Włodzimierz OWSIAK [1], **Aleksander OWSIAK** [2]
[1] POLITECHNIKA OPOLSKA, OPOLE I UKRAIŃSKA AKADEMIA DRUKARSTWA, LWÓW
[2] LWOWSKA FILIA KIJOWSKIEGO NARODOWEGO UNIWERSYTETU KULTURY I SZTUK PIĘKNYCH, LWÓW

Rozszerzenie algebry algorytmów

Prof. dr hab. inż. Włodzimierz OWSIAK

Studia ukończył na Wydziale Automatyki Politechniki Lwowskiej w roku 1974. Stopień doktora nauk technicznych uzyskał w roku 1989, stopień doktora habilitowanego w roku 1996 w dyscyplinie Informatyka, a tytuł profesora w roku 2001. Specjalizuje się w informatyce teoretycznej i stosowanej, teorii algorytmów, programowaniu, systemach informacyjnych, modelowaniu matematycznym.

e-mail: ovsyak@rambler.ru

Dr inż Aleksander OWSIAK

Studia ukończył na Wydziale Komputerowej Inżynierii Poligraficznej Ukraińskiej Akademii Drukarstwa w roku 1999. Stopień doktora nauk technicznych uzyskał w roku 2002 w dyscyplinie Informatyka. Specjalizuje się w informatyce teoretycznej i stosowanej, teorii algorytmów, programowaniu, systemach symulacji komputerowej, modelowaniu matematycznym.

e-mail: ovsjak@ukr.net

Streszczenie

W artykule za pomocą metody aksjomatycznej przedstawiono podstawy rozszerzonej algebry algorytmów. Algebra ta obejmuje operacje sekwencjonowania, eliminowania, zrównoleglenia, rewersowania oraz cyklicznego sekwencjonowania, eliminowania i zrównoleglenia, wykonywane na unitermach. Podano definicję algorytmu, do jakiego ma zastosowanie rozszerzona algebra algorytmów. Istotę zdefiniowanych operacji rozszerzonej algebry algorytmów zilustrowano za pomocą rysunków. Na przykładzie pokazano jej zastosowanie. Opis porównano z opisem algorytmów, otrzymywanym za pomocą klasycznej algebry algorytmów.

Słowa kluczowe: uniterm, operacja, sekwencjonowanie, cykliczne sekwencjonowanie, eliminowanie, cykliczne eliminowanie, zrównoleglenie, cykliczne zrównoleglenie, rewersowanie, algebra algorytmów.

The expansion of algorithm algebra

Abstract

Very often algorithms are described verbally or like a unit - diagram. The well known methods offering algorithms are: Post [1], Turing [2], Aho-Ullman-Hopcroft [3] or Schönhage [4] virtual machines, recursive functions (calculus λ, Church) [5], Markov algorithms [6], *b*-complexes of Kolmogorov (Kolmogorov machine) [7], Krinitski universal algorithms [8], and algorithm algebra [9]. It is obvious that verbal methods, and methods of unit - diagram, as well as, algorithm methods [1] - [8] are depicted by the intuition, not formally. Only by means of the algorithm algebra, the algorithm description is getting into the formulae form, on abstract and meaningful levels. The transformation and investigation of their trustworthiness can be made on formulae of algorithms with minimization target, by the specific operations. These advantages of algebra algorithms beyond other methods of algorithm description make a ground for it's using. Classical algorithm algebra [9] manipulates over conditional uniterms, which are delivered only two meanings (e.g. "yes" and "no" or "0" and "1").Very often conditional uniterm can deliver more than two meanings. For example, automation systems are operated in a plenty of regimes. Score parameters are controlled in checking systems. It is possible to describe the algorithms which contain more than 2 conditions by means of classical algebra algorithms. These formulae – algorithms are complicated for apprehension. To avoid possible mistakes, the expansion of the algorithm algebra is presented in the paper.

Keywords: uniterm, operation, sequencing, elimination, paralleling, reversing, cyclic sequencing, cyclic elimination, cyclic paralleling.

