Programowanie

Składnia języków programowania

Dodatkowe notatki do zajęć

Spis treści

1 Elementy teorii języków formalnych

1.1 Motywacja — notacja BNF

Przykład definicji języka przy pomocy notacji BNF:

$$
\begin{array}{rcl}
\langle \text{cyfra} \rangle & ::= & \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9} \\
\langle \text{liczba} \rangle & ::= & \langle \text{cyfra} \rangle \mid \langle \text{liczba} \rangle \langle \text{cyfra} \rangle \\
\langle \text{czynnik} \rangle & ::= & \langle \text{liczba} \rangle \mid \texttt{(} \langle \text{wyrażenie} \rangle \texttt{)} \\
\langle \text{składnik} \rangle & ::= & \langle \text{czynnik} \rangle \\
 & \mid & \langle \text{składnik} \rangle \langle \text{operator multiplikatywny} \rangle \langle \text{czynnik} \rangle \\
\langle \text{wyrażenie} \rangle & ::= & \langle \text{składnik} \rangle \\
 & \mid & \langle \text{wyrażenie} \rangle \langle \text{operator addytywny} \rangle \langle \text{składnik} \rangle \\
\langle \text{operator multiplikatywny} \rangle & ::= & \texttt{*} \mid \texttt{/} \\
\langle \text{operator addytywny} \rangle & ::= & \texttt{+} \mid \texttt{-}
\end{array}
$$

Główny problem: jak w skończony sposób opisywać nieskończone języki.

1.2 Symbole, słowa i języki

Definicja 1 (alfabet). Skończony niepusty zbiór Σ nazywamy *alfabetem*, a jego elementy *literami* lub *symbolami*. Alfabet jest *unarny*, jeśli zawiera tylko jedną literę, *binarny*, jeśli zawiera dwie litery.

Definicja 2 (słowo). Skończony ciąg liter (alfabetu Σ) nazywamy *słowem* (*nad alfabetem* Σ). Ciąg nie zawierający liter nazywamy *słowem pustym* i oznaczamy ϵ. Zbiór wszystkich słów nad alfabetem Σ oznaczamy Σ^*. Zbiór wszystkich niepustych słów nad alfabetem Σ oznaczamy Σ^+.

Definicja 3 (długość słowa). Liczbę liter słowa u nazywamy jego *długością* i oznaczamy $|u|$. Liczbę wystąpień litery $a \in \Sigma$ w słowie u oznaczamy $|u|_a$.

Mamy: $|\epsilon| = 0$ oraz $|u| = \sum_{a\in\Sigma} |u|_a$ dla dowolnego słowa $u \in \Sigma^*$.

Definicja 4 (konkatenacja słów). Napis uw oznacza słowo (zwane *konkatenacją* lub *złączeniem słów* u *i* w) powstałe przez wypisanie wszystkich liter słowa u, a następnie wszystkich liter słowa w. Operację konkatenacji słów będziemy niekiedy oznaczać symbolem „$\cdot$". Napis u^i, gdzie i jest dodatnią liczbą naturalną oznacza słowo powstałe przez złączenie i kopii słowa u ze sobą. Napis u^0 oznacza słowo puste ϵ.

Fakt 1 (własności konkatenacji). Konkatenacja słów jest operacją łączną. Słowo puste jest elementem neutralnym konkatenacji. Jeżeli alfabet zawiera więcej niż jedną literę, to konkatenacja nie jest operacją przemienną.

Struktura $\langle \Sigma^*, \cdot, \epsilon\rangle$ jest więc (przeważnie nieprzemienną) półgrupą z jednością.

Fakt 2 (długość jest homomorfizmem). Funkcje $|\ |$ oraz $|\ |_a$ dla $a \in \Sigma$ są homomorfizmami tej półgrupy w półgrupę $\langle \mathbb{N}, +, 0\rangle$ liczb naturalnych z operacją dodawania i zerem. W przypadku alfabetu jednoliterowego homomorfizmy te są izomorfizmami. Algebra słów nad alfabetem więcej niż jednoliterowym nie jest izomorficzna ze zbiorem liczb naturalnych.

Fakt 3 (alfabet dwuliterowy jest uniwersalny). Niech Σ_2 będzie dowolnym alfabetem dwuliterowym. Dla dowolnego alfabetu Σ istnieje homomorfizm $h : \langle \Sigma^*, \cdot, \epsilon\rangle \to \langle \Sigma_2^*, \cdot, \epsilon\rangle$, który jest różnowartościowy. Ponieważ zbiór $h(\Sigma^*)$ jest zamknięty względem konkatenacji i zawiera słowo puste, to algebra $\langle \Sigma^*, \cdot, \epsilon\rangle$ jest izomorficzna z podalgebrą $\langle h(\Sigma^*), \cdot, \epsilon\rangle$ algebry $\langle \Sigma_2^*, \cdot, \epsilon\rangle$. Powyższej własności nie ma alfabet jednoliterowy.

Wniosek: możemy ograniczyć nasze rozważania do alfabetu dwuliterowego.

Definicja 5 (podsłowo). Jeżeli dla danych słów u i v istnieją słowa w_1 i w_2 takie, że $u = w_1vw_2$, to słowo v nazywamy *podsłowem* słowa u. Podsłowo v jest *właściwe*, jeżeli $v \neq \epsilon$ oraz $v \neq u$. Jeżeli $w_1 = \epsilon$, to podsłowo v nazywamy *prefiksem słowa* u. Jeżeli $w_2 = \epsilon$, to podsłowo v nazywamy *sufiksem słowa* u.

Definicja 6 (język). Dowolny podzbiór zbioru słów (nad alfabetem Σ) nazywamy *językiem* (nad alfabetem Σ).

Zbiór słów Σ^* nad dowolnym alfabetem Σ jest zbiorem nieskończonym przeliczalnym. Rodzina języków nad dowolnym alfabetem jest więc zbiorem mocy kontinuum.

Definicja 7 (operacje na językach). W zbiorze języków wprowadzamy następujące operacje:

$$\begin{aligned} L^n &= \{u_1 \dots u_n \mid u_i \in L\} \\ L^* &= \bigcup_{i=0}^{\infty} L^i \\ L^+ &= \bigcup_{i=1}^{\infty} L^i \\ L_1L_2 &= \{u_1u_2 \mid u_1 \in L_1, u_2 \in L_2\} \end{aligned}$$

dla dowolnych języków $L, L_1, L_2 \subseteq \Sigma^*$.

Operacja L^* bywa nazywana *domknięciem Kleenego języka* L. Na językach wykonujemy też operacje mnogościowe sumy, przekroju, dopełnienia, różnicy i różnicy symetrycznej.

1.3 Procedury mechaniczne (efektywne)

Pozostaniemy przy intuicyjnym rozumieniu pojęcia *procedury mechanicznej* i nie będziemy go formalizować (podobnie jak nie formalizowaliśmy teorii mnogości póki nie natrafiliśmy na paradoksy). Możemy utożsamiać procedury mechaniczne z programami w pewnym języku programowania, takim jak Java, Pascal lub C.

Definicja 8 (język rekurencyjny). Język $L \subseteq \Sigma^*$ jest *rekurencyjny* (*rozstrzygalny*) jeżeli istnieje procedura mechaniczna, która dla dowolnego słowa $u \in \Sigma^*$ odpowiada na pytanie, czy $u \in L$. Zbiór języków rekurencyjnych nad alfabetem Σ będziemy oznaczać $\mathcal{R}(\Sigma)$ lub $\mathcal{R}$, gdy alfabet będzie ustalony.

Definicja 9 (język rekurencyjnie przeliczalny). Język $L \subseteq \Sigma^*$ jest *rekurencyjnie przeliczalny* (*pozytywnie rozstrzygalny*) jeżeli istnieje procedura mechaniczna, która (działając w nieskończoność) wypisuje wszystkie słowa języka L (być może z powtórzeniami) i nie wypisuje słów spoza języka L. Zbiór języków rekurencyjnie przeliczalnych nad alfabetem Σ będziemy oznaczać $\mathcal{E}(\Sigma)$ lub $\mathcal{E}$, gdy alfabet będzie ustalony.

Zbiór języków rekurencyjnie przeliczalnych nad dowolnym alfabetem jest przeliczalny. Ponieważ wszystkich języków jest kontinuum, zatem istnieją języki, które nie są rekurencyjnie przeliczalne.

1.4 Gramatyki generacyjne

Definicja 10 (gramatyka generacyjna). *Gramatyką generacyjną* nazywamy czwórkę $G = \langle \Sigma, V, S, P \rangle$, gdzie Σ i V są dwoma rozłącznymi alfabetami ($\Sigma \cap V = \emptyset$), Σ jest zwany alfabetem *terminalnym*, V zaś — *nieterminalnym*, S jest wyróżnionym symbolem nieterminalnym, zwanym *symbolem startowym*, zaś P jest skończonym zbiorem *produkcji*, tj. par słów $u, w \in (\Sigma \cup V)^*$ zapisywanych w postaci $u \to w$, gdzie słowo u zawiera co najmniej jeden symbol nieterminalny. Słowo nad alfabetem terminalnym Σ nazywamy *słowem terminalnym*. Symbole nieterminalne bywają też nazywane *symbolami pomocniczymi* lub *kategoriami gramatycznymi*.

Definicja 11 (relacja bezpośredniego wyprowadzenia). Na zbiorze słów nad alfabetem $\Sigma \cup V$ wprowadzamy binarną relację $\Rightarrow_G$, zwaną *relacją bezpośredniego wyprowadzenia* lub *relacją wyprowadzenia w jednym kroku* (*z gramatyki* G):

$$\Rightarrow_G \quad = \quad \{(uxw, uvw) \mid (x \to y) \in P,\ u, w, x, y \in (\Sigma \cup V)^*\}.$$

Definicja 12 (wyprowadzenie słowa). *Wyprowadzeniem słowa w ze słowa u w gramatyce G* nazywamy ciąg słów $u_0, \ldots, u_n \in (\Sigma \cup V)^*$, taki, że $u_0 = u$, $u_n = w$, oraz $u_{i-1} \Rightarrow_G u_i$ dla $i = 1, \ldots, n$. Liczbę n nazywamy *długością wyprowadzenia*. *Wyprowadzeniem słowa w z gramatyki G* nazywamy wyprowadzenie tego słowa z symbolu startowego S tej gramatyki.

Definicja 13 (relacja wyprowadzenia). Słowa u i v są w *relacji wyprowadzenia*, co oznaczamy $u \stackrel{*}{\Rightarrow}_G v$, jeżeli istnieje w gramatyce G wyprowadzenie słowa v ze słowa u.

Relacja wyprowadzenia jest zwrotnym i przechodnim domknięciem relacji bezpośredniego wyprowadzenia, tj. jest najmniejszą zwrotną i przechodnią relacją binarną na słowach ze zbioru $(\Sigma \cup V)^*$ zawierającą relację $\Rightarrow_G$.

