Розділ VIII. Граматики та автомати

У цьому розділі розглянуто головні поняття теорії формальних мов і теорії формальних граматик, показано зв'язок між граматиками й автоматами.

Формальні мови та граматики мають велике значення в побудові й реалізації мов програмування. Скінченні автомати та тісно пов'язані з ними конструкції, як, наприклад, регулярні граматики та регулярні вирази, належать до найважливіших понять інформатики. Різні варіанти скінченних автоматів використовують для опису й аналізу технічних пристроїв, різних систем і процесів, програм і алгоритмів. На базі теорії скінченних автоматів сформовано багато складних концепцій теоретичної інформатики. Ця теорія має чимало застосувань у технічній інформатиці та становить важливу частину теоретичної інформатики.

Ми розглядатимемо скінченні автомати як абстрактні моделі найпростіших пристроїв опрацювання даних. Спосіб викладення орієнтовано передусім на теорію формальних мов.

Лекція 32. Граматики

32.1. Мови

В цьому розділі ми будемо використовувати терміни, схожі за змістом на введені у попередньому розділі – теорії кодування. Серед них: алфавіт, буква, слово, конкатенація. Але деякі з них мають власні звучання в теорії граматик. Так слово тут називається **ланцюжок**.

Нагадаємо, що множину слів або ланцюжків називають **мовою**. Правила, що задають множину ланцюжків (слів, речень), утворюють **синтаксис** мови, а опис множини змістів і відповідність між реченнями та змістами – її **семантику**. Семантика мови залежить від характеру описуваних нею об'єктів, засоби її вивчення для різних типів мов різні. Семантика мови математики — формальні теорії. Дослідження семантики мов програмування стало окремою частиною теоретичного програмування. Спроби точно описувати семантику природних мов стосуються передусім машинного перекладу. Що ж до синтаксису, то його особливості значно менше залежать від призначення мови. Можна сформулювати поняття й методи дослідження синтаксису, які не залежать від змісту та призначення мов. Тому найбільших успіхів математична лінгвістика досягла у вивченні синтаксису, де із середини XX ст. розвинувся спеціальний математичний апарат — теорія формальних породжувальних граматик. Вона дуже важлива як така й ефективна в застосуваннях (мовах програмування, штучному інтелекті, машинному перекладі).

Зосередимо увагу на тому, як можна складати слова в речення, і зазначимо, що множина всіх речень, які мають зміст, утворює мову. Нас будуть цікавити здебільшого формальні мови, такі як мови програмування чи мови, що описують правильні математичні вирази. Спочатку наведемо приклад із природної мови.

Розглянемо речення «молодий нападник забив гол». Проаналізуємо його синтаксис. Розглянемо діаграму на рис. 32.1. Вона означає, що ⟨речення⟩ можна побудувати за допомогою злиття ⟨групи підмета⟩ й ⟨групи присудка⟩, хоча це потребує формального означення. Група підмета складається з ⟨означення⟩ та ⟨підмета⟩, а група присудка – із ⟨присудка⟩ та ⟨додатка⟩. Остаточно отримаємо ⟨означення⟩ «молодий», ⟨підмет⟩ нападник, ⟨присудок⟩ «забив», ⟨додаток⟩ «гол».

Перш ніж увести термінологію та позначення, потрібні для уточнення загальних понять у конкретній ситуації, яка зображена на рис. 32.1, зазначимо основні задачі теорії мов.

Нагадаємо, що для заданого алфавіту V мова L — довільна підмножина множини V*, проте довільні підмножини становлять незначний інтерес. Ми хочемо зосередити увагу на спеціальних мовах, що містять ланцюжки, які завдяки зовнішній інформації про їх семантику можна вважати осмисленими чи добре сконструйованими.

Найцікавіші мови нескінченні й, отже, їх неможливо виписати явно. Потрібно придумати способи породження таких мов; як породжувальну систему можна розглядати граматику G. Сформулюємо дві основні задачі формальної теорії мов.