1. Wprowadzenie

Systemy pomiarowe, automatyki i kontroli powinny funkcjonować zgodnie z algorytmami zaimplementowanymi w nich sprzętowo, programowo lub sprzętowo i programowo. Algorytmy ich działania wymagają ściśle określonej kolejności operacji.

Najczęściej algorytmy określa się werbalnie lub w postaci schematów blokowych. Istnieją także znane metody opisu algorytmów, takie jak metody wirtualnych maszyn Posta [1], Turinga [2], Aho-Ullmana-Hopcrofta [3], Schönhage [4], funkcji rekursywnych (rachunek λ, Churcha) [5], algorytmów Markova [6], *b*-kombleksów Kołmogorowa (maszyna Kolmogorova) [7], uniwersalnych algorytmów Krinitskiego [8] oraz algebra algorytmów [9]. Algorytmy tworzone tymi metodami, za wyjątkiem [9], są opisywane intuicyjnie, a nie formalnie.

Algebra algorytmów ma zalety, które wyróżniają ją spośród innych metod. Za jej pomocą otrzymuje się opisy algorytmów w postaci formuł, na poziomie abstrakcyjnym jak i treściowym. Formuły algorytmów mogą być minimalizowane formalnie, przy wykorzystaniu właściwości operacji. Przeprowadzono sprawdzenie poprawności tych przekształceń.

Klasyczna algebra algorytmów [9] operuje jednak tylko unitermami warunkowymi, przyjmującymi tylko dwie wartości (tak, nie lub 0 i 1). Dosyć często zdarza się, że uniterm warunkowy może przyjmować więcej niż dwie wartości. Np. wyniki otrzymane z badań mogą należeć do kilku zakresów pomiarowych, w automatyce często systemy funkcjonują w wielu reżimach (w maszynach drukarskich steruje się prędkością początkową, biegu jałowego i szeregiem prędkości roboczych), w systemach kontroli najczęściej kontroluje się dziesiątki parametrów. Za pomocą zasobów klasycznej algebry algorytmów można opisać algorytmy, które zawierają unitermy warunkowe z kilkoma wartościami. Taki algorytm–formuła jest jednak skomplikowany i trudny do odczytania. W pracy rozszerzono definicję algebry algorytmów w celu usunięcia tej wady..

2. Podstawowe pojęcia algebry algorytmów

Wprowadzone zostaną oznaczenia pojęć i operacji, na jakich jest oparta algebra algorytmów.

1. **Alfabet** algebry algorytmów tworzą:

1.1. **Unitermy**. To pojęcie oznacza symbole i znaki, na których wykonywane są operacje. Na przykład unitermami są 1, 7, 13, a, k, l, x, y, z, $y=x$, $y=kx$, $y>x$, $y=x^2$, $z=x\&y$, $z=x\vee y$, $p:=q$, 0+1, $x\cdot y$, $f(x,y)$, $A(x)$, $F(x,y,z)$.

Unitermy, które nie zależą od żadnej zmiennej, dzielą się na ***stałe*** (np. 0, 2, n, m, ...), ***parametry*** (np. a, k, l) i ***zmienne*** (np. x, y, z). Dla oznaczenia stałych, z indeksami lub bez, wykorzystuje się symbol c. Parametry, z indeksami lub bez, oznacza się początkowymi literami alfabetu łacińskiego. Do oznaczenia zmiennych, z indeksami lub bez, używa się końcowych liter alfabetu łacińskiego.

Unitermy, które zależą od co najmniej jednej zmiennej, dzielą się na ***przedmiotowe*** (np. $p:=q$, $y=x$, $y=a+kx$, $y=x^2$, $z=x\&y$, $y=x+z$) i ***abstraktowe*** (np. A, $F(x)$, $S(1,x)$, $R(x,y)$, $P(x,y,z)$), które oznaczamy wielkimi literami alfabetu łacińskiego z indeksami lub bez nich.

