konfiguracja taka istnieje ponieważ wobec $x \in L(G)$ i twierdzenia 2 q^n jest stanem końcowym. Tak znalezione p jest produkcją użytą w pierwszym kroku wywodu x. Przyjmij r = n.

2. Sprawdź czy ostatnio znaleziona produkcja p jest postaci $A \longrightarrow y$ dla pewnego A i y. Jeżeli tak, to zakończ proces.

3. Oznacz przez B pierwszy symbol prawej strony ostatnio znalezionej produkcji p, a przez l - jej długość.

4. Przyjmij r = r - l + 1.

5. W stanie q^r znajdź konfigurację $\langle p, n(p) + 1\rangle$, taką, że lewa strona p jest równa B. Tak znaleziona produkcja p jest produkcją zastosowaną w kolejnym kroku wywodu x. Przejdź do p. 2.

Przykład 3

Niech dla $G = \langle V_T, V_N, S, P\rangle$ jest

$$V_T = \{c, d\}$$
$$V_N = \{A, S\}$$
$$P = \{S \longrightarrow S\,c$$
$$S \longrightarrow A$$
$$A \longrightarrow A\,c\,d$$
$$A \longrightarrow \varepsilon\}.$$

Wynik zastosowania algorytmu 2 w przypadku tak zdefiniowanej G przedstawiono w tabeli 2.

Tabela 2

i	q_i	a	$\delta(q_i, a)$	rodzaj stanu
0	$\langle A \longrightarrow \varepsilon, 1\rangle$ } Q_0 $\langle A \longrightarrow A\ c\ d, 2\rangle$ $\langle S \longrightarrow A, 2\rangle$ $\langle S \longrightarrow S\ c, 2\rangle$	c	q_1	końcowy
1	$\langle A \longrightarrow A\ c\ d, 3\rangle$ $\langle S \longrightarrow S\ c, 3\rangle$ } $Q_{0c} = Q_{3c}$ $\langle S \longrightarrow S\ c, 2\rangle$	c d	q_2 q_3	końcowy
2	$\langle S \longrightarrow S\ c, 3\rangle$ } $Q_{1c} = Q_{2c}$ $\langle S \longrightarrow S\ c, 2\rangle$	c	q_2	końcowy
3	$\langle A \longrightarrow A\ c\ d, 4\rangle$ } Q_{1d} $\langle A \longrightarrow A\ c\ d, 2\rangle$ $\langle S \longrightarrow A, 2\rangle$ $\langle S \longrightarrow S\ c, 2\rangle$	c	q_1	końcowy

Weźmy teraz pod uwagę napis c d c c c, który - jak łatwo widać - jest słowem języka L(G). Jak to wynika z tabeli 2 automat M(G) analizując napis c d c c c znajdzie się kolejno w stanach:

$$q_0, q_1, q_3, q_1, q_2, q_2$$

Stąd, na podstawie algorytmu 21, otrzymujemy następujący ciąg produkcji zastosowanych w kolejnych krokach wywodu c d c c c:

$S \longrightarrow Sc$, $S \longrightarrow Sc$, $S \longrightarrow Sc$, $S \longrightarrow A$, $A \longrightarrow A\ c\ d$, $A \longrightarrow \varepsilon$.

Literatura

[1] BLIKLE A.: Automaty i Gramatyki, PWN 1971

[2] BRZOZOWSKI J.: Derivatives of Regular Expressions, Journal of ACM, paźdz. 1964, t. 11, nr 4

[3] CHOMSKY N., MILLER C.: Finite State Languages, Information and Control 1958, t. 1, nr 2, s. 91-112

[4] HOPCROFT J., ULLMAN J.: Formal Languages and their Relation to Automata, Addisson-Weseley, 1969