Definicja 14 (język generowany przez gramatykę). Język *generowany przez gramatykę* G, który oznaczamy $\mathfrak{L}(G)$, jest zbiorem słów terminalnych wyprowadzalnych z symbolu startowego S tej gramatyki:

$$\mathfrak{L}(G) \quad = \quad \{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow}_G w\}.$$

Słowo należy więc do języka generowanego przez gramatykę wtedy i tylko wtedy, gdy istnieje jego wyprowadzenie z tej gramatyki.

Zbiory gramatyk generacyjnych i języków przez nie generowanych są przeliczalne. Ponieważ języków nad dowolnym alfabetem jest kontinuum, to istnieją języki, które nie są generowane przez żadne gramatyki.

Definicja 15 (równoważność gramatyk). Mówimy, że gramatyki G_1 i G_2 nad tym samym alfabetem są *równoważne*, jeżeli $\mathfrak{L}(G_1) = \mathfrak{L}(G_2)$.

1.5 Hierarchia Chomsky'ego

Definicja 16 (hierarchia Chomsky'ego). Nakładając na postać produkcji gramatyki dodatkowe warunki, wyróżniamy następujące zbiory gramatyk generacyjnych:

Gramatyki kontekstowe (typu 1): każda produkcja ma postać

$$uAw \to uxw, \quad \text{gdzie } u, x, w \in (\Sigma \cup V)^*,\ A \in V, \text{ oraz } x \neq \epsilon.$$

Zauważmy, że żaden język generowany przez taką gramatykę nie zawierałby słowa pustego. Dlatego dopuszczamy dodatkowo jeden wyjątek: gramatyka może zawierać produkcję

$$S \to \epsilon,$$

ale wówczas S nie może wystąpić po prawej stronie żadnej produkcji tej gramatyki. Jeśli dla danego języka istnieje gramatyka kontekstowa, która go generuje, język ten nazywa się *językiem kontekstowym*.

Gramatyki bezkontekstowe (typu 2): każda produkcja gramatyki ma postać

$$A \to x, \quad \text{gdzie } A \in V, \text{ zaś } x \in (\Sigma \cup V)^*.$$

Jeśli dla danego języka istnieje gramatyka bezkontekstowa, która go generuje, język ten nazywa się *językiem bezkontekstowym*.

Gramatyki regularne (typu 3): każda produkcja gramatyki ma postać

$$A \to x, \quad \text{gdzie } A \in V,\ x \in \Sigma \cup (\Sigma V) \cup \{\epsilon\}.$$

Jeśli dla danego języka istnieje gramatyka regularna, która go generuje, język ten nazywa się *językiem regularnym*.

Mówimy też, że dowolna gramatyka generacyjna jest typu 0. Zbiór języków typu i (dla $i = 0, \ldots, 3$) nad alfabetem Σ będziemy oznaczać $\mathcal{L}_i(\Sigma)$ lub $\mathcal{L}_i$, jeżeli alfabet będzie ustalony.

Definicja 17 (inne klasy gramatyk). Poza powyższą klasyfikacją Chomsky'ego wyróżniamy także następujące typy gramatyk:

Gramatyki monotoniczne (nieskracające): dla każdej produkcji $u \to v \in P$ zachodzi $|u| \leq |v|$. Gramatyka monotoniczna może dodatkowo zawierać produkcję

$$S \to \epsilon,$$

ale wówczas S nie może wystąpić po prawej stronie żadnej produkcji tej gramatyki. Zbiór języków generowanych przez gramatyki monotoniczne nad alfabetem Σ oznaczamy $\mathcal{L}_{mon}(\Sigma)$ lub $\mathcal{L}_{mon}$, gdy alfabet jest ustalony.

Gramatyki liniowe: każda produkcja gramatyki ma postać

$$A \to x, \quad \text{gdzie } A \in V, \text{ zaś } x \in \Sigma^* \cup (\Sigma^* V \Sigma^*).$$

Jeśli dla danego języka istnieje gramatyka liniowa, która go generuje, język ten nazywa się *językiem liniowym*. Zbiór języków generowanych przez gramatyki liniowe nad alfabetem Σ oznaczamy $\mathcal{L}_{2\frac{1}{2}}(\Sigma)$ lub $\mathcal{L}_{2\frac{1}{2}}$, gdy alfabet jest ustalony.

Gramatyki prawostronnie liniowe: każda produkcja gramatyki ma postać

$$A \to x, \quad \text{gdzie } A \in V, \text{ zaś } x \in \Sigma^* \cup (\Sigma^* V).$$

Jeśli dla danego języka istnieje gramatyka prawostronnie liniowa, która go generuje, język ten nazywa się *językiem prawostronnie liniowym*. Zbiór języków generowanych przez gramatyki prawostronnie liniowe nad alfabetem Σ oznaczamy $\mathcal{L}_{rlin}(\Sigma)$ lub $\mathcal{L}_{rlin}$, gdy alfabet jest ustalony.

Gramatyki lewostronnie liniowe: każda produkcja gramatyki ma postać

$$A \to x, \quad \text{gdzie } A \in V, \text{ zaś } x \in \Sigma^* \cup (V \Sigma^*).$$

Jeśli dla danego języka istnieje gramatyka lewostronnie liniowa, która go generuje, język ten nazywa się *językiem lewostronnie liniowym*. Zbiór języków generowanych przez gramatyki lewostronnie liniowe nad alfabetem Σ oznaczamy $\mathcal{L}_{llin}(\Sigma)$ lub $\mathcal{L}_{llin}$, gdy alfabet jest ustalony.

1.6 Relacje między klasami języków

Twierdzenie 1. Dla dowolnego alfabetu Σ zachodzi

$$\mathcal{L}_3 = \mathcal{L}_{llin} = \mathcal{L}_{rlin} \subsetneq \mathcal{L}_{2\frac{1}{2}} \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 = \mathcal{L}_{mon} \subsetneq \mathcal{R} \subsetneq \mathcal{L}_0 = \mathcal{E}.$$

Idee dowodu. Równości $\mathcal{L}_3 = \mathcal{L}_{rlin}$ oraz $\mathcal{L}_1 = \mathcal{L}_{mon}$ są łatwe do udowodnienia (por. zadania 11 i 12). Idea dowodu równości $\mathcal{L}_{llin} = \mathcal{L}_{rlin}$ polega na zbudowaniu dla danej gramatyki lewostronnie liniowej $G_l = \langle \Sigma, V_l, S_l, P_l \rangle$ gramatyki prawostronnie liniowej $G_r = \langle \Sigma, V_r, S_r, P_r \rangle$ w której dla każdej produkcji $(X_i \to X_j u) \in P_l$ istnieje produkcja $(X_j \to u X_i) \in P_r$. Dowód równości $\mathcal{L}_0 = \mathcal{E}$ wymaga sprecyzowania definicji zbioru $\mathcal{E}$, pozostawimy go więc bez komentarza.

Inkluzje $\mathcal{L}_3, \mathcal{L}_{rlin}, \mathcal{L}_{llin} \subseteq \mathcal{L}_{2\frac{1}{2}} \subseteq \mathcal{L}_2$ oraz $\mathcal{R} \subseteq \mathcal{E}$ wynikają wprost z definicji. Uzasadnienie inkluzji $\mathcal{L}_2 \subseteq \mathcal{L}_{mon}$ nie jest trudne, ale wymaga pewnych narzędzi, np. *twierdzenia o postaci normalnej Chomsky'ego*, które mówi, że *dla dowolnej gramatyki bezkontekstowej istnieje równoważna jej gramatyka zawierająca wyłącznie produkcje postaci $X \to YZ$ oraz $X \to a$, gdzie X, Y, Z są symbolami nieterminalnymi, zaś a symbolem terminalnym; gramatyka ta może dodatkowo zawierać produkcję $S \to \epsilon$, gdzie S jest symbolem startowym tej gramatyki, ale wówczas S nie może wystąpić po prawej stronie żadnej produkcji.* Dowód tego twierdzenia jest elementarny, ale dosyć żmudny.

Zawieranie $\mathcal{L}_{mon} \subseteq \mathcal{R}$ uzasadniamy następująco: niech będzie dana gramatyka monotoniczna $G = \langle \Sigma, V, S, P \rangle$ i słowo $u \in \Sigma^*$. Rozważmy zbiór W_u ciągów różnowartościowych $\langle S, v_1, \ldots, v_n \rangle$, takich, że $|v_i| \leq |v_{i+1}|$ dla $i < n$. Zbiór W_u jest skończony i zawiera (pośród innych ciągów) wszystkie różnowartościowe wyprowadzenia słów nie dłuższych niż u. Zauważmy, że dla dowolnej gramatyki generacyjnej jeżeli istnieje wyprowadzenie słowa y ze słowa x, przy czym $x \neq y$, to istnieje różnowartościowe wyprowadzenie słowa y ze słowa x. Zatem słowo $u \in \Sigma^*$ należy do języka generowanego przez gramatykę G wtedy i tylko wtedy, gdy w zbiorze W_u istnieje ciąg będący wyprowadzeniem słowa u ze słowa S.

Ostatnia część, to dowód pięciu nierówności. Trzy pierwsze z nich najłatwiej uzasadnić przez wskazanie przykładów. Niech

- $L_{2\frac{1}{2}} = \{0^n 1^n \mid n \in \mathbb{N}\}$; wówczas $L_{2\frac{1}{2}} \in \mathcal{L}_{2\frac{1}{2}}$, ale $L_{2\frac{1}{2}} \notin \mathcal{L}_3$,
- L_2 będzie opisany gramatyką $\langle \{0,1\}, \{S\}, S, \{S \to \epsilon, S \to 0S1, S \to SS\} \rangle$; wówczas $L_2 \in \mathcal{L}_2$, ale $L_2 \notin \mathcal{L}_{2\frac{1}{2}}$,
- $L_1 = \{0^n 1^n 0^n \mid n \in \mathbb{N}\}$; wówczas $L_1 \in \mathcal{L}_1$, ale $L_1 \notin \mathcal{L}_2$.

Budując odpowiednią gramatykę łatwo pokazać, że $L_i \in \mathcal{L}_i$. Wszystkie trzy dowody, że $L_i \notin \mathcal{L}_j$ dla $j < i$ opierają się na tej samej idei, jednak zawierają zbyt wiele szczegółów, by je tu przytaczać.

Języki odróżniające klasy $\mathcal{L}_1$ i $\mathcal{R}$ oraz $\mathcal{R}$ i $\mathcal{E}$ są bardziej skomplikowane. Naszkicujemy konstrukcję pierwszego z nich, opartą na *metodzie przekątniowej*, znanej z dowodu twierdzenia Cantora.