1.2. **Znaki operacji:** $\frown$ - sekwencjonowania; $\vdash\!\dashv$ - eliminowania; $\sqcap$ - zrównoleglenia; $\overline{}$ - rewersowania; $\not\subset$ - sekwencjonowania cyklicznego; $\not\supset$ - eliminowania cyklicznego; $\not\emptyset$ - zrównoleglenia cyklicznego; = równa się;

1.3. Niezwiązane operacjami cyklu unitermy-zmienne, współczynniki i stałe, z indeksami lub bez nich, oznaczane małymi literami z początku alfabetu łacińskiego:

$$a, b, c_o, c_1, \ldots, c_j, c^0, c^1, \ldots, c^I, c^i_j, \ldots;$$

1.4. Związane operacjami cyklu unitermy-zmienne, z indeksami lub bez nich, oznaczane małymi literami z końca alfabetu łacińskiego:

$$x, y, z_o, z_1, \ldots, z_j, z^0, z^1, \ldots, z^I, z^i_j, \ldots;$$

1.5. Unitermy abstraktowe niezależne oraz zależne od jednej lub wielu zmiennych, z indeksami lub bez nich, oznaczane wielkimi literami alfabetu łacińskiego:

$$P(a), \ P(a,b), \ldots;$$

1.6. Unitermy warunków, przyjmujących dwie lub więcej wartości, występujących z indeksami lub bez nich, oznaczane małą literą u alfabetu łacińskiego:

$$u, u_o, u_1, \ldots, u_j, u^i_j;$$

1.7. * - uniterm pusty;

1.8. Nawiasy okrągłe, otwierający i zamykający – stosowane w unitermach (np. $F(x)$, $S(1,x)$, $R(x,y)$, $P(x,y,z)$, $(u - ?)$).

Klamrowy nawias otwierający ({) – stosowany w opisie operacji eliminowania.

Przecinek (,) – znak rozdzielający unitermy przemienne.

Średnik (;) – znak rozdzielający unitermy nieprzemienne.

Dwukropek (:) – znak rozdzielający unitermy przemienne lub nieprzemienne.

c_x – oznaczenie końca cyklu ze zmienną x.

2. Operacje algebry algorytmów

2.1. **„Równa się”** nazywamy operację posiadającą następujące właściwości:

Tożsamości: $A = A$, (1)

Symetryczności: jeśli $A = B$, to $B = A$, (2)

Tranzytywności: jeśli $A = B$ oraz $B = S$, to $A = S$. (3)

2.2. **Sekwencjonowaniem** nazywamy operację posiadającą następujące właściwości:

Pochłaniania unitermu:

$$\overbrace{S, S} = S \tag{4}$$

Pochłaniania pustego unitermu:

$$\overbrace{*: S} = S \tag{5}$$

Przemienności:

$$\overbrace{R, S} = \overbrace{S, R} \tag{6}$$

Łączności:

$$\overbrace{R, S}\ , T = R, \overbrace{S, T} \tag{7}$$

Pochłaniania unitermu różnych sekwencji:

$$\overbrace{A; B}\ , \overbrace{A; C} = \overbrace{A; B, C} \qquad \overbrace{A; B}\ , \overbrace{C; B} = \overbrace{A, C; B} \tag{8}$$

2.3. **Eliminowaniem** nazywamy operację posiadającą następujące właściwości:

Wyboru unitermu o dwu wartościach warunku:

$$\overbrace{\quad\quad}^{A\ ;\ B;\ u\ -\ ?} = \begin{cases} A, \text{jeśli } u = t_1 \\ B, \text{jesli } u \neq t_1, \end{cases} \tag{9}$$

gdzie t_1 - wartość unitermu warunkowego u.

Wyboru unitermu o warunku wielowartościowym:

$$\overbrace{\quad\quad\quad}^{R\ ;\ S;;\ldots\ ;\ Z;\ w\ -\ ?} = \begin{cases} R, \text{jeśli } w = v_0, \\ S, \text{jeśli } w = v_1, \\ . \\ . \\ . \\ Z, \text{jesli } w = v_{n-1}, \end{cases} \tag{10}$$

gdzie $v_0, v_1, \ldots, v_{n-1}$ – wartości unitermu warunkowego w.

