ДМ-2

Лекція 10

Тема 8. Теорія обчислень

Скінченні автомати

План лекції

- Скінченні автомати з виходом
- Скінченні автомати без виходу
- Подання мов
- Що це – мова чи проблема?
- Лема про накачування для регулярних мов
- Лема про накачування для контекстно вільних мов

Скінченні автомати з виходом

Скінченним автоматом називають систему $M=(S, I, O, f, g, s_0)$, у якій S, I, O – скінченні множини, а f: $S\times I\to S$ та g: $S\times I\to O$ – функції, визначені на декартовому добутку $S\times I$. Множину S називають *множиною станів*, I – *вхідним алфавітом*, O – *вихідним алфавітом*, f – *функцією переходів*, g – *функцією виходів*, виділений елемент $s_0\in S$ – *початковим станом*.

Елементи вхідного алфавіту називають *вхідними символами*, або *входами*, а вихідного – *вихідними символами*, або *виходами*. Рівність $f(s_i, x)=s_j$ означає, що в разі входу x автомат, який перебуває в стані s_i, переходить у стан s_j, а рівність $g(s_i, x)=y$, – що в цьому разі на виході з'являється y; тут $s_i, s_j\in S$, $x\in I$, $y\in O$.

Оскільки функції f та g визначено на скінченних множинах, то їх можна задати таблицями. Зазвичай дві таблиці зводять в одну й називають *таблицею станів*, або *автоматною таблицею*. Вона містить значення функції переходів f і функції виходів g для всіх пар (s, x), де $s\in S$, $x\in I$. Таблиця станів задає скінченний автомат.

Приклад 1. Табл. 1 задає функції переходів і виходів для автомата з множиною станів $S=\{s_0, s_1, s_2, s_3\}$ та вхідним і вихідним алфавітами $I = \{0, 1\}$, $O = \{0, 1\}$.

Таблиця 1

Стан	f		g	
	Вхід		Вхід	
	0	1	0	1
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0

Іще один поширений і наочний спосіб задати автомат – за допомогою орієнтованого мультиграфа, який називають *діаграмою станів*. Вершини графа відповідають станам; якщо $f(s_i, x) = s_j$ та $g(s_i, x) = y$ (тут $x \in I$, $y \in O$), то з вершини s_i у вершину s_j веде дуга з позначкою x, y.

Приклад 2. Діаграму станів для автомата, заданого табл. 1, наведено на рис. 1.

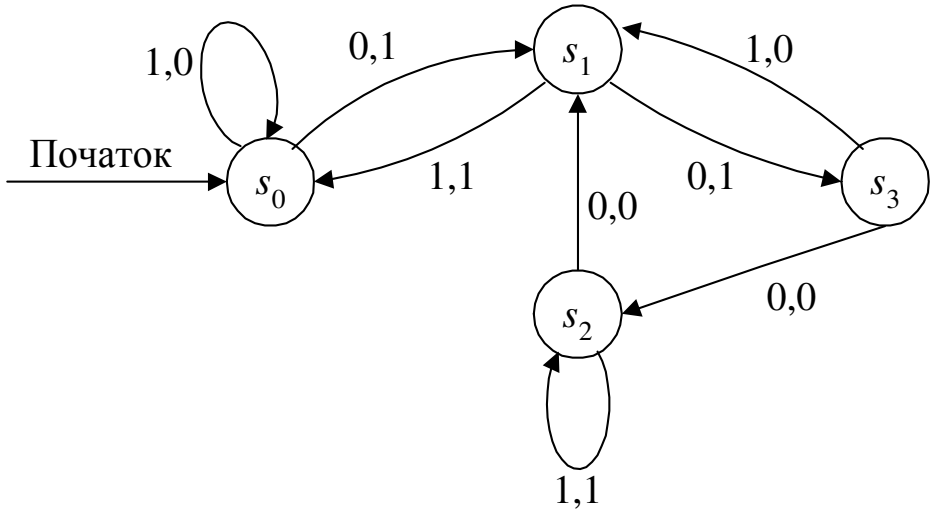

Рис. 1

Розглянемо скінченний автомат M. Кожному вхідному ланцюжку α поставимо у відповідність вихідний ланцюжок ω. Нехай вхідний ланцюжок $\alpha = x_1x_2\ldots x_k$. Тоді під час читання цього ланцюжка автомат спочатку переходить зі стану s_0 у стан s_1, де $s_1 = f(s_0, x_1)$, потім у стан s_2, де $s_2 = f(s_1, x_2)$, і цей процес триває до досягнення стану $s_k = f(s_{k-1}, x_k)$. Зазначимо, що тут x_k – останній символ вхідного ланцюжка. Ця послідовність переходів у нові стани формує вихідний ланцюжок $\omega = y_1y_2...y_k$, де $y_1 = g(s_0, x_1)$ – вихідний символ, який відповідає переходу з s_0 в s_1, $y_2 = g(s_1, x_2)$ – вихідний символ, що відповідає переходу з s_1 в s_2, і так до отримання вихідного символу $y_k = g(s_{k-1}, x_k)$. Загалом $y_j = g(s_{j-1}, x_j)$ для $j = 1, 2, \ldots, k$.