Dowolną gramatykę generacyjną nad ustalonym alfabetem $\Sigma = \{a_i\}_{i=1}^k$ można jednoznacznie opisać podając listy symboli nieterminalnych i produkcji tej gramatyki. Symbol startowy umieszczamy na początku listy symboli nieterminalnych. Elementy tak powstałych list można połączyć w jedno słowo oddzielając je dodatkowym symbolem, np. #. W ten sposób gramatykę $\langle \Sigma, V, V_1, \{u_i \to v_i\}_{i=1}^n \rangle$, gdzie $V = \{V_i\}_{i=1}^m$, możemy przedstawić w postaci słowa

$$\#V_1\#V_2\#\ldots\#V_m\#\#u_1 \to v_1\#\ldots\#u_n \to v_n\#$$

nad alfabetem $\Sigma' = \Sigma \cup V \cup \{\to, \epsilon, \#\}$. Niech a i b będą dwiema różnymi literami alfabetu Σ (alfabet jednoliterowy można rozważyć osobno). Zakodujemy słowa nad alfabetem Σ' przy pomocy słów złożonych z symboli a i b w następujący sposób: i-tej literze alfabetu Σ' przyporządkowujemy słowo ab^i. Jeżeli teraz u jest pewnym słowem nad alfabetem Σ, to istnieje procedura mechaniczna pozwalająca stwierdzić, czy u jest opisem pewnej gramatyki monotonicznej: słowo to powinno składać się wyłącznie z symboli a i b, żadne dwa symbole a nie powinny stać obok siebie, a symbole alfabetu Σ' odczytane na podstawie długości bloków symboli b oddzielonych literami a powinny ułożyć się w poprawny opis gramatyki monotonicznej. Rozważmy język

$$R \quad = \quad \{u \in \Sigma^* \mid u \text{ jest opisem pewnej gramatyki monotonicznej } G \text{ i } u \notin \mathfrak{L}(G)\}.$$

Język R jest rekurencyjny, ponieważ stwierdziliśmy, że dla dowolnego słowa u istnieje procedura mechaniczna, która sprawdza, czy u jest poprawnym opisem gramatyki monotonicznej i jeśli tak, to odtwarza tę gramatykę na podstawie u. Ponadto wiemy, że sprawdzenie, czy dane słowo należy do języka generowanego przez gramatykę monotoniczną również daje się wykonać przy pomocy odpowiedniej procedury

mechanicznej. Mamy więc procedurę mechaniczną, która odpowiada na pytanie, czy dane słowo u należy do języka R. Zauważmy ponadto, że dla dowolnej gramatyki monotonicznej G istnieje słowo u (będące jej opisem) o tej własności, że $u \in \mathfrak{L}(G)$ wtedy i tylko wtedy, gdy $u \notin R$. Zatem dla dowolnej gramatyki monotonicznej G mamy $R \neq \mathfrak{L}(G)$, czyli $R \notin \mathcal{L}_{mon}$.

1.7 Algebry

Definicja 18 (gatunek). Niech $\mathcal{S} \neq \emptyset$ będzie niepustym (przeważnie skończonym) zbiorem *gatunków* (*rodzajów*, ang. *sorts*). Jego elementy zwykle oznaczamy literami a, b itd, niekiedy z indeksami.

Definicja 19 (typ algebraiczny). Skończony niepusty ciąg gatunków $\langle a_1, \ldots, a_n, b\rangle$ dla $n \geq 0$ i $a_1, \ldots, a_n, b \in \mathcal{S}$ nazywamy *typem algebraicznym* (krócej *typem*), oznaczamy σ, τ, ρ itd, niekiedy z indeksami i zapisujemy w postaci $a_1 \times \ldots \times a_n \to b$ dla $n > 0$ oraz b dla $n = 0$. Liczbę n nazywamy *arnością* typu. Zbiór typów algebraicznych oznaczamy $\mathbb{T}_1(\mathcal{S})$.

Definicja 20 (sygnatura). Z każdym typem τ związujemy zbiór Σ^τ *symboli typu* τ. Symbole typu τ oznaczamy f^τ, g^τ itd, niekiedy z indeksami. *Arnością* (*liczbą argumentów*) symbolu nazywamy arność jego typu. Symbole o arności 0, tj. typu $\tau = a \in \mathcal{S}$ nazywamy *stałymi* i oznaczamy c^a, d^a itd, niekiedy z indeksami. Symbole o arności 1 nazywamy *unarnymi*, symbole o arności 2 zaś *binarnymi*. Z każdym gatunkiem $a \in \mathcal{S}$ związujemy (zwykle przeliczalny nieskończony) zbiór $\mathcal{X}^a$ *zmiennych gatunku* a. Zmienne gatunku a oznaczamy x^a, y^a, z^a itd, niekiedy z indeksami. Zakładamy, że zbiory Σ^τ i $\mathcal{X}^a$ są parami rozłączne. Rodzinę $\Sigma = \{\Sigma^\tau\}_{\tau \in \mathbb{T}_1(\mathcal{S})}$ nazywamy *sygnaturą algebraiczną* (krócej *sygnaturą*), rodzinę $\mathcal{X} = \{\mathcal{X}^a\}_{a \in \mathcal{S}}$ zaś *rodziną zmiennych*. Gdy nie prowadzi to do nieporozumień, pomijamy oznaczenie typu lub gatunku i piszemy f, c, x zamiast f^τ, c^a, x^a. Piszemy też $x \in \mathcal{X}$ na oznaczenie faktu, że $x \in \mathcal{X}^a$ dla pewnego $a \in \mathcal{S}$ itp.

Definicja 21 (term). Zbiory *termów* (*wyrażeń*) gatunku $a \in \mathcal{S}$ nad sygnaturą Σ i zbiorem zmiennych $\mathcal{X}$, oznaczane $\mathcal{T}^a(\Sigma, \mathcal{X})$ dla $a \in \mathcal{S}$, definiujemy indukcyjnie:

1. każda zmienna gatunku a jest termem tego gatunku, tj. $\mathcal{X}^a \subseteq \mathcal{T}^a(\Sigma, \mathcal{X})$;
2. jeżeli $t_i \in \mathcal{T}^{a_i}(\Sigma, \mathcal{X})$ dla $i = 1, \ldots, n$ i $n \geq 0$ oraz $f \in \Sigma^{a_1 \times \ldots \times a_n \to b}$, to para złożona z symbolu f i ciągu termów $\langle t_1, \ldots, t_n\rangle$ jest termem gatunku b, tj.
$$\langle f, \langle t_1, \ldots, t_n\rangle\rangle \in \mathcal{T}^b(\Sigma, \mathcal{X});$$
3. każdy term można zbudować używając reguł 1 i 2.

Definicja 22 (algebra wielogatunkowa). Niech $\Sigma = \{\Sigma^a\}_{a \in \mathcal{S}}$ będzie sygnaturą nad zbiorem gatunków $\mathcal{S}$. *Algebrą* (*strukturą algebraiczną*) o sygnaturze Σ nazywamy parę $\mathfrak{A} = \langle \mathcal{A}, \cdot^{\mathfrak{A}}\rangle$ złożoną z rodziny $\mathcal{A} = \{A^a\}_{a \in \mathcal{S}}$ zbiorów zwanych *dziedzinami* (*nośnikami*, *uniwersami*) algebry (po jednym dla każdego gatunku) i odwzorowania $\cdot^{\mathfrak{A}}$ zwanego *interpretacją symboli funkcyjnych*, które każdemu symbolowi sygnatury $f \in \Sigma^{a_1 \times \ldots \times a_n \to b}$ przyporządkowuje funkcję
$$f^{\mathfrak{A}} : A^{a_1} \times \ldots \times A^{a_n} \to A^b.$$

1.8 Składnia konkretna i abstrakcyjna

Składnia abstrakcyjna języka, to jego opis przy pomocy *abstrakcyjnych drzew rozbioru*. Do takiego opisu będziemy używać m. in. algebr termów nad odpowiednią sygnaturą wielogatunkową.

Składnia konkretna języka, to opis zbioru słów, które do niego należą. Do takiego opisu będziemy używać m. in. gramatyk bezkontekstowych.

1.9 Gramatyki bezkontekstowe

Definicja 23 (drzewo wyprowadzenia). *Drzewem wyprowadzenia* (*rozbioru*) słowa $u \in \Sigma^*$ w gramatyce bezkontekstowej $G = \langle \Sigma, V, S, P\rangle$ nazywamy drzewo o wierzchołkach wewnętrznych etykietowanych symbolami nieterminalnymi i liściach etykietowanych symbolami terminalnymi lub symbolem ϵ, którego korzeń ma etykietę S, etykiety liści wypisane od lewej do prawej tworzą słowo u i w którym dla każdego wierzchołka wewnętrznego albo wierzchołek ten ma etykietę X a jego jedyny syn etykietę ϵ i $X \to \epsilon$ jest produkcją gramatyki, albo X jest jego etykietą, $a_1, \ldots, a_k \in \Sigma \cup V$ — etykietami jego potomków w kolejności od lewego do prawego i istnieje w P produkcja $X \to a_1 \cdots a_k$.

Definicja 24 (gramatyka jednoznaczna). Gramatyka bezkontekstowa jest *jednoznaczna*, jeśli dla każdego słowa z języka przez nią generowanego istnieje dokładnie jedno drzewo wyprowadzenia tego słowa. W przeciwnym razie gramatyka jest *niejednoznaczna*.

Definicja 25 (język istotnie niejednoznaczny). Język bezkontekstowy jest *istotnie niejednoznaczny*, jeżeli nie istnieje jednoznaczna gramatyka, która go generuje.

Definicja 26 (gramatyka bezkontekstowa jako algebra wielogatunkowa). Rozważmy gramatykę bezkontekstową $G = \langle \Sigma, V, S, P \rangle$, przy czym $P = \{X_i \to r_i\}_{i=1}^n$. Niech $\mathcal{S} = V$. Produkcji $X_i \to r_i$, gdzie $r_i = u_1 Y_1 u_2 Y_2 \dots u_k Y_k u_{k+1} \in P$, przy czym $u_j \in \Sigma^*$ zaś $Y_j \in V$, przyporządkowujemy symbol funkcyjny $f_i : Y_1 \times \dots \times Y_k \to X_i$. Definiujemy algebrę $\mathfrak{G} = \langle \{L_i\}_{i=1}^n, \cdot^{\mathfrak{G}} \rangle$ nad sygnaturą $\Gamma = \{f_i\}_{i=1}^n$ następująco: $L_i = \mathfrak{L}(\langle \Sigma, V, X_i, P \rangle)$, zaś $f_i^{\mathfrak{G}}(v_1, \dots, v_k) = u_1 v_1 u_2 v_2 \dots u_k v_k u_{(k+1)}$.

Termy stałe nad sygnaturą Γ utożsamiamy z drzewami wyprowadzenia. Interpretacją termu stałego $t \in (\Gamma, \emptyset)$ w algebrze $\mathfrak{G}$ jest słowo wyprowadzane przez odpowiadające mu drzewo wyprowadzenia.