[5] IGLEWSKI M., KRZEMIEŃ R.: Analiza leksykalna; Prace CO PAN, 1973, nr 122

[6] WALIGÓRSKI S.: Algebraiczna teoria automatów, Algorytmy, 1969, t. VI, nr 11

ДВА АЛГОРИТМА КОНСТРУКЦИИ ДЕТЕРМИНИСТИЧЕСКОГО КОНЕЧНОГО АВТОМАТА НА ОСНОВЕ ЗАДАННОЙ ЛЕВОСТРОННОЙ ИЛИ ПРАВОСТРОННОЙ ЛИНЕЙНОЙ ГРАММАТИКИ

Резюме

В статье указаны два алгоритма конструкции детерминистических конечных автоматов воспринимающих регулярные языки генерированные заданными грамматиками соответственно правостронне или левостронне линейными. Указана правильность представленных алгоритмов, а также некоторые свойства полученных автоматов.

TWO ALGORITHMS CONSTRUCTING DETERMINISTIC FINITE STATE AUTOMATA FOR RIGHT OR LEFT LINEAR GRAMMARS

Summary

The paper presents two algorithms constructing deterministic finite state automata, which accept regular languages defined by given, left or right linear grammars, respectively. Correctness of these algorithms is proved and same properties of the obtained automata are demonstrated.

Prace IMM
Zeszyt 2
© 1975.04
681.322.06:800.92

LL — REGULAR GRAMMARS (SUFFICIENT CONDITIONS)

Stanisław JARZĄBEK
Tomasz KRAWCZYK

Received 2nd Dec. 1974

In the first part of our thesis on LLR grammars apart from definitions, some necessary conditions for a grammar to be LL-regular were given. This part contains some sufficient conditions for a grammar to be LLR.

INTRODUCTION

The syntactic analysis of context-free languages is continually being worked upon in the professional literature. The goal of those papers is to find possibly general and possibly effective algorithms for syntactic analysis. In general, these two aspects are contradictory. For this reason, as a rule, one chooses algorithm effective enough, which can analyse only a subclass of context-free languages. By the way of choosing such an algorithm the LL(k), LLS(k) and LR(k) languages were described. All algorithms for syntactic analysis can be divided into two classes: bottom-up algorithms and top-down algorithms. Following the example of K. Culik and R. Cohen in the case of bottom-up algorithms, we introduced a new class of languages - LLR languages, which can be parsed in top-down manner without backtrack and properly contains the class of LL(k) languages. This paper contains some sufficient conditions for a grammar to be LLR.

Below are some basic notations and definitions.

We represent a context-free grammar G by a quadruple (V_N, V_T, P, S), where V_N and V_T are finite disjoint sets of symbols called nonterminal and terminal alphabets respectively, P is a finite set of the productions of the form $A \longrightarrow \alpha$ with $A \in V_N$ and $\alpha \in (V_N \cup V_T)^*$ and $S \in V_N$ is the starting symbol. By V we denote $V_N \cup V_T$.

For γ_1 and $\gamma_2 \in V^*$ $\gamma_1 \underset{G}{\Longrightarrow} \gamma_2$ iff there exist β, $\delta \in V_T^*$ and production $A \longrightarrow \alpha$ in P such that $\gamma_1 = \beta A \delta$ and $\gamma_2 = \beta \alpha \delta$.

We write $\gamma_1 \underset{G}{\overset{}{\Longrightarrow}}_L \gamma_2$ if β is in V_T^*.

We assume that $\overset{*}{\underset{G}{\Longrightarrow}}$ ($\overset{*}{\underset{G}{\Longrightarrow}}_L$) represents a reflexive transitive completion of $\underset{G}{\Longrightarrow}$ ($\underset{G}{\Longrightarrow}_L$). The string $\alpha \in V^*$ such that $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha$ will be called <u>the sentential form.</u>

<u>The language</u> generated by G is the following set $L(G) = \{ x \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} x \}$, where S is the starting symbol. The set of productions with the same nonterminal symbol on the left side is called <u>a rule</u>.