Приклад 3. Знайдемо вихідний ланцюжок, який видає скінченний автомат, діаграму якого зображено на рис. 1, якщо вхідний ланцюжок – 101001.

За допомогою діаграми станів виписуємо послідовності станів і вихідних символів, їх наведено в табл. 2. Отже, автомат видає на виході ланцюжок 011110.

Таблиця 2

Вхід	1	0	1	0	0	1
Стан	s_0	s_0	s_1	s_0	s_1	s_3
Вихід	0	1	1	1	1	0
Новий стан	s_0	s_1	s_0	s_1	s_3	s_1

Розглянуту вище відповідність, яка відображає вхідні ланцюжки у вихідні, називають *автоматним відображенням*, а також *автоматною* (або *обмежено детермінованою*) *функцією*, яка *реалізується автоматом M*. Якщо результат застосування цього відображення до ланцюжка α – вихідний ланцюжок ω, то це позначають $M(\alpha)=\omega$. Кількість символів у ланцюжку α, як завжди, називають довжиною α та позначають $|\alpha|$ чи $l(\alpha)$.

Автоматне відображення має дві властивості.

1. Ланцюжки α та $\omega = M(\alpha)$ мають однакову довжину: $|\alpha|=|\omega|$ (властивість збереження довжини).

2. Якщо $\alpha=\alpha_1\alpha_2$ і $M(\alpha_1\alpha_2)=\omega_1\omega_2$, де $|\alpha_1|=|\omega_1|$, то $M(\alpha_1)=\omega_1$, тобто образ відрізка довжиною l дорівнює відрізку образу з такою самою довжиною.

Властивість 2 означає, що автоматні відображення – це відображення *без випередження*, тобто такі, котрі, перероблюючи ланцюжок зліва направо, «не підглядають уперед»: *i*-та буква вихідного ланцюжка залежить тільки від перших *i* букв вхідного ланцюжка. Приклад відображення **з випередженням** – те, яке ланцюжку $\alpha=x_1x_2...x_k$ ставить у відповідність ланцюжок $\omega = x_k...x_2x_1$; перша буква вихідного ланцюжка тут дорівнює останній букві вхідного ланцюжка. Зазначимо, що ці дві властивості – це не достатні умови автоматності відображення: існують відображення, які задовольняють умови 1 і 2, але не реалізуються в скінченному автоматі.

Розглянемо деякі корисні приклади скінченних автоматів. Ці приклади свідчать, що стани скінченного автомата дають змогу використовувати їх як скінченну пам’ять. Стани можна використовувати для запам’ятовування ситуацій або символів, які читає автомат. Проте через скінченну множину станів скінченні автомати не можна використовувати в деяких важливих застосуваннях.

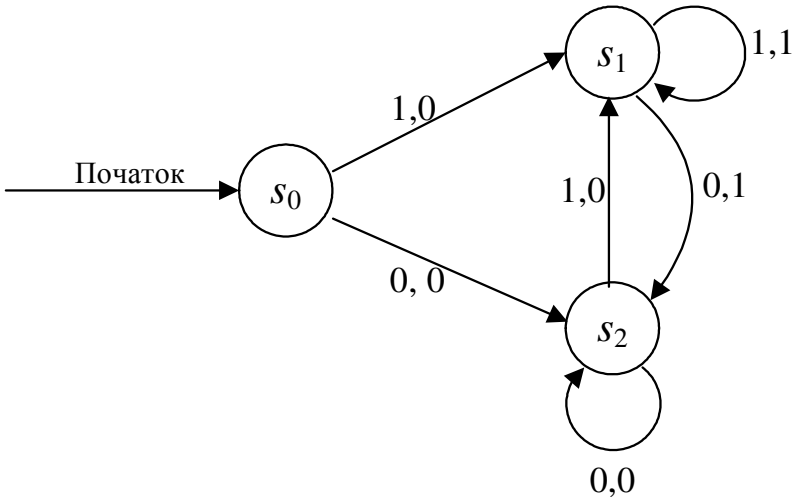

Рис. 2

Приклад 4. Важливий елемент багатьох пристроїв – автомат одиничної затримки. Він видає на виході вхідний ланцюжок, затриманий на одиницю часу. Отже, якщо на вході подано двійковий ланцюжок $x_1x_2...x_k$, то на виході буде ланцюжок $0x_1x_2...x_{k-1}$.

У такого автомата має бути два вхідні символи (нехай 0 і 1) і він має «пам'ятати», який із двох символів 0 або 1 був на вході в попередній момент. Отже, крім початкового стану s_0, потрібно ще два стани: нехай автомат перебуває в стані s_1, якщо попереднім вхідним символом була 1, і в стані s_2 – якщо 0. На виході в початковому стані завжди 0 незалежно від входу. Кожний перехід зі стану s_1 дає на виході 1, а зі стану s_2 – 0. Діаграму станів цього автомата зображено на рис. 2.

Приклад 5. Побудуємо скінченний автомат, який видає на виході 1 тоді й лише тоді, коли на вході останніми трьома символами були 1.

