ROCZNIKI POLSKIEGO TOWARZYSTWA MATEMATYCZNEGO
Seria II: WIADOMOŚCI MATEMATYCZNE X (1968)

Ministerstwo Nauki i Szkolnictwa Wyższego

A. Blikle (Warszawa)

O lingwistyce matematycznej

Współczesny matematyk zajmujący się teorią organizacji maszyn matematycznych, teorią programowania i pokrewnymi dziedzinami, spotyka się coraz częściej z terminem *lingwistyka matematyczna*. Niestety, termin ten bywa rozumiany opacznie jako dział językoznawstwa, a nie matematyki. Niniejszy artykuł stawia sobie za zadanie wyjaśnienie, że pogląd ten jest mylny. Postaramy się przedstawić tu zasadnicze kierunki badań oraz zasadnicze metody lingwistyki matematycznej. Ze względu na szczupłość miejsca ograniczymy się jedynie do twierdzeń i przykładów, rezygnując z dowodów. Zainteresowany czytelnik będzie je mógł odnaleźć w cytowanej literaturze.

Można bez przesady powiedzieć, że lingwistyka matematyczna jest jednym z najmłodszych działów matematyki. Pierwsze prace o większym znaczeniu, formujące podstawy modeli stosowanych obecnie w lingwistyce matematycznej ukazały się w latach 1956-1959. Wymienić tu należy takie pozycje jak Chomsky [3], [4], [5] oraz Rabin i Scott [22]. Inspiracji do ukazania się tych i następnych prac dostarczyły dwa następujące źródła: potrzeba wprowadzenia metod matematycznych do językoznawstwa oraz potrzeba badania, opisywania i klasyfikowania powstających w ogromnej ilości języków sztucznych, tzw. algorytmicznych, służących do programowania maszyn matematycznych.

Jeżeli chodzi o językoznawstwo, to okazało się, że wprowadzenie modelu matematycznego wskazane jest aż z dwóch względów: Po pierwsze, językoznawstwo osiągnęło ten stopień dojrzałości, na którym wszelki dalszy postęp bez wprowadzenia metod formalnych (matematycznych) jest poważnie utrudniony. Po drugie, rozwój techniki automatycznego przetwarzania danych za pomocą maszyn matematycznych stworzył potencjalne możliwości dokonywania automatycznych przekładów z języka na język, co nie może być jednak zrealizowane bez znajomości takich gramatyk języków naturalnych, których forma byłaby zrozumiała dla maszyn matematycznych. (Przez język naturalny rozumiemy język używany w mowie potocznej, taki jak polski, angielski itp., w przeciwień-

stwie do języków sztucznych, takich jak ALGOL, FORTRAN, formalizm nawiasowy etc.) Obecnie znane gramatyki opisowe nie spełniają oczywiście tego warunku.

Jeżeli chodzi o języki programowe, to w początkowym okresie swojego rozwoju były to tzw. języki wewnętrzne, charakteryzujące się bardzo prostą syntaktyką (składnią). Języki te były jednak mało czytelne i programowanie za ich pomocą stawało się niezmiernie uciążliwe, szczególnie przy bardziej złożonych zagadnieniach. W wyniku tej sytuacji powstawać poczęły (w latach 1958-1960) tzw. języki algorytmiczne, zwane również językami automatycznego programowania. Są one znacznie czytelniejsze dla programisty, mają jednak złożoną syntaktykę wymagającą wyraźnego określenia, które ze zdań tego języka uznajemy za poprawne, a które nie. Ponadto języki te, jako niezrozumiałe dla maszyny, wymagają dokonywania translacji na język wewnętrzny. Stwarza to konieczność posiadania jasnej i matematycznie precyzyjnej gramatyki takich języków. Na podkreślenie zasługuje fakt, że liczba języków algorytmicznych znanych i używanych dzisiaj na całym świecie sięga kilkudziesięciu. Istnieje więc rzeczywista potrzeba znalezienia jednolitej metody ich opisu, pozwalającej na porównywanie różnych języków, ich klasyfikację, optymalizację itp.

Wymienione problemy wskazują na konieczność zbudowania matematycznego modelu języka. Zadanie to, podjęte przed jedenastu laty, doprowadziło do powstania nowej, intensywnie rozwijającej się obecnie dziedziny matematyki, nazywanej przez jednych *lingwistyką matematyczną*, przez drugich zaś — na przykład we Francji — *teorią języków formalnych*. Zanim przystąpimy do bardziej szczegółowego omówienia jej poszczególnych aspektów, wprowadzimy kilka zasadniczych pojęć niezbędnych w dalszych rozważaniach.

Niech $V = \{v_1, \ldots, v_n\}$ będzie dowolnym skończonym zbiorem symboli. Zbiór ten nazwiemy *alfabetem*, a jego elementy *literami*. Ciągi skończone elementów zbioru V nazwiemy *słowami nad alfabetem* V. Będziemy je zapisywać bez przecinków pomiędzy pojedynczymi symbolami, a więc np. $v_1v_2v_5v_5$ zamiast v_1, v_2, v_5, v_5. Zbiór wszystkich słów nad alfabetem V oznaczymy przez V^*. Litery traktujemy jako słowa jednoelementowe, a więc $V \subseteq V^*$.

Należy podkreślić, że alfabet V stanowi w przypadku rozpatrywania języków naturalnych model nie alfabetu, ale słownika danego języka. Słowa nad alfabetem V odpowiadają więc zdaniom. Podobnie rzecz ma się przy budowaniu modelu dla języków algorytmicznych. Za elementy alfabetu uważamy wtedy wszystkie wyrażenia nierozkładalne w tym języku. Dla ALGOL'u na przykład elementami alfabetu będą między innymi wyrażenia begin, end, array, step, sin, sign, +, 7, itp.

W zbiorze V^* wprowadzamy dwuargumentową operację „$\circ$" zwaną

składaniem (używa się również terminu *konkatenacja*), określoną w sposób następujący:

$$v_{j_1} \dots v_{j_k} \circ v_{i_1} \dots v_{i_p} = v_{j_1} \dots v_{j_k} v_1 \dots v_{i_p}.$$

Widzimy więc, że złożenie dwóch słów należących do V^* jest dalej słowem należącym do V^* i jeżeli $x, y, z \in V^*$, to ma miejsce równość $x \circ (y \circ z) = (x \circ y) \circ z$, tzn. działanie składania jest łączne. Zbiór V^* tworzy więc z operacją „$\circ$" półgrupę, którą nazywa się zwykle *półgrupą wolną*. Przyjmujemy dodatkowo abstrakcyjne założenie, że do zbioru V^* należy ciąg pusty, który oznaczymy przez ε i który dla każdego $x \in V^*$ spełnia następujące dwa równania:

$$x \circ \varepsilon = x, \qquad \varepsilon \circ x = x.$$

Ciąg ten nazwiemy *słowem pustym*. Stanowi on jedność w naszej półgrupie.

W zbiorze V^* określimy ponadto funkcję $|x|$, o wartościach całkowitych nieujemnych, zwaną *długością słowa* i określającą liczbę występujących w nim symboli alfabetu. Na przykład $|v_1 v_5 v_3| = 3$, $|\varepsilon| = 0$.

Znak działania składania będziemy zwykle opuszczać, pisząc xy zamiast $x \circ y$.

Każdy podzbiór skończony lub nieskończony L zbioru V^* nazywać będziemy *językiem nad alfabetem* V. Jeżeli dla danego języka $L \subseteq V^*$ istnieje przepis pozwalający dla każdego słowa $x \in V^*$ rozstrzygnąć w skończonej liczbie prostych (efektywnie wykonywalnych) kroków, czy x należy do L, czy nie, to przepis ten nazywamy *gramatyką języka* L, a język L nazywamy *gramatykalizowalnym*. Wydaje się, że takie określenie gramatyki jest zgodne z zapotrzebowaniem, jakie praktyka stawia przed lingwistyką matematyczną, bowiem rzeczywiście każda dobra gramatyka języka musi mieć tę własność, by przy jej znajomości orzekać można było o każdym zdaniu zbudowanym ze słów rozważanego słownika, czy jest to zdanie poprawne, czy nie.

W logice matematycznej istnieje precyzyjne pojęcie odpowiadające temu, co nazwaliśmy powyżej „... przepisem pozwalającym dla każdego $x \in V^*$ rozstrzygać w skończonej liczbie prostych (efektywnie wykonalnych) kroków, czy x należy do L, czy nie". Jest to pojęcie *algorytmu*, znane pod wieloma postaciami (algorytmy Markowa, algorytmy Posta, funkcje obliczalne, maszyny Turinga itp.), którymi nie będziemy się tu jednak zajmować ze względu na ich złożoność. Zainteresowanego czytelnika odsyłamy do bogatej literatury na ten temat: Davis [3], Grzegorczyk [14], Kleene [15], Malcev [18].

Jak się okazuje, pojęcie algorytmu jako gramatyki jest w swej ogólnej postaci pojęciem zbyt obszernym dla lingwistyki matematycznej. Zawęża się je zatem przez nałożenie szeregu ograniczeń, otrzymując

w ten sposób pewną hierarchię gramatyk o różnych stopniach złożoności.

Najlepiej zbadanymi obecnie klasami gramatyk są automaty abstrakcyjne, zwane również *akceptorami*, oraz *gramatyki kombinatoryczne*. Obie te klasy omówimy w niniejszym artykule. Zarówno w jednej jak i w drugiej klasie mamy szereg typów różniących się złożonością. Wśród automatów abstrakcyjnych wyróżniamy — w porządku według rosnącej złożoności — automaty skończone, automaty ze stosem (push-down) i automaty liniowo ograniczone; natomiast wśród gramatyk kombinatorycznych rozważa się między innymi — też według rosnącej złożoności — gramatyki prawostronnie liniowe, gramatyki bezkontekstowe i gramatyki kontekstowe. Oba te ujęcia, aczkolwiek powstałe niezależnie od siebie, pozostają w tak ścisłym związku, że mówić możemy o dualności pojęć. Na przykład, pojęciem dualnym dla automatu skończonego jest gramatyka prawostronnie liniowa, tzn. każdy język opisywany pewnym automatem skończonym opisywany jest również pewną gramatyką prawostronnie liniową i na odwrót. Podobnie, pojęciem dualnym dla automatu ze stosem jest gramatyka bezkontekstowa, a pojęciem dualnym dla automatu liniowo ograniczonego jest gramatyka kontekstowa.

Skoro istnieje tak ścisła współzależność pomiędzy automatami abstrakcyjnymi i gramatykami kombinatorycznymi, można postawić pytanie, czy warto rozwijać te oba ujęcia jednocześnie — dotyczą one bowiem tych samych faktów. Okazuje się jednak, że jednoczesne badanie automatów i gramatyk jest celowe, ujęcia te bowiem uzupełniają się w sposób następujący: Automaty stanowią model dosyć intuicyjny, w którym wiele faktów dotyczących języków daje się łatwo zauważyć lub przewidzieć, a ponadto twierdzenia wypowiedziane dla automatów interpretują się w sposób jasny dla maszyn matematycznych. Zasadniczą wadą tego ujęcia jest jednak złożoność formalizmu, prowadząca do mało przejrzystej i matematycznie nieeleganckiej teorii. Gramatyki kombinatoryczne mają formalizm znacznie czytelniejszy i matematycznie elegantszy, a przy tym bliższy modelom języków naturalnych; jednakże wiele własności języków, które widać na pierwszy rzut oka w ujęciu „automatowym", nie jest widocznych w teorii gramatyk kombinatorycznych. Ponadto gramatyki te mają znacznie słabszą interpretację w terminach maszyn matematycznych.