Upper case Latin letters will denote nonterminals,
letters: a, b, c, d, ... - terminals
letters: u, v, w, x ... - will denote strings belonging to V_T^*
and $\alpha, \beta, \gamma, \delta, \ldots$ - will denote strings belonging to V^*.

For any string δ we define k: δ to be the string consisting of the first k symbols of δ, if δ has more than k symbols or δ in the other case.

<u>Definition 1</u>

Let $k > 0$. A grammar $G = (V_N, V_T, P, S)$ is said to be LLS(k) grammar iff for any strings $u_1, u_2, x_1, x_2 \in V_T^*$ and any strings $\alpha_1, \alpha_2, \delta_1, \delta_2 \in V^*$ from

$$S \overset{*}{\underset{G}{\Longrightarrow}} u_1 A \delta_1 \underset{G}{\Longrightarrow}_L u_1 \alpha_1 \delta_1 \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 x_1$$

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 A \delta_2 \underset{G}{\Longrightarrow}_L u_2 \alpha_2 \delta_2 \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 x_2$$

and

$k : x_1 = k : x_2$ we have $\alpha_1 = \alpha_2$

Let $\pi = \{R_1, \ldots, R_n\}$ be a partition of V_T^* into n disjoint sets R_i. Set π is called a regular partition if all sets R_i are regular.

Definition 2

Let $G = (V_N, V_T, P, S)$ be any CF grammar with k rules and let U denote a rule in P:

$$U = \{A \longrightarrow \alpha_1, \ldots, A \longrightarrow \alpha_n\}.$$

Let $\pi = \{R_1, \ldots, R_n\}$ denote a regular partition of V_T^*.

The rule U is called a LL (π) rule iff for any two derivations of the form

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 A \delta_1 \underset{G}{\Longrightarrow}_L u_1 \alpha' \delta_1 \overset{*}{\underset{G}{\Longrightarrow}}_L u_1 x_1$$

$$S \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 A \delta_2 \underset{G}{\Longrightarrow}_L u_2 \alpha'' \delta_2 \overset{*}{\underset{G}{\Longrightarrow}}_L u_2 x_2$$

where $u_1, u_2, x_1, x_2 \in V_T^*$ $\delta_1, \delta_2 \in V^*$, from the existence of $l : 1 \leqslant l \leqslant n$ such that x_1 is in R_l and x_2 is in R_l, we have $\alpha' = \alpha''$.

A grammar G is called LL ($\pi_1, \ldots, \pi_k$) if for any $1 \leqslant i \leqslant k$ the rule U_i is LL(π_i) rule.

A grammar G is called LL - regular (LLR) iff G is LL ($\pi_1, \ldots, \pi_k$) for some fixed regular partitions $\pi_1, \ldots, \pi_k$ of V_T^*.

A language L is LLR iff there exists a LLR grammar G such that $L = L(G)$.

$L(A, i)$ is a language defined by

$L(A, i) = \{ x \in V_T^* \mid$ there exist $y \in V_T^*$ and $\delta \in V^*$ such that $S \overset{*}{\underset{L}{\Longrightarrow}} y\, A\, \delta$ and $A\, \delta \Longrightarrow \alpha_i\, \delta \overset{*}{\underset{L}{\Longrightarrow}} x \}$

1. SUFFICIENT CONDITIONS

First, we introduce some definitions and prove some theorems, which allow easier formulating the sufficient conditions.

Definition 1

A production $p_i : A \longrightarrow \alpha_i$ in U is said to be a LLR (R_i) production iff there exists a regular set R_i, such that $L(A, i) \subseteq R_i$ and for each $1 \leqslant j \leqslant n$, $j \neq i$ $R_i \cap L(A, i) = \emptyset$. Then p_i is a LLR production.

Theorem 1

A rule U is a LLR (π) rule iff for each production $p_i : A \longrightarrow \alpha_i$ in U, p_i is a LLR production.

