Nina Gierasimczuk

Algorytmiczne podejście do problemu uczenia się języka

1. Wstęp

W swych pracach Chomsky silnie wiązał dociekania lingwistyczne z psychologią. Adekwatność generatywnego modelu języka uzasadniał argumentami z zakresu wyuczalności. Za szczególną cechę języków naturalnych uważał możliwość ich przyswojenia w okresie dziecięctwa przez człowieka. Postulował istnienie aparatu służącego nabywaniu zdolności językowych, znajdującego się „na wyposażeniu" umysłu ludzkiego – *LAD* (skrót od *Language Acquisition Device*)(Chomsky, 1965).

W artykule opisuję formalne ujęcie procesu przyswajania wiedzy językowej w paradygmacie wyznaczonym przez Chomsky'ego. Proces nabywania zdolności językowych ma tutaj charakter indukcyjnego uogólnienia na podstawie skończonej próbki danych dotyczących języka. Paradygmat ten ustalony został przez E.M. Golda w latach 60-tych w postaci modelu identyfikacji w granicy (Gold, 1967). Model Golda jest bardzo ogólny, dotyczy wyuczalności rozumianej bardzo szeroko, bez odwołania do poszczególnych praktycznych realizacji algorytmów uczących się. Bardziej szczegółowy opis modelu identyfikacji w granicy wraz z oceną jego psychologicznej adekwatności znajdzie czytelnik w (Gierasimczuk, 2007).

Wprowadzenie modelu identyfikacji w granicy zaowocowało rozwojem dziedziny zwanej teorią wyuczalności. Umożliwiło formalną analizę zjawiska uczenia się jako dokonywania uogólnień indukcyjnych. Proces przyswajania języka jest tu opisywany w następujący sposób. Z ustalonej klasy języków zostaje wybrany jeden język. Uczeń otrzymuje informacje na jego temat. Dane przedstawione są na jeden z możliwych sposobów — albo poprzez tzw. tekst (prezentowane są jedynie elementy zadanego języka) albo poprzez tzw. informant (prezentowane są wszystkie możliwe ciągi nad zadanym alfabetem wraz z informacją, czy należą do zadanego języka). Oba rodzaje ciągów treningowych są nieskończone nawet wtedy, gdy mamy do czynienia z językiem skończonym — dopuszczone jest powtarzanie się elementów. Zadanie ucznia

polega na odgadnięciu nazwy wybranego języka (na przykład gramatyki generującej ten język). Model identyfikacji w granicy polega na wyszukiwaniu odpowiedniej gramatyki opisującej zaprezentowany ciąg informacji dotyczącej wybranego języka. W każdym kolejnym kroku procedury uczniowi prezentowana jest jednostka danych dotycząca nieznanego języka. Każdorazowo uczeń ma do dyspozycji tylko skończony zbiór danych. W każdym kroku uczeń wybiera nazwę języka. Ta procedura przebiega w nieskończoność. Język jest identyfikowany w granicy, gdy po pewnym czasie uczeń wybiera poprawnie ciągle tę samą nazwę (gramatykę). Właściwa identyfikowalność dotyczy klas języków. Cała klasa języków jest identyfikowalna w granicy, gdy istnieje algorytm zgadujący (uczeń) taki, że identyfikuje on w granicy dowolny język z tej klasy. Warto zaznaczyć, że uczeń nie ma dostępu do informacji, kiedy jego zgadnięcia są prawidłowe. Przetwarza informacje w nieskończoność, ponieważ nie może stwierdzić, czy w kolejnym kroku nie dostanie informacji, która zmusi go do zmiany wyboru.

Przyjmując powyższy aparat możemy udowodnić szereg twierdzeń na temat identyfikowalności różnych klas z hierarchii Chomsky'ego. Wśród nich szczególne miejsce zajmuje następujące twierdzenie:

Twierdzenie 1. (Gold 1967) *Żadna klasa języków zawierająca wszystkie języki skończone i przynajmniej jeden nieskończony (tzw. nadskończona klasa języków) nie jest identyfikowalna w granicy z danych jedynie pozytywnych.*

Z powyższego twierdzenia wynika, że żadna klasa języków z hierarchii Chomsky'ego, która mogłaby zawierać język naturalny (który jest potencjalnie nieskończony), nie jest identyfikowalna w granicy przy użyciu informacji wyłącznie pozytywnej. Wynik ten na wiele lat zawiesił formalne rozważania nad wyuczalnością oraz sprawił, że nie powstały żadne związki matematycznej teorii z wynikami psycholingwistyki. W latach 80–tych oraz 90–tych ten początkowy pesymizm został przezwyciężony. W pracy (Angluin, 1980) wskazano na klasy języków, które przecinają hierarchię Chomsky'ego i są wyuczalne z danych jedynie pozytywnych. Shinohara zaś wykazał, że wybrane ograniczenia nakładane na gramatyki kontekstowe sprawiają, że ich podklasy stają się pozytywnie wyuczalne (Shinohara, 1990). Zwrócono też uwagę na rolę informacji negatywnej w naturalnym procesie przyswajania języka — badacze dopatrują się jej w interakcjach dziecka z dorosłymi użytkownikami języka (zob. np. (Gordon, 1990)). W związku z tymi obserwacjami zaproponowano rozmaite rozszerzenia algorytmicznej wyuczalności wprowadzając do procedury informację negatywną. Wśród tych propozycji poczesne miejsce zajmują rozstrzygnięcia Dany Angluin, która uzyskała wiele istotnych szczegółowych wyników w teorii wyuczalności. Należy do nich algorytm L^* uczący

się języków regularnych przez tzw. zapytania (Angluin, 1987). Algorytm ten jest bardzo efektywny, z jego pomocą wyuczalne są jednak tylko języki regularne. W hierarchii Chomsky'ego język naturalny należy umieszczać powyżej klasy języków regularnych — w obrębie klasy gramatyk bezkontekstowych lub kontekstowych (zob. np. (Gazdar Pullum, 1985)). W sposób naturalny powstaje pytanie, jak skonstruować efektywny algorytm dla wyższych klas, na przykład klasy gramatyk bezkontekstowych. Wysunięto kilka propozycji, wszystkie one posiadały jednak sporo nierealistycznych założeń i ograniczeń (zob. np. (Angluin, 1987)). Wreszcie, w latach 90-tych Yasubumi Sakakibara zaproponował algorytm analogiczny do L^*, nazywany LA. Efektywnie uczy się on, również przez zapytania, klasy gramatyk bezkontekstowych.

W artykule zaprezentuję wspomniane wyżej algorytmy uczące się gramatyk regularnych i bezkontekstowych. Pragnę zaznaczyć, że wybór ten jest arbitralny — algorytmów uczących się na rozmaite sposoby różnych klas języków jest bardzo wiele (zob. np. (Kanazawa, 1994)). Prezentowane tu rozwiązania są jednak wystarczająco reprezentatywne, aby dać czytelnikowi obraz dziedziny i narzędzi, jakie są tu szczególnie przydatne. W ostatniej części tekstu wyodrębnię założenia metodologiczne algorytmów i porównam je z naturalnym procesem przyswajania języka.

2. Wprowadzenie terminologii

Poniżej przedstawimy podstawowe definicje i ustalenia notacyjne.

Alfabet to skończony i niepusty zbiór symboli.

Na przykład: $A = \{a, b\}$ — alfabet binarny;

Słowo to skończony ciąg symboli wybranych z pewnego alfabetu.

Na przykład: ciąg „*aaabbbaababaa*" jest słowem nad alfabetem A.

Długość słowa to liczba symboli w nim występujących, oznaczamy ją przez $lh()$, np. $lh(aa)$=2.

Słowo puste, oznaczane symbolem λ, to ciąg zerowej długości, czyli: $lh(\lambda) = 0$.

Jeśli A jest alfabetem, to przez A^k oznaczamy zbiór słów długości k nad alfabetem A. Na przykład $\{a, b\}^2 = \{aa, ab, ba, bb\}$. Dla dowolnego alfabetu A mamy $A^0 = \{\lambda\}$.

Zbiór wszystkich słów nad alfabetem A oznaczamy przez A^*, np. $\{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$. Innymi słowy $A^* = \bigcup_{n\in\omega} A^n$.

Przez xy, dla $x, y \in A^*$, oznaczamy *konkatenację* słowa x oraz słowa y, czyli słowo powstałe z kopii słowa x, po której bezpośrednio następuje kopia słowa y. Na przykład: jeśli $x = bab$ oraz $y = a$, to $xy = baba$. Dla dowolnego

ciągu $\alpha \in A^*$ zachodzi $\lambda\alpha = \alpha\lambda = \alpha$, czyli λ jest elementem neutralnym dla konkatenacji.

Dla dowolnych języków $X, Y \subseteq A^*$ określamy ich konkatenację XY, jako $XY = \{\alpha\beta : \alpha \in X,\ \beta \in Y\}$.

Niech $x, y \in A^*$. Wtedy:

— x jest *prefiksem* y wtedy i tylko wtedy, gdy istnieje $z \in A^*$ takie, że $y = xz$.

— x jest *sufiksem* y wtedy i tylko wtedy, gdy istnieje $z \in A^*$ takie, że $y = zx$.