1. Як за заданою граматикою G (і пов'язаною з нею мовою L) породжувати речення $\alpha \in L$?
2. Як за заданими мовою $L \subset V^*$ та реченням $\alpha \in V^*$ з'ясувати, чи $\alpha \in L$?

Щоб перевірити, чи належить якийсь ланцюжок (речення) мові L, потрібно знати, як граматика G породжує L. Далі опишемо загальні принципи породжу-вальних граматик.

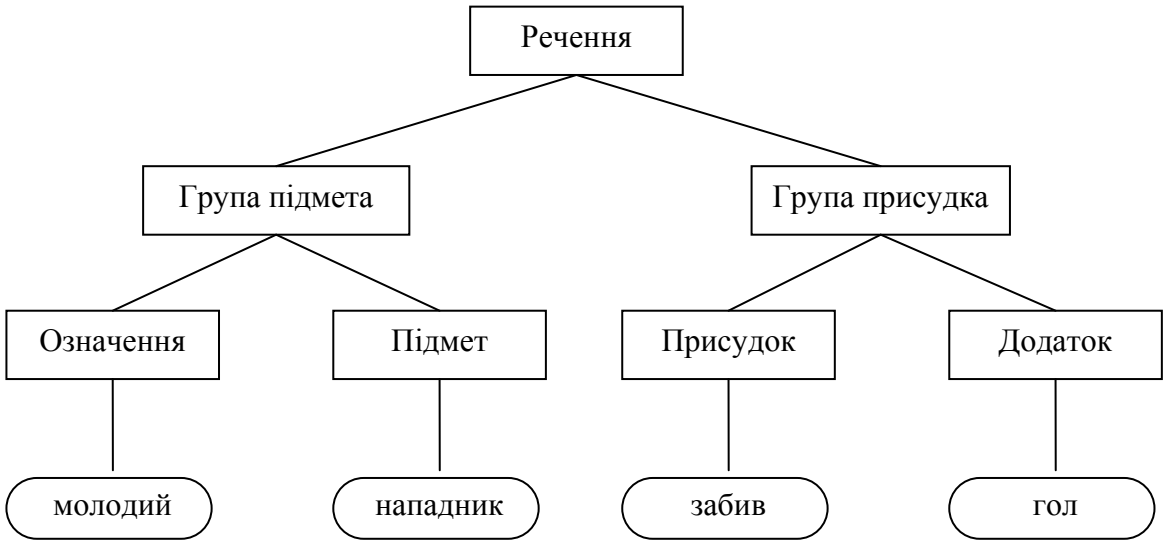

Рис. 32.1.

32.2. Формальні породжувальні граматики

У лінгвістиці природних мов терміни „речення" та „слово" мають різний зміст; тому в математичній лінгвістиці послідовність символів зазвичай називають нейтральним терміном „ланцюжок", а мову, яку розуміють як множину формальних ланцюжків, – формальною.

Означення 32.1. **Формальна породжувальна граматика** G (далі – **граматика** G*)* –це формальна система, задана четвіркою об'єктів G = (V, T, S, P)*,* де V — скінченна непорожня множина, яку називають **алфавітом** (або **словником**); T – її підмножина, елементи якої називають **термінальними** (**основними**) символами; S – **початковий символ** ($S \in V$), P – скінченна множина **продукцій** (або **правил перетворення**) вигляду $\xi \to \eta$, де ξ та η – ланцюжки над алфавітом V. Множину $V \setminus T$ позначають N, її елементи називають **нетермінальними** (**допоміжними**) символами.

Формальні породжувальні граматики часто називають **граматиками з фразовою структурою***,* **граматиками безпосередніх складових**. Термінальні символи часто називають **терміналами***,* а нетермінальні – **нетерміналами**.

У теорії формальних граматик усталилися традиції позначень, яких ми будемо дотримуватись. Символи термінального алфавіту позначають малими латинськими буквами чи цифрами, символи нетермінального алфавіту – великими латинськими буквами, ланцюжки над алфавітом V – грецькими буквами. Довжину ланцюжка α позначають $l(\alpha)$ або $|\alpha|$. Нагадаємо, що множину всіх ланцюжків у алфавіті V позначають V^*.