У цього автомата має бути три стани. Початковий стан s_0, його також використовують для запам'ятовування ситуації, коли попередній вхідний символ – 0. Стан s_1 відповідає ситуації, коли попередній символ на вході – 1, а символ перед ним – 0. Стан s_2 запам'ятовує ситуацію двох поспіль вхідних 1. Отже, якщо автомат перейшов у стан s_2, то на вхід подано дві 1 поспіль. Вхід 1 у стані s_2 означає, що це була третя поспіль 1, і на виході з'являється 1. У всіх інших ситуаціях на виході з'являється 0. Діаграму станів цього автомата наведено на рис. 3.

Автомат із прикладу 5 подає мову, оскільки він видає на виході 1 тоді й тільки тоді, коли вхідний ланцюжок (слово) має спеціальні властивості (у цьому прикладі мова складається з ланцюжків нулів і одиниць, які закінчуються трьома 1 поспіль). Подання мов – одне із найважливіших застосувань скінченних автоматів.

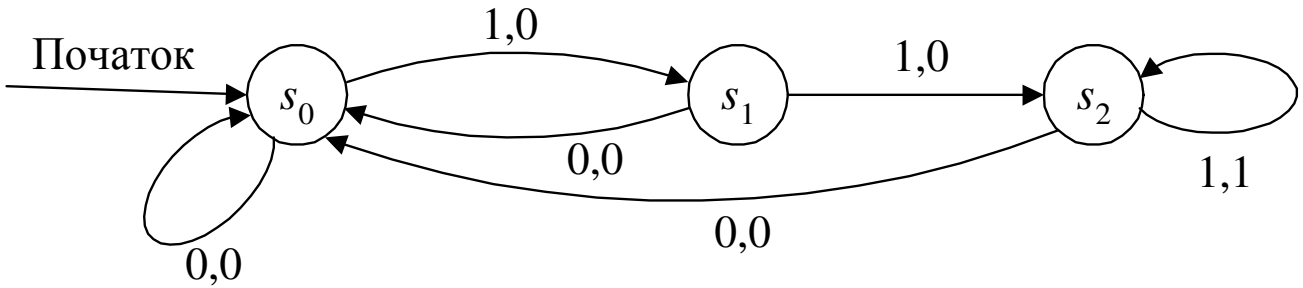

Рис. 3

Автомати, які ми розглянули, називають автоматами Мілі (G. Mealy), уперше їх уведено в 1955 р. Є також інший тип автоматів із виходом – так звані автомати Мура (E. Moore), запроваджені в 1956 р. У цих автоматах вихід визначається лише станом, тобто не залежить від вхідного сигналу.

У прикладі 5 описано, як автомат Мілі можна використати для розпізнавання мови. Проте для цього зазвичай застосовують інший тип автоматів – скінченні автомати без виходу. Такі автомати мають множину заключних (або приймаючих) станів і „приймають” ланцюжок тоді й лише тоді, коли цей ланцюжок переводить автомат без виходу з початкового стану в заключний.

Скінченні автомати без виходу

Одне з найважливіших застосувань скінченних автоматів – розпізнавання (подання) мов, яке має фундаментальне значення в дослідженні й побудові компіляторів для мов програмування. У прикладі 5 описано, як скінченний автомат із виходом може розпізнати мову: він видає на виході 1, якщо вхідний ланцюжок належить мові, і 0 – у протилежному випадку. Проте є інший тип скінченних автоматів, спеціально призначений для розпізнавання мов. Замість виходу ці автомати мають множину заключних станів. Ланцюжок допускається автоматом, якщо він переводить автомат із початкового стану в один із заключних станів.

Скінченним автоматом без виходу називають систему $M = (S, I, f, s_0, F)$, у якій S – скінченна *множина станів*, I – скінченний *вхідний алфавіт*, $f: S\times I\to S$ – *функція переходів*, визначена на декартовому добутку $S\times I$, $s_0\in S$ – *початковий стан*, $F\subset S$ – множина *заключних* (або *приймаючих*) *станів*.

Елементи вхідного алфавіту, як і раніше, називають *вхідними символами* чи *входами*.

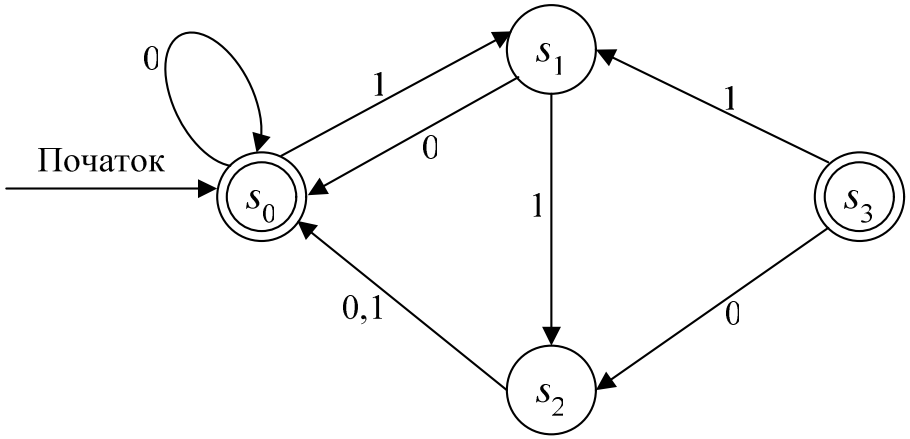

Рис. 4

Таблиця 3.