Niech $B \subseteq A^*$. Wtedy:

— Zbiór B jest *domknięty na prefiksy* wtedy i tylko wtedy, gdy:

$$\forall x, y \in A^*\ [(y \in B \wedge x \text{ jest prefiksem } y) \Longrightarrow x \in B].$$

— Zbiór B jest *domknięty na sufiksy* wtedy i tylko wtedy, gdy:

$$\forall x, y \in A^*\ [(y \in B \wedge x \text{ jest sufiksem } y) \Longrightarrow x \in B].$$

Dowolny zbiór słów wybranych z A^*, dla pewnego alfabetu A, nazywamy *językiem*. Jeśli A jest alfabetem oraz $L \subseteq A^*$, to mówimy, że L jest językiem nad A^*.

Definicja 1. *Gramatyka G jest to struktura postaci (A, Σ, S, P), gdzie:*

— *A jest alfabetem (alfabetem terminalnym);*

— *Σ jest zbiorem zmiennych (alfabetem nieterminalnym);*

— *$S \in \Sigma$ jest symbolem początkowym;*

— *P jest skończonym zbiorem par postaci $\alpha_i \longrightarrow \beta_i$ dla $i = 1, \ldots, n$ oraz $\alpha_i, \beta_i \in (A \cup \Sigma)^*$.*

Definicja 2. *Dla dowolnych $\gamma, \gamma' \in (A \cup \Sigma)^*$ powiemy, że γ' jest w gramatyce G bezpośrednio wyprowadzalna z γ, czyli $\gamma \longrightarrow_G \gamma'$ wtedy i tylko wtedy, gdy istnieją η_1, η_2 oraz $i = 1, \ldots, n$, takie, że $\gamma = \eta_1\alpha_i\eta_2$ oraz $\gamma' = \eta_1\beta_i\eta_2$.*

Definicja 3. *γ' jest w gramatyce G wyprowadzalna z γ, czyli $\gamma \longrightarrow^*_G \gamma'$ wtedy i tylko wtedy, gdy istnieje ciąg $\gamma_1, \ldots, \gamma_m \in (A \cup \Sigma)^*$ taki, że $\gamma = \gamma_1$, $\gamma' = \gamma_m$ oraz $\gamma_i \longrightarrow_G \gamma_{i+1}$, dla $i = 1, \ldots, m-1$.*

Definicja 4. *Język generowany przez gramatykę G jest to zbiór: $L(G) = \{\gamma \in A^* : S \longrightarrow^*_G \gamma\}$.*

Hierarchia języków

— Język L jest rekurencyjnie przeliczalny wtedy i tylko wtedy, gdy istnieje maszyna Turinga e taka, że

$$\alpha \in L \iff \{e\}(\alpha)\downarrow, \text{dla dowolnego } \alpha \in A^*,$$

czyli: maszyna Turinga e zatrzymuje się tylko na słowach należących do języka L.

— Język L jest rekurencyjny wtedy i tylko wtedy, gdy istnieje maszyna Turinga M licząca funkcję χ_L taką, że dla dowolnego $\alpha \in L$:

$$\chi_L(\alpha) = \begin{cases} 0 & \text{jeśli } \alpha \notin L \\ 1 & \text{jeśli } \alpha \in L. \end{cases}$$

— Język L jest pierwotnie rekurencyjny wtedy i tylko wtedy, gdy jego funkcja charakterystyczna jest pierwotnie rekurencyjna (w sprawie pojęć teorii rekursji zob. (Shoenfield, 1991), (Cutland, 1980)).

— Języki kontekstowe są postaci $L(G)$, gdzie wszystkie produkcje gramatyki G są postaci:

$$\eta_1 Y \eta_2 \longrightarrow_G \eta_1 \beta \eta_2, \text{ dla } Y \in \Sigma, \eta_1, \eta_2, \beta \in (A \cup \Sigma)^*.$$

— Języki bezkontekstowe są postaci $L(G)$, gdzie wszystkie produkcje gramatyki G są postaci:

$$Y \longrightarrow_G \beta, \text{ dla } Y \in \Sigma, \ \beta \in (A \cup \Sigma)^*.$$

— Języki regularne są postaci $L(G)$, gdzie wszystkie produkcje gramatyki G są postaci:

$$Y \longrightarrow_G \alpha Z \text{ lub } Y \longrightarrow_G \alpha, \text{ dla } Y, \ Z \in \Sigma, \ \alpha \in A^*.$$

3. Uczenie się języków regularnych z danych pozytywnych i negatywnych

Dana Angluin w pracy *Learning Regular Sets from Queries and Counterexamples* rozpatruje możliwość skończonej identyfikacji języka regularnego z danych pozytywnych i negatywnych. Identyfikacja odbywa się poprzez znalezienie odpowiadającego poszukiwanemu językowi deterministycznego automatu skończonego. Nabywanie wiedzy o nieznanym języku przebiega pod kontrolą nauczyciela zwanego minimalnie adekwatnym (*Minimally Adequate Teacher*). Nauczyciel odpowiada na pytania dotyczące należenia wyrażeń do języka oraz sprawdza, czy wysunięta przez ucznia hipoteza (uzyskany na podstawie danych automat skończony) jest poprawna. Jeśli hipoteza okazuje się błędna, to nauczyciel podaje kontrprzykład, który należy do symetrycznej różnicy zbioru poszukiwanego i zbioru generowanego przez automat–hipotezę. W dalszej części artykułu zaprezentuję algorytm L^*, który przyswaja dowolny język regularny przy pomocy dowolnego minimalnie adekwatnego nauczyciela. L^* działa w czasie wielomianowym względem liczby stanów minimalnego deterministycznego automatu skończonego

akceptującego język oraz względem maksymalnej długości kontrprzykładów podawanych przez nauczyciela.

3.1. Deterministyczne automaty skończone

Jak wspomnieliśmy na początku tego rozdziału, uczeń uzyskuje wiedzę na temat nieskończonego zbioru (języka) regularnego dzięki konstrukcji automatu odpowiadającego poszukiwanemu zbiorowi. Automat skończony odpowiadający danemu zbiorowi L to taki automat, który akceptuje wszystkie i tylko elementy zbioru L. Poniżej znajduje się definicja deterministycznego automatu skończonego oraz języka rozpoznawanego przez ten model obliczeń.

Definicja 5. *Deterministyczny automat skończony (DFA) jest to piątka postaci* (A, Q, q_0, F, δ)*, gdzie:*

- A *jest alfabetem wejściowym;*
- Q *jest skończonym zbiorem stanów;*
- $q_0 \in Q$ *jest wyróżnionym stanem początkowym;*
- $F \subseteq Q$ *jest zbiorem stanów akceptujących;*
- $\delta : Q \times A \longrightarrow Q$ *jest funkcją przejścia.*

Język rozpoznawany (akceptowany) przez DFA H to zbiór tych słów nad alfabetem A, które są akceptowane przez H, czyli:

$$L(H) = \{w \in A^* : \bar{\delta}(q_0, w) \in F\},$$

gdzie $\bar{\delta}$ to uogólniona funkcja przejścia określona następującymi warunkami rekurencyjnymi:

$$\bar{\delta}(q, \lambda) = q$$

$$\text{oraz dla dowolnego } w \in A^* \text{ i } a \in A,\ \bar{\delta}(q_0, wa) = \delta(\bar{\delta}(q_0, w), a).$$

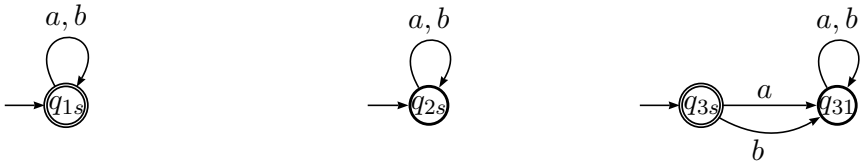
Przykład Poniżej znajduje się kilka prostych przykładów języków regularnych wraz z akceptującymi je automatami skończonymi.

Niech $A = \{a, b\}$. Rozważmy język $L_1 = A^*$. $L_1 = L(H_1)$, gdzie $H_1 = (Q_1, q_{1s}, F_1, \delta_1)$ taki, że: $Q_1 = \{q_{1s}\}$, $F_1 = \{q_{1s}\}$, $\delta_1(q_{1s}, a) = q_{1s}$ oraz $\delta_1(q_{1s}, b) = q_{1s}$.

Niech $L_2 = \emptyset$. Wówczas $L_2 = L(H_2)$, gdzie $H_2 = (Q_2, q_{2s}, F_2, \delta_2)$ taki, że: $Q_2 = \{q_{2s}\}$, $F_2 = \emptyset$, $\delta_2(q_{2s}, a) = q_{2s}$ oraz $\delta_2(q_{2s}, b) = q_{2s}$.