Zanim przystąpimy do omówienia kolejnych typów gramatyk, chcielibyśmy podkreślić, że w rozwijanej obecnie lingwistyce matematycznej wyodrębniają się dwie różne szkoły. Pierwsza z nich rozwijana głównie w Stanach Zjednoczonych, Izraelu, Francji i Czechosłowacji (Y. Bar-Hillel, N. Chomsky, K. Čulik, S. Ginsburg, M.P. Schützenberger i inni) przyjmuje za podstawę swoich modeli gramatyki kombinatoryczne i automaty abstrakcyjne. Reprezentuje ona kierunek zwany *syntetycznym*

(synteza języka przez gramatykę lub automat). Druga szkoła, która powstała i rozwija się głównie w Związku Radzieckim i Rumunii (A. Gładki, O. S. Kulagina, S. Marcus i inni) prowadzi rozważania głównie nad językami naturalnymi, używając do tego celu pewnych elementów algebry i teorii mnogości [13]. To ujęcie sami jego twórcy nazywają *analitycznym*.

W niniejszym artykule omówimy jedynie kierunek pierwszy, a to z tego względu, iż autor nie czuje się kompetentny do omówienia drugiego, jednak i pierwszy kierunek nie zostanie omówiony w sposób wyczerpujący. W części dotyczącej automatów pominiemy na przykład translatory, tj. automaty z wyjściem, których zadaniem jest odwzorowywanie języka na język, a w części drugiej, w paragrafie dotyczącym gramatyk bezkontekstowych nie wspomnimy, z braku miejsca, o problemie jednoznaczności.

W intencji autora leżało oddanie do rąk polskiego matematyka czegoś w rodzaju informatora o lingwistyce matematycznej, z którego można by zorientować się w zakresie jej wyników i potrzeb. Wynikom poświęcimy najbliższe dwa rozdziały; pomówmy zatem o potrzebach. Na pierwszy plan wysuwa się konieczność sformułowania i udowodnienia twierdzeń wiążących w jakiś sposób pojęcia lingwistyki matematycznej z pojęciami innych działów matematyki, w szczególności algebry, teorii mnogości i teorii funkcji rekurencyjnych. Pewne twierdzenia tego typu znane są jedynie dla klasy języków regularnych (tw. 2 Neroda i tw. 7 Kleene-Myhilla). Znalezienie analogicznych twierdzeń dla klas języków kontekstowych i bezkontekstowych pozwoliłoby najprawdopodobniej na uproszczenie wielu dowodów i na odkrycie nowych faktów. Następnym zagadnieniem czekającym jeszcze na rozwiązanie jest znalezienie dostatecznie adekwatnych modeli dla języków naturalnych, okazuje się bowiem, że zarówno gramatyki kombinatoryczne, jak i automaty, modeli takich nie stanowią. Pewne sugestie dotyczące kierunku badań nad tym zagadnieniem można znaleźć w pracy [2].

Oba wymienione problemy są zagadnieniami bardzo ogólnymi, odnoszącymi się do całości zagadnień lingwistyki matematycznej. Istnieje jednak szereg problemów szczegółowych, które również czekają na swoje rozwiązanie. Znaleźć je można we wszystkich prawie publikacjach dotyczących lingwistyki matematycznej. Szczególnie bogatą ich listę zawiera książka S. Ginsburga [11].

1. Automaty. Pojęcie automatu abstrakcyjnego, jakkolwiek używane dzisiaj z powodzeniem w lingwistyce matematycznej, powstało na gruncie potrzeb dosyć odległych od tego kierunku, a przy tym znacznie wcześniej niż inne jego pojęcia, takie jak np. gramatyka kombinatoryczna. Zacznijmy zatem od krótkiego wstępu historycznego.

W roku 1936 A. M. Turing [23] opisał pewną koncepcję maszyny abstrakcyjnej, zwanej dzisiaj *maszyną Turinga*. Była to pierwsza koncepcja automatu abstrakcyjnego i jakkolwiek jest ona bliska współczesnym abstrakcyjnym modelom maszyn matematycznych, to jednak w intencji jej autora leżało przede wszystkim znalezienie nowego podejścia do analizy pojęcia efektywności w matematyce. Dlatego też maszyny Turinga pomyślane zostały dostatecznie ogólnie na to, aby mogły opisywać funkcje obliczalne. Ogólność ta, jak się później okazało, stanowi jednak wątpliwą zaletę w momencie, gdy poszukujemy dostatecznie adekwatnego i prostego modelu maszyny cyfrowej.

Powstała zatem potrzeba określenia słabszej klasy automatów abstrakcyjnych. Jedną z prób zadośćuczynienia tej potrzebie była praca Rabina i Scotta [22], na której oprzemy się w najbliższym paragrafie. Od tego czasu powstały dalsze koncepcje automatów abstrakcyjnych; wszystkie one mieszczą się jednak pomiędzy koncepcjami Turinga oraz Rabina i Scotta w tym sensie, że każda z nich stanowi pewien szczególny przypadek maszyny Turinga — lub równoważną jej wersję, a automat Rabina-Scotta jest z tych wszystkich przypadków najprostszy.

Mówiąc o automatach skończonych, wspomnieć wypada o pierwszej pracy, jaka ukazała się na ten temat. Była to praca matematyka amerykańskiego S. C. Kleene [15]. Przedstawia ona jednak bardzo specjalne ujęcie i dlatego ma dla nas znaczenie raczej drugorzędne.

1.1. Automaty skończone Rabina-Scotta. Automaty Rabina-Scotta nazywać będziemy w niniejszej pracy RS-*automatami*. Poglądowo, RS-automat wyobrażamy sobie w postaci urządzenia, które może przyjmować pewną skończoną liczbę stanów wewnętrznych i które zaopatrzone jest w skończoną liczbę klawiszy odpowiadających symbolom pewnego alfabetu V. Wśród stanów wyróżniamy jeden stan zwany *początkowym* oraz podzbiór stanów zwanych *końcowymi*. Zadaniem automatu jest czytanie słów nad alfabetem V i wydawanie o nich „opinii". Czytanie to odbywa się przez wyciskanie na klawiaturze automatu klawiszy, odpowiadających kolejnym symbolom danego słowa. Przed przystąpieniem do czytania automat ustawia się w stanie początkowym. W trakcie czytania automat zmienia swoje stany, przy czym każdy kolejny stan zależy od stanu poprzedniego i od aktualnie wyciśniętego klawisza. Mówimy, że *automat zaakceptował dane słowo*, gdy po jego przeczytaniu znalazł się w jednym ze stanów końcowych. Zbiór wszystkich słów akceptowanych przez dany automat nazywamy *językiem definiowanym przez ten automat*.

Zajmiemy się teraz klasą wszystkich języków definiowanych przez RS-automaty. Wymaga to oczywiście wprowadzenia pewnego aparatu matematycznego.

DEFINICJA 1. *Deterministycznym RS-automatem* nazywamy system $\mathfrak{A} = \langle V, S, M, s_0, F\rangle$, gdzie:

$V = \{v_1, \ldots, v_n\}$ jest skończonym zbiorem symboli, zwanym *alfabetem*,

$S = \{s_0, \ldots, s_m\}$ jest skończonym zbiorem elementów zwanych *stanami automatu*,

$M\colon S \times V \to S$ jest funkcją dwuargumentową określoną na iloczynie kartezjańskim zbiorów S i V i o wartościach w zbiorze S,

s_0 jest wyróżnionym stanem ($s_0 \in S$) zwanym *stanem początkowym*,

F jest wyróżnionym podzbiorem zbioru stanów ($F \subseteq S$) zwanym *zbiorem stanów końcowych*.

W intuicyjnej interpretacji funkcja $M(s_j, v_i)$ określa nam stan, w jakim znajdzie się automat po wyciśnięciu klawisza v_i w stanie s_j. Funkcję tę, którą nazywamy *funkcją przejścia*, rozszerzamy zwykle do funkcji M^* określonej na $S \times V^*$, definiując indukcyjnie:

$M^*(s_j, \varepsilon) = s_j$ dla każdego $s_j \in S$,

$M^*(s_j, v_i y) = M^*\big(M(s_j, v_i), y\big)$ dla $s_j \in S$, $v_i \in V$, $y \in V^*$.

DEFINICJA 2. *Językiem definiowanym* (lub *rozpoznawanym*) *przez deterministyczny RS-automat* $\mathfrak{A}$, lub *językiem automatu* $\mathfrak{A}$, nazywamy zbiór tych wszystkich słów x nad alfabetem V, dla których $M^*(s_0, x) \in F$. Zbiór ten oznaczamy przez $L(\mathfrak{A})$.

Wprowadzimy teraz pewne założenie formalne, które pozwoli nam na poprawne zdefiniowanie klasy wszystkich języków rozpoznawalnych za pomocą RS-automatów. Założymy mianowicie, że wszystkie rozważane alfabety traktować będziemy (w tym i w następnych paragrafach) jako podzbiory pewnego ustalonego przeliczalnego alfabetu $U = \{u_1, u_2, \ldots\}$, zwanego *alfabetem uniwersalnym*.

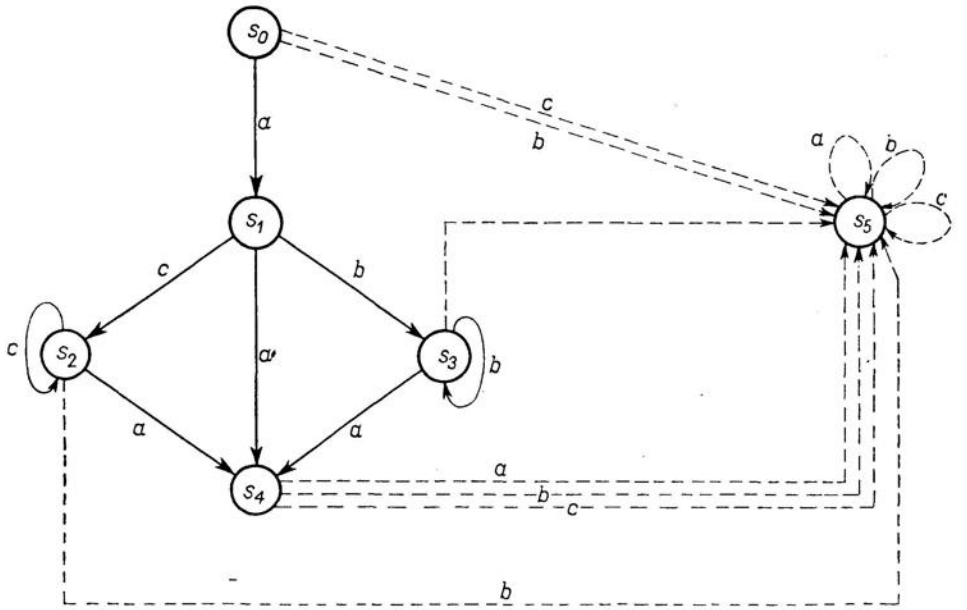
DEFINICJA 3. Język $L \subseteq V^*$, gdzie $V \subseteq U$, nazwiemy *językiem regularnym*, gdy istnieje taki RS-automat $\mathfrak{A}$, że $L = L(\mathfrak{A})$. Zbiór wszystkich języków regularnych oznaczymy przez $\mathscr{L}_R$.