Стан	f	
	Вхід	
	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1

Скінченні автомати без виходу можна задавати таблицями станів або діаграмами станів. Заключні стани на діаграмі зображають подвійними кружечками. Зазначимо, що в автоматах без виходу є тільки входи (символи вхідного алфавіту I), тому на дугах діаграми записують тільки їх.

Оскільки далі ми будемо розглядати лише скінченні автомати без виходу, то називатимемо їх *скінченними автоматами*, чи просто *автоматами*.

Приклад 6. На рис. 4 наведено діаграму станів для скінченного автомата $M = (S, I, f, s_0, F)$, де $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_0, s_3\}$, а функцію переходів задано табл. 3. Оскільки обидва входи 0 і 1 переводять автомат зі стану s_2 в стан s_0, то замість двох дуг від s_2 до s_0 використано лише одну дугу, на якій написано два входи: 0 і 1.

Функцію переходів f можна розширити й означити її для всіх пар станів і ланцюжків. У такому разі нехай $\alpha = x_1x_2\ldots x_k$ – ланцюжок із множини I^*. Тоді $f(s_1, \alpha)$ – стан, обчислений із використанням послідовних символів ланцюжка α зліва направо як вхідних символів, починаючи зі стану s_1. Процес відбувається так: $s_2 = f(s_1, x_1)$; $s_3 = f(s_2, x_2)$, … Нарешті вважаємо, що $f(s_1, \alpha) = f(s_k, x_k)$.

Говорять, що ланцюжок α *допускається* (*приймається*) скінченним автоматом $M = (S, I, O, f, s_0, F)$, якщо він переводить початковий стан s_0 у заключний стан; це означає, що стан $f(s_0, \alpha)$ – елемент множини F.

Мова, що *розпізнається* автоматом M, позначають $L(M)$, – це множина всіх ланцюжків, які допускаються автоматом M. Два автомати називають *еквівалентними*, якщо вони розпізнають одну й ту саму мову.

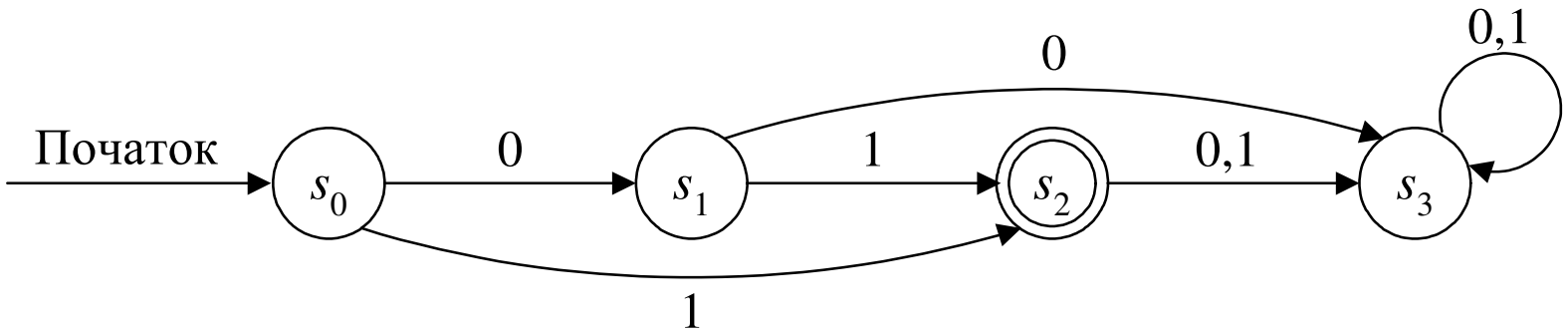

Рис. 5

Приклад 7. Знайдемо мову, яка розпізнається скінченним автоматом M_1 із діаграмою станів, зображеною на рис. 5. Автомат M_1 має тільки один заключний стан s_2. Тільки два ланцюжки переводять s_0 в s_2: 1 і 01. Отже, $L(M_1)=\{1,01\}$.

Приклад 8. Знайдемо мову, яка розпізнається скінченним автоматом M_2 із діаграмою станів, зображеною на рис. 6. Заключні стани автомата M_2 – s_0 та s_3. Стан s_0 переводять у самого себе порожній ланцюжок λ, а також ланцюжки з довільної кількості нулів: 0, 00, 000, ... Ланцюжки, які переводять стан s_0 в s_3, складаються з якоїсь кількості нулів, після яких є 10 і довільний ланцюжок β з нулів та одиниць. Отже,

$$L(M_2) = \{0^n, 0^n10\beta \mid n=0,1,2,\ldots;\ \beta \text{ – довільний ланцюжок}\}.$$

