dr inż. Stanisław Bocian
Instytut Pojazdów Szynowych „TABOR"

Nowy sposób wyznaczania najmniejszej wspólnej wielokrotności liczb naturalnych, jako model matematyczny automatu w technice komputerowej.

W pracy opisano nowy sposób wyznaczania Najmniejszej Wspólnej Wielokrotności (NWW) liczb naturalnych. Algorytm ten jest nowym narzędziem matematycznym optymalizującym obliczenia NWW liczb naturalnych. Algorytm ten umożliwia także uzyskanie wyników równoważności obliczeń półgrup charakterystycznych sumy prostej i iloczynu prostego pewnych klas automatów, które są modelami matematycznymi (algebraicznymi) odpowiednio sekwencyjnych i równoległych struktur tych klas automatów. Automat rozumiany jest tutaj jako model matematyczny opisujący sposób przetwarzania informacji przez różnorodne systemy cyfrowe.

1. Wprowadzenie

W konstrukcji i w eksploatacji pojazdów szynowych coraz częściej wykorzystuje się specjalistyczne komputery i systemy mikroprocesorowe (sterowniki). Funkcje realizowane przez sterowniki, algorytmy sterowania i graniczne wartości parametrów pracy mogą być zmodyfikowane przez zmiany w algorytmach i oprogramowaniu sterowników. Istotnym zagadnieniem w stosowaniu algorytmów i oprogramowaniu jest ich optymalizacja, co w konsekwencji zmniejsza złożoność pamięciową i czasową wykonywanych obliczeń.

W ramach realizowanego grantu 8T 12C 068 20 „Inteligentne podsystemy mechatroniczne w procesach diagnostycznych i sterowanie hamowaniem pojazdów szynowych" podjęto prace, których celem jest maksymalne wyeliminowanie układów mechaniczno – pneumatycznych podczas procesu hamowania pojazdów szynowych na rzecz układów elektryczno-elektronicznych. Aktualny stan wiedzy oraz uzasadnione współczesne tendencje wskazują na potrzebą podjęcia badań nad układami wprowadzającymi nowe techniki w konstrukcji i badaniach pojazdów szynowych w szeroko rozumianymi układami elektronicznymi i informatycznymi z elementami sztucznej inteligencji. Zakłada się, że w układach sterowania i diagnostyki hamowania pojazdów szynowych stosowane będą układy mikroprocesorowe (sterowniki), których działanie powinno odbywać się praktycznie w czasie rzeczywisty. Te wymagania narzucają dobór optymalnych parametrów technicznych w szczególności dotyczący czasu zadziałania poszczególnych elementów układu. Znalezienie metod i narzędzi optymalizujących algorytmy, według których działają systemy mikroprocesorowe powodują skracanie czasów działania systemów cyfrowych i umożliwiają przesyłanie sygnałów sterujących prawie, że w czasie rzeczywistym.

2. Modele matematyczne [7, 19]

Świat matematyki można sobie wyobrazić jako układ warstw otaczających jądro matematyki [19]. Jądro generuje bogactwem nowych idei, nowych struktur i teorii. Idee z jądra przenikają ciągle poprzez zewnętrzne warstwy nauk matematycznych, zapewniający stały dopływ wiedzy do niektórych dziedzin spośród problemów związanych z zastosowaniami matematyki. I na odwrót, problemy pojawiające się w warstwach zewnętrznych, gdzie matematyka czysta łączy się z naukami stosowanymi – zaopatruje centralne jądro w nowe struktury, nowe metody i pojęcia. Matematyka z jądra jest nauką samowystarczalną i przez swe warstwy zewnętrzne kontaktuje się z ludzkimi problemami. Teorie z tych warstw zewnętrznych nastawiane są bardziej na rozwiązywanie zagadnień niż na odkrywanie form podstawowych. Warstwy odległe od jądra posługują się matematyką raczej jako metaformą niż jako teorią: zastosowania matematyki zlewają się z techniką tak gruntownie, że powstaje całkiem odmienna dyscyplina wiedzy. Poprzez trudne do określenia granice miedzy tymi warstwami teoria i problemy przenikają się wzajemnie i wzbogacają się. Dostarcza to nowych źródeł inspiracji w matematyce i w innych naukach. W ostatnich latach świat matematyki rozwinął się z pojedynczej dyscypliny w grono splecionych dyscyplin nazywanych zazwyczaj naukami matematycznymi. Do nauk tych poza dziedzinami z jądra takimi jak: teoria liczb, logika matematyczna, topologia różniczkowa i geometria algebraiczna wchodzą także takie działy nauk stosowanych jak: badania operacyjne, statystyka, informatyka, kombinatoryka i programowanie matematyczne.

Model matematyczny [7] jest dowolnym, pełnym i niesprzecznym układem równań matematycznych, mających odpowiadać jakiejś innej wielkości, jego prototypowi. Ten prototyp musi być wielkością fizyczną, biologiczną, społeczną, psychologiczną czy pojęciową, a nawet innym modelem matematycznym. Zamiast równań można wprowadzić słowo „struktury", nie zawsze, bowiem mamy do czynienia z modelem numerycznym. Oto niektóre z celów, dla jakich konstruuje się modele:

1. uzyskanie odpowiedzi na to, co się zdarzy w świecie fizycznym,
2. ukierunkowanie dalszych eksperymentów czy obserwacji,
3. uzyskanie postępu koncepcyjnego i lepszego rozumienia,
4. aksjomatyzowanie sytuacji fizycznej,
5. rozwój matematyki i sztuki tworzenia modeli matematycznych.

Uświadomienie sobie, że teorie fizyczne mogą się zmieniać i ulegać modyfikacją, że mogą istnieć teorie konkurencyjne, że matematyka może nie wystarczyć do pełnego wykorzystania teorii – wszystko to doprowadziło do pragmatycznego uznania modelu jako „rzeczy chwilowej", bardziej dogodnego przybliżenia stanu rzeczy niż wyraźnie odwiecznej prawdy. Model może być uznawany za dobry lub zły, prosty lub skomplikowany, estetyczny lub okropny, użyteczny lub bezużyteczny, w mniejszym natomiast stopniu jako „prawdziwy" lub „fałszywy". Współczesna koncentracja na modelach w przeciwstawieniu do teorii doprowadziła do badania twórczości modelowej jako pracowitej sztuki i wynikającego stąd spadku zainteresowania specyficzną sytuacją fizyczną, którą się modeluje.

3. Złożoność algorytmów [13]

Wszystkie problemy, z jakimi można spotkać się w matematyce można podzielić na dwie klasy:

- takie, dla rozwiązania, których algorytm nie istnieje; problemy takie są nazywane nierozstrzygalnymi. Pewne przypadki szczególne tych problemów mogą być rozwiązane przez zgadnięcie, ale ogólna metoda ich rozwiązania nigdy nie może być znaleziona;
- takie problemy, które można rozwiązać za pomocą algorytmów.

Czas, jaki jest potrzebny do wykonania algorytmu, wyrażony w funkcji rozmiaru problemu, nazywa się złożonością czasową algorytmu.

Jeśli f (n) i g(n) są funkcjami określonymi w zbiorze liczb naturalnych, to mówimy, że f(n) jest 0(g(n)) („rzędu g(n)"), jeśli istnieją stałe k,m ∈ N, takie że $f(n) \leq k \cdot g(n)$ dla wszystkich $n \geq m$.