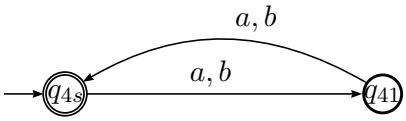
Niech $L_3 = \{\lambda\}$. Wówczas $L_3 = L(H_3)$, gdzie $H_3 = (Q_3, q_{3s}, F_3, \delta_3)$ taki, że: $Q_3 = \{q_{3s}, q_{31}\}$, $F_3 = \{q_{3s}\}$, $\delta_3(q_{3s}, i) = q_{31}$ oraz $\delta_3(q_{31}, i) = q_{31}$ dla $i = a, b$.

Automaty skończone często reprezentuje się w postaci grafu, w którym wierzchołki symbolizują stany, stan początkowy jest zaznaczony strzałką, stan akceptujący jest wyróżniony przez podwójne kółko a strzałki pomiędzy stanami opisują funkcję przejścia na literach reprezentowanych przez etykiety tych strzałek.



Rysunek 1. DFA akceptujące L_1, L_2 oraz L_3.

Niech symbol $n_x(w)$ oznacza liczbę wystąpień litery x w słowie w. Opiszemy teraz język, w którym występują tylko słowa o równej, co do parzystości, liczbie wystąpień liter a i b. Zatem $L_4 = \{w \in A^* : n_a(w) \equiv n_b(w) (\mathrm{mod}\, 2)\}$. $L_4 = L(H_4)$, gdzie $H_4 = (Q_4, q_{4s}, F_4, \delta_4)$ taki, że: $Q_4 = \{q_{4s}, q_{41}\}$, $F_4 = \{q_{4s}\}$, $\delta_4(q_{4s}, i) = q_{41}$ oraz $\delta_4(q_{41}, i) = q_{4s}$, dla $i = a, b$.



Rysunek 2. FA akceptujący $L_4 = \{w \in A^* : n_a(w) \equiv n_b(w) (\mathrm{mod} 2)\}$.

Zauważmy, iż język ten możemy opisać inaczej jako zbiór wszystkich słów o nieparzystej długości nad alfabetem binarnym.

Wcześniej zdefiniowaliśmy języki regularne jako języki generowane przez gramatyki regularne. Zachodzi następujące twierdzenie:

Twierdzenie 2. *(Kleene, 1956) Język $L \subseteq A^*$ jest regularny, wtedy i tylko wtedy, gdy istnieje DFA H taki, że $L = L(H)$.*

3.2. Zapytania algorytmu L*

Algorytm L^* (uczeń) w sposób istotny korzysta ze wskazówek nauczyciela, który zawsze odpowiada poprawnie (dlatego czasem nazywany jest wyrocznią) na zadawane przez ucznia pytania. Rozważymy dwa rodzaje pytań: pytania o należenie oraz pytania o równoważność. Pierwsza klasa pytań dotyczy poszczególnych ciągów nad pewnym ustalonym alfabetem. Pytania są postaci: Czy dany ciąg α należy do poszukiwanego języka? Nauczyciel odpowiada TAK, jeśli ciąg α należy do tego języka, w przeciwnym przypadku

odpowiada przecząco. Drugi rodzaj pytań dotyczy automatu M skonstruowanego przez algorytm L^*. Pytania te mają następującą postać: Czy zbiór, który jest akceptowany przez automat M jest identyczny z poszukiwanym zbiorem? Nauczyciel odpowiada TAK, jeśli zbiory te są identyczne, w przeciwnym przypadku pada odpowiedź negatywna. Jest ona każdorazowo stowarzyszona z kontrprzykładem — ciągiem należącym do symetrycznej różnicy zbioru poszukiwanego oraz zbioru odpowiadającego skonstruowanemu przez ucznia automatowi. Warto zauważyć, że o ile istnieje jedna poprawna odpowiedź na pytanie o należenie, o tyle poprawnych negatywnych odpowiedzi na pytanie o równoważność jest wiele. Pociąga to za sobą konkluzję, że różni nauczyciele będą tak samo odpowiadali na pytania pierwszego rodzaju. Różnice zaś będą się pojawiały w odpowiedziach na pytania drugiego rodzaju, nauczyciel może bowiem wybrać dowolny kontrprzykład spośród nieskończenie wielu możliwych.

Poniżej ścisły zapis obu kategorii zapytań:

1. Zapytania o należenie danej struktury do szukanej gramatyki:

$$T(\alpha) = \begin{cases} 1 & \text{jeśli } \alpha \in L \\ 0 & \text{jeśli } \alpha \notin L. \end{cases}$$

2. Zapytania o ekstensjonalną równoważność automatu skończonego otrzymanego przez algorytm L^* oraz automatu nauczyciela:

$$R(M) = \begin{cases} 1 & \text{jeśli } \mathbf{L}(M) = L \\ \langle 0, \alpha \rangle, \alpha \in L \div \mathbf{L}(M) & \text{jeśli } \mathbf{L}(M) \neq L. \end{cases}$$

3.3. Tablica obserwacyjna

W każdym kroku swego działania algorytm L^* dysponuje danymi na temat skończonego zbioru ciągów nad alfabetem. Ciągi te są sklasyfikowane według odpowiedzi nauczyciela na pytania o należenie. W każdym kroku działania algorytmu L^* mamy więc do czynienia z pewnym skończonym fragmentem informantu (w sensie Golda) dla poszukiwanego języka. Informant ten jest specyficzny ze względu na ciągi do niego należące — są one konstruowane z dotychczasowych danych językowych przez algorytm uczący się. Cechują się maksymalną prostotą względem otrzymywanych kontrprzykładów. Przechowywaną przez algorytm L^* informację można przedstawić w postaci dwuwymiarowej tabeli, zwanej tablicą obserwacyjną.

Definicja 6. ***Tablica obserwacyjna*** *jest to* (S, E, T)*, gdzie:*

1. *S jest niepustym skończonym zbiorem ciągów domkniętym na prefiksy;*
2. *E jest niepustym skończonym zbiorem ciągów domkniętym na sufiksy;*

3. T jest skończoną funkcją $(S \cup SA)E \to \{0,1\}$, gdzie[1]*:*

$$T(s) = 1 \iff s \in L.$$

(S, E, T) przedstawiamy za pomocą dwuwymiarowej tablicy, w której:

1. Wiersze są oznaczone elementami zbioru $(S \cup SA)$.
2. Kolumny są oznaczone elementami zbioru E.
3. Wartość w komórce o współrzędnych (s, e), gdzie $s \in (S \cup SA)$, $e \in E$, jest równa $T(se)$.
4. Niech $s \in (S \cup SA)$, wtedy $row(s)$ jest skończoną funkcją $f : E \longrightarrow \{0,1\}$ taką, że $f(e) = 1 \iff T(se) = 1$.

	T	e E
S	$s\ldots$	$\vdots$ $1(= T(se))$
(SA)	s_1	

Tabela 1. Tablica obserwacyjna

Definicja 7. *Tablica obserwacyjna (S, E, T) jest **domknięta** wtedy i tylko wtedy, gdy:*

$$\forall t \in SA\ \exists s \in S\ row(t) = row(s).$$

Definicja 8. *Tablica obserwacyjna (S, E, T) jest **spójna** wtedy i tylko wtedy, gdy:*

$$\forall s_1, s_2 \in S\ \forall a \in A\ [row(s_1) = row(s_2) \Longrightarrow row(s_1 a) = row(s_2 a)].$$

3.4. Konstrukcja automatu skończonego przy użyciu tablicy obserwacyjnej

Algorytm L^* używa tablicy obserwacyjnej do stworzenia odpowiedniego automatu skończonego. Wiersze tablicy obserwacyjnej oznaczone elementami zbioru S są kandydatami na stany automatu. Kolumny zaś, oznaczone elementami zbioru E, to wejścia wczytywane przez automat. Wiersze oznaczone elementami zbioru SA używane są w konstrukcji funkcji przejścia tego automatu.

[1] Zgodnie z powszechnym zwyczajem zakładamy, że operator konkatenacji wiąże silniej niż operator sumy.

Definicja 9. *Niech (S, E, T) będzie domkniętą i spójną tablicą obserwacyjną. Wówczas $M(S, E, T)$ jest automatem skończonym $M = (A, Q, q_0, F, \delta)$, gdzie:*

- *$Q = \{row(s) : s \in S\}$ jest zbiorem stanów automatu M;*
- *$q_0 = row(\lambda)$ jest stanem początkowym automatu M;*
- *$F = \{row(s) : s \in S \wedge T(s) = 1\}$ jest zbiorem stanów akceptujących;*
- *$\delta(row(s), a) = row(sa)$ jest funkcją przejścia automatu M.*

Definicja 10. *Automat skończony $M(S, E, T)$ jest zgodny ze skończoną funkcją T wtedy i tylko wtedy, gdy:*

$$\forall s \in (S \cup SA)\ \forall e \in E\ \bar{\delta}(q_0, se) \in F \Longleftrightarrow T(se) = 1.$$

Definicja 11. *Niech M oraz M^1 będą automatami deterministycznymi nad alfabetem A takimi, że: $M = (Q^M, q_0^M, \delta^M, F^M)$ oraz $M^1 = (Q^{M^1}, q_0^{M^1}, \delta^{M^1}, F^{M^1})$. Funkcja $\Gamma : Q^M \longrightarrow Q^{M^1}$ jest izomorfizmem automatów M i M^1, jeśli spełnione są następujące warunki:*