Rozważmy teraz dla przykładu pewien konkretny RS-automat. Niech $V = \{a, b, c\}$, $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$, $F = \{s_4\}$. Funkcję $M(s_i, v_j)$ opiszemy za pomocą następującej tabelki:

	s_0	s_1	s_2	s_3	s_4	s_5
a	s_1	s_4	s_4	s_4	s_5	s_5
b	s_5	s_3	s_5	s_3	s_5	s_5
c	s_5	s_2	s_2	s_5	s_5	s_5

Można sprawdzić, że tak określony automat definiuje język składający się ze wszystkich słów postaci aa, $acc \ldots ca$ oraz $abb \ldots ba$. Sprawdzenie to jest jednak dosyć żmudne nawet w tak prostym przypadku. Aby

opis automatu uczynić czytelniejszym, posługujemy się zwykle tzw. *diagramem Meley* (czyt. Mili). Diagram Meley jest to pewien graf, którego wierzchołki odpowiadają stanom automatu, pomiędzy zaś wierzchołkami biegną strzałki oznaczone symbolami alfabetu. Przyjmuje się zasadę, że od wierzchołka „s_i" biegnie strzałka „v_k" do wierzchołka „s_j" wtedy i tylko wtedy, gdy $M(s_i, v_k) = s_j$. W ten sposób funkcja $M(s_i, v_j)$ opisana jest jednoznacznie. Diagram Meley odpowiadający naszemu automatowi przedstawiony jest na rysunku 1. Część strzałek oznaczona została liniami przerywanymi. Są to strzałki wiodące do tych stanów (w tym przypadku do stanu s_5), z których nie ma powrotu do stanów końcowych (w tym przypadku stanu s_4). Strzałki te opisują więc niejako „pasywną" część automatu. Zwykle nie będziemy ich w ogóle rysowali na diagramie



Rys. 1

Parę $(s_i, v_j) \in S \times V$ nazywać będziemy *konfiguracją automatu*. Jak wynika z definicji 1 dla każdej konfiguracji (s_i, v_j) wyznaczony jest jednoznacznie pewien stan s_k, który interpretujemy jako stan następny. Właśnie ze względu na tę jednoznaczność, automaty określone w definicji 1 nazywamy *deterministycznymi*. Jeżeli odrzucimy założenie o jednoznaczności i przyjmiemy, że każda konfiguracja wyznacza zamiast jednego stanu pewien podzbiór zbioru stanów, z którego możemy wybrać w dowolny sposób stan następny, to otrzymamy nową klasę RS-automatów, zwanych automatami *wolnymi* lub *niedeterministycznymi*. Obrazowo można powiedzieć, że przy automacie deterministycznym rola opera-

tora (obsługującego) ograniczała się jedynie do naciskania odpowiednich klawiszy, w automacie wolnym natomiast po naciśnięciu kolejnego klawisza operator otrzymuje do dyspozycji pewien podzbiór zbioru stanów, z którego sam musi wybrać stan następny. Mówimy, że automat wolny akceptuje dane słowo x, gdy w trakcie czytania tego słowa daje się tak dobierać kolejne stany, aby wychodząc od stanu początkowego osiągnąć stan końcowy. Automat wolny można więc traktować jako urządzenie, w pewnym sensie, programowane.

DEFINICJA 4. *Niedeterministycznym RS-automatem* nazywamy system $\mathfrak{A} = \langle V, S, M, S_0, F\rangle$, gdzie V, S, F określone są jak w definicji 1, natomiast $M \subseteq S \times V \times S$, czyli M jest pewną relacją trójargumentową, oraz $S_0 \subseteq S$ jest zbiorem stanów początkowych. O relacji M zakładamy dodatkowo, że dla każdego $s_i \in S$ oraz każdego $v_j \in V$ istnieje stan $s_k \in S$ taki, że $M(s_i, v_j, s_k)$.

Podzbiór zbioru stanów wyznaczonych przez daną konfigurację (s_i, v_j) składa się z tych wszystkich stanów s_k, dla których zachodzi $M(s_i, v_j, s_k)$. Podobnie jak to czyniliśmy dla automatów deterministycznych, relację M rozszerzamy do relacji $M^* \subseteq S \times V^* \times S$ w sposób następujący: dla każdego $x \in V^*$, $x = v_1 v_2 \ldots v_k$ oraz dla każdych $s, s' \in S$ relacja $M(s, x, s')$ zachodzi wtedy i tylko wtedy, gdy istnieje ciąg stanów $s_1, s_2, \ldots, s_{k+1}$ spełniający następujące dwa warunki:

1. $s_1 = s$ oraz $s_{k+1} = s'$,
2. $M(s_p, v_p, s_{p+1})$ zachodzi dla $p = 1, 2, \ldots, k$.

Ciąg $s_1, \ldots, s_{k+1}$ nazywamy *drogą od s do s' w automacie* $\mathfrak{A}$. Słowo x nazywamy *etykietą* tej drogi.

DEFINICJA 5. *Językiem definiowanym* (lub *rozpoznawanym*) *przez niedeterministyczny RS-automat* $\mathfrak{A}$ lub po prostu *językiem tego automatu* nazywamy zbiór tych wszystkich słów x nad alfabetem V, dla których istnieją stany $s \in S_0$ oraz $s' \in F$ takie, że $M^*(s, x, s')$. Zbiór ten oznaczać będziemy przez $L(\mathfrak{A})$.

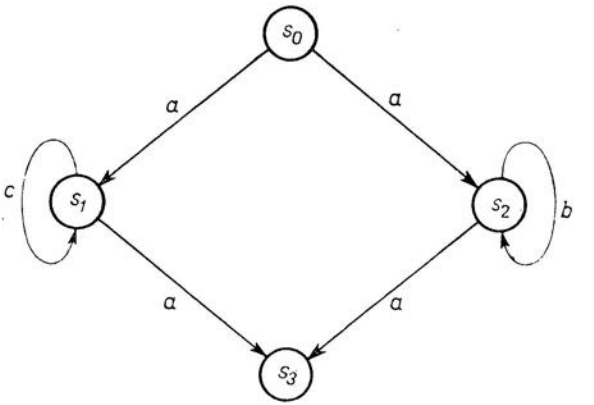
Pojęcie automatu deterministycznego stanowi szczególny przypadek pojęcia automatu niedeterministycznego, mający miejsce gdy dla każdej konfiguracji (s, v) istnieje dokładnie jeden stan s' taki, że $M(s, v, s')$. Z tego względu przez RS-automat rozumieć będziemy od tej pory dowolny RS-automat niedeterministyczny.

DEFINICJA 6. Dwa RS-automaty $\mathfrak{A}_1$ i $\mathfrak{A}_2$ są *równoważne*, jeżeli rozpoznają ten sam język; fakt ten zapisywać będziemy wyrażeniem $\mathfrak{A}_1 \sim \mathfrak{A}_2$.

Ten sam język może być rozpoznawany przez wiele różnych automatów. Rozpatrzmy dla przykładu automat niedeterministyczny, który przedstawimy od razu w postaci diagramu Meley (rys. 2). Niedeterministyczność tego automatu polega na istnieniu dwóch strzałek wychodzących z tego samego wierzchołka (wierzchołka s_0) i oznaczonych tym

samym symbolem. Jak łatwo zauważyć, automat ten jest równoważny automatowi deterministycznemu przedstawionemu na rysunku 1. Ogólnie, jest prawdziwe następujące twierdzenie:

TWIERDZENIE 1. *Dla każdego RS-automatu istnieje równoważny mu automat deterministyczny.*



Rys. 2

Automat deterministyczny, o którym mowa w twierdzeniu 1 można zbudować w sposób efektywny: Niech $\mathfrak{A} = \langle V, S, M, S_0, F\rangle$ będzie dowolnym RS-automatem. Równoważny mu automat deterministyczny $D(\mathfrak{A})$ określamy jako

$$D(\mathfrak{A}) = \langle V, T, N, t_0, G\rangle,$$

gdzie:

$T = 2^S$ jest rodziną wszystkich podzbiorów zbioru S,

$N: T \times V \to T$ jest funkcją określoną w sposób następujący: $N(t,v) = \bigcup_{s \in t} M(s,v)$, a więc dla każdego $t \in T$ i $v \in V$, $N(t,v)$ jest sumą zbiorów $M(s,v) = \{s' : M(s,v,s')\}$,

$t_0 = S_0$ oraz

$G = \{S' : S' \subseteq S \text{ i } S' \cap F \neq \emptyset\}$ czyli G jest rodziną tych wszystkich podzbiorów zbioru S, które mają elementy wspólne ze zbiorem F.

Automat $D(\mathfrak{A})$ ma więc znacznie większą liczbę stanów niż automat $\mathfrak{A}$. Jakkolwiek liczbę stanów automatu $D(\mathfrak{A})$ można zwykle poważnie zredukować, to jednak automat niedeterministyczny daje się na ogół zrealizować za pomocą mniejszej liczby stanów niż równoważny mu automat deterministyczny.

Zastanówmy się teraz ponownie nad klasą $\mathscr{L}_R$. Jedyne, jak na razie, kryterium przynależności języków do tej klasy daje nam definicja 3, która niewiele mówi o matematycznej strukturze tej klasy i jest ponadto

trudna do sprawdzenia. Aby scharakteryzować języki regularne w sposób algebraiczny wprowadzimy pewne pojęcia pomocnicze.

Dwuargumentową relację R określoną w V^* nazwiemy *relacją równoważności*, gdy spełnione są trzy następujące warunki:

1. $x R x$ dla każdego $x \in V^*$,
2. jeżeli $x R y$ to $y R x$ dla każdych $x, y \in V^*$,
3. jeżeli $x R y$ oraz $y R z$ to $x R z$ dla każdych $x, y, z \in V^*$.

Dla każdego $x \in V^*$, *klasą abstrakcji relacji R o reprezentancie x* (którą oznaczymy $[x]_R$) nazwiemy zbiór tych wszystkich y należących do V^*, dla których zachodzi $x R y$. Jak wynika z tego określenia, każdy element danej klasy abstrakcji może być uznany za jej reprezentanta oraz każdy element zbioru V^* jest reprezentantem pewnej klasy abstrakcji.

Indeksem w V^* relacji równoważności R nazwiemy liczbę klas abstrakcji R w V^*.

Powiemy, że relacja R w V^* jest *prawostronnie niezmiennicza*, gdy dla wszystkich $x, y, z \in V^*$ z warunku $x R y$ wynika warunek $xz R yz$.

Dla każdego języka $L \subseteq V^*$ określimy pewną relację dwuargumentową E_L w V^* w sposób następujący: dla każdych $x, y \in V^*$ relacja $x E_L y$ zachodzi wtedy i tylko wtedy, gdy dla każdego $z \in V^*$ warunek $xz \in L$ jest równoważny temu, że $yz \in L$.

Można udowodnić, że dla każdego $L \subseteq V^*$ E_L jest prawostronnie niezmienniczą relacją równoważności.

TWIERDZENIE 2 (NERODE). *Niech L będzie językiem nad alfabetem V. Wówczas następujące trzy warunki są równoważne*:

(i) *L jest językiem regularnym, czyli $L \in \mathscr{L}_R$;*

(ii) *istnieje prawostronnie niezmiennicza relacja równoważności R w V^* taka, że L jest sumą pewnej skończonej liczby klas abstrakcji relacji R w V^*;*

(iii) *relacja równoważności E_L ma w V^* skończony indeks.*

Największe znaczenie ma dla nas warunek (iii) tego twierdzenia, pozwala bowiem opisać przynależność języka L do $\mathscr{L}_R$ w sposób czysto algebraiczny. Okazuje się przy tym, że relacja E_L ma bardzo wyraźną interpretację teorio-automatową. Dla wyjaśnienia jej rozważmy dowolny deterministyczny RS-automat $\mathfrak{A}$ oraz relację E_L określoną dla języka $L(\mathfrak{A})$. Przypuśćmy teraz, że automat nasz czyta słowo xz należące do V^* i przeczytał już x. Strukturę przeczytanego słowa automat musi „zapamiętać" przez przyjęcie stanu $s = M^*(s_0, x)$, bowiem przyjmowanie odpowiedniego stanu jest w automacie skończonym jedyną możliwością zapamiętywania czegokolwiek. Od tej pory jedynie stan s reprezentuje słowo x w automacie i wespół ze słowem z decyduje, czy xz będzie, czy nie będzie zaakceptowane. Zatem, gdyby automat zamiast słowa x przeczytał słowo y takie, że $x E_L y$, to przyjąć powinien ten sam stan co dla

x, bowiem czy xz i yz należą do L zależy tylko od z i od tego, do jakiej klasy abstrakcji należą x i y. Z drugiej strony, jeżeli non $x E_L y$, to istnieje takie z, że $xz \in L$ i $yz \notin L$ lub na odwrót, równość $M^*(s_0, x) = M^*(s_0, y)$ nie może zatem mieć miejsca, gdyż w przeciwnym przypadku:

$$M^*(s_0, yz) = M^*\big(M^*(s_0, y), z\big) = M^*\big(M^*(s_0, x), z\big) = M^*(s_0, xz),$$

a więc bądź xz, $yz \in L$, bądź xz, $yz \notin L$.