1.10 Gramatyki atrybutowe

Definicja 27 (ścieżka). *Ścieżką* nazywamy skończony ciąg dodatnich liczb naturalnych $\rho \in \mathbb{N}_+^*$. Niech $\mathcal{T}(\Sigma, \mathcal{X})$ będzie algebrą termów i niech $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Ścieżka ρ jest *poprawna w* t jeżeli $\rho = \epsilon$ lub $\rho = i\rho'$, $t = f(t_1, \dots, t_n)$ i ścieżka ρ' jest poprawna w t_i. Zbiór poprawnych ścieżek w termie t oznaczamy $\mathrm{Paths}(t)$.

Definicja 28 (podterm). *Podtermem* termu $t \in \mathcal{T}(\Sigma, \mathcal{X})$ występującym na ścieżce $\rho \in \mathrm{Paths}(t)$ jest t, gdy $\rho = \epsilon$ lub podterm termu t_i występujący na ścieżce ρ', jeśli $t = f(t_1, \dots, t_n)$ i $\rho = i\rho'$. Podterm termu t występujący na ścieżce ρ będziemy oznaczać $t\restriction\rho$.

Definicja 29 (zbiór zmiennych termu). Zbiór $\mathrm{Var}(x)$ zmiennych występujących w termie t definiujemy indukcyjnie:

$$\begin{aligned} \mathrm{Var}(x) &= \{x\}, \quad \text{dla } x \in \mathcal{X}, \\ \mathrm{Var}(f(t_1, \dots, t_n)) &= \bigcup_{i=1}^{n} \mathrm{Var}(t_i). \end{aligned}$$

Definicja 30 (podstawienie). *Podstawienie* jest skończonym zbiorem par zmiennych i termów zapisywanym w postaci

$$\theta = [x_1/t_1, \dots, x_n/t_n],$$

gdzie $x_i \in \mathcal{X}^{a_i}$ i $t_i \in \mathcal{T}^{a_i}(\Sigma, \mathcal{X})$, dla $i = 1, \dots, n$. Gatunek zmiennej x_i musi się zgadzać z gatunkiem termu t_i. Wynik podstawienia $\theta = [x_1/t_1, \dots, x_n/t_n]$ w termie t oznaczamy $t\theta$ i definiujemy indukcyjnie:

$$\begin{aligned} x_i\theta &= t_i, \quad \text{dla } i = 1, \dots, n, \\ y\theta &= y, \quad \text{dla } y \in \mathcal{X},\ y \neq x_i \text{ dla } i = 1, \dots, n, \\ f(s_1, \dots, s_m)\theta &= f(s_1\theta, \dots, s_m\theta). \end{aligned}$$

Niech V będzie zbiorem gatunków, Π — sygnaturą nad zbiorem V i $\mathcal{T}(\Pi, \emptyset)$ zbiorem termów stałych nad sygnaturą Π. (Termy te będziemy utożsamiać ze zbiorem drzew wyprowadzenia z pewnej gramatyki bezkontekstowej).

Niech $\mathcal{S}$ będzie zbiorem gatunków, Γ — sygnaturą nad zbiorem $\mathcal{S}$ i $\mathcal{T}(\Gamma, \emptyset)$ zbiorem termów stałych nad sygnaturą Γ. (Termy te będziemy utożsamiać z abstrakcyjnymi drzewami rozbioru).

Zmierzamy do zdefiniowania mechanizmu przyporządkowywania termom z $\mathcal{T}(\Pi, \emptyset)$ termów z $\mathcal{T}(\Gamma, \emptyset)$ (tj. przyporządkowywania abstrakcyjnych drzew rozbioru drzewom wyprowadzenia z pewnej gramatyki).

Definicja 31 (zbiory atrybutów). Skończony zbiór Attr będziemy nazywać *zbiorem atrybutów*. Zbiór Attr jest rozłączną sumą zbiorów: Inh — *atrybutów dziedziczonych* i Syn — *atrybutów syntezowanych*. Z każdym atrybutem $a \in \mathrm{Attr}$ jest związany pewnien gatunek $\mathcal{S}(a)$. Z każdym gatunkiem $X \in V$ związujemy podzbiór atrybutów $\mathrm{Attr}(X) \subseteq \mathrm{Attr}$. Definujemy też $\mathrm{Syn}(X) = \mathrm{Attr}(X) \cap \mathrm{Syn}$ oraz $\mathrm{Inh}(X) = \mathrm{Attr}(X) \cap \mathrm{Inh}$.

Definicja 32 (wystąpienia atrybutów). Z każdym symbolem $p \in \Pi$ typu $X_1 \times \dots \times X_n \to X_0$ związujemy zbiór zmiennych $\mathrm{Attr}(p) = \bigcup_{i=0}^{n} \{a_i \mid a \in \mathrm{Attr}(X_i)\}$ taki, że a_i jest gatunku a i zmienne a_i dla różnych a oraz i są różne. Zbiór $\mathrm{Attr}(p)$ nazywamy *zbiorem wystapień atrybutów dla symbolu* p.

Definicja 33 (atrybuty wejściowe i wyjściowe). Zbiory *atrybutów wejściowych i wyjściowych* definiujemy następująco:

$$\begin{aligned} \mathrm{In}(p) &= \{a_i \in \mathrm{Attr}(p) \mid (a \in \mathrm{Inh}(X_0) \wedge i = 0) \vee (a \in \mathrm{Syn}(X_i) \wedge i > 0)\}, \\ \mathrm{Out}(p) &= \{a_i \in \mathrm{Attr}(p) \mid (a \in \mathrm{Syn}(X_0) \wedge i = 0) \vee (a \in \mathrm{Inh}(X_i) \wedge i > 0)\}. \end{aligned}$$

Definicja 34 (reguła semantyczna). Niech $p \in \Pi$. *Regułą semantyczną* (dla p) nazywamy parę złożoną ze zmiennej $a_i \in \mathrm{Out}(p)$ i termu $r_{p,i,a} \in T^a(\Gamma, \mathrm{Attr}(p))$. Regułę semantyczną zapisujemy w postaci równości $a_i = r_{p,i,a}$.

Definicja 35 (gramatyka atrybutowa). *Gramatyką atrybutową* nazywamy szóstkę $AG = \langle V, \Pi, \mathcal{S}, \Gamma, \mathrm{Attr}, \mathcal{R}\rangle$, gdzie $\mathcal{R} = \{(a_i = r_{p,i,a}) \mid p \in \Pi, a_i \in \mathrm{Out}(p)\}$ jest zbiorem reguł semantycznych dla wszystkich atrybutów wyjściowych wszystkich symboli z Π.

Definicja 36 (instancje atrybutów). Niech dany będzie term $t \in \mathcal{T}^X(\Pi, \emptyset)$. *Zbiorem instancji atrybutów termu t* nazywamy zbiór parami różnych zmiennych $\{a_\rho \mid \rho \in \mathrm{Paths}(t), a \in \mathrm{Attr}(V(t\restriction\rho))\}$ i takich, że a_ρ jest gatunku a. (Przez $V(s)$ oznaczamy gatunek termu s).

Definicja 37 (wartościowanie atrybutów termu). *Wartościowaniem atrybutów termu t w gramatyce AG* nazywamy najogólniejszy unifikator układu

$$\{(a_i = r_{p,i,a})[a_i/a_{\rho i} \mid a_i \in \mathrm{Attr}(p)] \mid \rho \in \mathrm{Paths}(t), t\restriction\rho = p(t_1, \ldots, t_n), a_i \in \mathrm{Out}(p)\}, \tag{1}$$

przy czym kładziemy $\rho 0 = \rho$.

Definicja 38 (gramatyka atrybutowa niecykliczna). Gramatyka atrybutowa jest *niecykliczna*, jeżeli dla dowolnego termu $t \in \mathcal{T}(\Pi, \emptyset)$ układ (1) jest unifikowalny i jego najogólniejszy unifikator jest stały.

Przykład 1 (Knuth '68). Rozważmy gramatykę bezkontekstową $G = \langle \Sigma, V, Z, P\rangle$, gdzie $\Sigma = \{0, 1, .\}$, $V = \{Z, N, B\}$ oraz

$$\begin{aligned} P = \{ & \\ & Z \to N.N, && (p_1) \\ & N \to NB, && (p_2) \\ & N \to \epsilon, && (p_3) \\ & B \to 1, && (p_4) \\ & B \to 0 && (p_5) \\ \}. & \end{aligned}$$

Gramatyka ta ma opisywać liczby zmiennopozycyjne w zapisie dwójkowym. Z produkcjami tej gramatyki związaliśmy symbole sygnatury Π:

$$\begin{aligned} &p_1 : N \times N \to Z, \\ &p_2 : N \times B \to N, \\ &p_3 : N, \\ &p_4 : B, \\ &p_5 : B. \end{aligned}$$

(Zauważmy, że w istocie sygnatura Π jest po prostu innym zapisem produkcji, w którym pomijamy postać symboli terminalnych). Skojarzoną z gramatyką G algebrą drzew wyprowadzenia jest algebra $\mathcal{T}(\Pi, \emptyset)$. Niech $\mathcal{S} = \{\mathbb{N}, \mathbb{R}\}$ oraz $\Gamma = \{0 : \mathbb{Z}, 0' : \mathbb{R}, - : \mathbb{Z} \to \mathbb{Z}, \mathrm{succ} : \mathbb{Z} \to \mathbb{Z}, + : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, \exp : \mathbb{Z} \to \mathbb{R}\}$. Niech $\mathrm{Syn} = \{l : \mathbb{Z}, v : \mathbb{R}\}$, $\mathrm{Inh} = \{r : \mathbb{Z}\}$ oraz $\mathrm{Attr}(Z) = \{v\}$, $\mathrm{Attr}(N) = \{r, l, v\}$, $\mathrm{Attr}(B) = \{r, v\}$. Zbiór reguł semantycznych jest następujący:

$$\begin{aligned} p_1 : \quad & v_0 = v_1 + v_2 \\ & r_1 = 0 \\ & r_2 = -l_2 \\ p_2 : \quad & v_0 = v_1 + v_2 \\ & l_0 = \mathrm{succ}(l_1) \\ & r_1 = \mathrm{succ}(r_0) \\ & r_2 = r_0 \\ p_3 : \quad & v_0 = 0' \\ & l_0 = 0 \\ p_4 : \quad & v_0 = \exp(r_0) \\ p_5 : \quad & v_0 = 0' \end{aligned}$$

Zbiór reguł semantycznych zapisuje się często łącznie z produkcjami gramatyki bezkontekstowej. Przyjmuje się wówczas nieco inną konwencję numerowania atrybutów: zamiast a_i (atrybut i-tego symbolu nieterminalnego licząc od lewej, ≥ 0) pisze się $a(X_j)$ (atrybut j-tego wystąpienia symbolu nieterminalnego X licząc od lewej, $j \geq 1$). Jeżeli w danej produkcji symbol nieterminalny ma tylko jedno wystąpienie, to jego indeks opuszczamy. Powyższą gramatykę atrybutową zapisujemy w tej konwencji następująco:

$$\begin{array}{rl}
Z \to N.N & v(Z) = v(N_1) + v(N_2) \\
 & r(N_1) = 0 \\
 & r(N_2) = -l(N_2) \\
N \to NB & v(N_0) = v(N_1) + v(B) \\
 & l(N_0) = \mathrm{succ}(l(N_1)) \\
 & r(N_1) = \mathrm{succ}(r(N_0)) \\
 & r(B) = r(N_0) \\
N \to \epsilon & v(N) = 0 \\
 & l(N) = 0 \\
B \to 1 & v(B) = \exp(r(B)) \\
B \to 0 & v(B) = 0'
\end{array}$$

Definicja 39 (postać normalna reguły semantycznej). Reguła semantyczna ma *postać normalną*, jeżeli $\mathrm{Var}(r_{p,i,a}) \subseteq \mathrm{In}(p)$ dla każdego symbolu $p \in \Pi$ i dla każdego atrybutu $a_i \in \mathrm{Out}(p)$.