Algorytm o złożoności czasowej wielomianowej, lub w skrócie algorytm wielomianowy, to taki algorytm, którego złożoność czasowa jest 0(p(n)), przy czym p(n) jest wielomianem, zaś n oznacza rozmiar problemu. Jeżeli powyższy warunek nie jest spełniony, to algorytm nazywa się algorytmem o złożoności czasowej wykładniczej lub po prostu algorytmem wykładniczym (eksponencjalnym).

Rozmiar pamięci urządzenia liczącego, potrzebny do wykonania algorytmu, wyrażony w funkcji rozmiaru problemu nazywa się złożonością pamięciową algorytmu. Definicje złożoności wielomianowej i wykładniczej algorytmów dla tego przypadku są analogiczna jak dla przypadku złożoności czasowej.

W klasyfikacji problemów rozwiązalnych stosuje się podział dychotomiczny: do jednej klasy zalicza się problemy „łatwe", a do drugiej „trudne". Problemy „łatwe" to takie, dla których istnieją algorytmy wielomianowe. Natomiast problemy „trudne" charakteryzują się wykładniczą złożonością algorytmów służących do ich rozwiązywania. Przypuśćmy, że mamy siedem algorytmów o następujących złożonościach:

Algorytm	Złożoność
A_1	0(n)
A_2	0(nlogn)
A_3	$0(n^2)$
A_4	$0(n^3)$
A_5	$0(2^n)$
A_6	$0(3^n)$
A_7	0(n!)

Algorytmy Al ÷ A4 są algorytmami wielomianowymi, natomiast algorytmy A5 ÷ A7 mają złożoność eksponencjalną (choć np. funkcja n! nie jest funkcję wykładniczą). Jest oczywiste, że w niektórych szczególnych przypadkach pewne algorytmy wykładnicze są efektywniejsze od algorytmów wielomianowych. Na przykład przypuśćmy, że złożoność algorytmu A3 wynosi w rzeczywistości $100 \cdot n^2$, natomiast złożoność algorytmu A6 jest równa $2 \cdot 3^n$. Wówczas algorytm A6 jest skuteczniejszy od algorytmu A3 dla wszystkich problemów o rozmiarze $n \leq 7$. Jednakże w ogólności, dla problemów o dowolnie dużych rozmiarach efektywniejsze są oczywiście algorytmy wielomianowe.

Maksymalne rozmiary problemów rozwiązywalnych w różnych jednostkach czasu.

Tabela 1

Złożoność czasowa	Rozmiar maksymalnego problemu rozwiązywalnego w ciągu		
	1 sekundy	1 minuty	1 godziny
n	1 000 000	60 000 000	3 600 000 000
$n \log_2 n$	62 746	2 801 417	133 378 058
n^2	1 000	7 745	60 000
n^3	100	391	1 532
2^n	19	25	30
3^n	12	16	20
10^n	6	7	9

Porównanie czasów wykonania algorytmów o różnych złożonościach czasowych

Tabela 2

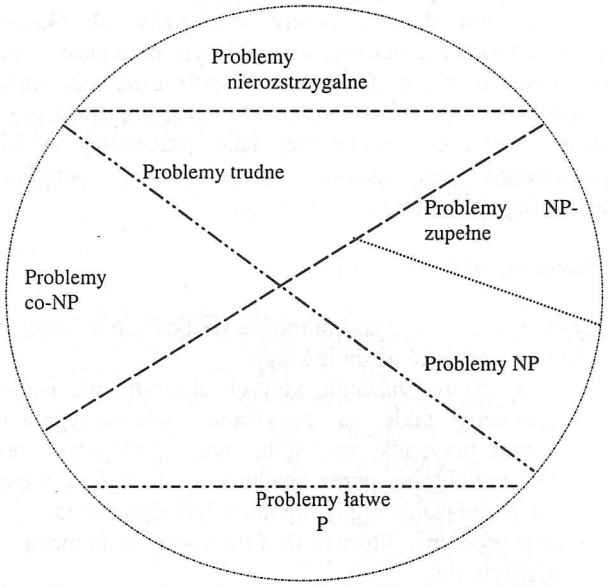
Rozmiar problemu n / Funkcja złożoności czasowej	10	20	30	40	50
n	$10 \cdot 10^{-6}$ sekundy	$20 \cdot 10^{-6}$ sekundy	$30 \cdot 10^{-6}$ sekundy	$40 \cdot 10^{-6}$ sekundy	$50 \cdot 10^{-6}$ sekundy
$n \log_2 n$	$33{,}2 \cdot 10^{-6}$ sekundy	$86{,}4 \cdot 10^{-6}$ sekundy	$147{,}2 \cdot 10^{-6}$ sekundy	$212{,}9 \cdot 10^{-6}$ sekundy	$282{,}5 \cdot 10^{-6}$ sekundy
n^2	$0{,}1 \cdot 10^{-3}$ sekundy	$0{,}4 \cdot 10^{-3}$ sekundy	$0{,}9 \cdot 10^{-3}$ sekundy	$1{,}6 \cdot 10^{-3}$ sekundy	$2{,}5 \cdot 10^{-3}$ sekundy
n^3	$1 \cdot 10^{-3}$ sekundy	$8 \cdot 10^{-3}$ sekundy	$27 \cdot 10^{-3}$ sekundy	$64 \cdot 10^{-3}$ sekundy	$125 \cdot 10^{-3}$ sekundy
2^n	0,001 sekundy	1 sekunda	17,9 minuty	12,7 dnia	35,7 lat
3^n	0,059 sekundy	58,1 minuty	6,53 roku	3 855 wieków	$2{,}3 \cdot 10^8$ wieków
10^n	2,8 godziny	31 710 wieków	$3{,}17 \cdot 10^{14}$ wieków	$3{,}17 \cdot 10^{24}$ wieków	$3{,}17 \cdot 10^{34}$ wieków

Efekt przyspieszenia obliczeń

Tabela 3

Złożoność czasowa	Rozmiar maksymalnego problemu rozwiązywalnego w ciągu 1 godziny na komputerze			
	aktualnym	10-krotnie szybszym	100-krotnie szybszym	1000-krotnie szybszym
n	n_1	$10\ n_1$	$100\ n_1$	$1\,000\ n_1$
n^2	n_2	$3{,}16\ n_2$	$10\ n_2$	$31{,}62\ n_2$
n^3	n_3	$2{,}15\ n_3$	$4{,}63\ n_3$	$10\ n_3$
2^n	n_4	$n_4 + 3{,}32$	$n_4 + 6{,}64$	$n_4 + 9{,}97$
3^n	n_5	$n_5 + 2{,}1$	$n_5 + 4{,}19$	$n5 + 6{,}29$
10^n	n_6	$n_6 + 1$	$n_6 + 2$	$n_6 + 3$

Przyjmijmy tutaj, że czas potrzebny do wykonania algorytmu będziemy mierzyć w umownych jednostkach, dla przykładu w mikrosekundach. W tabeli 1 podano rozmiary problemów jakie mogą być rozwiązane w ciągu 1 sekundy, 1 minuty i 1 godziny przez algorytmy o różnych złożonościach czasowych. Tempo wzrostu czasu wykonywania algorytmów o różnych złożonościach czasowych przedstawiono w tab 2. Szczególnie interesujące, wobec faktu, że wiek Ziemi szacuje się na $4{,}6 \cdot 10^9$ lat, wydają się być wyniki przedstawione w prawej dolnej części tej tabeli. W związku z tym rozpatrzmy jaki wpływ na szybkość wykonywania algorytmu na jakość stosowanego komputera. W tabeli 3 przedstawiono największe rozmiary szczególnych przypadków problemów, które mogę być rozwiązane w ciągu 1 godziny przy zastosowaniu komputerów o różnych szybkościach działania. Wynika z niej, że algorytmy o złożoności wykładniczej są w bardzo małym stopniu podatne na przyspieszenia polegające na stosowaniu do ich rozwiązania szybszych komputerów.