- *$\Gamma : Q^M \longrightarrow Q^{M^1}$ jest bijekcją;*
- *$\Gamma(q_0^M) = q_0^{M^1}$;*
- *dla dowolnego $q \in Q^M, a \in A$ $[\Gamma(\delta^M(q, a)) = \delta^{M^1}(\Gamma(q), a)]$;*
- *$F^M = \Gamma(F^{M^1})$.*

Twierdzenie 3. *Jeśli tablica obserwacyjna (S, E, T) jest domknięta i spójna to automat skończony $M(S, E, T)$ skonstruowany jak wyżej jest zgodny ze skończoną funkcją T. Każdy inny automat skończony zgodny z tą funkcją, lecz nieizomorficzny z automatem $M(S, E, T)$ musi mieć więcej stanów.*

3.5. Algorytm L*

Algorytm L^* działa na podstawie danych pozytywnych i negatywnych. Konstruuje początkową tablicę, następnie dąży do jej uspójnienia i domknięcia stosując zapytania o należenie. Gdy tablica stanie się domknięta i spójna, konstruuje z niej automat skończony na sposób opisany wcześniej. Wysuwa zapytanie o ekstensjonalną równoważność skonstruowanego automatu i automatu reprezentującego poszukiwany język. Jeśli otrzymuje odpowiedź pozytywną, to kończy pracę podając na wyjściu skonstruowany automat. Jeśli odpowiedź nauczyciela jest negatywna, to otrzymany kontrprzykład dodawany jest do tablicy obserwacyjnej, po czym algorytm wraca do poleceń uspójnienia i domknięcia tablicy. Poniżej znajduje się ścisły opis algorytmu L^* zapisany w tzw. pseudokodzie opartym na języku *Pascal* z elementami *C*.

```
begin
S := {λ} ∪ A;
```

$E := \{\lambda\}$;
Stosując zapytania o należenie utwórz tablicę obserwacyjną
(S, E, T):
  kolumna etykietowana λ
  wiersze etykietowane wszystkimi $a \in A$
  komórka o współrzędnych (a, λ) etykietowana wartościami $T(a)$,
  dla każdego $a \in A$
repeat
  while ((S, E, T) nie jest domknięta lub nie jest spójna) do
    { if ((S, E, T) nie jest spójna) then
      { znajdź $s_1, s_2 \in S$, $a \in A$, $e \in E$ takie, że:
        $row(s_1) = row(s_2)$ i $T(s_1ae) \neq T(s_2ae)$
        dodaj s_1a do zbioru E
        rozszerz T do $(S \cup SA)E$ używając zapytań o należenie
      }
      if ((S, E, T) nie jest domknięta) then
      { znajdź $s_1 \in S$ i $a \in A$ takie, że:
        dla dowolnego $s \in S$ $row(s_1a) \neq row(s)$
        dodaj s_1a do zbioru S
        rozszerz T do $(S \cup SA)E$ używając zapytań o należenie
      }
    }
  M:=$M(S, E, T)$;
  Zapytaj o poprawność M
  if odpowiedź=$\langle 0, \text{kontrprzykład } t\rangle$ then
    { dodaj kontrprzykład t i wszystkie jego prefiksy do S
      rozszerz T do $(S \cup SA)E$ używając zapytań o należenie
    }
until odpowiedź=TAK;
RETURN M;
end.

Warto zaznaczyć, że dla dowolnego tzw. minimalnie adekwatnego nauczyciela prezentującego nieznany regularny zbiór U, algorytm L^* podaje na wyjściu automat skończony H izomorficzny z minimalnym automatem akceptującym zbiór U.

Co więcej, jeśli n jest liczbą stanów minimalnego automatu dla zbioru U, zaś m jest górnym ograniczeniem długości kontrprzykładów, to całkowity czas działania algorytmu L^* jest wielomianowy względem n i m. Ponadto algorytm kończy obliczenie po najwyżej n zapytaniach o równoważność oraz najpóźniej po $n - 1$. wykonaniu głównej pętli.

Dyskusja Model Angluin eksplikuje pojęcie nauczyciela w teorii uczenia się języka. Taki opis stwarza możliwość modelowania roli nauczyciela w procesie uczenia się.

Ważną częścią modelu Angluin są dwojakiego rodzaju zapytania. Pytania o należenie nie wprowadzają niczego nowego w stosunku do modelu Golda. Jest to rozwinięcie idei przedstawiania danych językowych w postaci funkcji charakterystycznej danego języka, czyli informantu. Istotną nowością jest wprowadzenie zapytań o równoważność w celu uczynienia procesu uczenia się konkluzywnym. Założenie o udzielaniu przez nauczyciela poprawnej odpowiedzi na ten rodzaj pytań jest realistyczne. Problem ekstensjonalnej równoważności dwóch automatów skończonych jest bowiem rozstrzygalny (zob. (Hopcroft et al., 2001)).

Zadajmy sobie następujące pytania.

1. Czy możliwa jest redukcja zapytań?
2. Czy taki model wyuczalności można zastosować do języków wyższych klas, w szczególności do języków bezkontekstowych?

Ad. 1. Jeśli zrezygnujemy z zapytań o równoważność w procedurach identyfikacji języków nieskończonych, to uzyskamy algorytmy z dowolną dokładnością aproksymujące poszukiwany automat. Aproksymujący model wydaje się adekwatny do opisu naturalnego przyswajania języka. Rezygnacja z zapytań o równoważność prowadzi do zastosowania metod probabilistycznych w inferencji gramatyk. Modelom probabilistycznym poświęcono wiele uwagi; szczególnie istotnym rozwiązaniem jest tzw. PAC – learning (*probably approximately correct learning*) (zob. np. (Pitt, 1989), (Valiant, 1984), (Parekh Honavar, 1997)).

Gold wykazał, że niemożliwe jest identyfikowanie jakiejkolwiek klasy języków zawierającej wszystkie języki skończone oraz jakikolwiek język nieskończony przy użyciu jedynie pozytywnych danych (zob. Twierdzenie 1) . Tak więc, redukcja zapytań o należenie mogłaby być użyteczna tylko o tyle, o ile wśród danych językowych, z których korzysta uczeń są wadliwe dane wraz z informacją, że są wadliwe. Rozważania naturalnego procesu przyswajania języka nie prowadzą do konieczności redukcji tych zapytań. Pochodzenie informacji negatywnej wydaje się nieodłącznie związane z postacią wspomagającego ten proces nauczyciela. Co więcej forma zaproponowana przez Angluin: konstruowania wadliwych struktur przez ucznia, a nie przez nauczyciela, również odpowiada naturalnej sytuacji uczenia. Ciągi generowane przez ucznia są oceniane przez nauczyciela pod względem ich poprawności. Tak założenie o konieczności danych pozytywnych i negatywnych, jak i metoda prezentacji w postaci zapytań jest adekwatna do opisu naturalnego procesu przyswajania języka.

Ad. 2. Problem skończonej wyuczalności języków wyższych klas rozważymy w następnym rozdziale.

4. Uczenie się gramatyk bezkontekstowych ze strukturalnych danych pozytywnych i negatywnych

Nie można opisać składniowej struktury języka naturalnego środkami gramatyk regularnych (Chomsky, 1957). Zatem próby utożsamienia LAD z algorytmem uczącym się języków regularnych, opisywanym w poprzednim rozdziale muszą być nieskuteczne. Ludzką zdolność przyswajania języka można naśladować tylko przy pomocy algorytmu uczącego się klas języków, które zawierają język naturalny. Często twierdzi się, że język naturalny jest bezkontekstowy, to znaczy, że właśnie środkami gramatyk bezkontekstowych można opisać jego strukturę składniową (zob. (Gazdar Pullum, 1985), (Pullum Gazdar, 1982), (Shieber, 1985)).

W tym rozdziale opiszemy algorytmy uczące się języków bezkontekstowych, rozważymy trudności pojawiające się przy konstruowaniu takich algorytmów oraz zbadamy ich moc wyjaśniającą.

Algorytm $\mathbf{L^{cf}}$ Angluin w pracy (Angluin, 1987) zaproponowała metodę identyfikacji gramatyk bezkontekstowych przy pomocy algorytmu L^{cf}. Poniżej opiszemy pokrótce tę propozycję.

Niech G będzie poszukiwaną gramatyką bezkontekstową. Algorytm uczący się L^{cf} zna w momencie inicjacji:

1. zbiór A symboli terminalnych gramatyki G;
2. zbiór Σ symboli nieterminalnych gramatyki G;
3. symbol początkowy S gramatyki G.

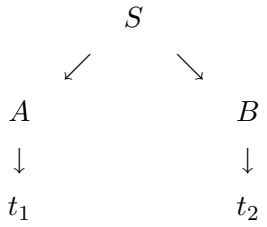
Algorytm korzysta z pomocy nauczyciela odpowiadającego na pytania dwóch kategorii:

1. Pytania o należenie. Niech x będzie ciągiem symboli terminalnych, A zaś symbolem nieterminalnym. Pytanie brzmi: Czy x może zostać wyprodukowany z A za pomocą G? Pytanie to będziemy skrótowo oznaczać: $MEMBER(x, A)$.
2. Pytania o równoważność. Niech H będzie gramatyką bezkontekstową. Pytanie brzmi: Czy H jest równoważna G? Skrótowo: $EQUIV(H)$. Odpowiedź negatywna na pytanie o równoważność stowarzyszona jest z kontrprzykładem z symetrycznej różnicy zbioru generowanego przez H i zbioru generowanego przez G.