Jak stąd wynika, automat w trakcie czytania danego słowa $v_1 \dots v_k$ powinien po przeczytaniu każdego kolejnego symbolu v_i $(1 \leqslant i \leqslant k)$ zapamiętać jedynie, do jakiej klasy należy słowo $v_1 \dots v_i$. W automacie $\mathfrak{A}$ musi więc być dokładnie tyle stanów, ile jest klas abstrakcji relacji E_L w V^* i musi istnieć pomiędzy nimi odpowiedniość wzajemnie jednoznaczna. Jak się ponadto okazuje, jeżeli dla jakiegoś języka L, dla którego E_L ma w V^* indeks skończony, znane są wszystkie klasy abstrakcji E_L, to potrafimy już zbudować automat rozpoznający język L. Co więcej, będzie to automat najekonomiczniejszy ze względu na ilość stanów, co wynika z następującego twierdzenia:

TWIERDZENIE 3. *Jeżeli dla danego języka L relacja E_L ma w V^* indeks skończony, to jest on równy najmniejszej liczbie stanów z jaką da się zrealizować automat rozpoznający język L.*

Konstrukcja automatu poprzez relację E_L przedstawiona jest w dowodzie twierdzenia Neroda w pracy Rabina i Scotta [22].

Zajmiemy się teraz pewnymi jedno- i dwuargumentowymi operacjami na językach zachowującymi regularność. Pozwoli to na podanie pełnej charakterystyki teoriomnogościowej klasy $\mathscr{L}_R$.

Przez *sumę dwóch języków* rozumieć będziemy ich sumę teoriomnogościową.

Złożeniem dwóch języków L_1 i L_2 nazwiemy język $L_1 \circ L_2$ określony w sposób następujący:

$$L_1 \circ L_2 = \{xy\colon\ x \in L_1 \text{ i } y \in L_2\}.$$

Znak działania składania będziemy zwykle opuszczać; n-tą potęgą języka nazwiemy n-krotne złożenie, tzn. $L^n = \underbrace{LL \dots L}_{n}$, przyjmiemy przy tym umowę, że $L^0 = \{\varepsilon\}$.

Domknięciem języka L nazwiemy zbiór

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Odbiciem lustrzanym języka L nazwiemy język

$$\hat{L} = \{v_n v_{n-1} \dots v_1\colon\ v_1 v_2 \dots v_n \in L\}.$$

Odbicie lustrzane języka L składa się zatem ze wszystkich słów języka L napisanych w odwrotnej kolejności.

TWIERDZENIE 4. *Klasa języków regularnych jest zamknięta ze względu na odbicie lustrzane, składanie i domknięcie języków.*

TWIERDZENIE 5. *Klasa języków regularnych jest zamknięta ze względu na sumę, przecięcie i różnicę języków.*

Zarówno twierdzenie 4 jak i twierdzenie 5 mogą być udowodnione w sposób efektywny, tzn. dla dowolnych dwóch języków regularnych L_1 i L_2, znając odpowiadające im automaty, potrafimy zbudować automaty rozpoznające języki L_1, $L_1 \cup L_2$, L_1^*, $L_1 \cap L_2$, $L_1 L_2$, $L_1 - L_2$.

Własność zamkniętości klasy $\mathscr{L}_R$ ze względu na operacje składania, domknięcia i sumy teoriomnogościowej jest wynikiem zamkniętości tej klasy ze względu na znacznie ogólniejszą operację, zwaną *operacją podstawiania*. Opiszemy tę operację dokładniej.

Przypuśćmy, że dla każdego $a \in V$ określony został pewien alfabet Σ_a. *Podstawieniem* nazywamy dowolne odwzorowanie

$$\tau\colon V^* \to 2^{\Sigma} \quad \text{gdzie} \quad \Sigma = \bigcup_{a \in V} \Sigma_a$$

spełniające następujące warunki:

(i) dla każdego $a \in V$, $\tau(a)$ jest określone i $\tau(a) \subseteq \Sigma_a^*$, a więc $\tau(a)$ jest językiem,

(ii) $\tau(\varepsilon) = \{\varepsilon\}$,

(iii) $\tau(a_1 \dots a_n) = \tau(a_1) \dots \tau(a_n)$, a więc każdemu słowu odwzorowanie τ przyporządkowuje pewien język będący złożeniem języków przyporządkowanych przez τ kolejnym literom tego słowa.

Jeżeli dla każdego a należącego do V zbiór $\tau(a)$ jest jednoelementowy, tzn. składa się z jednego tylko słowa, to τ nazywamy *homomorfizmem*.

Obrazem języka L ze względu na podstawienie τ nazwiemy sumę języków $\tau(x)$ dla wszystkich $x \in L$, czyli

$$\tau(L) \underset{\text{def}}{=} \bigcup_{x \in L} \tau(x).$$

TWIERDZENIE 6. *Jeżeli L jest językiem regularnym nad alfabetem V, a τ jest podstawieniem takim, że dla każdego $a \in V$, $\tau(a)$ jest językiem regularnym, to $\tau(L)$ jest również językiem regularnym. Innymi słowy, podstawianie języków regularnych do języków regularnych nie wyprowadza poza klasę $\mathscr{L}_R$.*

Warto może pokazać, w jaki sposób z twierdzenia 6 wynika zamkniętość klasy $\mathscr{L}_R$ ze względu na sumę, składanie i domknięcie. Rozważmy dowolne dwa języki L_1 i L_2 oraz alfabet $V = \{a, b\}$ i języki $P_1 = \{a, b\}$, $P_2 = \{ab\}$ i $P_3 = \{a\}^*$. Oczywiście P_1, P_2, P_3 są regularne. Niech τ będzie następującym podstawieniem: $\tau(a) = L_1$, $\tau(b) = L_2$. Jeżeli języki L_1

i L_2 są regularne, to z twierdzenia o podstawianiu wynika, że również języki $\tau(P_1) = L_1 \cup L_2$, $\tau(P_2) = L_1 L_2$, $\tau(P_3) = L_1^*$ są językami regularnymi.

TWIERDZENIE 7 (KLEENE, MYHILL). *Klasa języków regularnych jest najmniejszą klasą języków zawierającą wszystkie języki skończone (o skończonej liczbie słów) i zamkniętą ze względu na operacje sumy, składania i domknięcia.*

Warto zwrócić uwagę, że jest to drugie twierdzenie, obok twierdzenia 2 (Neroda), charakteryzujące w pełni klasę języków regularnych bez korzystania z pojęcia automatu.

Przypuśćmy teraz, że dany jest pewien RS-automat $\mathfrak{A}$, którego języka na razie nie znamy. Automat ten pozwala na rozstrzyganie w skończonej liczbie kroków, czy dane słowo $x \in V^*$ należy, czy nie należy do $L(\mathfrak{A})$. Może się jednak tak zdarzyć, że po przebadaniu bardzo dużej ilości słów nie natrafimy na żadne należące do $L(\mathfrak{A})$. Staniemy więc przed problemem: szukać dalej, czy poniechać tego, być może bezowocnego, postępowania. Oczywiście szukać dalej warto tylko wtedy, gdy istnieje gwarancja, że po skończonej liczbie kroków potrafimy rozstrzygnąć, czy $L(\mathfrak{A})$ jest puste, czy nie. Okazuje się, że gwarancja taka istnieje. W skończonej liczbie kroków potrafimy rozstrzygać o tej i innych jeszcze własnościach RS-automatów.

TWIERDZENIE 8. *Dla dowolnego RS-automatu $\mathfrak{A}$ potrafimy rozstrzygać w skończonej liczbie kroków, czy język tego automatu jest pusty, czy niepusty, skończony, czy nieskończony. Dla dwóch dowolnych RS-automatów $\mathfrak{A}_1$ i $\mathfrak{A}_2$ potrafimy rozstrzygnąć w skończonej liczbie kroków czy automaty te są równoważne.*

Aby rozstrzygnąć czy język danego n-stanowego automatu jest niepusty wystarczy, jak udawadniają Rabin i Scott [22], sprawdzić czy należy do niego jakieś słowo o długości nie większej od n. Słów takich jest oczywiście skończona ilość, zadanie da się więc wykonać efektywnie. Podobnie, aby rozstrzygnąć czy język danego automatu jest nieskończony wystarczy sprawdzić, czy należy do niego jakieś słowo o długości między n a $2n$.

Czy dwa dane automaty $\mathfrak{A}_1$ i $\mathfrak{A}_2$ są równoważne czy nie, rozstrzygamy badając, czy oba języki:

$$L(\mathfrak{A}_1) \cap [V^* - L(\mathfrak{A}_2)], \qquad L(\mathfrak{A}_2) \cap [V^* - L(\mathfrak{A}_1)]$$

gdzie $V = V_1 \cup V_2$, są puste czy nie. Jak wynika z uwagi do twierdzenia 5 potrafimy zbudować efektywnie automaty rozpoznające oba te języki, możemy zastosować tu zatem pierwszą część twierdzenia 8.

1.2. Automaty ze stosem (push-down). Jak zauważyliśmy przy omawianiu twierdzenia 2 (Neroda), automaty skończone mogą zapamię-

tywać informację jedynie poprzez zmianę stanu. Ponieważ liczba stanów jest skończona, liczba informacji, jaką automat może przechowywać w trakcie czytania każdego słowa, jest dla danego automatu ograniczona. Założenie to wpływa w sposób istotny na strukturę słów rozpoznawanych automatami skończonymi. Powoduje ono właśnie to, że zwykły formalizm nawiasowy nie jest rozpoznawalny przez takie automaty, weryfikacja poprawności formuł tego formalizmu wymaga bowiem porównywania ilości lewych i prawych nawiasów, a więc dla dostatecznie długich formuł zapamiętywania dostatecznie dużych ilości informacji.

Jedną, aczkolwiek nie jedyną, drogą prowadzącą do zwiększenia operatywności automatów jest przyjęcie nieskończonej pamięci. Można to oczywiście uczynić przez wyposażenie automatów w nieskończoną ilość stanów, takie rozwiązanie prowadzi jednak do modeli nieeleganckich, a przy tym nie mających dostatecznie jasnej interpretacji w istniejących maszynach matematycznych. Znacznie wygodniej jest przyjąć, że automat ma do swojej dyspozycji pamięć w postaci taśmy nieskończonej (w praktyce dostatecznie długiej), na której może robić notatki w trakcie czytania słów i z notatek tych następnie korzystać.