Rys.1. Klasyfikacja problemów

Na rysunku 1 przedstawiono schematycznie klasyfikację problemów. Wyróżniono tam problemy nierozstrzygalne, czyli takie, dla których udowodniono, że nie istnieje algorytm służący do ich rozwiązania. Nie oznacza to, że pewne przypadki nie nogę być rozwiązane przez zgadnięcie. Klasa problemów trudnych zawiera tylko problemy o złożoności wykładniczej. Natomiast problemy łatwe, czyli problemy wchodzące do klasy P (P-skrót angielskiego słowa "polynomial"), to problemy o złożoności wielomianowej. Tak wprowadzoną klasyfikację teraz rozbudujemy.

Klasa NP (NP to pierwsze litery angielskich słów "nondeterministic polynomial") zawiera wszystkie problemy, łatwe, a także problemy, których status jest mniej pewny: znane są tylko algorytmy o złożoności eksponencjalnej służące do ich rozwiązania, lecz nie dowiedziono, że nie istnieją dla nich algorytmy o złożoności wielomianowej. Klasę NP tworzy zbiór problemów, które mogą być rozwiązane za pomocą algorytmów niedeterministycznych o złożoności wielomianowej. Algorytm niedeterministyczny jest rozumiany tutaj jako algorytm składający się z dwóch etapów: etapu zgadnięcia i etapu weryfikacji. W pierwszym etapie po prostu zgadujemy rozwiązanie, a weryfikacja

polega na sprawdzeniu w sposób deterministyczny, z zastosowaniem algorytmu wielomianowego, czy rozwiązanie to spełnia nałożone na problem warunki.

Jeśli rozpatrywać problemy decyzyjne, to rozwiązania problemów z pomocą algorytmów niedeterministycznych różnią się od rozwiązań problemów uzyskiwanych za pomocą algorytmów deterministycznych brakiem pewnego rodzaju symetrii. Jeśli problem „Czy X jest prawdziwe dla zadanego przypadku I?" rozwiązujemy za pomocą wielomianowego algorytmu deterministycznego, to także za pomocą algorytmu tego samego typu można rozwiązać problem komplementarny „Czy X jest fałszywe dla zadanego przypadka I?" Inaczej sprawa przedstawia się z wielomianowymi algorytmami niedeterministycznymi. W tym przypadku wspomniana symetria nie zawsze występuje, stąd na rysunku 1 wyróżniono klasę co-NP, czyli klasę problemów komplementarnych (dopełnieniowych) do problemów typu NP.

Okazuje się, że przekrój mnogościowy klasy NP z klasą co-NP jest nie pusty. Przykładem problemu należącego zarówno do klasy NP jak i do klasy co-NP jest problem liczb złożonych, w którym pytamy czy liczba naturalna jest złożona, tzn. czy jest iloczynem dwóch liczb naturalnych od niej mniejszych.

Być może dla rozwiązania problemów typu NP istnieję algorytmy deterministyczne wielomianowe, lecz jak dotąd problem „Czy P = NP ?" jest nierozwiązany, choć przypuszcza się, że $P \neq NP$.

Zatrzymajmy się jeszcze przez moment przy klasie NP. Wyróżniono w niej podklasę problemów NP-zupełnych odznaczających się ciekawą własnością: jeśli choć jeden z tych problemów można by było rozwiązać za pomocą wielomianowego algorytmu deterministycznego, to także wszystkie pozostałe można by rozwiązać korzystając z takich algorytmów.

Istnieje ścisła zależność pomiędzy problemami NP-zupełnymi i hipotezą, że NP ≠ co-NP. Otóż jeśliby istniał problem NP-zupełny taki, że jego wersja komplementarna byłaby typu NP, to wówczas klasa NP i klasa co-NP stanowiłyby tę samą klasę problemów.

Obecnie przedstawimy jak różne operacje matematyczne mają wpływ na czas obliczeń procesorów[17]. Jako przykład analizuje się procesor specjalizowany iAPX 86/20 złożone z uniwersalnego mikroprocesora Intel 8086 (procesor główny) i specjalizowanego mikroprocesora numerycznego Intel 8087 (koproccsor). Mikroprocesor numeryczny wykonuje operacje arytmetyczne i oblicza funkcje elementarne od argumentów z zakresu 10^{-308} ... 10^{+308}, z dokładnością sięgającą 17 cyfr znaczących. Zestaw wykonywanych operacji obejmuje:

- działania arytmetyczne dodawania, odejmowania, mnożenia i dzielenia,
- obliczanie funkcji pierwiastka kwadratowego, tangens, e do potęgi.

Czasy wykonania przykładowych operacji zebrane są w tabeli 4. W czasie pracy zestawu iAPX 86/20 obydwa procesory współpracują ściśle przy wykonaniu tego samego programu. Kolejne rozkazy programu są wskazywane przez licznik rozkazów procesora głównego (8086). Koprocesor numeryczny (8087) nie ma własnego licznika rozkazów, lecz wychwytuje przeznaczone dla siebie rozkazy z ciągu rozkazów pobieranych przez procesor główny i wykonuje operacje matematyczne. Z tabeli 4 wynika, że operacje dodawania, odejmowania i mnożenia zużywają znacznie mniej czasu od operacji dzielenia, pierwiastkowania i potęgowania.

Czas wykonania przykładowych operacji koprocesora 8087

Tabela 4

Operacja	Czas wykonywania
Dodawanie	14 μs
Odejmowanie	18 μs
Mnożenie	19 μs
Dzielenie	39 μs
Pierwiastek	36 μs
Tangens	90 μs
e do potęgi	100 μs

W ostatnich latach problemami złożoności algorytmów i obliczeń coraz intensywniej zajmuje się informatyka. Informatyka (ang. computer science) [9] to usystematyzowana wiedza na temat obliczania. Jej początki sięgają projektowania algorytmów przez Euklidesa oraz wykorzystywania asymptotycznej złożoności i redukowalności przez Babilończyków [8]. Współczesne zainteresowania tą dziedziną zostały jednak ukształtowane przez dwa ważne wydarzenia: powstanie nowoczesnych maszyn liczących (komputerów), mogących wykonywać wiele milionów operacji na sekundę, oraz formalizację pojęcia efektywnej procedury.

Informatyka zawiera dwie główne składowe: po pierwsze, fundamentalne idee i modele leżące u podstaw obliczania, i po drugie, techniki inżynieryjne używane przy projektowaniu systemów obliczeniowych, zarówno sprzętowych, jak i oprogramowania, a w szczególności zastosowanie teorii do ich projektowania.

Informatyka teoretyczna ma źródła w wielu różnych dziedzinach. Biolodzy studiowali modele sieci neuronowych, inżynierowie elektrycy rozwijali teorię przełączników jako narzędzie projektowania sprzętu, matematycy pracowali nad podstawami logiki, lingwiści badali gramatyki języków naturalnych i ze wszystkich tych badań wyłoniły się modele o podstawowym znaczeniu dla informatyki teoretycznej.