We prove first the case if.

Let U be a LLR (π) rule for the regular partition $\pi = \{ R_1, \ldots, R_n \}$. From the definition a LLR (π) rule and a regular partition π, we have that $L(A, i) \subseteq R_i$ and $R_i \cap R_j = \emptyset$ for $j = 1, \ldots, n$, $j \neq i$. Thus $R_i \cap L(A, i) = \emptyset$ for $j = 1, \ldots, n$ and $j \neq i$, i.e. p_i is a LLR (R_i) production for $i = 1, \ldots, n$.

We prove now the case only if. Assume, that each production $p_i : A \longrightarrow \alpha_i$ in U is a LLR(R_i) production, i.e. $L(A, i) \subseteq R_i$ and $R_i \cap L(A, i) = \emptyset$, for $i = 1, \ldots, n$, $j \neq i$.

Then the rule U is a LLR(π) rule for the regular partition $\pi = \{ \bar{R}_1, \ldots, \bar{R}_n \}$, where $\bar{R}_i = R_i - \bigcup_{j \neq i} R_j$, $i = 1, \ldots, n$.

Definition 2

A production $p_i : A \longrightarrow \alpha_i$ in U is said to be a LLS(k, K) production iff there exists a finite set K of words over V_T, of length equal or less than k and $L(A, i) \subseteq K V_T^*$ and $L(A, j) \cap K V_T^* = \emptyset$ for each $j \neq i$. Then we say that p_i is LLS(k) production.

Theorem 2

If the production $p_i : A \longrightarrow \alpha_i$ is a LLS(k) production, then p_i is a LLR production.

Proof

Let p_i be a LLS(k, K) production. It is obvious that p_i is a LLR(R_i) production for the regular set $R_i = K V_T^*$.

We can introduce now a notion of separating set, which is a basic notion in the definition of the sufficient conditions.

Definition 3

Let K be a finite set of words over V_T. K is said to be a separating set for the production $p_i : A \longrightarrow \alpha_i$ in U iff $L(A, i) \subseteq V_T^* K V_T^*$ and $L(A, j) \cap V_T^* K V_T^* = \emptyset$ for each $j \neq i$.

Theorem 3

If there exists a separating set K for the production $p_i : A \longrightarrow \alpha_i$ in U then p_i is a LLR production.

Proof

Let K be a separating set for $p_i : A \longrightarrow \alpha_i$ and let us denote $R_i = V_T^* K V_T^*$. Obviously R_i is a regular set and from the definition of the separating set we have immediately that $L(A, i) \subseteq R_i$ and $R_i \cap L(A, j) = \emptyset$, for $j \neq i$, i.e. p_i is a LLR(R_i) production.

The production $p_i : A \longrightarrow \alpha_i$ in U may be a LLS(k)production while a separating set for p_i does not exist. On the other hand, for the production p_i a separating set may exist, while this production is not a LLS(k)production. To illustrate the above statements let us consider two examples.

Example 1

Let $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, where

$P = \{ S \longrightarrow A,$
$p_1 : A \longrightarrow Bcab,$
$p_2 : A \longrightarrow aBcab,$
$B \longrightarrow bB,$
$B \longrightarrow b\}$

Since $L(A, 1) = b^*cab$ and $L(A, 2) = ab^*cab$, p_1 and p_2 are LLS(1)productions, but there exist no separating sets for them.

Example 2

Suppose $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$ is a CF grammar, where

$P = \{ S \longrightarrow A,$
$p_1 : A \longrightarrow aBd,$
$p_2 : A \longrightarrow acd,$
$B \longrightarrow aB,$
$C \longrightarrow aC,$
$B \longrightarrow b,$
$C \longrightarrow c.$

Since $L(A, 1) = aa^*bd$ and $L(A, 2) = aa^*cd$, the productions p_1 and p_2 are not LLS(k) productions for any k, but obviously the separating sets K_1, K_2 exist and are equal: $K_1 = \{b\}$, $K_2 = \{c\}$.