Wyboru unitermu pustego o warunku wielowartościowym:

$$\overbrace{\quad\quad\quad}^{*\ ;\ S;;\ldots\ ;\ Z;\ w\ -\ ?} = \begin{cases} *, \text{jeśli } w = v_0, \\ S, \text{jeśli } w = v_1, \\ . \\ . \\ . \\ Z, \text{jesli } w = v_{n-1}, \end{cases} \tag{11}$$

Pochłaniania unitermu:

$$\overbrace{\quad\quad}^{A\ ;\ A\ ;\ u\ -\ ?} = A \tag{12}$$

Pochłaniania unitermu o warunku wielowartościowym:

$$\overbrace{\quad\quad\quad}^{S\ ;\ S\ ;\ \ldots\ ;\ S;\ \ w\ -\ ?} = S \tag{13}$$

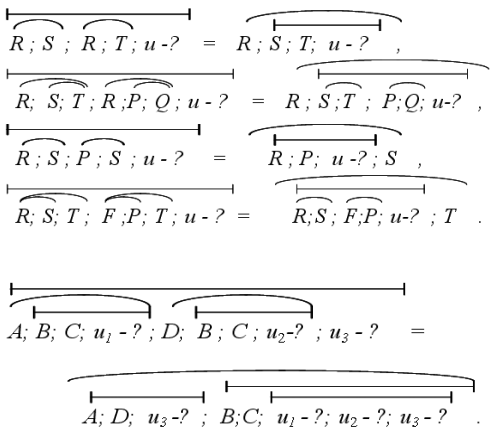
Wyboru warunku wielowartościowego:

$$\begin{aligned} &\overbrace{\overbrace{\quad}^{R;\ S\ ;\ u_1\ -\ ?}\ \overbrace{\quad}^{R;\ S\ ;\ u_2\ -\ ?}\ ;\ u_3\ -\ ?} = \overbrace{\quad\quad}^{R;\ S;\ u_1\ -\ ?;\ u_2\ -\ ?;\ u_3\ -\ ?} \\ &\overbrace{\overbrace{\quad\quad}^{R\ ;\ S\ ;\ \ldots\ ;\ Z\ ;\ w_1\ -\ ?}\ ;\ \overbrace{\quad\quad}^{R\ ;\ S\ ;\ \ldots\ ;\ Z\ ;\ w_2\ -\ ?}\ ;\ u_3\ -\ ?} = \\ &= \overbrace{\quad\quad}^{R;\ S\ ;\ \ldots\ ;\ Z\ ;\ \ w_1\ -\ ?} = \overbrace{\quad\quad}^{R;\ S\ ;\ \ldots\ ;\ Z\ ;\ \ w_2\ -\ ?}\ . \end{aligned} \tag{14}$$

Pochłaniania unitermów:

$$\begin{aligned} &\overbrace{\quad\quad}^{A\ ;\ \overbrace{\quad}^{B\ ;\ C\ ;\ u\ -?}\ ;\ u\ -?} = \overbrace{\quad}^{A;\ C;\ u\ -\ ?}, \\ &\overbrace{\quad\quad}^{\overbrace{\quad}^{A;\ B;\ u\ -?}\ ;\ C\ ;\ u\ -?} = \overbrace{\quad}^{A\ ;\ C\ ;\ u\ -\ ?}, \\ &\overbrace{\quad\quad\quad}^{A\ ;\ B\ ;\ \ldots\ ;\ Z\ ;\ w\ -?\ ;\ C;\ K;\ \ldots\ ;\ M;\ w\ -?} = \overbrace{\quad\quad}^{A\ ;\ C\ ;\ K\ ;\ \ldots\ ;\ M\ ;\ w\ -?}, \\ &\overbrace{\quad\quad\quad}^{A\ ;B;\ \ldots\ ;Z\ ;\ \ Q\ ;\ L\ ;\ \ldots\ ;\ M;\ \ u-?\ ;\ u\ -\ ?} = \overbrace{\quad\quad}^{A\ ;B\ ;\ \ldots\ ;Z\ ;\ M;\ u-?}\ . \end{aligned} \tag{15}$$