Definicja 40 (gramatyka atrybutowa czysto syntezowana). Gramatyka atrybutowa jest *czysto syntezowana*, jeżeli ma postać normalną oraz $\mathrm{Inh} = \emptyset$.

Definiowanie atrybutów przy pomocy gramatyki czysto syntezowanej sprowadza się do definiowania poprzez rekursję strukturalną.

Fakt 4. Dowolna gramatyka atrybutowa czysto syntezowana jest niecykliczna.

1.11 Notacje używane do zapisu wyrażeń

Rozważmy składnię abstrakcyjną wyrażeń z operatorami binarnymi:

$$W \quad ::= \quad A \mid W \oplus W,$$

gdzie A opisuje klasę wyrażeń atomowych, zaś $\oplus$ — klasę operatorów binarnych. Wyrażenia takie zapisujemy zwykle w *notacji infiksowej* (czyli jednym z możliwych wariantów składni konkretnej odpowiadającej tej składni abstrakcyjnej jest zapis infiksowy). Notację infiksową można opisać podając następującą regułę zapisywania wyrażeń: aby zapisać wyrażenie złożone z operatora $\oplus$ i ciągu jego dwóch argumentów $\langle e_1, e_2 \rangle$ należy:

- napisać „(”,
- wypisać wyrażenie e_1,
- napisać „$\oplus$”,
- wypisać wyrażenie e_2,
- napisać „)”.

Wadą tej notacji jest duża liczba nawiasów. Dlatego przyjmuje się pewne konwencje opuszczania nawiasów.

Wyrażenie postaci $e_1 \oplus e_2 \otimes e_3$, gdzie $\oplus$ i $\otimes$ są operatorami binarnymi zapisanymi infiksowo, nie jest jednoznaczne, tj. nie wiadomo, czy oznacza $(e_1 \oplus e_2) \otimes e_3$, czy też $e_1 \oplus (e_2 \otimes e_3)$. Aby nadać sens takim wyrażeniom wprowadza się dodatkowe reguły ich rozbioru. Z każdym operatorem związujemy liczbę naturalną, zwaną jego *priorytetem*, oraz ustalamy jego *łączliwość* (*kierunek wiązania*), tj. ustalamy, czy *łączy w lewo*, *w prawo*, czy może *nie jest łączny* w ogóle. Mówimy, że operator o *wyższym* (większym) priorytecie *wiąże silniej*. Algorytm „odtwarzania” nawiasów w wyrażeniu jest następujący: zaczynamy od operatorów o najwyższym priorytecie, jeśli wiążą w lewo, to od lewej strony wyrażenia, jeśli w prawo, to od prawej. Znajdujemy najmniejsze poprawnie zbudowane wyrażenie zawierające dany operator i ujmujemy je w nawiasy („wiążemy” jego argumenty). Następnie powtarzamy czynność dla operatorów o coraz niższych priorytetach. Np. jeśli operator $\oplus$ ma priorytet 3 i łączy w prawo, zaś $\otimes$ ma priorytet 17 i łączy w lewo, to w wyrażeniu $1 \oplus 2 \oplus 3 \otimes 4 \otimes 5 \oplus 6$ zaczynamy od operatora $\otimes$ od lewej strony:

$1 \oplus 2 \oplus (3 \otimes 4) \otimes 5 \oplus 6$, następnie $1 \oplus 2 \oplus ((3 \otimes 4) \otimes 5) \oplus 6$, potem dla operatora $\oplus$ od strony prawej: $1 \oplus 2 \oplus (((3 \otimes 4) \otimes 5) \oplus 6)$, na końcu $1 \oplus (2 \oplus (((3 \otimes 4) \otimes 5) \oplus 6))$. Powyższe reguły nie gwarantują jednoznaczności w przypadku, gdy w wyrażeniu $e_1 \oplus e_2 \otimes e_3$ operatory $\oplus$ i $\otimes$ mają ten sam priorytet, przy czym $\oplus$ wiąże w lewo, zaś $\otimes$ w prawo (lub odwrotnie). Można temu zaradzić albo zabraniając nadawania tego samego priorytetu operatorom wiążącym w różnych kierunkach (np. żądając, by priorytety operatorów wiążących w lewo były nieparzyste, zaś wiążących w prawo — parzyste), lub zakładając, że w takim przypadku wszystkie operatory wiążą np. w lewo. Dla niektórych operatorów możemy przyjąć, że w ogóle nie są łączne. Wówczas dwa takie operatory nie mogą wystąpić w jednym wyrażeniu nie rozdzielone jawnie napisanymi nawiasami.

1.12 Opis wyrażeń w notacji infiksowej przy pomocy gramatyki bezkontekstowej

Niech A będzie symbolem nieterminalnym opisującym (pewne) wyrażenia atomowe, zaś $\{\oplus_1, \ldots, \oplus_n\}$ zbiorem operatorów binarnych o ustalonych priorytetach i kierunkach łączności. Chcemy opisać za pomocą gramatyki bezkontekstowej język wyrażeń zbudowanych z wyrażeń atomowych, wymienionych operatorów binarnych i nawiasów. Następujące produkcje dla symbolu nieterminalnego W (wyrażenia) tworzą najprostszą gramatykę opisującą taki zbiór wyrażeń:

W	$\to$	A	wyrażenie atomowe jest wyrażeniem
W	$\to$	(W)	wyrażenie ujęte w nawiasy jest wyrażeniem
W	$\to$	$W \oplus W$	dwa wyrażenia rozdzielone operatorem binarnym są wyrażeniem
$\oplus$	$\to$	$\oplus_1 \mid \ldots \mid \oplus_n$	

Taka gramatyka jest niejednoznaczna. Chcielibyśmy na podstawie drzewa wyprowadzenia danego wyrażenia móc zbudować jego abstrakcyjne drzewo rozbioru. Aby zapewnić jednoznaczność, gramatykę musimy nieco skomplikować. Bez zmniejszania ogólności możemy przyjąć, że priorytetem operatora $\oplus_i$ jest i. Zbiór symboli nieterminalnych będzie się składał z $n+1$ symboli oznaczanych $W_0, \ldots, W_n$. Język wyprowadzony z symbolu W_i będzie zawierał wyrażenia atomowe, dowolne wyrażenia ujęte w nawiasy i te spośród wyrażeń złożonych, w których operator główny ma priorytet większy niż i. Symbol W_0 będzie zatem symbolem startowym gramatyki, symbol W_n zaś będzie opisywał język złożony wyłącznie z wyrażeń atomowych i dowolnych wyrażeń ujętych w nawiasy:

$$\begin{aligned} W_n &\to A \\ W_n &\to (W_0) \end{aligned}$$

Dla $i = 0, \ldots, n-1$ dodajmy produkcje:

$$\begin{aligned} W_i &\to W_i \oplus_{i+1} W_{i+1}, \quad \text{gdy } \oplus_{i+1} \text{ wiąże w lewo lub} \\ W_i &\to W_{i+1} \oplus_{i+1} W_i, \quad \text{gdy } \oplus_{i+1} \text{ wiąże w prawo,} \end{aligned}$$

oraz produkcję

$$W_i \to W_{i+1}.$$

Powyższy zbiór produkcji zadaje jednoznaczną gramatykę z symbolem startowym W_0 opisującą wyrażenia.

Przykład 2. Wyrażenia złożone z czterech podstawowych działań arytmetycznych, w których priorytety i łączność operatorów są takie, jak przyjęto w matematyce, opisujemy następującym zbiorem produkcji

$$\begin{aligned} W_2 &\to A \\ W_2 &\to (W_0) \\ W_1 &\to W_2 \\ W_1 &\to W_1 \times W_2 \\ W_1 &\to W_1 \ / \ W_2 \\ W_0 &\to W_1 \\ W_0 &\to W_0 + W_1 \\ W_0 &\to W_0 - W_1 \end{aligned}$$

1.13 Wyrażenia regularne

Definicja 41 (wyrażenia regularne). *Wyrażenia regularne* nad alfabetem Σ i zbiorem zmiennych $\mathcal{X}$ opisujemy następującą gramatyką abstrakcyjną:

$$e \quad ::= \quad a \mid X \mid \epsilon \mid \emptyset \mid e_1 + e_2 \mid e_1 e_2 \mid e^*,$$

gdzie $a \in \Sigma$, $X \in \mathcal{X}$. Przy zapisywaniu wyrażeń regularnych przyjmujemy następującą konwencję opuszczania nawiasów: operatory infiksowe łączą w lewo, najsłabiej wiąże „+", najsilniej zaś „*". Niech η będzie interpretacją zmiennych, tj. funkcją $\eta : \mathcal{X} \to \mathcal{P}(\Sigma^*)$. Znaczenie wyrażeń regularnych określamy zadając ich interpretację w zbiorze języków $\mathcal{P}(\Sigma^*)$:

$$\begin{aligned}
[\![a]\!]_\eta &= \{a\}, \quad a \in \Sigma \\
[\![X]\!]_\eta &= \eta(X), \quad X \in \mathcal{X} \\
[\![\epsilon]\!]_\eta &= \{\epsilon\} \\
[\![\emptyset]\!]_\eta &= \emptyset \\
[\![e_1 + e_2]\!]_\eta &= [\![e_1]\!]_\eta \cup [\![e_2]\!]_\eta \\
[\![e_1 e_2]\!]_\eta &= [\![e_1]\!]_\eta [\![e_2]\!]_\eta \\
[\![e^*]\!]_\eta &= [\![e]\!]_\eta^*
\end{aligned}$$

Zbiór wyrażeń regularnych nad alfabetem Σ i zbiorem zmiennych V oznaczamy $\mathfrak{R}(\Sigma, V)$.