Jednym z takich rozwiązań jest *automat ze stosem*, znany również pod nazwą „push-down" automatu. Jest to urządzenie mające w swojej dyspozycji:

1. dwie taśmy: Pz — *pamięć zewnętrzną* i Pw — *pamięć wewnętrzną* zwaną również *stosem*, obie podzielone na klatki i jednostronnie nieskończone,

2. dwie *głowice*: Gz — głowica czytająca z Pz oraz Gw — głowica czytająca z Pw i pisząca na Pw,

3. *rejestr* mogący przyjmować skończoną ilość stanów i mogący przesuwać taśmę Pz w lewo i taśmę Pw w lewo i w prawo (rys. 3).

Przyjmuje się ponadto następujące założenia:

1. w każdej klatce taśmy Pz może być zapisany dowolny pojedynczy symbol tzw. *alfabetu zewnętrznego* V_z, lub symbol pusty,

2. klatkę pamięci Pz stojącą aktualnie naprzeciwko głowicy Gz nazywamy *klatką obserwowaną w* Pz. W każdym cyklu pracy automatu taśmę Pz przesunąć możemy o co najwyżej jedno miejsce w lewo, zmieniając tym samym pozycję klatki obserwowanej,

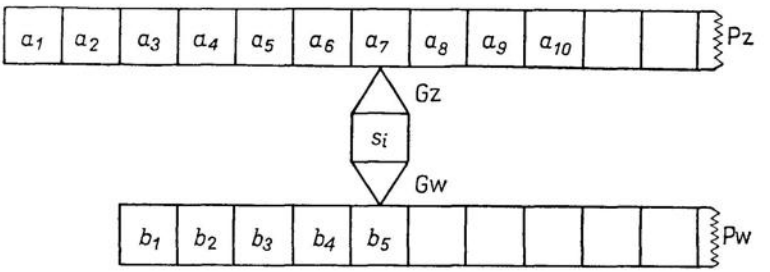
3. w każdej klatce taśmy Pw może być zapisany dowolny pojedynczy symbol tzw. *alfabetu wewnętrznego* V_w lub symbol pusty,

4. klatkę pamięci Pw stojącą aktualnie naprzeciwko głowicy Gw nazywamy *klatką obserwowaną w* Pw. W każdym cyklu pracy automatu możemy (a) wytrzeć klatkę obserwowaną i przesunąć taśmę Pw o jedno miejsce w prawo, lub (b) wpisać słowo nad V_w o długości k w k kolejnych

klatek, poczynając od klatki obserwowanej i przesunąć taśmę Pw o $k-1$ miejsc w lewo.

O alfabetach V_z i V_w nie zakłada się, że są rozłączne.

Konfiguracją automatu ze stosem nazywamy trójkę uporządkowaną $\langle x, s, y\rangle$, gdzie $x \in V_z^*$ i x jest słowem, którego pierwszy symbol stoi w aktualnie obserwowanej klatce pamięci Pz, natomiast pozostałe symbole stoją na prawo od niego; s jest aktualnym stanem automatu, $y \in V_w^*$ i y jest słowem, którego ostatni symbol stoi w aktualnie obserwowanej klatce pamięci Pw, a pozostałe symbole stoją na lewo od niego. Na przykład automat przedstawiony na rysunku 3 znajduje się w konfiguracji $\langle a_7 a_8 a_9 a_{10}, s_i, b_1 b_2 b_3 b_4 b_5\rangle$.



Rys. 3

Od jednej konfiguracji do drugiej automat przechodzić może posługując się *instrukcjami* postaci

$$\langle a, s, b\rangle \to \langle s', w\rangle,$$

gdzie $a \in V_z$, $s, s' \in S$ (S — zbiór stanów), $b \in V_w$, $w \in V_w^*$. Instrukcje takie czytamy w sposób następujący: jeżeli w Pz obserwowany jest symbol a, w Pw symbol b, a rejestr znajduje się w stanie s, to:

1. stan rejestru zmienić należy na s',
2. taśmę Pz przesunąć należy o jedno miejsce w lewo,
3. klatkę obserwowaną w Pw należy wyzerować (wytrzeć), a następnie słowo w wpisać na taśmę Pw, poczynając od klatki obserwowanej i taśmę Pw przesunąć o $|w| - 1$ miejsc w lewo (w przypadku gdy $w = \varepsilon$ oznacza to, że Pw przesunąć należy o jedno miejsce w prawo).

Ponadto dopuszcza się instrukcje postaci

$$\langle \varepsilon, s, b\rangle \to \langle s, w\rangle,$$

które czytamy jak następuje: cokolwiek byśmy obserwowali w Pz (może to być również symbol pusty), jeżeli w Pw obserwujemy b, a rejestr znajduje się w stanie s, to stan rejestru zmienić należy na s' i nie prze-

suwając Pz postępować dalej analogicznie jak dla instrukcji opisanych wyżej.

Każdy PD-automat określony jest przez pewną listę instrukcji, jest to przy tym automat niedeterministyczny, tzn. mogą istnieć instrukcje o identycznych wyrażeniach stojących na lewo od strzałki i różnych wyrażeniach stojących na prawo. W celu rozpoznania za pomocą takiego automatu dowolnego słowa $x = a_1 \dots a_p$ nad alfabetem V_z, wpisujemy to słowo w pierwsze p klatek taśmy Pz, wymazując jednocześnie klatki pozostałe. Następnie wymazujemy całą taśmę Pw i w pierwszą jej klatkę wpisujemy symbol pomocniczy σ. Na koniec ustawiamy automat w konfiguracji $\langle a_1 \dots a_p, s_0, \sigma\rangle$, gdzie s_0 jest wyróżnionym i określonym dla danego automatu stanem początkowym. Mówimy teraz, że automat rozpoznaje słowo $a_1 \dots a_p$, gdy w oparciu o określającą go listę instrukcji można ułożyć program przejścia od wyżej opisanej konfiguracji do konfiguracji postaci $\langle \varepsilon, s_f, y\rangle$, gdzie s_f należy do zbioru stanów końcowych automatu, natomiast y jest dowolnym słowem nad alfabetem V_w.

DEFINICJA 7. *Automatem ze stosem* lub PD-*automatem* nazywamy system

$$\mathfrak{A} = \langle V_z, V_w, S, I, s_0, F, \sigma\rangle,$$

gdzie

V_z i V_w są skończonymi alfabetami,

S jest skończonym zbiorem elementów, zwanych *stanami*,

$I \subseteq [(V_z \cup \{\varepsilon\}) \times S \times V_w] \times [S \times (V_w - \{\sigma\})^*]$ i I jest zbiorem skończonym, zwanym *listą instrukcji*,

$s_0 \in S$ jest *stanem początkowym*,

$F \subseteq S$ jest *zbiorem stanów końcowych*,

$\sigma \in V_w$ jest *symbolem początkowym*.

Przyjmiemy umowę, że fakt $\langle\langle a, s, b\rangle, \langle s', w\rangle\rangle \in I$ zapisywać będziemy w postaci $\langle a, s, b\rangle \to \langle s', w\rangle$.

DEFINICJA 8. Niech dane będą dwie konfiguracje $\langle ax, s, yb\rangle$ i $\langle x, s', yv\rangle$ automatu $\mathfrak{A}$, gdzie $a \in V_z \cup \{\varepsilon\}$, $x \in V_z^*$, $y, v \in V_w^*$, $b \in V_w$, $s, s' \in S$. Mówimy, że *konfiguracja* $\langle ax, s, yb\rangle$ *wyprowadza bezpośrednio konfigurację* $\langle x, s', yv\rangle$ *w automacie* $\mathfrak{A}$, co zapiszemy

$$\langle ax, s, yb\rangle \underset{\mathfrak{A}}{\Rightarrow} \langle x, s', yv\rangle,$$

gdy $\langle a, s, b\rangle \to \langle s', v\rangle$ w automacie $\mathfrak{A}$.

Mówimy, że *konfiguracja* $\langle x, s, y\rangle$ *wyprowadza* (*pośrednio*) *konfigurację* $\langle x', s', y'\rangle$ *w automacie* $\mathfrak{A}$, co zapiszemy

$$\langle x, s, y\rangle \overset{*}{\underset{\mathfrak{A}}{\Rightarrow}} \langle x', s', y'\rangle,$$

gdy istnieje taki ciąg konfiguracji automatu $\mathfrak{A}$

$$\langle x_1, s_{i_1}, y_1\rangle, \quad \dots, \quad \langle x_p, s_{i_p}, y_p\rangle,$$

że:

(i) $\langle x_1, s_{i_1}, y_1\rangle = \langle x, s, y\rangle, \quad \langle x_p, s_{i_p}, y_p\rangle = \langle x', s' y'\rangle,$

(ii) $\langle x_j, s_{i_j}, y_j\rangle \underset{\mathfrak{A}}{\Rightarrow} \langle x_{j+1}, s_{i_{j+1}}, y_{j+1}\rangle, \quad j = 1, \ldots, p-1.$

DEFINICJA 9. *Językiem definiowanym* lub *rozpoznawanym przez dany automat ze stosem* $\mathfrak{A}$ nazywamy zbiór tych wszystkich słów $x \in V_z^*$, dla których istnieje stan $s_f \in F$ oraz słowo $y \in V_w^*$ takie, że

$$\langle x, s_0, \sigma\rangle \overset{*}{\underset{\mathfrak{A}}{\Rightarrow}} \langle \varepsilon, s_f, y\rangle.$$

Podobnie jak to czyniliśmy dla RS-automatów, przyjmiemy i w tym przypadku, że dla każdego automatu ze stosem alfabety V_z i V_w są podzbiorami alfabetu uniwersalnego U. Język $L \subseteq V^*$ $(V \subseteq U)$ nazwiemy *S-językiem*, gdy istnieje rozpoznający go automat ze stosem. Klasę wszystkich S-języków oznaczymy przez $\mathscr{L}_S$. Stanowi ona istotne rozszerzenie klasy języków regularnych, tzn. zachodzi następujące

TWIERDZENIE 9. $\mathscr{L}_R \subseteq \mathscr{L}_S$ *oraz* $\mathscr{L}_R \neq \mathscr{L}_S$.

Przykładem języka należącego do $\mathscr{L}_S$, a nie należącego do $\mathscr{L}_R$ może być język $L = \{x\hat{x}\colon x \in V^* - \{\varepsilon\}\}$, gdzie $\hat{x}$ jest odbiciem lustrzanym słowa x (patrz paragraf o RS-automatach). (Założenie, że $x \neq \varepsilon$ przyjęto jedynie dla uproszczenia przykładu.) Opiszemy automat rozpoznający ten język. Dla uproszczenia przyjmiemy, że alfabet V jest dwuelementowy. Automat będzie wtedy następujący:

$$V_z = \{a, b\}, \quad V_w = \{a, b, \sigma\}, \quad S = \{s_0, s_1, s_2\}, \quad F = \{s_2\}$$

wraz z listą instrukcji:

1. $\langle \alpha, s_0, \beta\rangle \to \langle s_0, \beta\alpha\rangle$ dla wszystkich $\alpha \in V_z$ i $\beta \in V_w$,
2. $\langle \alpha, s_0, \beta\rangle \to \langle s_1, \beta\alpha\rangle$ dla wszystkich $\alpha \in V_z$ i $\beta \in V_w$,
3. $\langle \alpha, s_1, \alpha\rangle \to \langle s_1, \varepsilon\rangle$ dla wszystkich $\alpha \in V_z$,
4. $\langle \varepsilon, s_1, \beta\rangle \to \langle s_0, \beta\rangle$ dla wszystkich $\beta \in V_w$,
5. $\langle \alpha, s_1, \sigma\rangle \to \langle s_0, \varepsilon\rangle$ dla wszystkich $\alpha \in V_z$,
6. $\langle \varepsilon, s_1, \sigma\rangle \to \langle s_2, \sigma\rangle$.