4. Automaty skończone jako modele matematyczne przetwarzania informacji

4.1. Automaty skończone w informatyce

Pojęcia automatu skończonego i wyrażenia regularnego [9] zostały początkowo stworzone z myślą o sieciach neuronowych i układach przełączających. Języki akceptowane przez automaty skończone można łatwo opisać za pomocą prostych wyrażeń zwanych wyrażeniami regularnymi. Klasa języków akceptowalnych przez automaty skończone pokrywa się dokładnie z klasą języków opisywalnych wyrażeniami regularnymi. Później posłużyły one jako pożyteczne narzędzia do projektowania analizatorów leksykalnych, czyli tych części składowych kompilatorów, które grupują symbole w tokeny —

niepodzielne jednostki będące np. nazwami zmiennych lub słowami kluczowymi. Pewne systemy generowania kompilatorów przekształcają w sposób automatyczny wyrażenia regularne na automaty skończone, używane jako analizatory leksykalne. Inne zastosowania wyrażeń regularnych i automatów skończonych znaleziono w edytorach tekstu, przetwarzaniu obrazów, różnych programach przetwarzających tekst i przeszukujących pliki. Traktowane jako pewne pojęcia matematyczne, znalazły one również zastosowanie w innych dziedzinach, takich jak np. logika.

Pojęcia gramatyki bezkontekstowej i odpowiadającego jej automatu ze stosem były niezwykle pomocne przy specyfikowaniu języków programowania i projektowaniu analizatorów syntaktycznych (parserów), będących następną kluczową składową kompilatorów [9]. Języki generowane przez gramatykę bezkontekstową są szeroko stosowane do opisu języków sztucznych w szczególności języków programowania [12]. Obecnie, dzięki powszechnej znajomości całej gamy technik opartych na gramatykach bezkontekstowych, projektowanie analizatorów syntaktycznych nie jest już problemem, a analiza syntaktyczna zajmuje tylko mały procent czasu przeznaczanego na typową kompilację.

Przedmiotem badań w teorii automatów jest konstrukcja modeli matematycznych opisujących sposoby przetwarzania informacji przez różnorodne systemy cyfrowe. W szczególności przedmiotem badań w teorii automatów są abstrakcyjne modele układów technicznych pracujących z pomocą sygnałów dyskretnych, zwanych również sygnałami cyfrowymi. Przedmiot szczególnego zainteresowania stanowią cyfrowe maszyny matematyczne, cyfrowe układy sterowania procesami technologicznymi i cyfrowe układy transmisji danych. Z drugiej strony teoria automatów jest jedną z gałęzi ogólnej teorii systemów i jako taka daje narzędzie potrzebne do stawiania i rozwiązywania problemów ogólnych, które może być zastosowane zarówno rozwiązywaniu problemów już znanych, jak również w przyszłości. Teoria automatów stanowi w swej istocie teorię obliczalności obejmującą problemy obwodów komputerowych, oprogramowania, działania układu nerwowego i procesy wzrostu organizmów żywych. Automat, w szczególności skończony, jest abstrakcyjny pojęciem matematycznym, modelem obliczeń lub modelem pewnego procesu. Do badań w teorii automatów stosuje się algebrę ogólną i teorię grafów. Pewne pojęcia z algebry mają swoje początki w sieciach komputerowych.

Historycznie i merytorycznie patrząc, teoria automatów jest gałęzią informatyki mającą silne związki z lingwistyką matematyczną, matematyką dyskretną i inżynierią systemów cyfrowych. Związki te wynikają z natury pojęcia automatu, który może być rozumiany jako akceptor języków, algebra abstrakcyjna, graf skierowany, przetwornik sygnałów cyfrowych i model matematyczny systemów cyfrowych. Powyższe fakty dowodzą interdyscyplinarności teorii automatów, a także jej walorów ogólno poznawczych w zakresie modelowania procesów technicznych. W początkowym okresie, głównie w latach pięćdziesiątych i pierwszej połowie lat sześćdziesiątych, teoria automatów obejmowała głównie problemy związane z lingwistyką matematyczną. Druga połowa lat sześćdziesiątych i lata siedemdziesiąte dotyczą głównie badań w zakresie strukturalnej teorii automatów, a także algebraicznej teorii automatów w zakresie mającym znaczenie dla rozwiązywania ważnych problemów analizy i syntezy. Lata siedemdziesiąte i późniejsze przynoszą znaczący wpływ analizy złożonościowej w teorii automatów, a także rozwój teorii automatów zmiennych w czasie jako adekwatnych modeli matematycznych dla wielu procesów obliczeniowych i technicznych.

Ważne i znaczące są też wpływy teorii automatów na rozwój podstaw techniki cyfrowej, projektowania i użytkowania sprzętu informatycznego, które wyrażają się w powstaniu wielu nowych metod projektowych wdrożonych do praktyki inżynierskiej w procesie dydaktycznym. W ostatnim dziesięcioleciu największe znaczenie miały badania w zakresie strukturalnej teorii automatów, w szczególności zmiennych w czasie.

Teoria automatów jest skutecznym narzędziem umożliwiającym formalne projektowanie prostych i złożonych układów cyfrowych z zastosowaniem standardowych układów elementarnych. Rozwój technologii obwodów scalonych i wzrost funkcjonalnej złożoności podzespołów cyfrowych oraz tendencje wprowadzania maszyn cyfrowych przyczyniły się do znacznego zainteresowania złożonością obliczeniową dla wielu zagadnień. Automat jest abstrakcyjnym pojęciem matematycznym, modelem obliczeń lub modelem pewnego procesu. Algebraiczna teoria automatów z jednej strony jest teoretycznym uogólnieniem teorii układów logicznych, z drugiej strony może być traktowana jako dział algebry. Pojęcia z algebry w postaci sformalizowanej są analizowane i przekształcane do postaci dogodnych do optymalizacji.
Z postaci abstrakcyjnej, w procesie syntezy, można je przekształcić w schemat logiczny, oprogramowanie lub ich kombinację. Tym samym uczy teoria automatów jak koncepcyjnie i obliczeniowo rozważać wzrastające co do wielkości i złożoności problemy w informatyce. Równocześnie teoria automatów kształci też pierwiastek kreatywności, bowiem proces myślowy ma tu zasadniczy charakter heurystyczny.

Rozwój teorii automatów był stymulowany przez dwie uzupełniające się tendencje :

a) konstruowanie modeli bliżej związanych ze współczesnymi systemami i oprogramowaniem
b) znajdowanie poprawnych narzędzi matematycznych (języka matematycznego), w którym można wyrazić procesy obliczeniowe o dużej różnorodności.

Równocześnie rozwój ten związany był ze wzrostem znaczenia cyfrowych maszyn matematycznych w różnych gałęziach wytwarzania, jak również z doskonaleniem metod użytkowania, analizy i syntezy cyfrowych układów sterowania procesami technologicznymi z uwzględnieniem skali scalenia i złożoności funkcjonalnej podzespołów cyfrowych.

4.2. Struktury algebraiczne i ich reprezentacje w teorii automatów

Od blisko czterdziestu lat jesteśmy świadkami intensywnego rozwoju teorii automatów, szczególnie algebraicznej teorii automatów rozwijanej na gruncie teorii

półgrup. Komplementarność strukturalno-językowa automatów zaowocowała głębokimi pracami z zakresu akceptowalności języków przez automaty jak i automatowej reprezentacji poszczególnych klas języków. Definicje relacji i równoważności Myhilla [3, 9, 12] na zbiorze stanów automatu oraz półgrupy charakterystycznej automatu pozwoliły wydobyć zeń możliwości obliczeniowe. Tak więc na automat spojrzeć można zarówno pod kątem widzenia systemu algebraicznego o charakterze strukturalno - językowym jak i modelu obliczeń.