Нас цікавитимуть ланцюжки, які можуть бути породжені продукціями граматики.

Означення 32.2. Нехай G = (V, T, S, P) – граматика, і нехай $\alpha_0 = \sigma\xi\tau$, $\xi \neq \varepsilon$ (тобто α_0 – конкатенація ланцюжків σ, ξ та τ), $\alpha_1 = \sigma\eta\tau$ - ланцюжки над V. Якщо $\xi \to \eta$ - продукція граматики G, то говорять, що α_1 **безпосередньо виводиться** з α_0, і записують $\alpha_0 \Rightarrow \alpha_1$.

Означення 32.3. Якщо $\alpha_0, \alpha_1, \ldots, \alpha_n$ – ланцюжки над алфавітом V такі, що $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$, то говорять, що α_n **виводиться з** α_0, і використовують запис $\alpha_0 \overset{*}{\Rightarrow} \alpha_n$. Послідовність кроків для отримання α_n з α_0 називають **виведенням**.

Речення української мови з попереднього прикладу можна вивести в граматиці G, де

V = {⟨речення⟩, ⟨група підмета⟩, ⟨група присудка⟩, ⟨означення⟩, ⟨підмет⟩, ⟨присудок⟩, ⟨додаток⟩, молодий, нападник, забив, гол},

T = {молодий, нападник, забив, гол},
S = ⟨речення⟩,
P = {⟨речення⟩ → ⟨група підмета⟩ ⟨група присудка⟩,
⟨група підмета⟩ → ⟨означення⟩ ⟨підмет⟩,
⟨група присудка⟩ → ⟨присудок⟩ ⟨додаток⟩,
⟨означення⟩ → молодий,
⟨підмет⟩ → нападник,
⟨присудок⟩ → забив,
⟨додаток⟩ → гол}.

Ця схема породжує тільки одне речення, тому її можна замінити на

L = {молодий нападник забив гол}.

Якщо ми захочимо розширити мову, щоб увести до неї речення з означеннями «талановитий», «надійний», присудком «віддав», і додатками «пас», «м'яч» (тоді мова L матиме вісімнадцять речень), то це можна зробити відповідним розширенням алфавіту та додаванням лише п'яти продукцій до множини P.

Означення 32.4. **Мовою, породжуваною граматикою** G = (V, T, S, P), називають множину всіх ланцюжків терміналів, які виводяться з початкового символу S. Її позначають L(G). Отже,

$$L(G) = \{\omega \in T^* \mid S \overset{*}{\Rightarrow} \omega\}$$

де T — множина всіх ланцюжків терміналів, включаючи порожній ланцюжок.

Нехай G — граматика з алфавітом V = {S, A, a, b}, множиною терміналів T = {a, b}, початковим символом S і множиною продукцій P = {S → aA, S → b, A → aa}. Знайдемо мову L(G), породжувану цією граматикою.

Із початкового символу S можна вивести ланцюжок aA за допомогою продукції S → aA чи застосувати продукцію S → b, щоб вивести b. З aA, скориставшись продукцією A → aa, можна вивести ланцюжок aaa. Ніяких інших ланцюжків вивести неможливо. Отже, L(G) = {b, aaa}.

Нехай G – граматика з алфавітом V = {S, 0 ,1}, множиною терміналів T = {0, 1}, початковим символом S і множиною продукцій P = {S → 11S, S → 0}. Знайдемо L(G).

Із початкового символу S одержимо 0 (за допомогою другої продукції) чи 11S (за допомогою першої). З 11S можна отримати 110 або 1111S, з 1111S виводиться 11110 або 111111S. Отже, після кожного виведення ми додаємо дві одиниці в кінець ланцюжка чи закінчуємо ланцюжок нулем, тобто L(G) = {0, 110, 11110, 1111110, …} – це множина всіх ланцюжків із парною кількістю тільки 1, після яких (у кінці) обов'язково є один 0.