Rozdzielności:



(16)

(17)

2.4. **Zrównolegleniem** nazywamy operację posiadającą następujące właściwości:

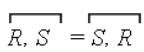
Pochłaniania unitermu:

(18)

Pochłaniania pustego unitermu:

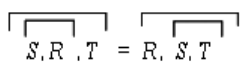
(19)

Przemienności:



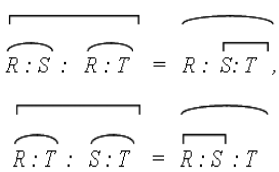
(20)

Łączności:



(21)

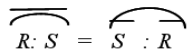
Wyniesienia unitermu poza operację zrównoleglenia:



(22)

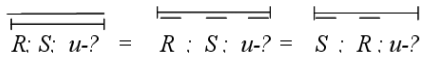
2.5. **Rewersowaniem** nazywamy operację posiadającą następujące właściwości:

Przestawiania unitermów w sekwencji:



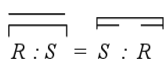
(23)

Przestawiania unitermów w eliminowaniu:



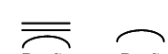
(24)

Przestawiania unitermów w zrównolegleniu:

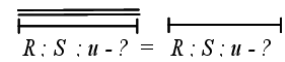


(25)

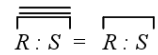
Podwójnego przestawiania unitermów:



(26)

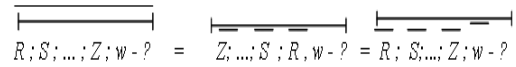


(27)



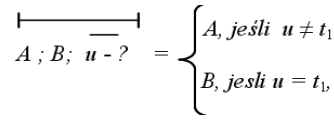
(28)

Przestawiania unitermów w eliminowaniu o warunku wielowartościowym *w*:

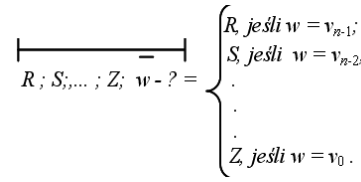


(29)

Wyboru unitermu na podstawie warunku rewersywnego:

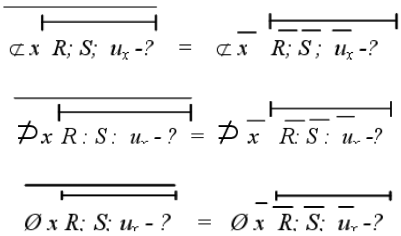


(30)



2.6. **Sekwencjonowaniem cyklicznym**, **eliminowaniem cyklicznym i zrównolegleniem cyklicznym** nazywamy operacje posiadające następujące właściwości:

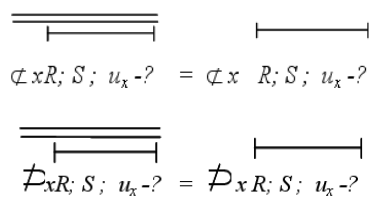
Rewersowanie zmiennej związanej:



(31)

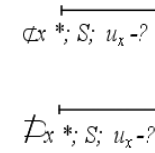
gdzie x –zmienna związana, a R i S - unitermy, będące w zasięgu operacji;

Podwójnego rewersowania:



(32)

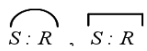
Cykl pusty:



(33)

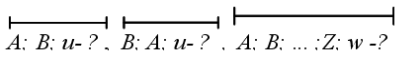
3. Indukcyjna definicja algorytmu

3.1. Jeśli S i R są unitermami, to



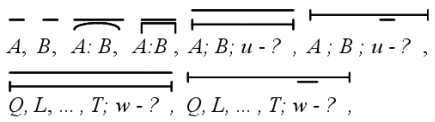
są algorytmami.