W algebrze wyróżniamy siedem głównych typów struktur algebraicznych: półgrupy, grupy, pierścienie, ciała, moduły, przestrzenie wektorowe i algebry [16].

Zbiór składający się z elementów $a_1, a_2, ..., a_n$ będziemy oznaczali $\{ a_1, a_2, ..., a_n\}$. To, że x jest elementem zbioru A, będziemy zapisywali krótko $x \in A$ [15]. Zbiór A nazywamy skończonym, gdy istnieje liczba naturalna n taka, że dany zbiór jest równoliczny ze zbiorem liczb naturalnych: 1, 2, 3, ..., n. Mówimy wówczas, że zbiór A ma n elementów [11]. Dwa zbiory A i B nazywamy równolicznym jeżeli element jednego zbioru można połączyć z element drugiego zbioru w pary (a, b) tak, by każdy element $a \in A$ był złączony z dokładnie jednym elementem $b \in B$ i odwrotnie; by każdy element zbioru B był złączony z dokładnie jednym elementem zbioru A [11]. Struktury algebraiczne na zbiorze A nazywamy każdy zespół $(A, F_1, ..., F_m; h_1, ..., h_n; g_1, ..., g_n)$ złożony na zbiorze A, z pewnej ilości działań wewnętrznych $h_1: A \times A \rightarrow A, ..., h_n: A \times A \rightarrow A$, i z pewnych ilości działań zewnętrznych $g_1: F_1 \times A \rightarrow A, ..., g_m: F_m \times A \rightarrow A$. Działania wewnętrzne lub krótko działania na zbiorze A nazywamy każde odwzorowanie iloczynu kartezjańskiego $A \times A$ w zbiór A. Innymi słowy działanie w zbiorze A jest każde odwzorowanie $h: A \times A \rightarrow A$, przyporządkowujące dowolnej parze elementów a, b zbioru A w ściśle określony element c zbioru A.

Odwzorowanie iloczynu kartezjańskiego $F \times A$ w zbiorze A nazywamy działaniem zewnętrznym określonym w zbiorze A. Działaniem zewnętrznym jest więc odwzorowanie g przyporządkowujące każdej parze elementów $x \in F$ i $a \in A$ ściśle określony element zbioru A, który oznaczać będziemy symbolem g(x,a). Najczęściej zbiory A i F są najzupełniej różne. Jeżeli A = F to definicje działanie zewnętrznego pokrywa się z definicją działania wewnętrznego. W rozpatrywanych przez algebrę strukturach algebraicznych ilość działań wewnętrznych wynosi jeden lub dwa; w niektórych z nich nie ma w ogóle działań zewnętrznych, w innych zaś jest tylko jedno takie działanie. Działanie wewnętrzne oznaczamy przy tym z reguły jako addytywne i multiplikatywne i nazywamy je odpowiednio dodawaniem lub mnożeniem. Jeżeli określone jest jedno działanie, to oznaczamy je na ogół multiplikatywnie, a addytywnie jedynie wtedy, gdy jest to działanie przemienne. W dalszych rozważaniach zajmiemy się strukturą algebraiczna – półgrupą i określonej na niej operacją wewnętrzną mnożenia. Grupoidem nazywamy parę uporządkowaną $(A, \circ)$, gdzie A – niepusty zbiór, $(\circ)$ operacja binarna na zbiorze. Operację binarną na zbiorze A nazywamy przekształcenie niepustego podzbioru zbioru $A \times A$ w zbiór A. Binarną operacje $(\circ)$ na zbiorze A nazywamy łączną (asocjatywną), jeśli

$a \circ (b \circ c) = (a \circ b) \circ c$ dla wszystkich a, b, $c \in A$. Półgrupa, to taki grupoid $(A, \circ)$, w którym operacja $(\circ)$ jest asocjatywna. Niech Σ będzie dowolnym zbiorem niepustym. Zbiór Σ będziemy nazywać alfabetem, a jego elementy literami. Słowem x w alfabecie Σ nazywamy dowolny ciąg liter alfabetu Σ napisanych obok siebie, a długość słowa (oznaczoną przez $|x|$) nazywamy liczbą tych liter. Operacje połączenia dwóch słów, polegająca na napisaniu ich obok siebie w celu otrzymania nowego słowa nazywamy konkatenacją.

Skończonym automatem zdeterminowanym bez wyjść lub krótko automatem nazywamy 3-kę uporządkowań (S, Σ, δ) gdzie: S jest skończonym, niepustym zbiorem stanów, Σ jest skończonym, niepustym zbiorem wejść, $\delta: S \times \Sigma \rightarrow S$ jest funkcję przejść [3, 12]. Symbolem Σ^+ oznaczać będziemy przeliczalny nieskończony zbiór ciągów o skończonej długości utworzonych z elementów zbioru Σ. Zbiór Σ^+ razem z operacją konkatenacji tworzy półgrupę wolną zwaną półgrupą wejściową. Symbolem Σ^* oznaczać będziemy monoid wejściowy, czyli $\Sigma = \Sigma^* \cup \lambda$, gdzie λ jest słowem pustym. Funkcję δ rozszerzamy do obszaru określoności $S \times \Sigma^+$ w następujący sposób: niech δ (s, x) będzie zdefiniowane, wtedy: $\delta(s, x\sigma) = \delta(\delta(s, x), \sigma)$ dla każdego $s \in S$, $x \in \Sigma^+$, $\sigma \in \Sigma$. Na zbiorze Σ^* zdefiniujemy relację: x Ry wtedy i tylko wtedy, gdy $\forall_{s \in S} \delta(s, x) = \delta(s, y)$, (gdzie $\forall$ kwantyfikator ogólny mówiący, że dla każdego $s \in S$ istnieje przedstawione równanie). R jest relacją równoważności (relacja Myhilla) [3, 12]. Klasę równoważności zawierającą element $x \in \Sigma^*$ oznaczać będziemy $\overline{x}$, a zbiór wszystkich klas równoważności oznaczać będziemy $\overline{I}$. Zbiór $\overline{I}$ łącznie z operacją $\circ$, gdzie $\overline{x} \circ \overline{y} = \overline{xy}$ tworzy półgrupę (odpowiednio, monoid) zwaną półgrupą charakterystyczną (odpowiednio, monoidem charakterystycznym). Półgrupę charakterystyczną automatu A oznaczać będziemy $\overline{I(A)}$.