Oczywiście instrukcje 1-5 są w istocie rzeczy schematami instrukcji, występują w nich bowiem zmienne α i β.

Niech teraz $x = abba$. Podamy przykład drogi rozpoznawania tego słowa przez automat. Cyfry w nawiasach oznaczać będą numery zastosowanych instrukcji.

$$\langle abba, s_0, \sigma\rangle \underset{(1)}{\Rightarrow} \langle bba, s_0, \sigma a\rangle \underset{(2)}{\Rightarrow} \langle ba, s_1, \sigma ab\rangle \underset{(2)}{\Rightarrow}$$

$$\underset{(2)}{\Rightarrow} \langle a, s_1, \sigma a\rangle \underset{(2)}{\Rightarrow} \langle \varepsilon, s_1, \sigma\rangle \underset{(6)}{\Rightarrow} \langle \varepsilon, s_2, \sigma\rangle.$$

Innym przykładem S-języka może być język $L = \{x\hat{x}\colon x \in R\}$, gdzie R jest dowolnym językiem regularnym. Łatwo również pokazać, że język

prostego formalizmu nawiasowego opisany w poprzednim paragrafie, jest S-językiem.

Ginsburg i Rice [13] (1) udowodnili, że S-językami są także tak zwane języki *typu ALGOL* (w języku angielskim ALGOL-like), tzn. takie, które dają się zdefiniować analogicznymi jak ALGOL systemami wyrażeń metajęzykowych [21]. Z tego powodu klasa S-języków jest obecnie intensywnie badana, co wywołuje jednak pewne kontrowersje, gdyż jak utrzymują niektórzy schemat języka typu-ALGOL jest zbyt uproszczony i w związku z tym, ich zdaniem, ALGOL nie jest językiem typu-ALGOL.

Podamy teraz kilka twierdzeń o zamkniętości klasy $\mathscr{L}_S$ ze względu na różne operacje.

TWIERDZENIE 10 [1]. *Jeżeli L jest S-językiem nad alfabetem V, a τ jest podstawieniem, takim że dla każdego $a \in V$, $\tau(a)$ jest S-językiem, to $\tau(L)$ jest również S-językiem.*

Stąd, podobnie jak dla języków regularnych, otrzymujemy następujący wniosek:

TWIERDZENIE 11. *Klasa $\mathscr{L}_S$ jest zamknięta ze względu na sumę, składanie i domknięcie języków.*

Klasa $\mathscr{L}_S$ nie jest zamknięta ze względu na przecięcie, ani ze względu na uzupełnianie języków, jest natomiast prawdziwe następujące słabsze twierdzenie:

TWIERDZENIE 12 [1]. *Jeżeli $L \in \mathscr{L}_S$ oraz $R \in \mathscr{L}_R$, to $L \cap R \in \mathscr{L}_S$ oraz $L - R \in \mathscr{L}_S$.*

Języki regularne mogą reprezentować w pewien sposób S-języki:

TWIERDZENIE 13 [6]. *Dla każdego alfabetu V istnieje taki alfabet Σ, S-język $D \subseteq \Sigma^*$ i homomorfizm $h\colon \Sigma^* \to V^*$, że każdy S-język $L \subseteq V^*$ daje się przedstawić w postaci $L = h(D \cap R_L)$, gdzie $R_L \subseteq \Sigma^*$ jest pewnym językiem regularnym określonym dla L.*

Na zakończenie niniejszego paragrafu podamy dwa twierdzenia o własnościach rozstrzygalnych i nierozstrzygalnych automatów ze stosem.

TWIERDZENIE 14 [1]. *Istnieją efektywne metody pozwalające dla dowolnego automatu ze stosem $\mathfrak{A}$ udzielać odpowiedzi na następujące pytania:*

1. *czy dane słowo x należy do $L(\mathfrak{A})$?*
2. *czy $L(\mathfrak{A})$ jest językiem pustym?*
3. *czy $L(\mathfrak{A})$ jest językiem skończonym?*

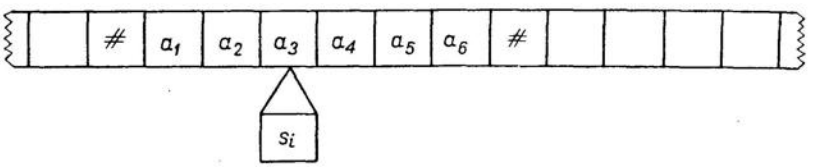
---

(1) Noty bibliograficzne podane przy twierdzeniach tego paragrafu odpowiadają równoważnym twierdzeniom (patrz wstęp) sformułowanym dla gramatyk bezkontekstowych. Wszystkie je można znaleźć ponadto w książce Ginsburga [11].

TWIERDZENIE 15 [1]. *Jeżeli V jest alfabetem co najmniej dwuelementowym, to nie istnieją metody efektywne pozwalające dla dwóch dowolnych automatów ze stosem $\mathfrak{A}_1$ i $\mathfrak{A}_2$ nad V odpowiedzieć na następujące pytania:*

1. *czy $L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2)$ jest S-językiem?*
2. *czy $L(\mathfrak{A}_1) \subseteq L(\mathfrak{A}_2)$?*
3. *czy $L(\mathfrak{A}_1) = L(\mathfrak{A}_2)$?*
4. *czy $L(\mathfrak{A}_1)$ jest językiem regularnym?*

1.3. Automaty liniowo ograniczone. Automaty ze stosem czytając zadane słowo posuwają się zawsze w jednym kierunku, mianowicie od lewej do prawej. Do raz odczytanego fragmentu słowa nie mogą już powrócić inaczej jak przez zanotowanie tego fragmentu w pamięci wewnętrznej (Pw). Jednak i w tym przypadku powrót oznacza zawsze wytarcie notatki, a więc zniszczenie informacji. Z jednej notatki korzystać można zatem tylko raz. Dlatego właśnie automaty ze stosem nie mogą na przykład rozpoznawać języka $L = \{a^i b^i a^i\colon i = 1,2,\ldots\}$. Jak się przekonamy, język ten jest rozpoznawany przez automaty liniowo ograniczone.



Rys. 4

Pojęcie automatu *liniowo ograniczonego* wprowadzone zostało po raz pierwszy przez J. Myhill'a [20]. Automat ten składa się z taśmy podzielonej na klatki, oraz z rejestru mogącego przyjmować skończoną ilość stanów, obserwować w każdym momencie pojedynczą klatkę taśmy, wpisywać w obserwowaną klatkę symbol z pewnego określonego alfabetu oraz przesuwać taśmę w każdym momencie o jedno miejsce w lewo, lub jedno miejsce w prawo, albo pozostawiać taśmę nieruchomą. Słowa, które automat będzie czytał, pisane są w pewnym alfabecie zewnętrznym V_z. Ponadto automat dysponuje pewną ilością symboli pomocniczych, tworzących tzw. alfabet wewnętrzny V_w. Symbole alfabetu wewnętrznego mogą pojawiać się na taśmie jedynie w następstwie wpisania ich tam przez automat w trakcie pracy. Ponadto automat dysponuje pewnym symbolem specjalnym $\#$ należącym do V_w, który występuje zawsze na początku i na końcu słowa, wyznaczając jego krańce.

Przed przystąpieniem do rozpoznawania (czytania) słowa $x \in V_z^*$ na taśmę wpisane zostaje słowo $\# x \#$ i taśmę ustawia się tak, aby rejestr obserwował pierwszy od lewej symbol słowa x. Sam rejestr ustawia się

w pewnym stanie początkowym s_0. Następnie automat zaczyna pracować posługując się pewną niedeterministyczną listą instrukcji postaci:

$$\langle a, s_j\rangle \to \langle b, s_i, k\rangle,$$

gdzie $a, b \in V_z \cup V_w$, $s_i, s_j \in S$ (S — zbiór stanów), $k \in \{-1, 0, +1\}$. Instrukcje te rozumieć należy w sposób następujący: jeżeli aktualnie obserwowanym symbolem jest a, natomiast aktualnym stanem rejestru jest s_j, to w miejsce a wpisz b, zmień stan rejestru na s_i i przesuń taśmę o k miejsc w prawo, tzn. o jedno miejsce w lewo, gdy $k = -1$, o jedno miejsce w prawo, gdy $k = +1$, i pozostaw nieruchomą, gdy $k = 0$. Przyjmuje się bardzo istotne założenie, że w czasie pracy rejestr nie może wyjść poza słowo $\# x \#$. Założenie to jest równoważne temu, że za każdym razem automat otrzymuje do dyspozycji taśmę o długości równej długości słowa $\# x \#$. Stąd właśnie bierze się termin: liniowo ograniczony. Uważamy, że automat zaakceptował słowo x, gdy rejestr automatu przesunął się tak, aby obserwować pierwszy z prawej symbol $\#$ oraz przyjął stan należący do wyróżnionego podzbioru stanów końcowych. Nie odgrywa przy tym roli, jakie słowo znajduje się wtedy na taśmie. Wprowadzimy teraz definicję formalną.

DEFINICJA 10. *Automatem liniowo ograniczonym lub LO-automatem* nazwiemy system

$$\mathfrak{A} = \langle V_z, V_w, S, I, s_0, F, \#\rangle,$$

gdzie

V_z i V_w są skończonymi alfabetami, zwanymi odpowiednio alfabetem zewnętrznym i alfabetem wewnętrznym,

S jest skończonym zbiorem stanów,

$I \subseteq [(V_z \cup V_w) \times S] \times [(V_z \cup V_w) \times S \times \{-1, 0, +1\}]$ jest *listą instrukcji*, o której zakłada się, że spełnia następujące dwa warunki:

(i) dla każdej pary $\langle a, s\rangle \in (V_z \cup V_w) \times S$ istnieje trójka $\langle b, s', k\rangle \in \in [(V_z \cup V_w) \times S \times \{-1, 0, 1\}]$, taka że $\langle\langle a, s\rangle, \langle b, s', k\rangle\rangle \in I$,

(ii) jeżeli $\langle\langle a, s\rangle, \langle b, s', k\rangle\rangle \in I$ oraz $a = \#$, to $b = \#$,

$s_0 \in S$ jest *stanem początkowym*,

$F \subseteq S$ jest *zbiorem stanów końcowych*,

$\# \in V_w$ jest wyróżnionym symbolem służącym do zaznaczania krańców słowa.

Podobnie jak dla automatów ze stosem przyjmujemy umowę, że fakt $\langle\langle a, s\rangle, \langle b, s', k\rangle\rangle \in I$ zapisywać będziemy w postaci $\langle a, s\rangle \to \to \langle b, s', k\rangle$.

Konfiguracją automatu liniowo ograniczonego nazwiemy słowo nad alfabetem $V_z \cup V_w \cup S$ (zbiór S traktujemy tu formalnie jako pewien alfabet składający się z symboli $s_0, s_1, \ldots, s_n$) będące postaci $\# x s_i y \#$, gdzie $x, y \in (V_z \cup V_w)^*$. Konfigurację tę interpretujemy w sposób nastę-

pujący: na taśmie automatu znajduje się aktualnie słowo $\#xy\#$, rejestr znajduje się w stanie s_j i obserwuje pierwszy symbol słowa y jeżeli $y \neq \varepsilon$, lub obserwuje prawy symbol $\#$ jeżeli $y = \varepsilon$. Dopuszcza się również konfiguracje postaci $s_j\#x\#$, tzn. takie, w których automat obserwuje lewy symbol $\#$.