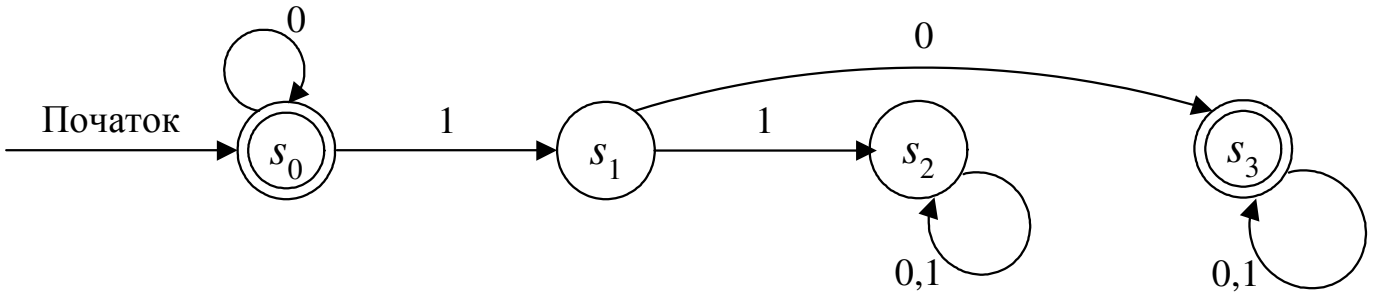

Рис. 6

Розглянуті скінченні автомати без виходу називають *детермінованими*, бо для кожної пари „стан – вхід" існує єдиний наступний стан, заданий функцією переходів. Є й інший тип автоматів без виходу – це недетерміновані автомати. У них може бути декілька можливих наступних станів для кожної пари «стан – вхід».

Недетермінованим скінченним автоматом без виходу називають систему $M = (S, I, f, s_0, F)$, у якій S – скінченна *множина станів*, I – скінченний *вхідний алфавіт*, f – *функція переходів*, яка кожній парі «стан – вхід» ставить у відповідність множину станів, $s_0 \in S$ – *початковий стан*, $F \subset S$ – множина *заключних* (або *приймаючих*) *станів*.

Зазначимо, що єдина відмінність між недетермінованим і детермінованим автоматами – тип значень функції переходів *f*. Для недетермінованого автомата це множина станів (вона може бути й порожньою), а для детермінованого – один стан. Недетермінований скінченний автомат задають таблицею або діаграмою станів. У таблиці для кожної пари «стан – вхід» записують множину всіх можливих наступних станів (якщо вона порожня, то ставлять прочерк). У діаграмі переходів проводять дуги з кожного стану до всіх можливих наступних станів, на цих дугах записують входи, які спричинюють переходи одного стану в інший.

Приклад 9. На рис. 7 і в табл. 4 подано відповідно діаграму та таблицю станів недетермінованого автомата.

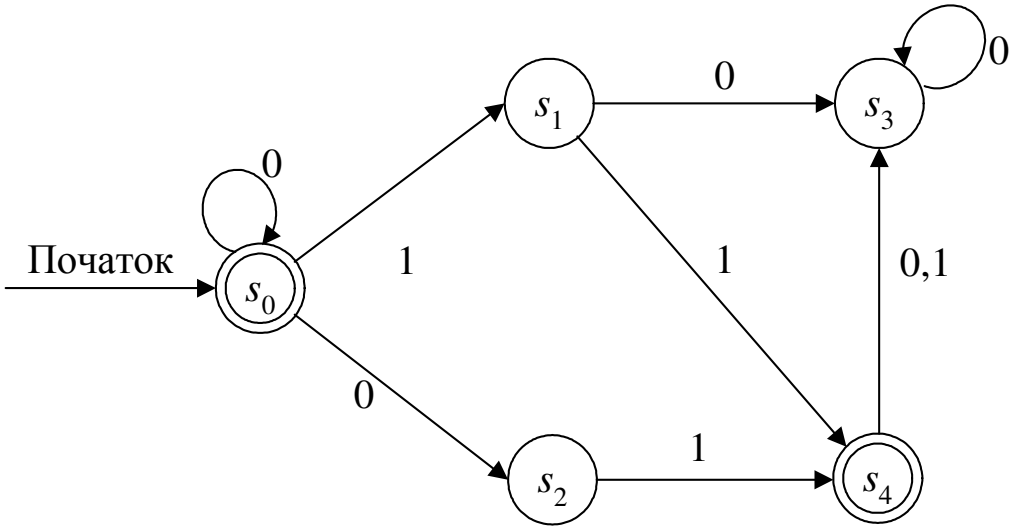

Рис. 7

Таблиця 4

Стан	*f*	
	Вхід	
	0	1
s_0	s_0, s_2	s_1
s_1	s_3	s_4
s_2	–	s_4
s_3	s_3	–
s_4	s_3	s_3

Тепер визначимо, як недетермінований скінченний автомат допускає (приймає) ланцюжок $\alpha=x_1x_2\ldots x_k$. Перший вхідний символ x_1 переводить стан s_0 в множину S_1, яка може містити більше одного стану. Наступний вхідний символ x_2 переводить **кожний** зі станів множини S_1 у якусь **множину** станів, і нехай S_2 – об'єднання цих множин. Цей процес продовжують, вибираючи на кожному кроці всі стани, отримані з використанням поточного вхідного символу й усіх станів, одержаних на попередньому кроці. Ланцюжок α *допускається* (*приймається*) недетермінованим скінченним автоматом, якщо в множині станів, отриманій з початкового стану s_0 під дією ланцюжка α, є заключний стан.