Najczęściej rozważa się wyrażenia regularne z pustym zbiorem zmiennych. Wówczas wartościowanie zmiennych w interpretacji opuszczamy.

Twierdzenie 2 (wyrażenia regularne definiują języki regularne). Dla dowolnego języka $L \subseteq \Sigma^*$ język ten jest regularny wtedy i tylko wtedy, gdy istnieje wyrażenie regularne $e \in \mathfrak{R}(\Sigma, \emptyset)$, takie, że $L = [\![e]\!]$.

Napis $[x_i/y_i]_{i=1}^n$ oznacza funkcję określoną na zbiorze $\{x_i\}_{i=1}^n$ o wartościach w zbiorze $\{y_i\}_{i=1}^n$, która wartości x_i przyporządkowuje wartość y_i. Zakładamy przy tym, że $x_i \neq x_j$ dla $x \neq j$. Niech $\eta : X \to Y$ będzie pewną funkcją i niech $\{x_i\}_{i=1}^n \subseteq X$ oraz $\{y_i\}_{i=1}^n \subseteq Y$. Napis $\eta[x_i/y_i]_{i=1}^n$ oznacza taką funkcję, że

$$\eta[x_i/y_i]_{i=1}^n(x) \quad = \quad \begin{cases} y_i & \text{gdy } x = x_i \text{ dla pewnego } i = 1, \ldots, n, \\ \eta(x) & \text{gdy } x \neq x_i \text{ dla wszystkich } i = 1, \ldots, n. \end{cases}$$

Niech Σ będzie alfabetem. Na $(\mathcal{P}(\Sigma^*))^n$ wprowadzamy porządek $\preccurlyeq$ wzorem

$$(U_1, \ldots, U_n) \preccurlyeq (V_1, \ldots, V_n) \iff \forall\, i{\in}\{1, \ldots, n\}\ U_i \subseteq V_i.$$

Definicja 42 (gramatyka EBNF). Niech Σ będzie alfabetem i niech $V = \{X_i\}_{i=1}^n$ będzie zbiorem zmiennych. *Gramatyką EBNF* nazywamy układ równań na zbiorach postaci

$$\{X_i = e_i\}_{i=1}^n,$$

gdzie $e_i \in \mathfrak{R}(\Sigma, V)$. Krotką języków definiowanych przez powyższą gramatykę EBNF nazywamy najmniejszą (w sensie relacji $\preccurlyeq$) krotkę $(L_1, \ldots, L_n) \in (\mathcal{P}(\Sigma^*))^n$, taką, że

$$L_i = [\![e_i]\!]_{[X_i/L_i]}.$$

Fakt 5 (poprawność definicji EBNF). Definicja powyższa jest poprawna, gdyż zbiór uporządkowany $\langle (\mathcal{P}(\Sigma^*))^n, \preccurlyeq \rangle$ jest kratą zupełną, a operator $f((L_i)_{i=1}^n) = ([\![e_i]\!]_{[X_i/L_i]})_{i=1}^n$ jest monotoniczny, posiada więc najmniejszy punkt stały.

Fakt 6 (gramatyki EBNF definiują języki bezkontekstowe). Jeżeli krotka języków $(L_1, \ldots, L_n) \in (\mathcal{P}(\Sigma^*))^n$ jest zadana pewną gramatyką EBNF, to języki $L_1, \ldots, L_n$ są bezkontekstowe. Dla dowolnej gramatyki bezkontekstowej $G = \langle \Sigma, V, S, P \rangle$ istnieje gramatyka EBNF definująca krotkę $(\mathfrak{L}(\langle \Sigma, V, X, P \rangle))_{X \in V}$.

Podobnie jak w przypadku „zwykłych" gramatyk bezkontekstowych, które w praktyce zapisuje się w notacji BNF, gramatyki EBNF również zapisuje się w wygodnej z punktu widzenia praktyki notacji. Zamiast „+" pisze się „|", zamiast „+" używa się dwóch rodzajów nawiasów: $[e]$ oznacza $e + \epsilon$, zaś $\{e\}$

oznacza ee^* (zatem e^* można zapisać jako $[\{e\}]$). Aby odróżnić symbole terminalne od nieterminalnych, te pierwsze ujmuje się w cudzysłowy. Dla przykładu język ze strony 1 opiszemy w notacji EBNF następująco:

$$\begin{array}{rcl} \text{liczba} & = & \{\,\text{"0"} \mid \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"}\,\} \\ \text{czynnik} & = & \text{liczba} \mid \text{"("}\ \text{wyrażenie}\ \text{")"} \\ \text{składnik} & = & \text{czynnik}\,[\{\,(\text{"+"} \mid \text{"}-\text{"})\,\text{czynnik}\,\}] \\ \text{wyrażenie} & = & \text{składnik}\,[\{(\text{"+"} \mid \text{"}-\text{"})\,\text{składnik}\,\}] \end{array}$$

2 Zadania

Zadanie 1. Udowodnij, że funkcje $|\ |$ oraz $|\ |_a$ dla $a \in \Sigma$ są homomorfizmami algebry $\langle \Sigma^*, \cdot, \epsilon\rangle$ słów z konkatenacją i słowem pustym w algebrę $\langle \mathbb{N}, +, 0\rangle$ liczb naturalnych z operacją dodawania i zerem. Wykaż, że w przypadku alfabetu jednoliterowego homomorfizmy te są izomorfizmami. Udowodnij, że algebra słów nad alfabetem więcej niż jednoliterowym nie jest izomorficzna ze zbiorem liczb naturalnych.

Zadanie 2. Niech Σ_2 będzie dowolnym alfabetem dwuliterowym. Udowodnij, że dla dowolnego alfabetu Σ istnieje homomorfizm $h : \langle \Sigma^*, \cdot, \epsilon\rangle \to \langle \Sigma_2^*, \cdot, \epsilon\rangle$, który jest różnowartościowy. Pokaż, że zbiór $h(\Sigma^*)$ jest zamknięty względem konkatenacji i zawiera słowo puste. Pokaż, że powyższej własności nie posiada alfabet jednoliterowy.

Zadanie 3. Udowodnij, że dla dowolnych języków L_1, L_2 i L_3 są prawdziwe następujące równości:

1. $(L_1L_2)L_3 = L_1(L_2L_3)$ (konkatenacja języków jest łączna)
2. $(L_1 \cup L_2)L_3 = L_1L_3 \cup L_2L_3$ i $L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$ (prawa rozdzielności konkatenacji względem sumy mnogościowej)
3. $L_1\{\epsilon\} = L_1$ i $\{\epsilon\}L_1 = L_1$ ($\{\epsilon\}$ jest obustronnym elementem neutralnym konkatenacji)
4. $\emptyset^* = \{\epsilon\}$
5. $(L_1^*)^* = L_1^*$ (domknięcie Kleene'go jest idempotencją)
6. $(L_1^*L_2^*)^* = (L_1 \cup L_2)^*$
7. $(L_1 \cup \{\epsilon\})^* = L_1^*$ i $(\{\epsilon\} \cup L_1)^* = L_1^*$
8. $(L_1L_2 \cup L_1)^*L_1 = L_1(L_2L_1 \cup L_1)^*$

Zadanie 4. Rozważmy gramatykę $G = \langle \Sigma, V, X_0, P\rangle$, gdzie

$$\begin{array}{rcl} \Sigma & = & \{0,1\}, \\ V & = & \{X_0, X_1, X_2\}, \\ P & = & \{\ X_0 \to X_00,\ X_0 \to X_11,\ X_0 \to 0, \\ & & \quad X_1 \to X_20,\ X_1 \to X_01,\ X_1 \to 1, \\ & & \quad X_2 \to X_10,\ X_2 \to X_21\ \}. \end{array}$$

Ciągi zero-jedynkowe traktujemy jako zapisy liczb naturalnych w postaci binarnej, tj. dla $w = w_1 \ldots w_n \in \{0,1\}^*$ przyjmujemy

$$\text{val}(w) \quad = \quad \sum_{i=1}^{n} w_i 2^{n-i}.$$

Udowodnij, że dowolne słowo $w \in \{0,1\}^*$ należy do $\mathfrak{L}(G)$ wtedy i tylko wtedy, gdy val(w) jest podzielne przez 3. *Wskazówka:* pokaż przez indukcję względem długości słowa, że dla dowolnego słowa $w \in \{0,1\}^*$ oraz $i \in \{0,1,2\}$ zachodzi $w \in \mathfrak{L}(G_i)$ wtedy i tylko wtedy, gdy val$(w) \equiv i \bmod 3$, gdzie $G_i = \langle \Sigma, V, X_i, P\rangle$ dla $i = 0, 1, 2$.

Zadanie 5. Słowo x nad alfabetem $\{(,)\}$ jest *ciągiem poprawnie rozstawionych nawiasów*, jeśli bilans$(x) = 0$ i bilans$(y) \geq 0$, dla każdego prefiksu y słowa x, gdzie bilans$(\epsilon) = 0$, bilans$(z() =$ bilans$(z) + 1$, oraz bilans$(z)) =$ bilans$(z) - 1$. Dla przykładu `(((()))())` jest ciągiem poprawnie rozstawionych nawiasów, zaś `(()))(()` nim nie jest. Udowodnij, że gramatyka $G = \langle \{(,)\}, \{S\}, S, \{S \to (S), S \to SS, S \to \epsilon\}\rangle$ generuje zbiór wszystkich ciągów poprawnie rozstawionych nawiasów.

Zadanie 6. Niech $|w|_0$ i $|w|_1$ oznaczają liczbę wystąpień odpowiednio zer i jedynek w słowie $w \in \{0,1\}^*$. Napisz gramatykę generującą język

$$L \quad = \quad \{w \in \{0,1\}^* \ : \ |w|_0 = 2|w|_1 \ \wedge \ \forall v \leq w . |v|_0 \leq 2|v|_1\},$$

gdzie $v \leq w$ oznacza, że v jest prefiksem słowa w. Proszę napisać gramatykę na tyle prostą, by autor rozwiązania mógł w rozsądnym czasie przekonać widownię, że jego gramatyka generuje język, o który nam chodzi. *Wskazówka:* należy wzorować się na zadaniu 5, przy czy teraz definiujemy bilans$(\epsilon) = 0$, bilans$(z0) =$ bilans$(z) - 1$ i bilans$(z1) =$ bilans$(z) + 2$.

Zadanie 7. Udowodnij, że gramatyka $G = \langle \Sigma, V, X_0, P \rangle$, gdzie

$$\begin{aligned} \Sigma &= \{a, b, c\}, \\ V &= \{X_0, X_1, X_2\}, \\ P &= \{\ X_0 \to abc,\ X_0 \to aX_1bc, \\ &\quad\ X_1b \to bX_1,\ X_1c \to X_2bcc \\ &\quad\ bX_2 \to X_2b,\ aX_2 \to aaX_1,\ aX_2 \to aa\ \} \end{aligned}$$

generuje język $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$.