Ми отримали нескінченну мову L(G), яка складається з нескінченної кількості ланцюжків. Щоб граматика G породжувала нескінченну мову, у множині продукцій має бути принаймні одне рекурсивне правило (наприклад, S → 11S).

Дуже важлива проблема побудови граматики для заданої мови.

Наприклад, знайдемо граматику, яка породжує мову $\{0^m1^m \mid m = 0, 1, 2,...\}$. Потрібні дві продукції, щоб побудувати ланцюжок із якоїсь кількості 0, після яких є така сама кількість 1. Перша продукція збільшує ланцюжок зсередини, додаючи зліва 0, а справа – 1. Друга продукція замінює початковий символ S на порожній ланцюжок ε. Одержимо граматику G = {V, T, S, P}, де V = {0, 1, S}, T= {0, 1}, S – початковий символ, P = {S → 0S1, S → ε}.

Знайдемо породжувальну граматику для мову $\{0^m1^n \mid m, n = 0, 1, 2,...\}$. Наведемо дві такі граматики:

G_1: V = {S, 0, 1}, T = {0, 1}, P = {S → 0S, S → S1, S → ε};
G_2: V = {S, A, 0, 1}, T = {0, 1}, P = {S → 0S, S → 1A, S → 1, A → 1A, A → 1, S → ε}.

Означення 32.5. Граматики G_1 та G_2 називають **еквівалентними**, якщо $L(G_1) = L(G_2)$.

32.3. Типи граматик (ієрархія Хомскі)

Продукція граматики дає змогу заміняти одну послідовність символів іншою. Граматики класифікують за типами продукцій. Розглянемо класифікацію, яку запропонував американський математик і лінгвіст Н. Хомскі (табл. 32.1).

Тип граматики	Обмеження на продукції $\xi \to \eta$
0	немає обмежень
1	$\lvert\xi\rvert \le \lvert\eta\rvert$ чи $\eta=\varepsilon$
2	$\xi = A$, де A – нетермінальний символи
3	$\xi = A$, причому $\eta = aB$ чи $\eta = a$, де A, B – нетермінальні символи, a – термінальний символ, або $S \to \varepsilon$

Табл. 32.1.

Принцип цієї класифікації полягає в тому, що на продукції накладено певні обмеження. Із цих означень випливає, що кожна граматика типу n, де n = 1, 2, 3, являє собою граматику типу n - 1. Граматики типу 2 називають **контекстно вільними**, бо нетермінал *A* в лівій частині продукції $A \to \eta$ можна замінити ланцюжком η в довільному оточенні щоразу, коли він зустрічається, тобто незалежно від контексту. Мову, яку породжує граматика типу 2, називають **контекстно вільною**.

Якщо в множині продукцій P є продукція вигляду $\gamma\mu\delta \to \gamma\nu\delta$ (але не $\mu \to \nu$), $|\mu| \le |\nu|$, то граматику називають граматикою типу 1, або **контекстно залежною**, оскільки μ можна замінити на ν лише в оточенні ланцюжків $\gamma\ldots\delta$, тобто у відповідному контексті. Мову, яку породжує граматика типу 1, називають **контекстно залежною**.

Граматику типу 3 називають **регулярною**. У ній можуть бути лише продукції $A \to aB$, $A \to a$, $S \to \varepsilon$, де A, B – нетермінальні символи, a – термінальний символ. Мову, яку породжує граматика типу 3, називають **регулярною**.

На рис. 32.2 наведено діаграму, яка показує співвідношення між граматиками різних типів.

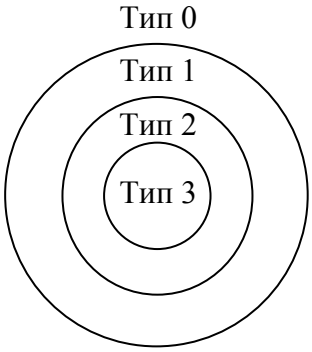

Рис. 32.2.