Мова, що *розпізнається* недетермінованим скінченним автоматом – це множина всіх ланцюжків, які допускаються автоматом *M*.

Приклад 10. Знайдемо мову, яка розпізнається недетермінованим скінченим автоматом із рис. 7.

Оскільки s_0 – заключний стан і вхід 0 переводить його в себе, то ланцюжки λ, 0, 00, 000, 0000, ... допускаються цим автоматом. Стан s_4 також заключний, і нехай s_4 є в множині станів, що досягаються зі стану s_0 із ланцюжком α на вході. Тоді ланцюжок α допускається. Такими ланцюжками є 0^n01 та 0^n11, де n=0,1,2,… . Інших заключних станів немає, тому мова, яка розпізнається цим недетермінованим автоматом, така: $\{0^n, 0^n01, 0^n11 | n=0, 1, 2, \ldots\}$.

Зазначимо, що коли мова розпізнається недетермінованим автоматом, то вона також розпізнається детермінованим автоматом.

Теорема 1. Якщо мова *L* розпізнається недетермінованим скінченним автоматом M_0, то *L* розпізнається також детермінованим скінченним автоматом M_1.

Подання мов

Спочатку розглянемо деякі фундаментальні питання, що стосуються множин ланцюжків. Нехай A та B – підмножини множини V^*, де V – алфавіт. *Конкатенацією множин A та B* (позначають AB) називають множину всіх ланцюжків вигляду $\alpha\beta$, де α – ланцюжок із множини A, а β – ланцюжок із множини B.

Приклад 11. Нехай $A=\{0, 11\}$ і $B=\{1, 10, 110\}$. Знайдемо множини AB та BA. Множина AB містить усі конкатенації ланцюжка з A та ланцюжка з B, тому

$AB = \{01, 010, 0110, 111, 1110, 11110\}$.

Множина BA містить усі конкатенації ланцюжка з B і ланцюжка з A:

$BA = \{10, 111, 100, 1011, 1100, 11011\}$.

Загалом $AB \neq BA$.

Використавши означення конкатенації двох множин ланцюжків, можна означити степінь множини ланцюжків A^n для $n = 0, 1, 2, \ldots$. Це роблять за допомогою рекурсії:

$A^0=\{\lambda\}$;

$A^{n+1}=A^nA$ для $n = 0, 1, 2, \ldots$.

Тут, як і раніше, λ – порожній ланцюжок.

Приклад 12. Нехай $A = \{1, 00\}$. Знайдемо A^n для $n = 1, 2, 3$.

Маємо $A^0 = \{\lambda\}$ та $A^1 = A^0A = \{\lambda\}A = \{1, 00\}$. Щоб знайти A^2, візьмемо конкатенацію всіх пар елементів з A. Це дасть $A^2 = \{11, 100, 001, 0000\}$. Для знаходження A^3 потрібно взяти конкатенації елементів з A^2 й A:

$A^3 = \{111, 1100, 1001, 10000, 0011, 00100, 00001, 000000\}$.

Нехай A – підмножина множини V^*. *Замиканням Кліні множини* A (позначають A^*) називають множину всіх тих ланцюжків, які можна утворити конкатенацією довільної кількості ланцюжків з A. Отже, $A^* = \bigcup_{k=0}^{\infty} A^k$.

Приклад 13. Знайдемо замикання Кліні для множин $A = \{0\}$, $B = \{0, 1\}$ та $C = \{11\}$. Замикання Кліні множини A – конкатенація ланцюжка 0 із собою довільну скінченну кількість разів. Отже, $A^*=\{0^n \mid n=0,1,2,\ldots\}$. Замикання Кліні множини B – конкатенація довільної кількості ланцюжків, де кожний ланцюжок являє собою 0 або 1. Це не що інше, як множина всіх ланцюжків над алфавітом $V=\{0,1\}$; тобто, $B^* = V^*$. Нарешті, замикання Кліні множини C – конкатенація ланцюжка 11 із собою довільну скінченну кількість разів. Отже, C^* – множина ланцюжків із парною кількістю одиниць: $C^*=\{1^{2n} \mid n=0,1,2,\ldots\}$.

Скінченні автомати можна використовувати для подання (розпізнавання) мов (множин ланцюжків). Але які саме множини розпізнаються скінченними автоматами? Цю проблему вперше розв'язав американський математик Кліні (S. Kleene) 1956 р. Він довів, що скінченний автомат, який розпізнає множину ланцюжків, існує тоді й лише тоді, коли її можна побудувати з порожньої множини (що не містить жодного ланцюжка), множини, що містить тільки порожній ланцюжок, і множин, що містять один односимвольний ланцюжок, за допомогою використання в довільному порядку операцій конкатенації, об'єднання та замикання Кліні. Множини ланцюжків, які можуть бути побудовані таким способом, називають *регулярними*.

Перш ніж точно означити регулярні множини, необхідно дати *означення регулярних виразів*.

Регулярний вираз над множиною I рекурсивно означають так:

- символ $\varnothing$ – регулярний вираз;
- символ $\boldsymbol{\lambda}$ – регулярний вираз;
- символ $\boldsymbol{x}$ – регулярним вираз, якщо $x \in I$;

- вирази **(AB)**, **(A∪B)** та **A*** регулярні, якщо **A** та **B** регулярні.