Powiemy, że *konfiguracja* $\#xas_jby\#$ *wyprowadza bezpośrednio konfigurację* $\#xs_icby\#$, gdy istnieje instrukcja $\langle a, s_j\rangle \to \langle c, s_i, +1\rangle$; *konfiguracja* $\#xas_jby\#$ *wyprowadza bezpośrednio konfigurację* $\#xcbs_iy\#$, gdy istnieje instrukcja $\langle a, s_j\rangle \to \langle c, s_i, -1\rangle$; *konfiguracja* $\#xas_jby\#$ *wyprowadza bezpośrednio konfigurację* $\#xcs_iby\#$, gdy istnieje instrukcja $\langle a, s_j\rangle \to \langle c, s_i, 0\rangle$.

Oczywiście analogiczne określenia formułuje się również dla przypadku szczególnego konfiguracji typu $s_j\#x\#$.

Mówimy, że *konfiguracja* α *wyprowadza konfigurację* β *w automacie* $\mathfrak{A}$, co zapisujemy $\alpha \overset{*}{\underset{\mathfrak{A}}{\Rightarrow}} \beta$, gdy istnieje taki ciąg konfiguracji pośrednich $\alpha_0, \alpha_1, \ldots, \alpha_m$, że $\alpha_0 = \alpha$, $\alpha_m = \beta$ oraz α_i wyprowadza bezpośrednio α_{i+1} dla każdego $i = 1, 2, \ldots, m-1$.

DEFINICJA 11. *Językiem rozpoznawalnym lub definiowalnym* przez dany LO-automat $\mathfrak{A}$ nazywamy zbiór tych wszystkich słów $x \in V_z^*$, dla których istnieje takie słowo $y \in (V_z \cup V_w)^*$ i taki stan $s_f \in F$, że $\#s_0x\# \overset{*}{\underset{\mathfrak{A}}{\Rightarrow}} \#ys_f\#$.

Podobnie jak uprzednio założymy, że dla każdego LO-automatu $V_z, V_w \subseteq U$. Język $L \subseteq V^*$, gdzie $V \subseteq U$ nazywać będziemy LO-*językiem*, gdy istnieje rozpoznający go LO-automat. Klasę wszystkich *LO*-języków oznaczymy przez $\mathscr{L}_{LO}$. Jest ona istotnie szersza od klasy $\mathscr{L}_S$, tzn. zachodzi następujące twierdzenie:

TWIERDZENIE 16. $\mathscr{L}_S \subseteq \mathscr{L}_{LO}$, *przy czym* $\mathscr{L}_S \neq \mathscr{L}_{LO}$.

Jako przykłady LO-języków nie należących do $\mathscr{L}_S$ posłużyć mogą języki $L_1 = \{xx\colon x \in V_z\}$ i $L_2 = \{a^ib^ia^i\colon i = 1, 2, \ldots\}$, gdzie v^i oznacza oczywiście i-krotne złożenie symbolu v. Przykładów automatów rozpoznających te języki nie będziemy podawać, gdyż są one dosyć złożone (np. automat odpowiadający językowi L_2 ma 29 instrukcji).

Spośród rozważanych przez nas klas języków, klasa LO-języków jest obecnie najmniej zbadana. Fakt ten jest prawdopodobnie wynikiem szczególnego zainteresowania, jakim dotąd cieszyły się S-języki. W chwili obecnej sytuacja ta ulega powolnej zmianie. Może o tym świadczyć ukazanie się pod koniec roku 1966 pracy S. Ginsburga i S.A. Greibach [12], przynoszącej interesujące wyniki o klasie $\mathscr{L}_{LO}$. Jednym z nich jest twierdzenie o podstawianiu:

TWIERDZENIE 17. *Jeżeli* L *jest LO-językiem nad alfabetem* V, *a* τ *jest podstawieniem takim, że dla każdego* $a \in V$, $\tau(a)$ *jest LO-językiem oraz* $\varepsilon \notin \tau(a)$, *to* $\tau(L)$ *jest również LO-językiem.*

Założenie, że $\varepsilon \notin \tau(a)$ dla każdego $a \in V$ jest założeniem istotnym, nie dającym usunąć się z twierdzenia bez naruszenia jego prawdziwości. Niemniej założenie to można nieco osłabić.

TWIERDZENIE 18 [12]. *Niech V będzie dowolnym alfabetem i niech $c \in V$. Określimy na V^* homomorfizm $h_c\colon V^* \to V^*$ w sposób następujący: $h_c(a) = a$ dla każdego $a \in V$ i $a \neq c$ $h_c(c) = \varepsilon$.*

Jeżeli dla danego LO-języka $L \subseteq V^*$ homomorfizm ten spełnia warunek: istnieje liczba naturalna k, taka że dla każdego słowa $x \in L$ zachodzi $k|h_c(x)| \geqslant |x|$ (gdzie $|x|$ oznacza długość słowa x), to $h_c(L)$ jest LO-językiem.

Twierdzenie to pozwala na konstruowanie operacji podstawiania, w których część symboli ulega wytarciu pod warunkiem jednak, że miejsc podlegających wytarciu nie jest w żadnym słowie zbyt wiele. Z twierdzenia 17 otrzymujemy jako prosty wniosek:

TWIERDZENIE 19. *Klasa $\mathscr{L}_{LO}$ jest zamknięta ze względu na operacje sumy, składania i domknięcia.*

TWIERDZENIE 20. [17]. *Klasa $\mathscr{L}_{LO}$ jest zamknięta ze względu na operację przecięcia języków.*

Nie wiadomo natomiast, czy klasa LO-języków jest zamknięta ze względu na operację różnicy teoriomnogościowej.

2. Gramatyki kombinatoryczne

2.1. Semisystemy Thuego. Gramatyki kombinatoryczne, a ściślej mówiąc ich szczególna podklasa — gramatyki struktur frazowych, powstały w wyniku pewnych rozważań prowadzonych nad językami naturalnymi [3]. Kiedy gramatyki te opisano w sposób formalny, okazało się, że stanowią one szczególny przypadek pojęcia znanego w logice matematycznej pod nazwą *kombinatorycznych semisystemów Thuego.*

DEFINICJA 12. *Semisystem Thuego* nazywamy system $T = \langle V, R, x\rangle$, gdzie V jest alfabetem skończonym, $R \subseteq V^* \times V^*$ jest relacją skończoną, której elementy nazywamy produkcjami systemu T (fakt, że (v, w) należy do R zapisujemy w postaci $v \to w$), x jest wyróżnionym słowem nad alfabetem V, zwanym aksjomatem systemu.

Definicja ta odbiega nieco od sformułowania przyjmowanego zazwyczaj w logice matematycznej, w tej jednak postaci będzie znacznie wygodniejsza dla naszych celów. Czytelnika zainteresowanego szerzej semisystemami Thuego odsyłamy do książki M. Davisa [9]. Określimy teraz pewne pojęcia pochodne, które stanowić będą podstawę do dalszych rozważań.

DEFINICJA 13. Niech $\boldsymbol{T} = \langle V, R, x\rangle$ będzie semisystemem Thuego i niech $u, y \in V^*$. Powiemy, że słowo *u wyprowadza bezpośrednio w $\boldsymbol{T}$ słowo*

y, co zapiszemy $u \overset{}{\underset{T}{\Rightarrow}} y$, gdy istnieją słowa $z_1, z_2, v, w \in V^*$ takie, że $u = z_1 v z_2$, $y = z_1 w z_2$ oraz $v \to w$ jest produkcją systemu $\boldsymbol{T}$.

Powiemy, że słowo u *wyprowadza w* $\boldsymbol{T}$ *słowo* y lub że y jest *u-wyprowadzalne* w $\boldsymbol{T}$, co zapiszemy $u \overset{*}{\underset{T}{\Rightarrow}} y$, gdy istnieje ciąg słów $u_0, \ldots, u_n$ nad alfabetem V spełniający warunki:

1. $u_0 = u$, $u_n = y$,
2. $u_i \overset{}{\underset{T}{\Rightarrow}} u_{i+1}$ dla $i = 0, 1, \ldots, n-1$.

Ciąg $u_0, \ldots, u_n$ nazywamy *u-wyprowadzeniem* słowa y.

DEFINICJA 14. *Językiem definiowanym* przez dany semisystem Thuego $\boldsymbol{T} = \langle V, R, x\rangle$ nazywamy zbiór $L(\boldsymbol{T})$ wszystkich x-wyprowadzalnych słów nad V, tzn.

$$L(\boldsymbol{T}) = \{y\colon y \in V^* \text{ i } x \overset{*}{\underset{T}{\Rightarrow}} y\}.$$

Rozważmy dla przykładu semisystem

$$\boldsymbol{T} = \langle\{a, b, c\}, \{ab \to aabb, bb \to c\}, ab\rangle.$$

Jak łatwo sprawdzić, $L(\boldsymbol{T})$ składa się ze wszystkich słów postaci $a^n x$, gdzie $|x| \leqslant n$, oraz, jeżeli n jest parzyste, to x jest dowolnym słowem nad $\{c, b\}$, natomiast jeżeli n jest nieparzyste, to x jest dowolnym słowem nad $\{c, b\}$ zawierającym co najmniej jeden symbol b.

Semisystemy Thuego mają, z punktu widzenia lingwistyki matematycznej, dwie zasadnicze wady. Po pierwsze, jeżeli rozważymy jakieś słowo x języka $L(\boldsymbol{T})$ dowolnego systemu $\boldsymbol{T}$, to na ogół do słowa tego dadzą się jeszcze zastosować pewne produkcje z $\boldsymbol{T}$. Słowa definiowalne przez semisystemy Thuego nie mają więc jakiejś formy ostatecznej, można je bowiem dalej przekształcać. Z punktu widzenia zarówno teorii jak i modeli lingwistycznych, wskazanym jest jednak, aby słowa definiowalne nie mogły być już dalej przekształcane, tzn. aby były słowami końcowymi wyprowadzeń nieprzedłużalnych. Druga wada semisystemów Thuego jest znacznie poważniejsza i wynika z następującego twierdzenia.

TWIERDZENIE 21 [9]. *Każdy język definiowalny semisystemem Thuego jest językiem rekurencyjnie przeliczalnym i każdy język rekurencyjnie przeliczalny jest definiowalny pewnym semisystemem Thuego.*

Jak wiadomo, dla języka rekurencyjnie przeliczalnego nie istnieje na ogół metoda efektywna pozwalająca sprawdzać, czy dane słowo x należy czy nie należy do tego języka. Języki rekurencyjnie przeliczalne stanowią więc dla lingwistyki zły, bo zbyt ogólny, model matematyczny. Aby złu temu zaradzić, modyfikuje się pojęcie semisystemu Thuego w sposób następujący:

DEFINICJA 15. *Semisystemem Thuego z alfabetem pomocniczym* nazywamy system $\boldsymbol{T} = \langle V, \Sigma, R, x\rangle$, gdzie Σ jest pewnym alfabetem (zwanym pomocniczym) rozłącznym z V, natomiast $\langle V \cup \Sigma, R, x\rangle$ tworzy

zwykły semisystem Thuego. *Językiem definiowanym* przez taki system nazywamy zbiór wszystkich x-wyprowadzalnych słów nad V. Język ten oznaczamy przez $L(\boldsymbol{T})$.

Zauważmy, że systemy z alfabetem pomocniczym dysponują bogatszym arsenałem środków, mogą bowiem do definiowania danego języka używać symboli nie należących do jego alfabetu. Pozwala to oczywiście na ominięcie pierwszej z wymienionych trudności, nie zmienia jednak w niczym drugiej. Na semisystemy Thuego z alfabetem pomocniczym nakłada się zatem dalsze ograniczenia.