Мова $\{0^m1^n \mid m, n = 0, 1, 2,...\}$ регулярна, бо її породжує регулярна граматика G_2. Мова $\{0^m1^m \mid m = 0, 1, 2,...\}$ контекстно вільна, оскільки вона породжена граматикою з продукція ми $S \to 0S1$ та $S \to \varepsilon$.

Приклад контекстно вільної мови – мова булевих формул зі змінними a, b, c, яку породжує контекстно вільна граматика G = (V, T, S, P), де V = {S, a, b, c, $\vee$, $\wedge$, $\neg$, (,)}, T = {a, b, c, $\vee$, $\wedge$, $\neg$, (,)}, а множина продукцій

$$P = \{ S \to (S \vee S), S \to (S \wedge S), S \to \neg S, S \to a, S \to b, S \to c\}.$$

32.4. Дерева виведення

У мовах, породжених контекстно вільними граматиками, виведення можна зображати графічно за допомогою орієнтованих кореневих дерев. Їх називають **деревами виведення**, або **деревами синтаксичною розбору**. Кореню цього дерева відповідає початковий символ, внутрішнім вершинам – нетермінальні символи, що зустрічаються у виведенні, листкам –

термінальні символи. Нехай γ – ланцюжок і $A \to \gamma$ – продукція, використана у виведенні. Тоді вершина, що відповідає нетермінальному символу A, має синами вершини, які відповідають кожному символу ланцюжка γ в порядку зліва направо.

Наприклад, визначимо, чи ланцюжок cbab належить мові, породженій граматикою G = (V, T, S, P), де V = {a, b, c, A, B, C, S}, T = {a, b, c}, S – початковий символ, а множина продукцій

$$P = \{ S \to AB,\ A \to Ca,\ B \to Ba,\ B \to Cb,\ B \to b,\ C \to cb,\ C \to b\}.$$

Розв'язати цю задачу можна двома способами.

1. Розбір зверху вниз. Оскільки є лише одна продукція з початковим символом S у лівій частині, то почнемо виведення з $S \Rightarrow AB$. Далі використаємо продукцію $A \to Ca$. Отже, $S \Rightarrow AB \Rightarrow CaB$. Позаяк ланцюжок cbab починається із символів cb, то, використавши продукцію $C \to cb$, одержимо $S \Rightarrow AB \Rightarrow CaB \Rightarrow cbaB$. Завершуємо виведення застосуванням продукції $B \to b$:

$$S \Rightarrow AB \Rightarrow CaB \Rightarrow cbaB \Rightarrow cbab$$

Отже, ланцюжок cbab належить мові L(G).

2. Розбір знизу вверх. Почнемо з ланцюжка cbab, який потрібно вивести. Можна використати продукцію $C \to cb$; отже, $Cab \Rightarrow cbab$. Застосувавши продукцію $A \to Ca$, отримаємо $Ab \Rightarrow Cab \Rightarrow cbab$. Тепер використаємо продукцію $B \to b$; отже, $AB \Rightarrow Ab \Rightarrow Cab \Rightarrow cbab$. Нарешті, застосуємо продукцію $S \to AB$;

$$S \Rightarrow AB \Rightarrow Ab \Rightarrow Cab \Rightarrow cbab$$

Дерево виведення для рядка cbab в граматиці G зображено на рис. 32.3.

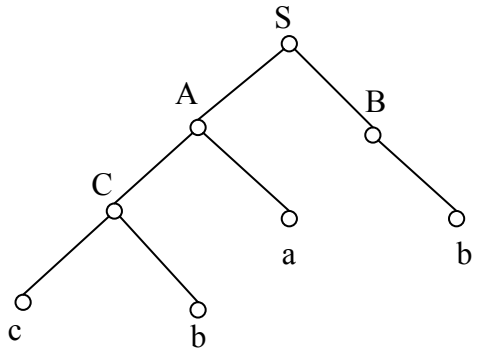

Рис 32.3.