Кожний регулярний вираз задає множину ланцюжків, визначену за такими правилами:

- ∅ – порожню множину, тобто таку, що не містить жодного ланцюжка;
- **λ** – множину {λ}, що містить тільки порожній ланцюжок;
- ***x*** – множину {*x*}, яка має один ланцюжок, що складається з одного символу *x*;
- **(AB)** – конкатенацію множин, поданих виразами **A** та **B**;
- **(A∪B)** – об'єднання множин, поданих виразами **A** та **B**;
- **A*** – замикання Кліні множини, поданої виразом **A**.

Множину, подану регулярним виразом, називають *регулярною множиною*.

Відтепер регулярні вирази будемо використовувати для опису регулярних множин. Це означає таке: посилаючись на регулярну множину **A**, ми будемо мати на увазі регулярну множину, подану регулярним виразом **A**.

Нижче наведено приклад того, як регулярні вирази використовують для того, щоб подавати регулярні множин.

Приклад 14. Покажемо, з яких ланцюжків утворено регулярні множини, задані регулярними виразами **10***, **(10)***, **0∪01**, **0(0∪1)*** та **(0*1)***. Описи регулярних множин, поданих цими регулярними виразами, наведено в табл. 5.

Таблиця 5

Вираз	Ланцюжки, з яких складається відповідна регулярна множини
10*	1 і після неї довільна кількість 0 (або нулів немає)
(10)*	Довільна кількість повторень 10 (включно з порожнім ланцюжком)
0∪01	Ланцюжок 0 і ланцюжок 01
0(0∪1)*	Довільний ланцюжок, який починається з 0
(0*1)*	Довільний ланцюжок, який закінчується 1 (або порожній).

Теорема 2 (теорема Кліні). Для того щоб множина була регулярною, необхідно й достатньо, щоб її розпізнавав скінченний автомат.

Доведення – див підручник «Дискретна математика», автори Ю. Нікольський, В. Пасічник, Ю. Щербина. Видавництво «Магнолія-2006», Львів, 2010, 2013, 2016, 2019, 2021 рр. Стор. 363 – 365.

Є тісний зв'язок між регулярними множинами та регулярними граматиками. Його розкрито в теоремі 3.

Теорема 3. Регулярна множина й лише вона породжується регулярною граматикою.

Доведення. Спочатку доведемо, що множина, породжена регулярною граматикою, регулярна. Нехай $G = (V, T, S, P)$ – регулярна граматика, яка породжує множину $L(G)$. Доведемо, що $L(G)$ – регулярна множина. Для цього побудуємо недетермінований скінченний автомат $M=(S, I, f, s_0, F)$, який розпізнає множину $L(G)$. Множина станів S автомата M містить стан s_A для кожного нетермінального символу A граматики G та заключний стан s_F. Початковий стан s_0 відповідає початковому символу S граматики G.

Переходи в автоматі M утворимо за допомогою продукцій граматики G таким способом. Якщо в граматиці є продукція $A \to a$, то в автоматі M для вхідного символу a має бути перехід зі стану s_A до заключного стану s_F. За наявності продукції $A \to aB$ для вхідного символу a утворюють перехід зі стану s_A до стану s_B. Множина F заключних станів автомата M містить стан s_F і, якщо $S \to \lambda$ – продукція граматики G, то ще й стан s_0.

Тепер неважко переконатись, що мова $L(M)$, яка розпізнається побудованим автоматом M, збігається з мовою $L(G)$, породженою граматикою G. Отже, $L(M) = L(G)$. Це можна зробити, перевіривши ланцюжки, які переводять автомат у заключний стан. Отже, ми побудували недетермінований скінченний автомат, який розпізнає множину, породжену регулярною граматикою G. Тепер регулярність цієї множини випливає з теореми 2.

Перш ніж завершити доведення, наведемо приклад описаної вище методики побудови скінченного автомата, який розпізнає ту саму множину, що й породжує регулярна граматика.

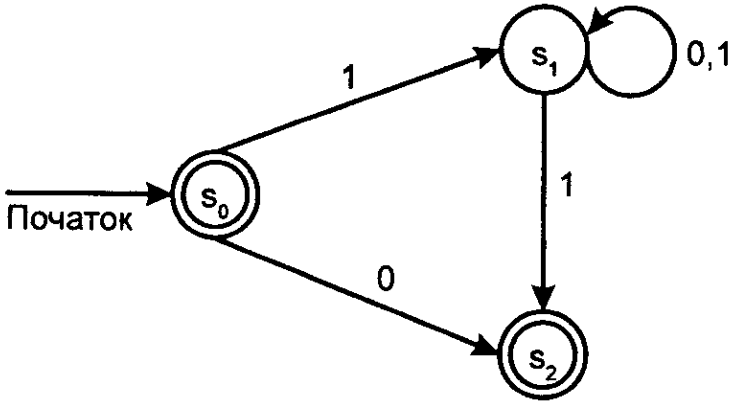

Рис. 8

Приклад 15. Побудуємо недетермінований скінченний автомат для розпізнавання мови, яка породжується регулярною граматикою $G = (V, T, S, P)$, де $V = (0, 1, A, S)$, $T = \{0, 1\}$, а множина P складається з таких продукцій: $S \to 1A$; $S \to 0$; $S \to \lambda$; $A \to 0A$; $A \to 1A$ та $A \to 1$.