Sumę prostą automatów $A = ({}^A S, \Sigma, {}^A\delta)$ i $B = ({}^B S, \Sigma, {}^B\delta)$ jest trójka uporządkowana

$A \cup B = ({}^{A \cup B} S, \Sigma, {}^{A \cup B}\delta)$, gdzie: ${}^{A \cup B} S = {}^A S \cup {}^B S$;

${}^{A \cup B}\delta: {}^{A \cup B} S \times \Sigma \rightarrow {}^{A \cup B} S$ i dla każdego $s \in {}^{A \cup B} S$ i $\delta \in \Sigma$ zachodzi

$$ {}^{A \cup B}\delta(s, \sigma) = \begin{cases} {}^A\delta({}^A s, \sigma), & {}^A s \in {}^A S \\ {}^B\delta({}^B s, \sigma), & {}^B s \in {}^B S \end{cases} $$

Iloczyn prosty automatów $A = ({}^A S, \Sigma, {}^A\delta)$ i $B = ({}^B S, \Sigma, {}^B\delta)$ jest trójka uporządkowana

$A \times B = ({}^{A \times B} S, \Sigma, {}^{A \times B}\delta)$, gdzie: ${}^{A \times B} S = {}^A S \times {}^B S$;
${}^{A \times B}\delta: {}^{A \times B} S \times \Sigma \rightarrow {}^{A \times B} S$,

a funkcja przejść jest definiowana jak następuje
${}^{A \times B}\delta(({}^A s, {}^B s), (\sigma)) = ({}^A\delta({}^A s, \sigma), ({}^B\delta({}^B s, \sigma))$.

Półgrupa charakterystyczna jest szczególnie istotnym pojęciem w teorii automatów; jest ona nośnikiem ważnych informacji, tak w aspekcie strukturalnym, jak i lingwistycznym. Ma to bezpośrednie ważkie konsekwencje praktyczne w sferze projektowania optymalnych układów logicznych.

Półgrupa charakterystyczna określa zdolność do przetwarzania informacji. Suma prosta i iloczyn prosty półgrup charakterystycznych automatów skończonych można uważać za realizację odpowiednio sekwencyjną i równoległą obliczeń. W pracy [3] udowodniono, że suma prosta i iloczyn prosty półgrup charakterystycznych automatów asynchronicznych, zdeterminowanych, skończonych o alfabecie dwuwejściowym są izomorficzne czyli równoważne. Wziąwszy pod uwagę iż półgrupa charakterystyczna określa zdolność do przetwarzania informacji, zaś sumę prostą i iloczyn prosty można uważać za realizację odpowiednio sekwencyjną i równoległa obliczeń uzyskane rezultaty oznaczają iż owa zdolność nie zależy od realizacji sekwencyjnej lub równoległej (taka sama liczba klas abstrakcji odpowiednich półgrup charakterystycznych). Wynik ten uzyskano stosując nowy sposób wyznaczania NWW liczb naturalnych przedstawiony poniżej. Nowy sposób wyznaczania NWW liczb naturalnych umożliwia minimalizację obliczeń w stosunku do dotychczas stosowanych algorytmów [10, 11, 14].

5. Tradycyjny sposób wyznaczania NWW z wykorzystaniem techniki komputerowej

Znane są metody obliczania NWD (największy wspólny dzielnik) i NWW dwóch liczb całkowitych [10,14]. Dowolne dwie liczby całkowite m_0 i m_1 różne od zera można napisać w postaci iloczynu tych samych liczb pierwszych:

$$m_0 = \pm p_1^{r_i} \cdots p_n^{r_n}, \quad m_1 = \pm p_1^{s_i} \cdots p_n^{s_n}$$

jeśli dopuścimy wykładniki zerowe (oczywiście $p_i^0 = 1$)

Dla dwóch liczb całkowitych m_0 i m_1

$$NWD(m_0, m_1) = p_1^{a_i} \cdots p_n^{a_n},$$

$$NWW(m_0, m_1) = p_1^{b_i} \cdots p_n^{b_n}$$

gdzie : $a_i = \min(r_i, s_i)$, $b_i = \max(r_i, s_i)$, $i = 1, \cdots, n$.

Dodatkowo przyjmujemy $NWD(0, m_1) = |m_1|$,

$NWW(0, m_1) = 0$ oraz $|m_0| > |m_1|$

Bardzo często realizuje się obliczanie NWD i NWW z wykorzystaniem komputera. Sposób według [10, 14], ze względu na operacje potęgowania i wyszukiwania a_i i b_i jest niewygodny. Należy szukać takich metod aby korzystać przy obliczaniu NWD i NWW z operacji dodawania, odejmowania i mnożenia, co ułatwia tworzenie algorytmów minimalizujących czas obliczeń na komputerze.

Obliczanie NWD na komputerze, które jest oczywiste to branie kolejnych liczb od jedynki do min ($|m_0|$, $|m_1|$) sprawdzenie czy dzielą m_0 i m_1 bez reszty, a następnie wybranie liczby o bezwzględnej wartości największej spełniającej ten warunek. Sposób ten nazwiemy algorytmem "siłowym" obliczania NWD. Algorytm ten jest niezwykle prosty, ale zachodzi pytanie, czy nie można wykonać mniej operacji osiągając ten sam rezultat. Okazuje się, że można, zrobił to kiedyś Euklides. Algorytm Euklidesa obliczania NWD jest prosty, a liczba operacji jest o wiele mniejsza w porównaniu z „siłowym" algorytmem obliczania NWD.

Liczba kroków w "siłowym" algorytmie obliczania NWD jest równa $|m_1| - 1$. W algorytmie Euklidesa [14,18] liczba kroków jest równa lub mniejsza od $|m_1| - 1$ (od 1 do $|m_1| - 1$). Liczba kroków w algorytmie Euklidesa zależy od wartości liczb m_0 i m_1.

Do obliczenia NWW na komputerze istnieje również algorytm, który podobnie jak przy obliczaniu NWD nazwiemy "siłowym". Algorytm ten jest niezwykle prosty. Obliczanie NWW na komputerze polega na tym, że dla $|m_0| > |m_1| > 0$ bierzemy kolejno liczby $|m_0+k|$ gdzie $k = 1, 2, \ldots, n$ i sprawdzamy czy liczby m_0 i m_1 dzielą się przez (m_0+k) bez reszty. Wybrane liczby o bezwzględnej wartości najmniejszej spełniający ten warunek, przy założeniu, że jest ona równa lub większa od $|m_0|$ kończy algorytm. Liczba ta jest NWW (m_0 , m_1).

Podczas rozwiązywania problemów dotyczących złożoności obliczeniowych półgrup charakterystycznych automatów skończonych [3] znaleziono bardzo prosty algorytm obliczania NWW liczb naturalnych. Charakteryzuje się on tym, że dla jego realizacji podobnie jak w algorytmie Euklidesa przy obliczaniu NWD wystarczające są operacje dodawania, odejmowania i mnożenia. Liczba operacji w czasie realizacji tego algorytmu jest o wiele mniejsza w porównaniu z „siłowym" algorytmem obliczania NWW. Liczba kroków w "siłowym" algorytmie obliczania NWW jest równa $p = |m_0| \cdot |m_1| - |m_0| - 1$. W algorytmie obliczania NWW liczb naturalnych przedstawionym w artykule liczba kroków zależy od wartości liczb naturalnych m_0 i m_1 i zmienia się od 1 do $m_0 - 1$.

Maksymalna liczba kroków tego algorytmu przy obliczaniu NWW jest równa $m_0 - 1$ i występuje wtedy, gdy $m_1 = m_0 - 1$. Dotychczas stosowano dwie metody wyznaczania Najmniejszej Wspólnej Wielokrotności (NWW) i Największego Wspólnego Dzielnika (NWD) liczb naturalnych, które przedstawiono w literaturze [14, 18]. Metoda pierwsza polega na wyznaczaniu NWW i NWD za pomocą rozkładu liczb naturalnych na czynniki pierwsze [18]. Druga metoda polega na wyznaczaniu NWD stosując algorytm Euklidesa i potem korzystając z zależności, że NWW ($m_0, m_1, \ldots, m_n$) · NWD ($m_0, m_1, \ldots, m_n$) = $m_0, m_1, \ldots, m_n$

wyznaczamy $NWW = \frac{m_0 m_1 \cdots m_n}{NWD}$, gdzie $m_0, m_1, \ldots, m_n$ liczby naturalne [14,18]. Proponowane nowe rozwiązanie jest bardzo proste i do jego zrealizowania wystarczy wykonać proste operacje odejmowania i mnożenia. Metody stosowane dotychczas [14, 18] są bardziej skomplikowane obliczeniowo od przedstawionej nowej metody.