Означення 32.6. Виведення називають **еквівалентними**, якщо їм відповідають однакові дерева.

Перевага дерева виведення порівняно з виведенням – його компактність. Дерево на рис. 32.3 має вісім вершин, а відповідні виведення – 18 або 16 символів.

Взаємно однозначної відповідності між ланцюжками мови L і деревами виведення в граматиці G, яка породжує L, може й не бути.

Означення 32.7. Контекстно вільну граматику G називають **неоднозначною**, якщо існує хоча б один ланцюжок у мові L(G), який має в G більше одного дерева виведення.

Розглянемо граматику G = (V, T, S, P), де V = {S, a, b, c, +, –, *, /, (,)}, T = {a, b, c, +, –, *, /, (,)}, P = {$S \to S+S$, $S \to S–S$, $S \to S*S$, $S \to S/S$, $S \to (S)$, $S \to a$, $S \to b$, $S \to c$}. Ця граматика породжує мову арифметичних виразів, але вона неоднозначна. Вираз a*b + c має в ній два дерева виведення (рис. 32.4).

Неоднозначність мови призводить до незручностей у її використанні. Річ у тому, що дерево виведення – основний засіб інтерпретації ланцюжка; тому синтаксична неоднозначність (тобто наявність декількох дерев виведення) ланцюжка спричинює її семантичну неоднозначність – наявність різних інтерпретацій. Наприклад, для ланцюжка a*b+c з попереднього прикладу різні дерева виведення інтерпретовано як різні способи розставлення дужок: (a*b) + c в першому випадку й a * (b + c) в другому. Це зумовлює різну послідовність операцій і, відповідно, різні результати обчислень. Щоб забезпечити однозначність, достатньо явно ввести дужки після кожної неодномісної операції в мовах

формул (логічних, арифметичних, алгебричних тощо). Граматика з дужками хоча й призводить до надлишкових дужок у формулах, однак дає змогу зменшити кількість нетермінальних символів і тим самим спростити синтаксичну структуру формули, задану її деревом.

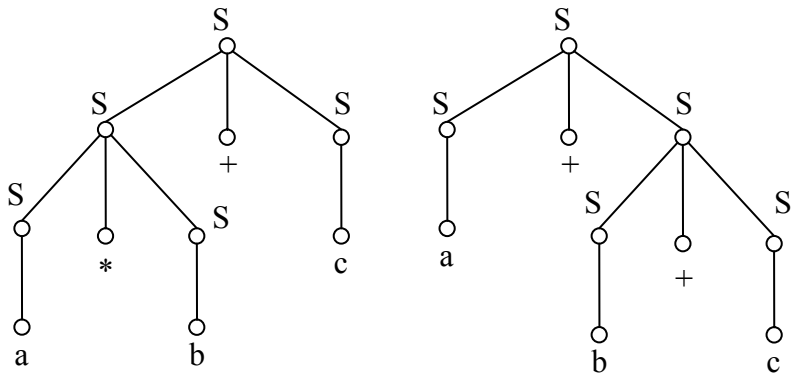

Рис. 32.4.

32.5 Форми Бекуса-Наура

Для граматик типу 2 (контекстно вільних), окрім звичайного, є й інший спосіб подання – **форми Бекуса-Наура**. У лівій частині продукцій граматик типу 2 один символ (нетермінальний). Замість того щоб виписувати окремо всі продукції, можна об'єднати в один вираз продукції з однаковими символами в лівій частині. Тоді замість символу $\rightarrow$ в продукціях використовують символ ::=. Усі нетермінали в цьому разі беруть у трикутні дужки ⟨⟩. Праві частини продукцій в одному виразі відокремлюють одну від одної символом |.

Наприклад, три продукції $A \rightarrow Aa$, $A \rightarrow a$, $A \rightarrow AB$ можна подати одним таким виразом у формі Бекуса-Наура: ⟨A⟩ ::= ⟨A⟩a | a | ⟨A⟩⟨B⟩.