W momencie inicjacji algorytm umieszcza wszystkie możliwe produkcje G w zbiorze hipotetycznych produkcji — P. Następnie zadaje pytanie

$EQUIV(H)$ gdzie $H = \{A, V, S, P\}$. Jeśli $EQUIV(H) = 1$, to L^{cf} podaje na wyjściu gramatykę H. Jeśli $EQUIV(H) = 0$, to podany przez nauczyciela kontrprzykład powoduje, że ze zbioru produkcji P usunięta zostaje przynajmniej jedna produkcja. Łatwo zauważyć, że H do momentu osiągnięcia równoważności z G zawsze będzie nadzbiorem G. Kontrprzykład będzie więc zawsze pochodził ze zbioru $H - G$.

Pozostaje wyjaśnić jak przebiega analiza kontrprzykładu t zakończona decyzją, która z produkcji powinna zostać wyeliminowana ze zbioru P. Otóż L^{cf} znajduje drzewo derywacyjne t z symbolu startowego S w gramatyce H. Załóżmy, że $t = t_1 t_2$ oraz, że drzewo derywacyjne ciągu t w gramatyce H wygląda tak:



Rysunek 3. Drzewo derywacyjne słowa t z S w gramatyce H.

W takiej sytuacji L^{cf} zadaje pytanie: $MEMBER(t_1, A)$. Jeśli odpowiedź jest negatywna, to t_1 może być derywowane z A w H, ale nie w G. L^{cf} usuwa z P produkcję $A \longrightarrow t_1$. Jeśli zaś odpowiedź jest pozytywna, to L^{cf} zadaje pytanie $MEMBER(t_2, B)$. Jeśli na to pytanie odpowiedź jest negatywna, to L^{cf} usuwa z P produkcję $B \longrightarrow t_2$. Jeśli odpowiedź jest pozytywna znaczy to, że do G należą produkcje $A \longrightarrow t_1$ oraz $B \longrightarrow t_2$, nie zaś $S \longrightarrow t$. Pozostaje więc usunąć z P produkcję $S \longrightarrow AB$. W ten sposób algorytm L^{cf} dochodzi do wyznaczenia wszystkich i tylko produkcji G.

Rozwiązanie opisane powyżej jest problematyczne. Największą trudność sprawia fakt, że pytanie $EQUIV(H)$ jest nierozstrzygalne (zob.(Sudkamp, 1988), (Hopcroft et al., 2001)). W związku z tym nie można liczyć na realistyczną implementację nauczyciela poprawnie odpowiadającego na to pytanie. Kolejny problem związany jest ze złożonością kontrprzykładów. W ogólnym przypadku nie ma rekurencyjnego oszacowania rozmiaru wymaganych kontrprzykładów.

Ponadto założenie o tak dużej wiedzy początkowej ucznia (choćby znajomość symboli nieterminalnych) jest zbyt mocne by adekwatnie opisywać naturalny proces uczenia się języka. To samo tyczy się zapytań o symbole nieterminalne.

Podsumowując algorytm L^{cf} jest mniej realistyczny niż L^* z powodów, których z zaproponowanego przez Angluin modelu nie można wyeliminować.

Algorytm LA Rozwinięcie koncepcji skończonego uczenia się zaproponował Yasubumi Sakakibara (Sakakibara, 1990). Jest to bezpośrednie przełożenie algorytmu L^* opisanego w rozdziale 3. na problem uczenia się gramatyk bezkontekstowych. Zaproponowany algorytm uczy się danej gramatyki bezkontekstowej G na podstawie tzw. danych strukturalnych. Są nimi szkielety drzew derywacyjnych gramatyki G. Szereg następujących faktów pozwoli przybliżyć w zarysie zasadę działania zapowiadanego algorytmu.

1. Zbiór drzew derywacyjnych danej gramatyki bezkontekstowej jest regularnym zbiorem drzew.
2. Regularny zbiór drzew to zbiór drzew rozpoznawany przez pewien automat drzewiasty.
3. Procedura tworzenia z drzew derywacyjnych ich opisów strukturalnych zachowuje regularność zbioru.
4. Problem uczenia się gramatyki bezkontekstowej z opisów strukturalnych jest więc redukowalny do problemu uczenia się pewnego automatu drzewiastego.

Należy podkreślić, iż celem „kształcenia" naszego algorytmu L^* w wersji dla gramatyk bezkontekstowych nie jest język bezkontekstowy, lecz pewna gramatyka bezkontekstowa. Wiadomo bowiem, że dla każdego języka bezkontekstowego istnieje nieskończenie wiele gramatyk go opisujących.

4.1. Gramatyki bezkontekstowe

Od gramatyk regularnych, którymi zajmowaliśmy się w poprzednim rozdziale gramatyki bezkontekstowe różnią się postacią produkcji. Dla przypomnienia:

1. Produkcje w gramatykach regularnych:

$$Y \longrightarrow_G \alpha Z \text{ lub } Y \longrightarrow_G \alpha \text{ dla } Y,\ Z \in \Sigma,\ \alpha \in A^*.$$

2. Produkcje w gramatykach bezkontekstowych:

$$Y \longrightarrow_G \beta \text{ dla } Y \in \Sigma,\ \beta \in (A \cup \Sigma)^*.$$

Definicja 12. *Niech G_1, G_2 będą gramatykami bezkontekstowymi, wtedy:*

$$G_1 \equiv G_2 \ \textit{wtw}\ L(G_1) = L(G_2).$$

Czyli: Gramatyki bezkontekstowe są równoważne wtedy i tylko wtedy, gdy generują ten sam język.

Przykład Gramatyka bezkontekstowa G^{PAL}.

$$G^{PAL} = \{A, \Sigma, R, P\}, \text{ gdzie:}$$

1. $A = \{0, 1\}$ jest zbiorem symboli terminalnych,
2. $\Sigma = \{R\}$ jest zbiorem symboli nieterminalnych,
3. R jest nieterminalnym symbolem początkowym,
4. P jest zbiorem produkcji takim, że:

$$P = \{R \longrightarrow \lambda, R \longrightarrow 0, R \longrightarrow 1, R \longrightarrow 0R0, R \longrightarrow 1R1\}.$$

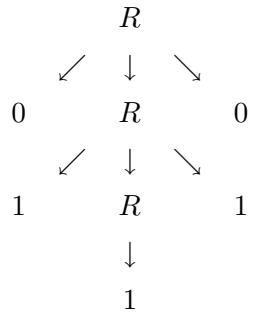
Nietrudno zauważyć, że gramatyka G^{PAL} generuje wszystkie palindromy nad alfabetem $\{0, 1\}$.

4.2. Drzewa derywacyjne gramatyki bezkontekstowej

Niech $G = (A, \Sigma, P, S)$. Każdemu ciągowi z języka generowanego przez gramatykę G można przypisać drzewo derywacyjne. Drzewa derywacyjne gramatyki G to drzewa spełniające następujące warunki:

1. W korzeniu drzewa znajduje się symbol S.
2. Każdy wierzchołek nie będący liściem jest etykietowany symbolem nieterminalnym ze zbioru Σ.
3. Każdy liść jest etykietowany symbolem terminalnym ze zbioru A lub λ.
4. Jeśli wierzchołek nie będący liściem jest etykietowany symbolem B zaś jego synowie etykietowani są kolejno od lewej strony symbolami $X_1, X_2, \ldots, X_k$, to $B \longrightarrow X_1X_2 \ldots X_k$ jest produkcją w P.

Przykład Aby podać przykład drzewa derywacyjnego pewnego ciągu w danej gramatyce bezkontekstowej możemy wykorzystać gramatykę G^{PAL}. Wybierzmy w tym celu prosty przykład palindromu nad alfabetem $\{0, 1\}$: 01110. Ciąg ten ma w gramatyce G^{PAL} drzewo derywacyjne następującej postaci:



Rysunek 4. Drzewo derywacyjne ciągu 01110 w gramatyce G^{PAL}.

Oczywiście, elementy zbioru A, czyli symbole terminalne, znajdziemy w etykietach liści, zaś elementy zbioru Σ, czyli symbole nieterminalne, w pozostałych węzłach drzewa derywacyjnego.

W dalszych rozważaniach będzie nas interesował zbiór wszystkich drzew derywacyjnych gramatyki G, zwany dalej $D_S(G)$, gdzie S jest symbolem początkowym gramatyki G. Będziemy dalej pomijać indeks dolny S. Pisząc $D(G)$ mamy na myśli zbiór drzew derywowanych w gramatyce G z symbolu początkowego S.