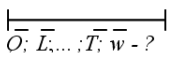
3.2. Jeśli A , B, ..., Z są unitermami lub algorytmami, a u, w są unitermami warunkowymi i liczba wartości w jest równa liczbie unitermów eliminowania przy warunku w, to



są algorytmami.

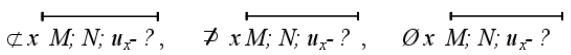
3.3. Jeśli A , B, Q, L, ... , T są unitermami lub algorytmami, a u, w są unitermami warunkowymi i liczba wartości w jest równa liczbie unitermów eliminowania przy warunku w, to

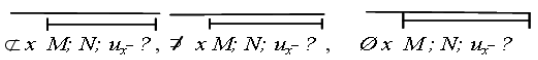


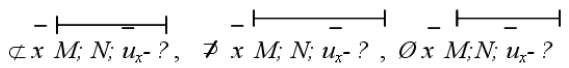


są algorytmami.

3.4. Jeśli M i N są unitermami lub algorytmami, x jest zmienną związaną z operacją cyklu, a u_x – warunkiem związanym z operacją cyklu, to







są algorytmami.

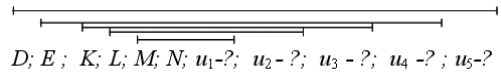
3.5. Jeśli F, S, T są unitermami lub algorytmami, takimi że $F = S$, a $S = T$, to $F = T$ jest algorytmem.

3.6. Algorytmem jest tylko takie wyrażenie, które można przedstawić w postaci skończonej liczby realizacji operacji przedstawionych w punktach 3.1– 3.5.

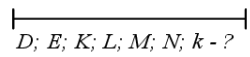
Algebra algorytmów jest nauką o właściwościach operacji sekwencjonowania, eliminowania, zrównoleglenia, rewersowania oraz operacji cyklicznych przeprowadzanych na unitermach i algorytmach.

4. Przykład wykorzystania rozszerzonej algebry algorytmów

Na platforme Microsoft Visual Studio .NET w języku C# istnieje instrukcja ***switch*** [10] zawierająca etykiety ***case*** z warunkami oraz blokami kodu, wykonywanymi przy wykonaniu warunków. Instrukcja ***switch*** za pomocą klasycznej algebry algorytmów [9] może być opisana następująco



Stosując do tej formuły eliminowanie przy warunku k, przyjmującym pięć wartości, otrzymujemy wyrażenie



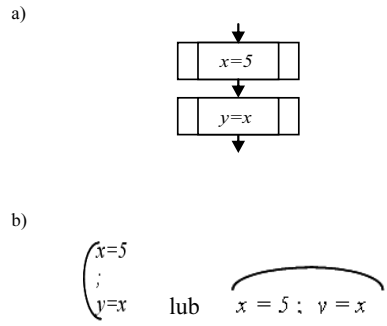
które, w porównaniu z poprzednią formułą, zawiera 5 razy mniej znaków operacji eliminowania.

5. Przykłady intuicyjnego wyjaśnienia operacji

Dla intuicyjnego wyjaśnienia operacji rozszerzonej algebry algorytmów stosujemy schematy blokowe.

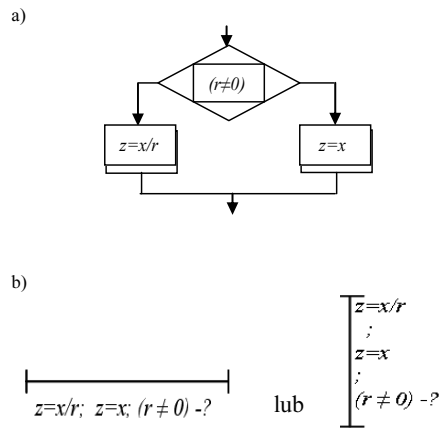
1. Operacja „równa się” jest operacją klasyczną i nie wymaga specjalnego wyjaśnienia.