Zadanie 8. Zdefiniuj gramatykę monotoniczną generującą język $L = \{a^{n^2} \mid n \geq 0\}$ nad alfabetem $\Sigma = \{a\}$.

Zadanie 9. Zdefiniuj gramatykę typu 0 generującą język $L = \{a^{2^n} \mid n \geq 0\}$ nad alfabetem $\Sigma = \{a\}$.

Zadanie 10. Zdefiniuj gramatykę typu 0 generującą język $L = \{ww \mid w \in \{a,b\}^*\}$, tj. zbiór wszystkich słów będących dwukrotnym powtórzeniem jakiegoś słowa.

Zadanie 11. Pokaż, że $\mathcal{L}_3 = \mathcal{L}_{rlin}$.

Zadanie 12. Pokaż, że $\mathcal{L}_1 = \mathcal{L}_{mon}$.

Zadanie 13. Zaprojektuj algorytm, który dla podanej gramatyki bezkontekstowej rozstrzyga, czy język przez nią generowany jest niepusty.

Zadanie 14. Napisz gramatyki bezkontekstowe opisujące następujące języki nad alfabetem $\Sigma = \{0,1\}$:

1. $\{0^n 1^m 0^n \mid n, m \in \mathbb{N}\}$;
2. $\{0^n 1^n 0^m \mid n, m \in \mathbb{N}\}$;
3. $\{0^n 1^m 0^k \mid n, m, k \in \mathbb{N}, n \leq m\}$;
4. $\{(01)^n 0^{2n} \mid n \in \mathbb{N}\}$;
5. zbiór ciągów zerojedynkowych, w których liczba zer i jedynek jest taka sama;
6. zbiór ciągów zerojedynkowych zawierających dwukrotnie więcej zer niż jedynek.

Zadanie 15. Zdefiniuj gramatykę liniową generującą język wszystkich *palindromów* nad alfabetem $\{\mathtt{a}, \mathtt{b}\}$. Słowo jest palindromem, jeśli przeczytane wspak jest sobie równe. *Informacja:* nie istnieje gramatyka regularna (typu 3) generująca ten język.

Zadanie 16. Napisz gramatyki regularne opisujące następujące języki nad alfabetem $\Sigma = \{0,1\}$:

1. $\{0^{2n} \mid n \in \mathbb{N}\}$;
2. $\{0^n 1^m \mid n, m \in \mathbb{N}\}$;
3. zbiór ciągów zerojedynkowych, które nie zawierają trzech kolejnych jedynek;
4. zbiór ciągów zerojedynkowych, w których liczba zer jest parzysta, a jedynek — dowolna;
5. zbiór ciągów zerojedynkowych, w których liczba zer jest parzysta, a jedynek — nieparzysta;
6. zbiór ciągów zerojedynkowych, w których różnica liczby zer i jedynek jest parzysta.

Czy można napisać gramatyki bezkontekstowe posiadające prostsze zbiory produkcji i opisujące powyższe języki?

Zadanie 17. Udowodnij, że jeżeli Σ jest alfabetem unarnym, to $\mathcal{L}_3(\Sigma) = \mathcal{L}_{2\frac{1}{2}}(\Sigma)$.

Zadanie 18. Wzorując się na szkicu dowodu nierówności $\mathcal{L}_{mon} \neq \mathcal{R}$ pokaż, że nie istnieje procedura mechaniczna odpowiadająca na pytanie, czy dana gramatyka typu 0 generuje język rekurencyjny.

Zadanie 19. Jaki język opisuje gramatyka $G = \langle \Sigma, V, S, P \rangle$, gdzie $\Sigma = \{0, 1\}$, $V = \{S\}$, zaś $P = \{S \to 0S1, S \to 0S, S \to \epsilon\}$?. Wykaż, że jest ona niejednoznaczna. Zdefiniuj jednoznaczną gramatykę opisującą ten sam język.

Niech $\Sigma = \{a, b\}$,

$$\begin{aligned} \text{bilans}(\epsilon) &= 0 \\ \text{bilans}(wa) &= \text{bilans}(w) + 1 \\ \text{bilans}(wb) &= \text{bilans}(w) - 1 \end{aligned}$$

dla $w \in \Sigma^*$ i

$$\begin{aligned} L &= \{w \in \Sigma^* \mid \forall v \leq w.\text{bilans}(v) \geq 0\} \\ G_1 &= \langle \Sigma, \{S\}, S, \{S \to \epsilon, S \to aSbS, S \to aS\} \rangle \\ G_2 &= \langle \Sigma, \{N\}, N, \{N \to \epsilon, N \to aNbN\} \rangle \\ G_3 &= \langle \Sigma, \{N, T\}, T, \{N \to \epsilon, N \to aNbN, T \to \epsilon, T \to aNbT, T \to aT\} \rangle \end{aligned}$$

gdzie $v \leq w$ oznacza, że słowo v jest prefiksem słowa w.

Zadanie 20. Udowodnij, że

1. $\mathfrak{L}(G_1) = L$,
2. $\mathfrak{L}(G_3) = L$,
3. $\mathfrak{L}(G_2) = L \cap \{w \in \Sigma^* \mid \text{bilans}(w) = 0\}$.

Zadanie 21. Udowodnij, że

1. G_1 nie jest jednoznaczna,
2. G_2 jest jednoznaczna,
3. G_3 jest jednoznaczna (*wskazówka*: rozważ dwa drzewa wyprowadzenia pewnego słowa; pokaż że ich korzenie muszą być takie same; dalej przez indukcję względem struktury drzew pokaż, że drzewa te są równe).

Zadanie 22. Z dwóch poprzednich zadań wynika następujący wniosek: $\mathfrak{L}(G_1) = \mathfrak{L}(G_3)$ i G_3 jest jednoznaczna.

Opcjonalna fraza `else` w gramatyce

$$\begin{aligned} \langle \textit{instrukcja} \rangle \quad ::= \quad & \texttt{if } \langle \textit{wyrażenie logiczne} \rangle \texttt{ then } \langle \textit{instrukcja} \rangle \texttt{ else } \langle \textit{instrukcja} \rangle \\ \mid \quad & \texttt{if } \langle \textit{wyrażenie logiczne} \rangle \texttt{ then } \langle \textit{instrukcja} \rangle \\ \mid \quad & \langle \textit{instrukcja prosta} \rangle \end{aligned}$$

powoduje, że gramatyka ta jest niejednoznaczna. Korzystając z powyższego wniosku napisz jednoznaczną gramatykę opisującą ten sam język, w której fraza `else` jest wiązana z najbliższym „wolnym” `if`.

Zadanie 23. Pokaż, że istnieje gramatyka *nieskończenie niejednoznaczna*, tj. taka, że istnieje słowo posiadające nieskończenie wiele drzew wyprowadzenia z tej gramatyki.

Pokaż więcej, że istnieje gramatyka opisująca nieskończony język, taka, że dowolne słowo albo nie ma wcale, albo ma nieskończenie wiele drzew wyprowadzenia w tej gramatyce.

Zadanie 24. Zaprojektuj algorytm, który dla podanej gramatyki bezkontekstowej rozstrzyga, czy język przez nią generowany jest niepusty.

Zadanie 25. Wyprowadzenie $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_n$ z gramatyki bezkontekstowej jest *skrajnie lewe*, jeżeli dla każdego $i < n$ jeśli $u_i = u'_i A u''_i$ oraz $u'_{i+1} = u'_i w u''_i$, gdzie $A \to w$ jest produkcją gramatyki, to u'_i nie zawiera symboli nieterminalnych (innymi słowy przepisaniu zawsze podlega skrajny lewy symbol nieterminalny). Pokaż, że każde słowo ma tyle samo drzew rozbioru i skrajnie lewych wyprowadzeń (innymi słowy gramatyka jest jednoznaczna wtedy i tylko wtedy, gdy każde słowo posiada co najwyżej jedno skrajnie lewe wyprowadzenie).

Zadanie 26 (ciąg dalszy zadania 23). Czy istnieją języki bezkontekstowe, dla których każda gramatyka je opisująca jest nieskończenie niejednoznaczna? Jeśli nie, to czy istnieje algorytm, który dla danej gramatyki G buduje gramatykę G' opisującą ten sam język, jednak taką, że każde słowo posiada tylko skończenie wiele drzew wyprowadzenia?

Zadanie 27. Pokaż, że każdy język bezkontekstowy nie zawierający słowa pustego można opisać gramatyką bezkontekstową w tzw. *postaci Chomsky'ego*, w której wszystkie produkcje są postaci:

$$X \to YZ \qquad \text{lub} \qquad X \to a$$

gdzie X, Y i Z są symbolami nieterminalnymi, zaś a jest symbolem terminalnym.

Zadanie 28. Zaprojektuj algorytm, który dla zadanej gramatyki w postaci Chomsky'ego sprawdza, czy podane słowo $a_1 \dots a_n$ należy do języka opisanego tą gramatyką. Wykorzystaj w tym celu technikę *programowania dynamicznego*. Algorytm powinien wyznaczyć dla każdego podsłowa $a_i \dots a_j$ $(1 \leq i \leq j \leq n)$ zbiór symboli nieterminalnych, z których daje się to słowo wyprowadzić. Jest to łatwe dla podsłów długości 1. Dalej należy postępować indukcyjnie. Słowo należy do języka, jeśli symbol startowy gramatyki należy do zbioru symboli, z których dane słowo daje się wyprowadzić.

Zadanie 29. Udowodnij, że język bezkontekstowy nad alfabetem $\Sigma = \{a, b, c, d\}$:

$$L \quad = \quad \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

jest istotnie niejednoznaczny.

Zadanie 30. Gramatyka jest *nieograniczenie niejednoznaczna*, jeżeli dla każdej liczby $n \in \mathbb{N}$ istnieje słowo posiadające co najmniej n drzew wyprowadzenia. Język jest *nieograniczenie niejednoznaczny*, jeżeli każda gramatyka, która go opisuje jest nieograniczenie niejednoznaczna. Pokaż, że istnieje taki język.

Zadanie 31. Zaprojektuj algorytm, który dla podanej gramatyki bezkontekstowej rozstrzyga, czy język przez nią generowany jest nieskończony.

Zadanie 32. Gramatyka bezkontekstowa $G = \langle \Sigma, V, S, P \rangle$ jest *w postaci operatorowej*, jeżeli każda jej produkcja jest postaci $X \to w$, gdzie $w \neq \epsilon$ i w nie zawiera dwóch sąsiadujących symboli nieterminalnych. Pokaż, że każdy język bezkontekstowy nie zawierający słowa pustego posiada generującą go gramatykę operatorową.

Zadanie 33. Udowodnij, że dowolna gramatyka atrybutowa czysto syntezowana jest niecykliczna. Opisz efektywny algorytm wyznaczania atrybutów wierzchołków drzewa rozbioru dla takiej gramatyki.