Operacja

Definicja 13. *Niech V będzie skończonym zbiorem symboli funkcyjnych takim, że:*

$$V = V_0 \cup V_1 \cup \ldots \cup V_n, \text{ gdzie:}$$

1. *V_0 jest zbiorem stałych, odpowiadającym zbiorowi symboli terminalnych gramatyki;*
2. *V_i jest zbiorem symboli funkcyjnych o liczbie argumentów równej i, dla $i = 1, \ldots, n$, odpowiadających symbolom nieterminalnym gramatyki.*

Etykietami węzłów w drzewie derywacyjnym są elementy zbioru V. Etykietami liści mogą być jedynie elementy zbioru V_0, czyli stałe funkcyjne. Skończone drzewa nad V mogą być identyfikowane z dobrze zbudowanymi termami nad V. Jako takie mogą być zapisywane w formie liniowej.

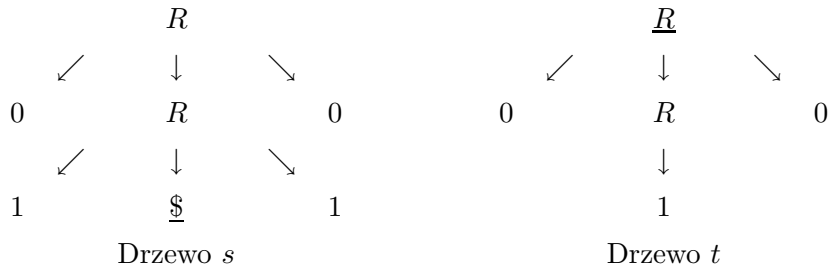
Definicja 14. *Niech V^T będzie zbiorem wszystkich skończonych drzew nad V. Niech $\$ \notin V$ będzie nowym symbolem funkcyjnym o liczbie argumentów równej 0. Wtedy:*

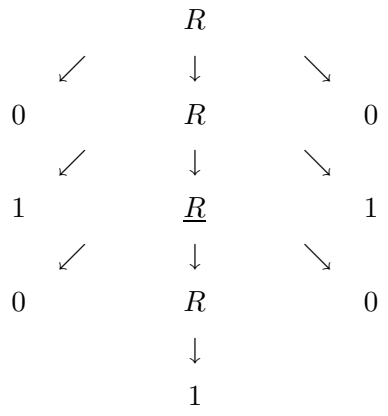
$$V_\$^T = \{t \in (V \cup \{\$\})^T : t \text{ zawiera dokładnie jeden symbol } \$\}.$$

Definicja 15. *Niech $s \in V_\$^T$ oraz $t \in (V^T \cup V_\$^T)$. Definiujemy operację podmieniania liścia etykietowanego symbolem $\$$ w drzewie s przez drzewo t:*

$$s\#t(x) = \begin{cases} s(x) & \text{jeśli } x \in s \text{ oraz } s(x) \neq \$ \\ t(y) & \text{jeśli } x = z \cdot y, s(z) = \$, y \in t. \end{cases}$$

Przykład Poniżej znajduje się ilustracja operacji podmieniania liścia etykietowanego $ w drzewie s drzewem t.





Drzewo $s\#t$

Rysunek 5. Konstrukcja drzewa $s\#t$.

Definicja 16. *Niech $S \subseteq V_{\$}^{T}$ oraz $T \subseteq (V^{T} \cup V_{\$}^{T})$. Definiujemy:*

$$S\#T = \{s\#t : s \in S, t \in T\}.$$

Szkielety drzew derywacyjnych Symbolem $K(D(G))$ będziemy określać zbiór szkieletów drzew derywacyjnych gramatyki G. Szkielety różnią się od drzew derywacyjnych jedynie tym, że w miejscach symboli nieterminalnych występuje symbol σ. Struktura i etykiety liści drzewa derywacyjnego pozostają bez zmian. Każdy taki szkielet drzewa derywacyjnego gramatyki G nazywać będziemy **opisem strukturalnym tej gramatyki**.

Definicja 17. *Niech $Sk = \{\{\sigma\} \times \{1, 2, \ldots, m\}\}$, m jest maksymalną liczbą argumentów symboli funkcyjnych w Sk.*

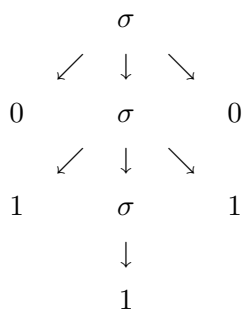
Definicja 18. *Niech $t \in V^{T}$. Definiujemy opis strukturalny (szkielet) drzewa t:*

$$s(t)(x) = \begin{cases} t(x) & \text{jeśli } x \text{ jest liściem w } t \\ \sigma & \text{jeśli } x \text{ jest różnym od liścia węzłem w drzewie } t. \end{cases}$$

Definicja 19. *Niech T będzie zbiorem drzew. Odpowiadający mu zbiór szkieletów to:*

$$K(T) = \{s(t) : t \in T\}.$$

Przykład Opisem strukturalnym omawianej gramatyki G^{PAL} odpowiadającym przedstawionemu wcześniej drzewu derywacyjnemu będzie:



Rysunek 6. Opis strukturalny (szkielet) gramatyki G odpowiadający ciągowi 01110.

Definicja 20. *Niech G_1, G_2 będą gramatykami bezkontekstowymi, wtedy:*

$$G_1 \equiv_{str} G_2 \text{ wtw } K(D(G_1)) = K(D(G_2)).$$

Czyli: Dwie gramatyki bezkontekstowe są strukturalnie równoważne wtedy i tylko wtedy, gdy odpowiadające im zbiory szkieletów drzew derywacyjnych są identyczne.

Rozważanie strukturalnej równoważności gramatyk pozwoli nam ominąć nierozstrzygalny problem zwykłej równoważności gramatyk bezkontekstowych.

4.3. Automaty drzewiaste

Algorytm L^* w wersji dla gramatyk bezkontekstowych identyfikuje nieskończony zbiór szkieletów drzew derywacyjnych gramatyki bezkontekstowej dzięki konstrukcji automatu drzewiastego odpowiadającego poszukiwanemu zbiorowi. Automat drzewiasty odpowiadający danemu zbiorowi szkieletów $K(D(G))$ to taki automat, który akceptuje wszystkie i tylko elementy zbioru $K(D(G))$. Poniżej znajduje się definicja deterministycznego automatu drzewiastego.

Definicja 21. *Deterministyczny automat drzewiasty (inaczej: DFTA), jest to:*

$$A = \left[\left(Q, F_1, \ldots, F_n, q_1, \ldots, q_m \right), Q_f \right], \text{ gdzie :}$$

— $F_i : Q^{k_i} \mapsto Q$, *dla* $i = 1, \ldots, n$;
— $q_1, \ldots, q_m \in Q$; $Q_f \subseteq Q$;
— *Niech t będzie termem nad słownikiem* $\left(\{f_1, \ldots, f_n\}, k, \{c_1, \ldots, c_m\} \right)$
$wart_A(t)$ *jest wartością termu t w A.*

Funkcja przejścia w DFTA A:

1. $wart_A(c_i) = Q_i$;
2. $wart_A\left(f_i(t_1, \ldots, t_{k_i})\right) = F_i\left(wart_A(t_1), \ldots, wart_A(t_{k_i})\right)$.

Fakt 1. *DFTA A akceptuje term t wtedy i tylko wtedy, gdy $wart_A(t) \in Q_f$.*

4.4. Zapytania algorytmu LA

Algorytm LA (uczeń) w sposób istotny korzysta ze wskazówek nauczyciela, który odpowiada poprawnie na zadawane przez ucznia pytania. Tak jak w przypadku algorytmu LA dla języków regularnych mamy tutaj do czynienia z dwoma rodzajami pytań: o należenie oraz o równoważność. Pierwsza klasa pytań dotyczy poszczególnych szkieletów drzew. Pytania są postaci: Czy dany szkielet s należy do zbioru szkieletów drzew poszukiwanej gramatyki? Nauczyciel odpowiada TAK, jeśli s należy do tego zbioru, w przeciwnym przypadku odpowiada przecząco. Drugi rodzaj pytań dotyczy automatu drzewiastego B skonstruowanego przez algorytm LA. Pytania te mają następującą postać: Czy zbiór szkieletów drzew, który jest akceptowany przez automat B jest identyczny z poszukiwanym zbiorem szkieletów? Nauczyciel odpowiada TAK, jeśli zbiory te są identyczne, w przeciwnym przypadku pada odpowiedź negatywna. Jest ona każdorazowo stowarzyszona z tzw. kontrprzykładem — ciągiem należącym do symetrycznej różnicy poszukiwanego zbioru szkieletów oraz zbioru odpowiadającego skonstruowanemu przez ucznia automatowi drzewiastemu. Warto zauważyć, że tak jak w przypadku algorytmu uczącego się języków regularnych różni nauczyciele będą różnili się między sobą w odpowiedziach na pytania drugiego rodzaju, nauczyciel może bowiem wybrać dowolny kontrprzykład spośród nieskończenie wielu możliwych. Poniżej opis obu kategorii pytań:

1. Zapytania o należenie danej struktury do szukanej gramatyki (G_u):

$$T(s) = \begin{cases} 1 & \text{jeśli } s \in K(D(G_u)) \\ 0 & \text{jeśli } s \notin K(D(G_u)). \end{cases}$$

2. Zapytania o równoważność struktury wyjściowej (G') algorytmu i struktury szukanej (G_u):

$$R(A) = \begin{cases} 1 & \text{jeśli } K(D(G_u)) = K(D(G')) \\ \langle 0, s \rangle, s \in K(D(G_u)) \div K(D(G')) & \text{jeśli } K(D(G_u)) \neq K(D(G')). \end{cases}$$

4.5. Tablica obserwacyjna

Podczas swego działania algorytm LA dysponuje danymi na temat skończonego zbioru szkieletów drzew derywacyjnych. Szkielety te są sklasyfikowane według odpowiedzi nauczyciela na pytania o należenie. Przechowywaną przez algorytm LA informację można przedstawić w postaci dwuwymiarowej tabeli, zwanej tablicą obserwacyjną.