2.2 Gramatyki monotoniczne i kontekstowe

DEFINICJA 16. *Gramatyką monotoniczną* nazywamy semisystem Thuego z alfabetem pomocniczym $\boldsymbol{G} = \langle V, \Sigma, R, s\rangle$, gdzie s jest pojedynczym symbolem z alfabetu Σ, natomiast wszystkie produkcje $v \to u$ należące do R spełniają następujące warunki:

1. $|v| \leqslant |u|$, gdy $u \neq \varepsilon$,
2. jeżeli $v \neq s$, to $u \in (V \cup \Sigma)^* - \{\varepsilon\}$; jeżeli $v = s$, to $u \in (V \cup \Sigma)^*$.

TWIERDZENIE 22 [5]. *Dla każdego języka L nad V definiowalnego gramatyką monotoniczną istnieje efektywna metoda pozwalająca rozstrzygać czy zadane słowo $x \in V^*$ należy do języka L.*

Jak widzimy, gramatyki monotoniczne definiują języki, które nazwaliśmy we wstępie gramatykalizowalnymi. Okazuje się jednak, że istnieją języki gramatykalizowalne niedefiniowalne takimi gramatykami. Niemniej gramatyki monotoniczne stanowią obecnie najszerszą znaną klasę gramatyk definiujących języki gramatykalizowalne.

Każdą gramatykę monotoniczną sprowadzić można do pewnej specjalnej postaci, w której wszystkie produkcje $v \to u$ dla $u \neq \varepsilon$ mają postać $z_1 A z_2 \to z_1 y z_2$, gdzie $A \in \Sigma$, $z_1, z_2 \in (V \cup \Sigma)^*$, $y \in (V \cup \Sigma)^* - \{\varepsilon\}$.

Gramatyki takie nazywamy *kontekstowymi*. Jeżeli więc przez *gramatyki równoważne* rozumieć będziemy gramatyki definiujące identyczne języki, to otrzymamy następujące twierdzenie:

TWIERDZENIE 23 [5]. *Dla każdej gramatyki monotonicznej istnieje równoważna jej gramatyka kontekstowa.*

Gramatyki kontekstowe są przedmiotem specjalnego zainteresowania lingwistów, a to z tego względu, że prowadzonym w nich wyprowadzeniom przyporządkować można diagramy opisujące w pewnym stopniu strukturę wyprowadzanych słów. Z tego względu gramatyki te nazywa się również *gramatykami struktur frazowych*. Ponadto były one najwcześniej znanymi gramatykami kombinatorycznymi. Zwyczajowo przyjmuje się więc języki definiowalne gramatykami monotonicznymi nazywać *językami kontekstowymi*. Przyjmując, podobne jak w poprzednich paragrafach, założenie, że alfabety wszystkich gramatyk monotonicznych są

podzbiorami alfabetu uniwersalnego U, wprowadzić możemy klasę $\mathscr{L}_K$ wszystkich języków kontekstowych.

TWIERDZENIE 24 [17], [16]. *Dla każdego języka kontekstowego istnieje rozpoznający go automat liniowo ograniczony i dla każdego LO-języka istnieje definiująca go gramatyka kontekstowa. Inaczej mówiąc,* $\mathscr{L}_K = \mathscr{L}_{LO}$.

Twierdzenie to ma konstruktywny dowód, to znaczy dla każdej zadanej gramatyki kontekstowej potrafimy zbudować odpowiadający jej *LO*-automat i dla każdego zadanego *LO-automatu* potrafimy określić odpowiadającą mu gramatykę. Wynika z niego ponadto, że wszystkie twierdzenia dotyczące *LO*-języków przenoszą się w sposób automatyczny na języki kontekstowe. W rzeczywistości zostały one w oryginale sformułowane właśnie dla tych ostatnich języków, a to z tego względu, iż dowody prowadzi się znacznie prościej dla gramatyk niż dla automatów.

Rozpatrzmy teraz przykład gramatyki monotonicznej. Niech $V = \{a, b, c, d\}$, $\Sigma = \{S, A, B\}$, S — symbol początkowy, a lista produkcji jest następująca:

$$\begin{array}{lll} S \to AB, & AB \to BA, & A \to a, \\ AB \to ABc, & BA \to BAd, & B \to b. \\ AB \to dAB, & BA \to cBA, & \end{array}$$

Gramatyka ta definiuje język, w którym wszystkie słowa są postaci $d^i abc^j$, $i, j = 0, 1, \ldots$ lub $c^i bad^j$, $i, j = 0, 1, \ldots$

Nie jest to oczywiście gramatyka kontekstowa, bowiem produkcja $AB \to BA$ nie spełnia wymaganego warunku. Jeżeli jednak do alfabetu Σ dodamy symbol niekońcowy C, a powyżej zdefiniowaną produkcję zastąpimy równoważną jej listą

$$AB \to AC, \quad AC \to BC, \quad BC \to BA,$$

to otrzymamy gramatykę kontekstową równoważną naszej gramatyce. Należy tu zwrócić uwagę, że wyżej opisana konstrukcja możliwa jest jedynie dlatego, że do definiowania języka wolno używać symboli nie należących do alfabetu tego języka. W przeciwnym przypadku, np. dla zwykłych semisystemów Thuego, każde wprowadzenie nowej produkcji zwiększa na ogół język definiowany.

2.3 Gramatyki bezkontekstowe

DEFINICJA 17. *Gramatyką bezkontekstową* nazywamy semisystem Thuego z alfabetem pomocniczym $\boldsymbol{G} = \langle V, \Sigma, R, s\rangle$, gdzie s jest pojedynczym symbolem z alfabetu Σ, natomiast wszystkie produkcje są postaci $A \to x$, gdzie $A \in \Sigma$ oraz $x \in (V \cup \Sigma)^*$.

Języki definiowalne gramatykami bezkontekstowymi nazywamy *językami bezkontekstowymi*. Ich klasę oznaczać będziemy przez $\mathscr{L}_{BK}$.

Zauważmy, że gramatyki bezkontekstowe nie stanowią prostego

przypadku szczególnego gramatyk kontekstowych, mogą występować w nich bowiem produkcje typu $A \to \varepsilon$, gdzie $A \neq s$, niedozwolone w gramatykach kontekstowych. Jest niemniej prawdziwe następujące

TWIERDZENIE 25 [1]. *Dla każdej gramatyki bezkontekstowej* $\boldsymbol{G}$ *istnieje gramatyka bezkontekstowa* $\boldsymbol{G'}$ *nie zawierająca produkcji postaci* $A \to \varepsilon$, *taka że* $L(\boldsymbol{G'}) = L(\boldsymbol{G}) - \{\varepsilon\}$.

Jak już wspominaliśmy na wstępie, pojęciu gramatyki bezkontekstowej odpowiada w teorii automatów automat ze stosem:

TWIERDZENIE 26 [6], [10]. *Dla każdego języka bezkontekstowego istnieje rozpoznający go automat ze stosem i dla każdego S-języka istnieje definiująca go gramatyka bezkontekstowa. Inaczej mówiąc* $\mathscr{L}_{BK} = \mathscr{L}_S$.

Wynika stąd, że wszystkie twierdzenia o teoriomnogościowych i algebraicznych własnościach S-języków przenoszą się w sposób bezpośredni na języki bezkontekstowe. Ponadto, ponieważ twierdzenie 26 ma dowód efektywny przez konstrukcję, więc przenoszą się również twierdzenia o rozstrzygalności (tw. 14 i 15). Podobnie jak to miało miejsce dla języków kontekstowych, większość twierdzeń o językach bezkontekstowych została sformułowana i udowodniona właśnie przy użyciu aparatu gramatyk, a nie automatów.

Skala problemów dotyczących gramatyk bezkontekstowych jest bardzo szeroka. Wiele z nich jest już rozwiązanych, ale wiele jest tylko postawionych lub naszkicowanych. Czytelnika zainteresowanego tymi problemami odsyłamy do [11]. Znajdują się tam prawie wszystkie wyniki opublikowane do końca 1966 roku.

2.4. Gramatyki prawostronnie liniowe. Wśród gramatyk bezkontekstowych rozważa się wiele szczególnych podtypów gramatyk, takich jak gramatyki *sekwencyjne*, *normalne*, *metaliniowe*, *liniowe*, *lewo- i prawostronnie liniowe* [7].

DEFINICJA 19. Gramatykę bezkontekstową $\boldsymbol{G} = \langle V, \Sigma, R, s\rangle$ nazwiemy *prawostronnie liniową*, gdy wszystkie jej produkcje są postaci $A \to x$ lub $A \to xB$, gdzie $A, B \in \Sigma$ oraz $x \in V^*$.

TWIERDZENIE 27 [11]. *Język jest regularny wtedy i tylko wtedy, gdy jest definiowalny przez gramatykę prawostronnie liniową.*

Dowód tego twierdzenia przebiega przez efektywną konstrukcję automatu dla gramatyki i na odwrót.

Prace cytowane

[1] Y. Bar-Hillel, M. Perles and E. Shamir, *On formal properties of simple phrase-structure grammars*, Zeit. Phonetic Sprach. Kommunikationsforsch. 14 (1961), str. 143-172.

[2] J. Bellert, *Relational phrase structure grammar and its tentative applications*, Information and Control, 8 (1965), str. 503-530.

[3] N. Chomsky, *Three models for the description of language*, IRE Trans. Information Theory, IT2 (1956), str. 113-124.

[4] — *Syntactic structures*, Haga 1957.

[5] — *On certain formal properties of grammars*, Information and Control, 2 (1959), str. 137-167.

[6] — *Context free grammars and pushdown storage*, MIT Res. Lab. Electr. Quart. Prog. Rep. 65 (1962).

[7] — *Formal properties of grammars*, rozdz. 12, vol. 2, *Handbook of Mathematical Psychology*, John Wiley and Sons, Inc. (1963).

[8] — M. P. Schutzenberger, *The algebraic theory of context free languages*, *Computer programming and formal systems*, Amsterdam 1963, str. 118-161.

[9] M. Davis, *Computability and un solvability*, New York 1958.

[10] R. J. Evey, *The theory and application of pushdown store machines*, *Mathematical linguistics and automatic translation*, Harvard Univ., Computation Lab. Rep. NSF-IO, May 1963.

[11] S. Ginsburg, *The mathematical theory of context free languages*, McGraw-Hill Book Comp. (1966).

[12] — S. A. Greibach, *Mappings which preserves context sensitive languages*, Information and Control, 9 (1966), str. 563-582.

[13] — H. G. Rice, *Two families of languages related to* ALGOL, J. Assoc. Comput. Mach., 9 (1962), str. 350-371.

[14] A. Grzegorczyk, *Zarys logiki matematycznej*, Warszawa 1961.

[15] S. C. Kleene, *Introduction to metamathematics*, Groningen 1952.

[16] S. Y. Kuroda, *Classes of languages and linear bounded automata*, Information and Control, 7 (1964), str. 207-223.

[17] P. S. Landweber, *Three theorems on phrase structure grammars of type* 1, Information and Control, 6 (1963).

[18] A. J. Malcev, *Algoritmy i Rekursivnye Funkcii*, Moskva 1965.

[19] S. Marcus, *Algebraic linguistics, analytical models*, Academic Press 1966.

[20] J. Myhill, *Linear bounded automata*, Wright Air Dev. Division, Techn. Note No. 60-165 (1960). Cincinnati, Ohio.

[21] P. Naur (edit.), *Report on the algorithmic language* ALGOL-60, Comm. ACM, 3 (1960), str. 299-314.

[22] M. O. Rabin, D. Scott, *Finite automata and their decision problems*, IBM J. Res. Develop. (1959), 3, str. 144-125.

[23] A. M. Turing, *On computable numbers with an applications to the Entscheidungsproblem*, Proc. London Math. Soc., ser. 2, vol. 42, str. 230-265 (1936-1937); errata ibid., vol. 43, str. 544-546 (1937).