Let us give now a sufficient condition for a grammar to be LLR. This condition is connected with the two conditions stated above and it generalizes them.

Definition 4

Let $K_0, K_1, \ldots, K_p$ be finite sets of words over V_T. The set $K \subseteq K_0 \times K_1 \times \ldots \times K_p$ is said to be a separating product for $p_i : A \longrightarrow \alpha_i$ in U iff

$$L(A, i) \subseteq \bigcup_{(t_0, t_1, \ldots, t_p) \in K} t_0 V_T^* t_1 V_T^* \ldots t_p V_T^*$$

$$L(A, j) \cap \bigcup_{(t_0, t_1, \ldots, t_p) \in K} t_0 V_T^* t_1 V_T^* \ldots t_p V_T^* = \emptyset \text{ for each } j \neq i$$

Theorem 4

If there exists a product separating set for the production $p_i : A \longrightarrow \alpha_i$ in U, then p_i is a LLR production.

Proof

Let $K \subseteq K_0 \times K_1 \times \ldots \times K_\ell$ be a product separating set for a production $p_i : A \longrightarrow \alpha_i$.

Denote $R_i = \bigcup_{(t_0, t_1, \ldots, t_\ell) \in K} t_0 V_T t_1 V_T^* \ldots V_T^* t_\ell V_T^*$

Since R_i is a regular set, $L(A, i) \subseteq R_i$ and $R_i \cap L(A, j) = \emptyset$ for $j \neq i$, the production p_i is LLR (R_i).

Assuming that either $K_i = \{\varepsilon\}$ for $i = 1, 2, \ldots, \ell$, or the set K_0 and 1-1 sets from $K_1, \ldots, K_\ell$ consist only of empty word ε, we immediately get the following two conclusions:

Conclusion 1

If the production p_i in U is a LLS(k) production, then there exists a product separating set for p_i.

Conclusion 2

If there exists the separating set for the production p_i in U, then a product separating set for p_i exists.

Example 3 below shows that for the production p_i in U may exist a product separating set, while p_i is neither a LLS(k) production, nor does exist a separating set for p_i.

Example 3

Let us consider a CF-grammar $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, where $P = \{ S \longrightarrow aAb,$

$p_1 : A \longrightarrow aCd,$

$p_2 : A \longrightarrow cCg,$

$p_3 : A \longrightarrow bCd,$

$C \longrightarrow aCa,$

$C \longrightarrow c,$

$B \longrightarrow bB,$

$B \longrightarrow b\}$

The production p_1 is not a LLS(k) production ($L(A, 1) = \{aa^n ca^n d, n \geqslant 0\}$, $L(A, 2) = \{aa^n ca^n g, n \geqslant 0\}$) and it does not exist a separating set for p_1. But there exists a product separating set for p_1: $K = \{(a, d)\}$, where $K_0 = \{a\}$, $K_1 = \{d\}$.

Let us give now an extension of the definition of product separating sets.

Definition 5

Let $K_0, K_1, \ldots, K_l$ be regular sets of words over V_T. The set $K \subseteq K_0 \times K_1 \times \ldots \times K_l$ is said to be a regular product separating set for $p_i : A \longrightarrow \alpha_i$ in U iff

$$L(A, i) \subseteq \bigcup_{(t_0, t_1, \ldots, t_l) \in K} t_0 V_T^* t_1 V_T^* \ldots t_l V_T^*$$

$$L(A, j) \cap \bigcup_{(t_0, t_1, \ldots, t_l) \in K} t_0 V_T^* t_1 V_T^* \ldots t_l V_T^* = \emptyset \text{ for each } j \neq i.$$

Theorem 5

If there exists a regular product separating set for the production $p_i : A \longrightarrow \alpha_i$ in U, then p_i is a LLR production.