Definicja 22. *Zbiór drzew A nazywamy domkniętym na poddrzewa wtedy i tylko wtedy, gdy jeśli s należy do A, to wszystkie poddrzewa drzewa s o głębokości niemniejszej niż 1 należą do zbioru A.*

Definicja 23. *Zbiór drzew B nazywamy domkniętym na \$–prefiksy ze względu na zbiór A wtedy i tylko wtedy, gdy jeśli $e \in B - \{\$\}$, to istnieje $e' \in B$, $s_1, \ldots, s_{k-1} \in A \cup \Sigma$ takie, że: $e = e'\#\sigma(s_1, \ldots, s_{i-1}, \$, s_i, \ldots, s_{k-1})$.*

Definicja 24. ***Tablica obserwacyjna*** *jest to (S, E, T), gdzie:*

1. *S jest niepustym skończonym zbiorem szkieletów drzew derywacyjnych o głębokości $\geqslant 1$;*
2. *$X(S) = \{\sigma(u_1, \ldots, u_k) : \ \sigma \in Sk;\ u_1, \ldots, u_k \in S \cup \Sigma, \sigma(u_1, \ldots, u_k) \notin S,\ k \geqslant 1\}$;*
3. *E jest niepustym skończonym zbiorem szkieletów ze zbioru $(Sk \cup \Sigma)^T_\$$ domkniętym na \$–prefiksy ze względu na zbiór S;*
4. *T jest skończoną funkcja $(E\#(S \cup X(S))) \to \{0, 1\}$, gdzie $T(s) = 1$ wtedy i tylko wtedy, gdy s jest opisem strukturalnym poszukiwanej gramatyki G.*

(S, E, T) można przedstawić za pomocą dwuwymiarowej tablicy:

1. Wiersze oznaczone elementami zbioru $(S \cup X(S))$.
2. Kolumny oznaczone elementami zbioru E.
3. Wartość w komórce o współrzędnych (s, e), gdzie $s \in (S \cup X(S))$, $e \in E$, jest równa $T(s\#e)$.
4. Niech $s \in (S \cup X(S))$, wtedy $row(s)$ jest wektorem złożonym z wartości $T(s\#e)$, dla wszystkich $e \in E$.

	T	e E
		$\vdots$
S	$s \ldots$	$1 (= T(s\#e))$
$X(S)$	s_1	

Tabela 2. Tablica obserwacyjna

Definicja 25. *Tablica obserwacyjna (S, E, T) jest* ***domknięta*** *wtedy i tylko wtedy, gdy:*

$$\forall t \in X(S)\ \exists s \in S(row(t) = row(s)).$$

Definicja 26. *Tablica obserwacyjna* (S, E, T) *jest* ***spójna*** *wtedy i tylko wtedy, gdy:*

$$\forall s_1, s_2 \in S, \sigma \in Sk, u_1, \ldots, u_k \in S \cup \Sigma$$

$$[(row(s_1) = row(s_2)) \Rightarrow (row(\sigma(u_1, \ldots, u_{i-1}, s_1, u_i, \ldots, u_{k-1}))$$

$$=$$

$$row(\sigma(u_1, \ldots, u_{i-1}, s_2, u_i, \ldots, u_{k-1})].$$

4.6. Konstrukcja skończonego automatu drzewiastego przy użyciu tablicy obserwacyjnej

Algorytm LA używa tablicy obserwacyjnej do stworzenia odpowiedniego deterministycznego automatu drzewiastego.

Definicja 27. *Niech* (S, E, T) *będzie domkniętą i spójną tablicą obserwacyjną. Możemy zdefiniować skończony automat drzewiasty* $A(S, E, T)$ *nad* $Sk \cup \Sigma$ *taki, że:*

1. $Q = \{row(s) : s \in S\}$ *jest zbiorem stanów automatu* A*;*
2. $F = \{row(s) : s \in S \wedge T(s) = 1\}$ *jest zbiorem stanów akceptujących;*
3. $\delta(row(s_1), \ldots, row(s_k)) = row(\sigma(s_1, \ldots, s_k))$ *oraz* $\delta(a) = a$*, dla* $a \in \Sigma$ *jest funkcją przejścia automatu* A*.*

4.7. Algorytm LA

Algorytm LA działa na podstawie pozytywnych i negatywnych danych na temat szkieletów drzew derywacyjnych. Konstruuje początkową tablicę, następnie dąży do jej uspójnienia i domknięcia stosując zapytania o należenie. Gdy tablica stanie się domknięta i spójna, konstruuje z niej automat drzewiasty na sposób opisany wcześniej. Wysuwa zapytanie o ekstensjonalną równoważność skonstruowanego automatu i automatu reprezentującego poszukiwany zbiór szkieletów. Jeśli otrzymuje odpowiedź pozytywną, to kończy pracę podając na wyjściu skonstruowany automat. Jeśli odpowiedź nauczyciela jest negatywna, to otrzymany kontrprzykład dodawany jest do tablicy obserwacyjnej, po czym algorytm wraca do poleceń uspójnienia i domknięcia tablicy. Poniżej znajduje się ścisły opis algorytmu LA.

```
begin
S := ∅;
E := $;
G := ({S}, Σ, ∅, S)
Zapytaj o poprawność G
if (odpowiedź=TAK) then
   RETURN G
```

```
else dodaj kontrprzykład t i wszystkie jego poddrzewa o
głębokości przynajmniej 1 do S
Stosując zapytania o należenie utwórz tablicę obserwacyjną
(S, E, T):
   kolumna etykietowana $
   wiersze etykietowane kontrprzykładem t i wszystkimi jego
   poddrzewami o głębokości ⩾ 1
   komórka o współrzędnych (s, $) etykietowana wartościami T(s),
   dla każdego s ∈ S
repeat
  while ((S, E, T) nie jest domknięta lub nie jest spójna) do
     { if ((S, E, T) nie jest spójna) then
        { znajdź s_1, s_2 ∈ S, e ∈ E, u_1, ..., u_{k-1} ∈ S ∪ Σ
          oraz i ∈ ℕ takie, że:
          row(s_1) = row(s_2) i
          T(e#σ(u_1, ..., u_{i-1}, s_1, u_i, ..., u_{k-1}))
          ≠ T(e#σ(u_1, ..., u_{i-1}, s_2, u_i, ..., u_{k-1}))
          dodaj (e#σ(u_1, ..., u_{i-1}, $, u_i, ..., u_{k-1})) do zbioru E
          rozszerz T do E#(S ∪ X(S))
          używając zapytań o należenie
        }
       if ((S, E, T) nie jest domknięta) then
        { znajdź s_1 ∈ X(S) takie, że:
          dla dowolnego s ∈ S row(s_1) ≠ row(s)
          dodaj s_1 do zbioru S
          rozszerz T do E#(S ∪ X(S))
          używając zapytań o należenie
        }
     }
   G:=G(A(S, E, T))
   Zapytaj o poprawność G
   if odpowiedź=⟨0, kontrprzykład t⟩ then
     { dodaj kontrprzykład t i wszystkie jego poddrzewa
       o głębokości ⩾ 1 do S
       rozszerz T do E#(S ∪ X(S))
       używając zapytań o należenie.
     }
until odpowiedź=TAK
RETURN G
end.
```

Warto zaznaczyć, że dla dowolnej gramatyki bezkontekstowej G_U algorytm LA zatrzymuje się podając na wyjściu gramatykę bezkontekstową strukturalnie równoważną gramatyce G_U.

Co więcej, jeśli n jest liczbą stanów minimalnego automatu drzewiastego dla zbioru opisów strukturalnych G_U, zaś m jest górnym ograniczeniem długości kontrprzykładów, to całkowity czas działania algorytmu LA jest wielomianowy względem n i m (Sakakibara, 1990).

Dyskusja Powyższy model jest pierwszym, który daje efektywną możliwość wyuczalności całej klasy gramatyk bezkontekstowych. Wcześniejsze próby dotyczyły pewnych podklas języków bezkontekstowych (Crespi–Reghizzi, 1972) lub były nieefektywne (Levy Joshi, 1978), (Fass, 1983). Poniżej zastanowimy się, czy model wyuczalności gramatyk bezkontekstowych opisany w tym rozdziale może mieć znaczenie w eksplikacji naturalnego procesu uczenia się.