6. Nowy sposób wyznaczania Najmniejszej Wspólnej Wielokrotności (NWW) liczb naturalnych.

Przedstawimy nowy sposób wyznaczania najmniejszej wspólnej wielokrotności $[m_0, m_1, \ldots, m_n]$ liczb naturalnych $m_0, m_1, \ldots, m_n$.

Niech: $m_0, m_1, \ldots, m_n \in N$; $m_0 \geq m_1$; wtedy:

$d_0 = m_0 - k \cdot m_1$; $d_0 > 0$; $\equiv$ relacja równoważności

k – najmniejsza wielokrotność m_1 w m_0.

Jeśli $d_0 = 0$, to $m_0 = m_1$ i wtedy $[m_0, m_1] = m_0$
W przypadku $d_0 \neq 0$ wyznaczamy $[m_0, m_1]$ następująco :

$d_0 = m_0 - k \cdot m_1$ $\qquad$ $b_0 = m_1 - d_0$
$d_1 = m_0 - b_0 - k \cdot m_1$ $\qquad$ $b_1 = m_1 - d_1$
$d_2 = m_0 - b_1 - k \cdot m_1$ $\qquad$ $b_2 = m_1 - d_2$
. $\qquad$.
. $\qquad$.
. $\qquad$.
$d_{w-2} = m_0 - b_{w-3} - k \cdot m_1$ $\qquad$ $b_{w-2} = m_1 - d_{w-2}$
$d_{w-1} = m_0 - b_{w-2} - k \cdot m_1 = 0$

W przypadku, gdy $d_x < 0$; gdzie: $0 < x < w - 1$, to w miejsce b_x wpisujemy bezwzględną wartość liczby d_x i obliczenia kontynuujemy. Przedstawiony sposób umożliwia obliczenie najmniejszej wspólnej wielokrotności według następującego wzoru: $[m_0, m_1] = m_0 \cdot w$.

Dowód

Załóżmy, że $0 < m_1 < m_0$ są liczbami naturalnymi. Wtedy $m_0 = k \cdot m_1 + d_0$, gdzie k jest liczbą naturalną i $0 \leq d_0 < m_1$.

Gdyby $d_0 = 0$, to $[m_0, m_1] = m_0$.

Załóżmy więc, że $d_0 > 0$

Określmy ciąg liczb całkowitych b_i i d_i w następujący sposób:

$$b_0 = m_1 - d_0 \quad ; \quad d_1 = d_0 - b_0$$

$$b_i = \begin{cases} m_1 - d_i & \text{gdy} \quad 0 < d_i < m_1 \\ -d_i & \text{gdy} \quad d_i < 0 \end{cases}$$

$d_{i+1} = d_0 - b_i$ $(i = 1, 2, 3, \ldots)$

Gdyby dla pewnego i, $d_i = 0$, to kończymy proces wyznaczania d_i

Załóżmy zatem, że $d_i \neq 0$.

I. Pokażemy najpierw, że

$0 < b_i < m_1$ oraz $|d_i| < m_1$ (1)

Oczywiście $0 < b_0 < m_1$, gdyż $0 < d_0 < m_1$

$-m_1 < d_1 = d_0 - b_0 < d_0 < m_1$

$$b_1 = \begin{cases} m_1 - d_1 & \text{gdy} \quad d_1 > 0 \\ -d_1 & \text{gdy} \quad d_1 < 0 \end{cases}$$

zatem $0 < b_1 < m_1$

Załóżmy teraz, że (1) jest prawdziwa dla pewnego i.

Wtedy $-m_1 < d_0 - m_1 < d_{i+1} = d_0 - b_i < d_0 < m_1$

$$b_{i+1} = \begin{cases} m_1 - d_{i+1} & \text{gdy} \quad m_1 > d_{i+1} > 0 \\ -d_{i+1} & \text{gdy} \quad -m_1 < d_{i+1} < 0 \end{cases}$$

Stąd widać, że $0 < b_{i+1} < m_1$.

Zatem potwierdziliśmy, że z prawdziwości (1) dla i wynika prawdziwość (1) dla (i+1).

II. Pokażemy teraz, że

$d_i \equiv (i+1) \cdot d_0 \pmod{m_1}$; $\equiv$ relacja równoważności (2)

$m_0 \equiv d_0 \pmod{m_1}$; $b_0 = m_1 - d_0 \equiv -d_0 \pmod{m_1}$,

wynika z faktu, że $\left.\begin{array}{l} m_1 \equiv 0 \pmod{m_1} \\ -d_0 \equiv -d_0 \pmod{m_1} \end{array}\right\} +$

to $\overline{m_1 - d_0 \equiv -d_0 \pmod{m_1}}$

$d_1 = d_0 - b_0 \equiv d_0 + d_0 \pmod{m_1} = 2\, d_0 \pmod{m_1}$

Zatem (2) jest prawdziwa dla i = 0, i = 1

Załóżmy, że (2) zachodzi dla pewnego i

Wtedy $$b_i = \begin{cases} m_1 - d_i & \text{gdy} \quad d_i > 0 \\ -d_i & \text{gdy} \quad d_i < 0 \end{cases}$$

Zatem $b_i \equiv -d_i \pmod{m_1} \equiv -(i+1) \cdot d_0 \pmod{m_1}$ (z założenia indukcyjnego).

Stąd $d_{i+1} = d_0 - b_i \equiv d_0 + (i+1) \cdot d_0 \pmod{m_1} \equiv (i+2) \cdot d_0 \pmod{m_1}$.

To dowodzi (2) dla (i + 1).

III. Z równości $w \cdot m_0 = w \cdot k \cdot m_1 + w \cdot d_0$ wynika, że $w \cdot m_0 = [m_0, m_1]$ wtedy i tylko wtedy „w" jest najmniejszą liczbą naturalną taką, że $w \cdot d_0 \equiv 0 \pmod{m_1}$.

Niech w będzie najmniejszą liczbą naturalną taką, że $w \cdot d_0 \equiv 0 \pmod{m_1}$.

Wtedy z (2) ; $d_{w-1} \equiv w \cdot d_0 \pmod{m_1} \equiv 0 \pmod{m_1}$.

Ponieważ z (1) wynika, że $|d_{w-1}| < m_1$, więc musi być $d_{w-1} = 0$.

Odwrotnie, jeśli „w" jest najmniejszą liczbą naturalną taką, że $d_{w-1} = 0$, to z powyższej konstrukcji najmniejszej wspólnej wielokrotności wynika, że $w \cdot d_0 \equiv 0 \pmod{m_1}$ i $w \cdot m_0 = [m_0, m_1]$.

W przypadku większej ilości liczb obliczenia wykonujemy sekwencyjnie

$[m_0, m_1] = a_1$
$a_2 = [a_1, m_2]$
$a_3 = [a_2, m_3]$
. $\qquad$.
. $\qquad$.
. $\qquad$.
$a_n = [a_{k-1}, m_n]$

Zatem $a_n = [m_0, m_1, \ldots, m_n]$

C.B.D.O.