2. Operacja sekwencjonowania jest przeznaczona dla opisywania kolejek, co zostało pokazane na rys. 1, gdzie najpierw zmiennej x przypisuje się wartość 5, a potem przypisuje się ją do y.



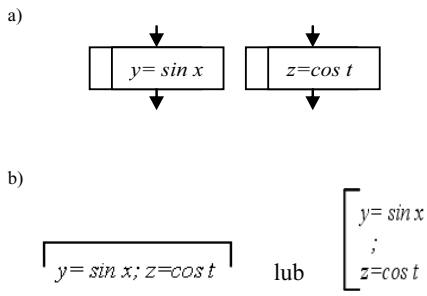
Rys. 1. Fragment schematu blokowego algorytmu (a) i jego opis za pomocą operacji sekwencjonowania (b)
Fig. 1. A part of the block diagram of an algorithm (a), and its description by the sequencing operation (b)

3. Na rys. 2 przedstawiono fragment schematu blokowego algorytmu z blokiem warunkowym (a) i jego opis za pomocą operacji eliminowania (b)



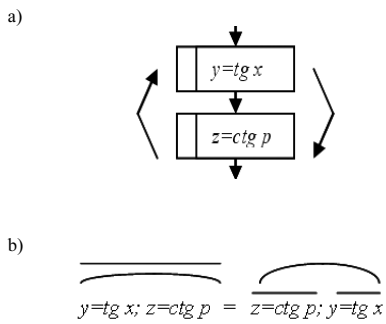
Rys. 2. Fragment schematu blokowego algorytmu (a) i jego opis za pomocą operacji eliminowania (b)
Fig. 2. A part of the block diagram of an algorithm (a), and its description by the elimination operation (b)

4. Operacja zrównoleglenia przeznaczona jest do opisu procesów równoległych, co pokazano na rys. 3.



Rys. 3. Fragment schematu blokowego algorytmu równoległego (a) i jego opis za pomocą operacji zrównoleglenia (b)
Fig. 3. A part of the block diagram of parallell algorithm (a), and its description by the paralleling operation (b)

5. Istota operacji rewersowania polega na przestawianiu unitermów. Po wykonaniu operacji pierwszy uniterm staje się ostatnim, a ostatni - pierwszym (patrz rys. 4). Rewersowanie operacji eliminowania i zrównoleglenia ma ten sam sens co rewersowanie operacji sekwencjonowania.



Rys. 4. Fragment schematu blokowego algorytmu (a) i jego opis za pomocą operacji rewersowania (b)
Fig. 4. A part of the block diagram of an algorithm (a), and its description by the reversing operation (b)

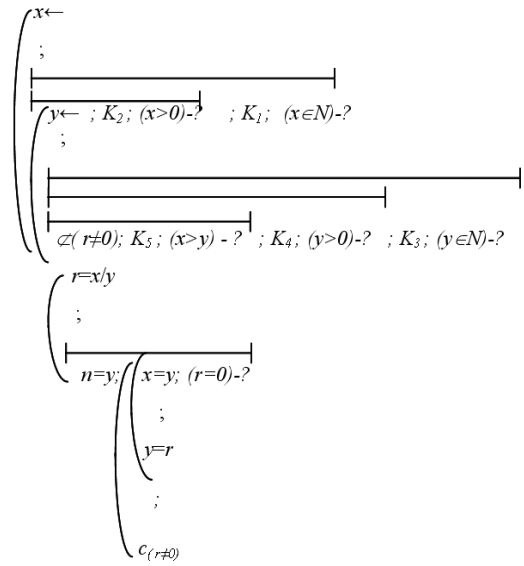
6. Operacje cyklicznego sekwencjonowania, eliminowania i zrównoleglenia przeznaczone są do opisu cykli.