Zaletą omawianego modelu jest to, że nie zakłada on „wrodzonej" wiedzy ucznia o symbolach nieterminalnych. Nie zakłada również, że symbole nieterminalne są uczniowi *explicite* podawane. Cechy te sprawiają, że w ograniczonym zakresie model Sakakibary odpowiada naturalnemu procesowi nabywania umiejętności językowych. Dziecko najpierw uczy się budować poprawnie ciągi, dużo później zaś dowiaduje się co jest nazwą a co zdaniem. Alternatywą mogłoby być przypuszczenie, że symbole nieterminalne (kategorie gramatyczne) są człowiekowi wrodzone. Jest to jednak założenie bardzo ryzykowne i zbyt daleko idące. Okazuje się bowiem, iż istnieje możliwość przyswojenia gramatyki bezkontekstowej bez uprzedniej znajomości jej symboli nieterminalnych.

Algorytm LA uczy się gramatyki bezkontekstowej w postaci strukturalnej. Nie operuje ciągami, a raczej termami zbudowanymi z symboli terminalnych i symboli funkcyjnych wiążących pewną liczbę argumentów. Zakłada więc wcześniejsze wyuczenie się ciągów terminalnych, z których będzie można później korzystać budując zdania. To założenie również wydaje się zgodne z naturalnym procesem uczenia się języka. Najpierw dziecko przyswaja poszczególne słowa ostensywnie wiążąc je z przedmiotami, dopiero później buduje konstrukcje składniowe. Ponadto eliminacja symboli nieterminalnych z procesu uczenia się języka sprzyja adekwatności modelu. Wydaje się, że wiedza jaki konkretnie nieterminalny symbol reprezentuje dany ciąg nie jest dziecku udostępniana.

Właśnie ten strukturalny aspekt uczenia się gramatyk bezkonstekstowych rodzi pewną trudność przy porównywaniu z naturalnym procesem przyswajania języka. Model Angluin, zaprezentowany w rozdziale 3. w przy-

padku zapytań o należenie przypisywał nauczycielowi i uczniowi realistyczne role. W modelu Sakakibary również ten rodzaj zapytań staje się nierealistyczny. Otóż dotyczą one tutaj szkieletów gramatyk. Pytanie stawiane jest w formie: Czy dany szkielet należy do zbioru opisów strukturalnych? Trudno znaleźć jego odpowiednik w naturalnym procesie uczenia się języka.

Rozwiązanie zaproponowane przez Sakakibarę może być użyteczne do opisu związków pomiędzy uczeniem się semantyki a uczeniem się składni. Przyswajanie szkieletów drzew derywacyjnych umożliwia przyporządkowanie wyrażeniom językowym struktury, która pozwala wyznaczyć znaczenia tych wyrażeń. Zakładając bowiem zasadę kompozycyjności (zob. (Janssen, 2003)) istnieje odpowiedniość pomiędzy strukturą składniową wyrażenia a jego strukturą semantyczną. Zatem szkielety drzew derywacyjnych reprezentują wyrażenia nie tylko pod względem składniowym, lecz również zawierają część informacji o znaczeniu (semantyce) wyrażeń.

5. Podsumowanie

Zaprezentowane w artykule modele wyuczalności języka w różnym stopniu odpowiadają naturalnemu procesowi przyswajania języka przez człowieka. Model identyfikacji w granicy szczególnie dobrze oddaje to, że uczeń nigdy nie dowiaduje się kiedy przyswojona przez niego gramatyka jest „prawidłowa". Inną zaletą metodologiczną modelu Golda jest fakt, że pojęcie identyfikowalności dotyczy określonych klas języków. Założenie to pozwala wnioskować o strukturze języka naturalnego (jego miejscu w hierarchii języków) na podstawie struktury kompetencji językowej.

Model wyuczalności przez zapytania podkreśla rolę nauczyciela w procesie uczenia się. Jej istotność wykazana została pośrednio przez Golda, w dowodzie nieidentyfikowalności nadskończonych klas języków z danych pozytywnych. Eksplikacja pojęcia nauczyciela stwarza możliwość formalnego opisu i modelowania roli nauczyciela w procesie uczenia się. Daje również ciekawą wskazówkę ogólnopoznawczą: tzw. informacja negatywna okazuje się niezbędna dla wielu procesów poznawczych.

Algorytm Sakakibary ukazuje możliwość rozszerzania zasięgu modeli uczenia się oraz wskazuje na możliwość korzystania z drzew derywacyjnych w badaniu wyuczalności języka, co może okazać się użyteczne w modelowaniu uczenia się semantyki.

Ważną cechą wszystkich zaprezentowanych modeli jest to, że nie zakładają one istnienia wrodzonego zrębu wszystkich gramatyk, w sensie gramatyki uniwersalnej postulowanej przez Chomsky'ego. Okazuje się więc, że przyjęcie algorytmicznego modelu kompetencji językowej nie wymaga postulowania

części wspólnej wszystkich języków naturalnych. Ich podobieństwo może być spowodowane przynależnością do takiej klasy, którą wrodzony mechanizm uczenia się jest w stanie identyfikować, na przykład klasy wszystkich języków bezkontekstowych.

Literatura

Angluin, D. (1980): *Inductive inference of formal languages from positive data*. Information and Control 45(1980), 117–135.

——— (1987): *Learning regular sets from queries and counterexamples*. Information and Computation 75/2(1987), 87–106.

Chomsky, N. (1957): *Syntactic structures*, Mouton, 1957.

——— (1965): *Aspects of the theory of syntax*, The M.I.T. Press, 1965.

Cutland, N. (1980): *Computability. An introduction to recursive function theory*, Cambridge University Press, 1980.

Crespi–Reghizzi, S. (1972): *An effective model for grammatical inference*, [w:] Information Processing 71, (B. Gilchrist, red.), Elsevier Science Publishers, 1972, str. 524–529.

Fass, L. F. (1983): *Learning context–free languages from their structured sentences*. SIGACT News 15/3(1983), 24–35.

Gazdar, G., G. K. Pullum (1985): *Computationally relevant properties of natural languages and their grammars*. New Generation Computing 3(1985), 273–306.

Gierasimczuk, N. (2007): *Psychologiczna adekwatność modelu identyfikacji w granicy*. Poznańskie studia z filozofii humanistyki (w druku).

Gold, E. M. (1965): *Limiting recursion*. The Journal of Symbolic Logic 30(1965), 28–48.

——— (1967): *Language identification in the limit*. Information and Control 10(1967), 447–474.

Gordon, P. (1990): *Learnability and Feedback*. Developmental Psychology 26(1990), 217-220.

Hopcroft, J., R. Motwani, J. Ullman (2001): *Introduction to automata theory, languages, and computation*, Addison-Weslley Publishing Company, 2001.

Janssen, T. (2003): *Compositionality*, [w:] Handbook of Logic and Linguistics, (J. van Benthem , A. ter Meulen, red.), Elsevier Science Publishers, 2003, str. 417–473.

Kanazawa, M. (1994): *Learnable Classes of Categorial Grammars*, ILLC Dissertation Series 1994–8.

Kleene, S.C. (1956): *Representation of events in nerve nets and finite automata*, [w:] Automata Studies, (C. E. Shannon , J. McCarthy, red.), Princeton University Press, 1956, str. 3–40.

Levy, L. S., A. K. Joshi (1978): *Skeletal structural descriptions*. Information and Control 39/2(1978), 192–211.

Lyons, J. (1977): *Semantics*, Cambridge University Press, 1977.

Macnamara, J. (1986): *A Border dispute. The place of logic in psychology*, The MIT Press, 1986.

Parekh, Y., V. Honavar (1997): *Learning DFA from simple examples*, [w:] Algorithmic Learning Theory, Proceedings from 8th International Workshop, Sendai, Japan, vol. 1316, Springer–Verlag, 1997, str. 116–131.

Partee, B., A. ter Meulen, R. E. Wall (1993): *Mathematical methods in linguistics*, Kluwer Academic Publishers, 1993.

Pinker, S. (2000): *The language instinct. The new science of language and mind*, Penguin Books, 2000.

Pitt, L. (1989): *Inductive inference, DFA's and computational complexity*, [w:] Analogical and Inductive Inference, Lecture Notes in Artificial Intelligence, 397, Springer–Verlag, 1989, str. 18–44.

Pullum, G. K., G. Gazdar (1982): *Natural languages and context–free grammars*. Linguistics and Philosophy 4, 471–504.

Sakakibara, Y. (1990): *Learning context-free grammars from structural data in polynomial time*. Theoretical Computer Science 75(1990), 223 – 242.

Shieber, S. M. (1985): *Evidence against the context–freeness of natural language*. Linguistics and Philosophy 8(1985), 333–343.

Shoenfield, J. R. (1991): *Recursion theory*, Springer–Verlag, 1991.

Shinohara, T. (1990): *Inductive Inference from Positive Data is Powerful*, [w:] The 1990 Workshop on Computational Learning Theory, San Mateo, California: Morgan Kaufmann, str. 339–351.

Sudkamp, T.A. (1988): *Languages and machines. An introduction to the theory of computer science*, Addison–Wesley Publishing Company, 1988.

Valiant, L. (1984): *A theory of the learnable*. Computation of the ACM 27/11(1984), 1134–1142.