The proof is quite similar as to that of the theorem 4.

Example 4

Consider the grammar $G = (\{S, A, B, C, D, E\}, \{a, b, c\}, P, S)$, where $P = \{S \longrightarrow A,$

$p_1 : A \longrightarrow BcBcD,$

$p_2 : A \longrightarrow BaE,$

$p_3 : A \longrightarrow EaEaE,$

$p_4 : A \longrightarrow BcDa,$

$B \longrightarrow bBb,$

$B \longrightarrow a,$

$D \longrightarrow bD,$

$D \longrightarrow b,$

$E \longrightarrow bEb,$

$E \longrightarrow c\}.$

In this grammar:

$L(A, 1)\ \{b^n a b^n c b^k a b^k c b^l,\ n \geqslant 0,\ l \geqslant 0, k \geqslant 0\}.$

$L(A, 2)\ \{b^n a b^n a b^k c b^k,\ n \geqslant 0, k \geqslant 0\}.$

$L(A, 3)\ \{b^n c b^n a b^k c b^k a b^l c b^l,\ n \geqslant 0, k \geqslant 0,\ l \geqslant 0\}.$

$L(A, 4)\ \{b^n a b^n c b^l a,\ n \geqslant 0,\ l \geqslant 0\}.$

The set $K = (b^*a, cb^*a, b)$ is a regular product separating set for the production p_1 (set $K_0 = b^*a$ differs p_1 from p_3, $K_1 = cb^*a$, differs p_1 from p_2 and $K_2 = \{b\}$ differs p_1 from p_4). In this example does not exist the finite sets K_0 and K_1, so these sets have to be regular sets.

CONCLUSION

In this paper certain sufficient conditions for a grammar to be LLR were derived but no effective method checking any of them was given. Further studies should to be performed to find such methods which would enable us to construct an effective parsing algorithm.

References

[1] JARZĄBEK S., KRAWCZYK T.: LL - Regular Grammars, Prace IMM, 1974, z. 1

LL - REGULARNE GRAMATYKI
(warunki dostateczne)

Streszczenie

Niniejsza praca jest kontynuacją badań prowadzonych nad gramatykami LLR. Zawiera ona szereg warunków dostatecznych jakie musi spełniać gramatyka na to, aby była LLR.

LL - РЕГУЛЯРНЫЕ ГРАММАТИКИ
(достаточные условия)

Резюме

Настоящая работа является конструкцией исследований грамматики LLR . Она заключает ряд достаточных условий которые должна выполнить грамматика для того чтобы стать LLR .

AUTORZY ARTYKUŁÓW ZAMIESZCZONYCH W ZESZYCIE 2 "PRAC IMM"

WYBRANE ZAGADNIENIA SYMULACJI SYSTEMU KIEROWANIA RUCHEM LOTNICZYM. WYNIKI WSTĘPNE

Jerzy DAŃDA mgr inż. Studia ukończył na Wydziale Łączności Politechniki Warszawskiej w roku 1959. Od roku 1956 pracuje w Instytucie Maszyn Matematycznych, zajmując się /do roku 1970/ zagadnieniami sprzętu cyfrowego i jego niezawodności. Obecnie zajmuje się zagadnieniami oprogramowania i systemami wieloprocesorowymi . Jest kierownikiem Pracowni Prognoz Zakładu Studiów i Analiz. Autor licznych publikacji z dziedziny maszyn matematycznych.

Jan PIETRALA mgr ukończył studia na Wydziale Matematyki i Fizyki Uniwersytetu Warszawskiego w roku 1965. Problematyką automatyzacji procesów związanych z ruchem lotniczym zajmował się ciągle pracując w Instytucie Technicznym Wojsk Lotniczych, Polskich Liniach Lotniczych, a od 1974 r. w Instytucie Maszyn Matematycznych. Jest głównym projektantem automatycznych systemów kierowania obiektów ruchowych w Zakładzie Studiów i Analiz IMM.