Zadanie 34. Napisz czysto syntezowaną gramatykę atrybutową dla wyrażeń opisanych w notacji BNF na stronie 1. Algebra abstrakcyjnych drzew rozbioru jest jednogatunkowa i ma sygnaturę $\Gamma = \{+, -, \times, /, n\}$, gdzie $n \in \mathbb{N}$.

Zadanie 35. Niech $\Sigma = \{1, +, \times, (,)\}$ oraz

$$\begin{aligned} G_1 &= \langle \Sigma, \{W, O\}, W, P_1 \rangle, \\ G_2 &= \langle \Sigma, \{W_0, W_1, W_2\}, W_0, P_2 \rangle, \end{aligned}$$

gdzie

$$\begin{aligned} P_1 &= \{W \to 1, W \to (W), W \to WOW, O \to +, O \to \times\}, \\ P_2 &= \{W_0 \to W_0 + W_1, W_0 \to W_1, W_1 \to W_1 \times W_2, W_1 \to W_2, W_2 \to 1, W_2 \to (W_0)\}. \end{aligned}$$

(Gramatyka G_2 powstała z G_1 przy użyciu „transformacji ujednoznaczniającej" przy założeniu, że oba operatory wiążą w lewo i „$\times$" ma wyższy priorytet, niż „$+$"). Udowodnij, że a) $\mathfrak{L}(G_1) = \mathfrak{L}(G_2)$ i b) gramatyka G_2 jest jednoznaczna.

Zadanie 36. Zdefiniuj gramatykę atrybutową opartą na gramatyce G_2 z poprzedniego zadania, która rozbieranemu wyrażeniu przyporządkowuje atrybut będący słowem nad alfabetem Σ, który jest zapisem tego wyrażenia nie zawierającym zbędnych nawiasów. Dla przykładu atrybutem drzewa wyprowadzenia napisu $((1 \times 1) + (1 + 1))$ powinien być napis $1 \times 1 + (1 + 1)$.

Zadanie 37. Formuły notacji BNF interpretujemy jako równania, w których zmiennymi są języki: nazwy kategorii składniowych to zmienne oznaczające języki, symbole terminalne oznaczają języki jednoelementowe złożone z tych symboli, symbol ϵ oznacza język $\{\epsilon\}$, konkatenacja słów oznacza operację konkatenacji języków, a symbol metaalternatywy „|" — operację sumowania zbiorów. Formalnie, niech Σ będzie pewnym alfabetem, a $\{X_i\}_{i=1}^n$ — zmiennymi oznaczającymi języki. Rozważmy układ równań

$$\{X_i = \bigcup_{j=1}^{n_i} L_{i,j,1} L_{i,j,2} \dots L_{i,j,k_{i,j}}\}_{i=1}^n, \tag{2}$$

gdzie $L_{i,j,k}$ oznacza $\{\epsilon\}$, $\{a\}$, dla $a \in \Sigma$ lub X_i. Na $(\mathcal{P}(\Sigma^*))^n$ wprowadzamy porządek $\preccurlyeq$ wzorem

$$(U_1, \dots, U_n) \preccurlyeq (V_1, \dots, V_n) \iff \forall\, i{=}1, \dots, n\; U_i \subseteq V_i.$$

Udowodnij, że $\preccurlyeq$ jest porządkiem i pokaż, że $\langle (\mathcal{P}(\Sigma^*))^n, \preccurlyeq \rangle$ jest kratą zupełną. *Rozwiązaniem* układu (2) nazywamy najmniejszą w sensie relacji $\preccurlyeq$ krotkę języków spełniającą wszystkie równości tego układu. Pokaż, że każdy układ posiada rozwiązanie. Pokaż, że jeżeli $(U_1, \dots, U_n)$ jest rozwiązaniem układu (2), to $U_i = \mathfrak{L}(\langle \Sigma, \{X_i\}_{i=1}^n, X_i, P \rangle)$, gdzie zbiór P jest zbiorem produkcji odpowiadającym zbiorowi formuł BNF.

Zadanie 38. Udowodnij, że jedynym językiem $L \subseteq \{0,1\}^*$ spełniającym równość

$$L \quad = \quad \{\epsilon\} \cup \{0\} L \{1\}$$

jest język

$$L \quad = \quad \{0^n 1^n \mid n \in \mathbb{N}\}.$$

Zadanie 39. Niech $L_1, L_2 \subseteq \{0,1\}^*$ i niech $\epsilon \notin L_1$. Znajdź język $L \subseteq \{0,1\}^*$ spełniający równość $L = L_1 L \cup L_2$. Wykaż, że jest to jedyne rozwiązanie. Pokaż, że jeżeli $\epsilon \in L_1$, to równość $L = L_1 L \cup L_2$ ma więcej niż jedno rozwiązanie.

Zadanie 40. Oto fragment gramatyki opisującej deklaracje klas w pewnym języku programowania:

⟨class declaration⟩	::=	⟨modifiers⟩ ⟨class declarator⟩
⟨modifiers⟩	::=	⟨modifiers⟩ ⟨modifier⟩
	|	⟨empty⟩
⟨modifier⟩	::=	⟨access modifier⟩
	|	⟨inheritance modifier⟩
	|	⟨storage modifier⟩
⟨access modifier⟩	::=	`public` | `private`
⟨inheritance modifier⟩	::=	`abstract` | `final`
⟨storage modifier⟩	::=	`persistent` | `ephemeral`

Właściwy deklarator może być poprzedzony ciągiem przymiotników. Semantyka statyczna języka zawiera dodatkowo regułę mówiącą, że przymiotniki mogą pojawić się w deklaracji klasy w dowolnej kolejności, lecz może wystąpić co najwyżej jeden przymiotnik z każdej grupy. Powyższa gramatyka nie uwzględnia tego ograniczenia. Uzasadnij, że decyzja twórców języka o przeniesieniu tego ograniczenia do semantyki statycznej jest słuszna, gdyż znacznie upraszcza opis gramatyki języka. Dokładniej, rozważmy następujący problem. Niech $\{a_i, b_i\}$ dla $i = 1, \dots, n$ będą parami różnych liter i niech $\Sigma = \bigcup_{i=1}^n \{a_i, b_i\}$. Udowodnij, że dowolna gramatyka bezkontekstowa generująca język

$$L \quad = \quad \{u \in \Sigma^* \mid |u|_{a_i} + |u|_{b_i} \leq 1\}$$

ma co najmniej $2^{\Theta(n)}$ produkcji.

Zadanie 41. Niech Σ będzie alfabetem jak w poprzednim zadaniu i niech

$$G = \langle \Sigma, \{S_i\}_{i=0}^n, S_0, \{S_0 \to \epsilon\} \cup \{S_0 \to S_0 S_i\}_{i=1}^n \cup \bigcup_{i=1}^n \{S_i \to a_i, S_i \to b_i\} \rangle.$$

Zbuduj gramatykę atrybutową nad gramatyką G, która każdemu wyprowadzeniu przypisuje atrybut będący formułą logiczną mówiącą, że słowo przez nie wyprowadzane należy do języka L z poprzedniego zadania.

Zadanie 42. Niech, podobnie jak w zadaniu 5

$$\text{bilans}(\epsilon) = 0 \quad \text{bilans}(w0) = \text{bilans}(w) + 1 \quad \text{bilans}(w1) = \text{bilans}(w) - 1.$$

Pokaż, że gramatyka $\langle \{0,1\rangle, \{S\}, S, \{S \to 0S1S, S \to \epsilon\}\rangle$ generuje język

$$\{w \in \{0,1\}^* \mid \text{bilans}(w) = 0 \wedge \forall v{\in}\{0,1\}^* \, (v \leq w \Rightarrow \text{bilans}(v) \geq 0)\}.$$

Pokaż, że ta gramatyka jest jednoznaczna.

Zadanie 43. Rozważmy gramatykę bezkontekstową $G = \langle \Sigma, V, S, P\rangle$. Mówimy, że gramatyka jest *lewostronnie rekurencyjna*, jeżeli dla pewnego symbolu $A \in V$ oraz pewnego słowa $u \in (\Sigma \cup V)^*$ jest $A \stackrel{+}{\Rightarrow}_G Au$. Pokaż, że dla dowolnego języka bezkontekstowego istnieje gramatyka, która go generuje i która nie jest lewostronnie rekurencyjna. *Wskazówka:* pokaż, że z każdej gramatyki G, takiej, że dla dowolnego symbolu $A \in V$ jest $(A \to \epsilon) \notin P$ oraz nieprawda, że $A \stackrel{+}{\Rightarrow}_G A$, można usunąć produkcje, które prowadzą do rekursji lewostronnej.

Zadanie 44. Udowodnij fakt 5.

Zadanie 45. Udowodnij fakt 6. Wskazówka: pokaż najpierw, że operator $f((L_i)_{i=1}^n) = (\llbracket e_i \rrbracket_{[X_i/L_i]})_{i=1}^n$ jest ciągły.

Definicja 43 (wyrażenia bezkontekstowe). *Wyrażenia bezkontekstowe* nad alfabetem Σ i zbiorem zmiennych $\mathcal{X}$ opisujemy następującą gramatyką abstrakcyjną:

$$e \quad ::= \quad a \mid X \mid \epsilon \mid \emptyset \mid e_1 + e_2 \mid e_1 e_2 \mid \mu X.e,$$

gdzie $a \in \Sigma$, $X \in \mathcal{X}$. Przy zapisywaniu wyrażeń bezkontekstowych przyjmujemy następującą konwencję opuszczania nawiasów: zasięg kwantyfikatora μ rozciąga się maksymalnie na prawo (zatem ma on najniższy priorytet), operatory infiksowe łączą w lewo, konkatenacja wiąże silniej niż „+". Niech η będzie interpretacją zmiennych, tj. funkcją $\eta : \mathcal{X} \to \mathcal{P}(\Sigma^*)$. Znaczenie wyrażeń bezkontekstowych określamy zadając ich interpretację w zbiorze języków $\mathcal{P}(\Sigma^*)$ w taki sam sposób, jak wyrażeń regularnych, przy czym $\llbracket \mu X.e \rrbracket_\eta$ jest z definicji najmniejszym (w sensie zawierania zbiorów) językiem $L \subseteq \Sigma^*$ takim, że $L = \llbracket e \rrbracket_{\eta[X/L]}$. Zbiór wyrażeń bezkontekstowych nad alfabetem Σ i zbiorem zmiennych V oznaczamy $\mathfrak{CF}(\Sigma, V)$.

Zadanie 46. Udowodnij, że definicja wyrażeń bezkontekstowych jest poprawna.

Zadanie 47. Udowodnij, że język $L \subseteq \Sigma^*$ jest bezkontekstowy wtedy i tylko wtedy, gdy istnieje wyrażenie bezkontekstowe, które go opisuje.