6. Formuła algorytmu Euklidesa

Algorytm został stworzony przez Euklidesa w trzecim wieku naszej ery. Uważa się że to jest pierwszy algorytm. Jest przeznaczony do znalezienia największego wspólnego dzielnika dwóch liczb naturalnych. Formułą algorytmu Euklidesa jest wzór (34). W formule tej wprowadzono następujące oznaczenia:

$x\leftarrow$ - wprowadzenie pierwszej liczby, a $y\leftarrow$ - wprowadzenie drugiej liczby, $(x\in N)$-*?*- sprawdzenie, czy wprowadzona liczba x jest liczbą naturalną, oraz $(y\in N)$-*?* – sprawdzenie, czy wprowadzona liczba y jest liczbą naturalną, komunikaty: K_1 –x nie jest liczbą naturalną, K_2 – x jest mniejsze od zera, K_3 – y nie jest liczbą naturalną, K_4 – y jest mniejsze od zera, K_5 – x jest mniejsze lub równe y, r=x/y – znalezienie reszty z dzielenia x przez y.

Po wprowadzeniu x i y oraz wykonaniu sprawdzenia warunków szuka się reszty z dzielenia x przez y. Jeżeli reszta równa się zero (r=0), to y jest największym wspólnym dzielnikiem (n=y). W przeciwnym przypadku liczbie x przypisuje się wartość y, a liczbie y – wartość reszty, i ponownie oblicza się resztę z dzielenia (cykl $\propto(r\neq 0)$).



(34)

7. Wnioski

1. Klasyczną algebrę algorytmów rozszerzono przez wprowadzenie aksjomatów operacji eliminowania z unitermem warunkowym, przyjmującym więcej niż dwie wartości. Tymi aksjomatami są: wybór i pochłanianie unitermu, wybór warunku wielowartościowego, pochłanianie unitermów, wybór unitermów na podstawie warunku rewersywnego, związek pomiędzy warunkiem rewersywnym i nierewersywnym.

2. Zastosowanie wielowartościowej operacji eliminowania upraszcza i zwiększa przejrzystość opisywanych formuł algorytmów.

8. Literatura

[1] Post E.L.: Finite Combinatory Processes˜ Formulation 1. Journal of Symbolic Logic, 1, pp. 103-105, 1936. Reprinted in The Undecidable, pp. 289ff.
[2] Turing A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of London Mathematical Society, series 2, vol. 42 (1936-1937), pp. 230-265; correction, ibidem, vol. 43, pp. 544-546. Reprinted in [13 Davis M., pp. 155-222] and available online at http://www.abelard.org/turpap2/tp2-ie.asp
[3] Aho A.V., Hopcroft J.E., Ullman J.D.: The design and analysis of computer algorithms. Addison-Wesley Publishing Company, 1974.
[4] Schönhage A.: Universelle Turing Speicherung. In J. Dörr and G. Hotz, Editors, Automatentheorie und Formale Sprachen, Bibliogr. Institut, Mannheim, 1970, pp. 369-383.
[5] Church A.: An unsolvable problem of elementary number theory. American Journal of Mathematics, vol. 58 (1936), pp. 345-363.
[6] Markov A.A.: Theory of algorithms (in Russian). Editions of Academy of Sciences of the USSR, vol. 38, 1951, pp. 176-189; translated into English in American Mathematical Society Transactions, 1960, series 2, 15, pp. 1-14.
[7] Kolmogorov A.N.: On the concept of algorithm (in Russian). Uspekhi Mat. Nauk 8:4 (1953), pp. 175176; translated into English in Uspensky V.A., Semenov A.L.: Algorithms: Main Ideas and Applications, Kluwer, 1993.
[8] Krinitski N.A.: Algorithms around us (in Russian). Mir, Moscow, 1988; also translated to Spanish (Algoritmos a nuestro alrededor).
[9] Owsiak W., Owsiak A., Owsiak J.: Teoria algorytmów abstrakcyjnych i modelowanie matematyczne systemów informacyjnych. Wyd. Pol. Opolskiej, Opole, 2005.
[10] Perry S.C.: C# i .NET. Wyd. Helion, Gliwice, 2006.

otrzymano / received: 21.09.2009
przyjęto do druku / accepted: 04.01.2010 *artykuł recenzowany*