Zbigniew POZNAŃSKI ukończył studia na Wydziale Mechanicznym i Energetyki Lotnictwa Politechniki Warszawskiej w 1974 r. Pracuje w Zakładzie Studiów i Analiz IMM zajmując się zagadnieniami automatyzacji kierowania ruchem lotniczym.

KONCEPCJA SYNTEZY FONEMATYCZNEJ (BEZTRANSIENTOWEJ)

Ryszard PATRYN ukończył w 1952 r. Politechnikę Śląską. W Instytucie Maszyn Matematycznych pracuje od 1967 r. Od kilku lat zajmuje się pracami związanymi z komunikacją foniczną z komputerem. Opracowana przez mgr inż. Patryna i przedstawiona w artykule metoda fonematycznej syntezy sygnału mowy jest nieznana dotąd w literaturze fachowej. Jest autorem kilku publikacji oraz autorem i współautorem 8 patentów.

WYBRANE PROCEDURY GRAFICZNE W SYSTEMACH KOMPUTEROWYCH.

Andrzej ZIEMKIEWICZ ukończył w 1964 r. Wydział Elektroniki Politechniki Warszawskiej. W 1963 r. podjął pracę w Instytucie Maszyn Matematycznych w Zakładzie Organizacji Maszyn Cyfrowych, gdzie brał udział m.in. w projektowaniu jednostki centralnej ZAM-41. W latach 1970-73 pracował w Pionie Rozwojowym Maszyn Cyfrowych ZWPP "ERA". Od 1973 r. pracuje jako główny specjalista w Zakładzie Doświadczalnym Oprogramowania IMM zajmując się komputerowymi systemami graficznymi. Jest autorem kilku publikacji z zakresu algorytmów oraz struktur maszyn cyfrowych.

ALGORYTM PLANOWANIA OBLICZEŃ DWUPROCESOROWYCH

Andrzej ROWICKI ukończył Wydział Elektroniki Politechniki Warszawskiej. Od roku 1956 pracuje w IMM, zajmując się konstrukcją i teorią maszyn matematycznych. Opublikował szereg prac z dziedziny maszyn matematycznych.

DWA ALGORYTMY KONSTRUKCJI
DETERMINISTYCZNEGO AUTOMATU SKOŃCZONEGO NA PODSTAWIE
ZADANEJ GRAMATYKI LEWOSTRONNIE LUB PRAWOSTRONNIE LINIOWEJ

Jacek WITASZEK ukończył w 1959 r. Wydział Matematyki, Fizyki, Chemii Uniwersytetu Jagiellońskiego. Od 1960 r. pracuje w Instytucie Maszyn Matematycznych. Początkowo zajmował się programowaniem maszyn XYZ, ZAM-2, ZAM-21, ZAM-41, a od roku 1971 pracuje nad metodami translacji. Obecnie zajmuje się konstrukcją metatranslatora. Jest autorem kilku publikacji z teorii programowania.

LL - REGULAR GRAMMARS (sufficient conditions)

Stanisław JARZĄBEK ukończył Wydział Matematyki Uniwersytetu Warszawskiego w 1972 r. W tym samym roku podjął pracę w Zakładzie Metod Translacji Instytutu Maszyn Matematycznych. Obecnie pracuje w Zakładzie Teorii Translatorów zajmując się problemami optymalizacji programów.

Tomasz KRAWCZYK w 1968 r. ukończył Wydział Matematyki Uniwersytetu Warszawskiego i w tym samym roku rozpoczął pracę w Zakładzie Metod Programowania Instytutu Maszyn Matematycznych. Obecnie jest kierownikiem pracowni w Zakładzie Teorii Translatorów. Zajmuje się metodami analizy składniowej i z tej tematyki ma kilka publikacji.

Cena 60 zł.