7. Programy komputerowe nowego sposobu wyznaczania NWW

7.1. Program w języku Qbasic

```
10      CLS : INPUT "N" ; N
20      DIM M(N)
30      FOR X = 1 TO N
40      PRINT "M ("; X; ")" ; : INPUT M (X-1)
50      NEXT X
60      Z = 2
70      IF M (0) > = M (1) THEN IF M (1) > 0 THEN
        GOTO 110
80      IF M (1) > M (0) THEN A = M (0) : M (0) = M
        (1) : M (1) = A: GOTO 70
90      PRINT "THE numbers M0 and M1 are not the
        natural numbers"
100     STOP
110     K = INT (M (0) / M (1))
120     DO = M (0) – K * M (1)
130     I = 0
140     IF DO = 0 THEN GOTO 200
150     B0 = M (1) – D0
160     I = I + 1
170     D = D0 – B0
180     IF D < 0 THEN B1 = -D: B0 = B1: GOTO 160
190     IF D > 0 THEN B1 = M (1) – D: B0 = B1:
        GOTO 160
200     W = (I + 1) * M(0)
210     IF Z = N THEN PRINT W: END
220     M0 = W: Z = Z + 1
230     M (1) = M(Z) : GOTO 70
```

7.2. Program w języku Pascal

```
Uses CRT;
var
  N, X, Z, K, D0, I, B0, D, W, A : integer;
  M : array [0..20] of integer;
begin
  clrscr;
  write('N ? '); readln(N);
  if N<2 then begin
    writeln('The number ',N,' is too small');
    exit;
    end;
  for X:=1 to N do begin
    write('M (',X,') ? '); readln(M[X-1]);
    if M[X-1]<1 then begin
      writeln('The number ',(X-1),' is not the natural
number');
      exit;
      end;
    end;
  Z := 2;
  while Z<=N do begin
    if M[1]>M[0] then begin A:=M[0]; M[0]:=M[1];
M[1]:=A; end;
  K:=M[0] div M[1];
  D0:=M[0]-K*M[1];
  I:=0;
  if D0<>0 then begin
    B0:=M[1]-D0;
    repeat
      I:=I+1;
      D:=D0-B0;
      if D<0 then B0:=-D
      else if D>0 then B0:=M[1]-D;
      until D=0;
    end;
  W:=(I+1)*M[0];
  M[0]:=W;
  if Z<N then M[1]:=M[Z];
  Z:=Z+1;
  end;
 Writeln(W);
 end.
```

8. WNIOSKI

W rozdziale 2 przedstawiono jak świat matematyki składający się z jądra i wielu warstw zewnętrznych przenika do problemu świata zewnętrznego i na odwrót. Podczas badań nad złożonością półgrup charakterystycznych automatów A i B asynchronicznych zdeterminowanych skończonych uzyskano wyniki z których wynika, że złożoność ta zależy między innymi od wartości NWW liczb stanów tych automatów. Wyniki te nie byłyby możliwe bez znalezienia nowej metody wyznaczania NWW liczb naturalnych. Wynika stąd, że badania w warstwach zewnętrznych matematyki pozwoliły na uzyskanie wyników, które powinny powstać w jądrze matematyki (teoria liczb). Co więcej wynik ten nie byłby możliwy do uzyskania gdyby takich lub podobnych badań nie przeprowadzono. Wynika stąd, że badania zjawisk natury technicznej umożliwiło uzyskanie wyniku w naukach podstawowych – w matematyce teoretycznej.

Nowy sposób wyznaczania NWW liczb naturalnych umożliwia:

1. przeprowadzenie dowodu dla automatów skończonych asynchronicznych zdeterminowanych dla alfabetu dwuwejściowego z którego wynika, że suma prosta i iloczyn prosty półgrup charakterystycznych automatów, które można uważać za realizację odpowiednio sekwencyjnych i równoległych obliczeń są izomorficzne czyli równoważne [3].
2. algorytm ten, który może być stosowany do wszelkiego rodzaju obliczeń i w różnych programach komputerowych jest znacznie prostszy od obecnie stosowanych algorytmów NWW liczb naturalnych, co znacznie zmniejsza złożoność czasową i pamięciową komputerów.

LITERATURA

[1] *Bocian S.: Złożoność półgrupy charakterystycznej automatów asynchronicznych i ich rozszerzeń. Praca Instytutu Podstaw Informatyki Polskiej Akademi Nauk, nr 552,s 1 – 46, Warszawa 1984.*

[2] *Bocian S., Mikolajczak B.: Complexity of a characteristic semigroup of some classes of automata, Comference on Theory and Applications of Semigroups, Greiswald , 12-16.11.1984, s. 11.*

[3] *Bocian S., Mikolajczak B.: Computational aspects of assigning characterstic semigroup of asynchronous automata and their extensions, Theory of algorithms, pod red. L. Lavosz and E. Szemeredi, Colloquia Matematica Societatis Janos Bólyai 44, North - Holland, Amsterdam - Oxford - New York, Budapest Hungary 1985 s. 37 - 47.*

[4] *Bocian S.: Badania nad złożonością obliczeniową półgrupy charakterystycznej automatów skończonych. Praca doktorska, Poznań 1986.*

[5] *Bocian S.: A new method of calculating the smallest common multiple, Computational Topology and Geometry and Computation in Teaching Mathematics, pod red. Eladio Dominquez Murillo, Antonio Quintero Toscano, Jose Luis Vincente Cordoba, Universidad de Sevilla 1987 s.25-41.*

[6] *Bocian S.: The complexity of semigroup characteizarion of asynchronous strongly connected automation and their extensions, Computational Topology and Geometry and Computation Teaching Mathematics, pod red. Eladio Dominquez Murillo, Antonio Quintero Toscano, Jose Luis Vincente Cordoba, Universidad de Sevilla 1987 s. 41 - 49.*

[7] *Davis P. J., Hersh R.: Świat matematyki. PWN, Warszawa 1994.*

[8] *Hogben L.: The Wonderful World of Mathematics. Garden City, N.Y., 1955, Garden City Books.*

[9] *Hopcroft J.E., Ullman J.D.: Wprowadzenie do teorii automatów języków i obliczeń. PWN, Warszawa,1994*

[10] *Kostrykin A. I. : Wstęp do algebry. PWN, Warszawa 1984.*

[11] *Leja F.: Rachunek różniczkowy i całkowy PWN, Warszawa 1967.*

[12] *Mikołajczak B. Red.: Algebraiczna i strukturalna teoria automatów. PWN, Warszawa –Łódź, 1985.*

[13] *Mikołajczak B., Stokłosa J. : Złożoność obliczeniowa algorytmów. Wydawnictwo Politechniki Poznańskiej, Poznań 1984.*

[14] *Mostowski A., Stark M.: Elementy algebry wyższej. PWN, Warszawa 1968.*

[15] *Musielak J.: Wstęp do matematyki. PWN, Warszawa 1970.*

[16] *Opial Z.: Algebra wyższa. PWN, Warszawa 1967.*

[17] *Sacha K., Rydzewski A.: Mikroprocesory w pytaniach i odpowiedziach. PWN, Warszawa 1968.*

[18] *Sierpiński W.: Arytmetyka teoretyczna. PWN, Warszawa 1968.*

[19] *Stern L. A.: Matematyka współczesna – 12 esejów. WNT, Warszawa 1983.*