Діаграму переходів для автомата, який розпізнає мову $L(G)$, зображено на рис. 8. Тут s_0 – стан, який відповідає початковому символу S граматики G, s_1 – стан, який відповідає нетерміналу A, s_2 – заключний стан.

Завершення доведення теореми 3. Доведемо, що для регулярної множини існує регулярна граматика, яка її породжує. Нехай M – недетермінований скінченний автомат, який розпізнає цю множину (він існує за теоремою 8.2). Можна вважати, що в ньому немає переходу в початковий стан s_0 (див. задачу 35). Побудуємо регулярну граматику $G=(V, T, S, P)$. Кожний символ алфавіту V в граматиці G вводимо відповідно або до символу стану, або до вхідного символу автомата M. Множина T термінальних символів граматики G якраз і відповідає символам вхідного алфавіту I автомата M. Початковий символ S граматики G відповідає символу початкового стану s_0 автомата M.

Множину P продукцій граматики G сформуємо на основі переходів у автоматі M. Якщо стан s у разі вхідного символу a переходить у кінцевий стан, то до множини P додають продукцію $A_s \to a$, де A_s – нетермінальний символ граматики G, що відповідає стану s. Якщо стан s переходить у стан t в разі вхідного символу a, то до множини P додають продукцію $A_s \to aA_t$. Продукцію $S \to \lambda$ додають до P, якщо й тільки якщо $\lambda \in L(M)$. Отже, продукції в граматиці G відповідають переходам в автоматі M. Тепер легко переконатись, що $L(G)=L(M)$. Теорему доведено.

Приклад 16. На рис. 9 зображено діаграму переходів скінченного автомата. Знайдемо регулярну граматику, яка породжує розпізнавану ним регулярну множину.

Граматика $G = (V, T, S, P)$ має алфавіт $V=\{S, A, B, 0, 1\}$, множину термінальних символів $T = \{0, 1\}$; нетермінальні символи S, A та B відповідають станам s_0, s_1 та s_2 автомата. У граматиці G початковий символ S і такі продукції: $S \to 0A$, $S \to 1B$, $S \to 1$, $S \to \lambda$, $A \to 0A$, $A \to 1B$, $A \to 1$, $B \to 0A$, $B \to 1B$ та $B \to 1$. Усі ці продукції отримано так, як це описано в завершенні доведення теореми 3.

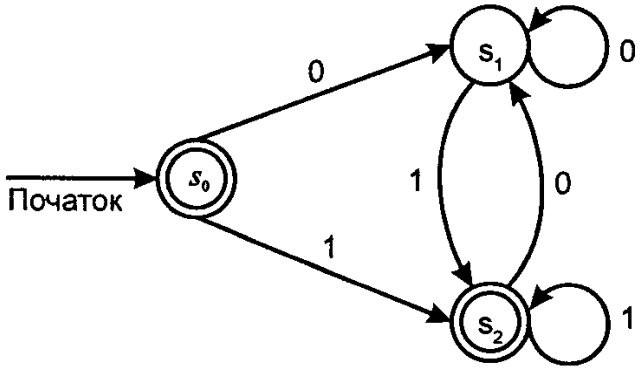

Рис. 9

Теорема 4. Множина, розпізнавана скінченним автоматом, є точно такою, ніби її породжувала регулярна граматика. Інакше кажучи, **для того щоб мова була регулярною, необхідно й достатньо, щоб вона розпізнавалась скінченним автоматом**.

Доведення безпосередньо випливає з теорем 2 та 3.

З’ясуємо, що означає задати питання щодо деякої мови. Зазначимо, що типова мова нескінченна, і тому немає сенсу наводити комусь ланцюжки цієї мови й задавати питання, яке потребує перевірки нескінченної множини ланцюжків. Набагато розумніше використовувати один зі способів скінченного подання мови, а саме – **детерміновані скінченні автомати, недетерміновані скінченні автомати, регулярні вирази**. Очевидно, що **подані одним із цих способів мови регулярні**.

Отже, для мов типу 3 (регулярних) скінченний автомат – адекватна модель. Для складніших мов адекватні інші автоматні моделі, які відрізняються від скінченних автоматів **нескінченністю** пам’яті. Проте на цю нескінченність накладаються різні обмеження залежно від типу моделі та пов’язаної з нею мови. Пам’ять може бути *магазинною,* тобто доступною лише з одного кінця (стек); такий автомат – адекватне подання мов типу 2 (контекстно вільних мов). Пам’ять може бути *лінійно обмеженою*, тобто такою, що лінійно залежить від довжини розпізнаваного слова. Зокрема, такі автомати розпізнають мови типу 1 (контекстно залежні мови). Усі названі обмеження на нескінченність пам’яті обмежують можливості цих моделей порівняно з *машинами Тьюрінга*. Машини Тьюрінга можна вважати автоматною моделлю мов типу 0, однак їх використовують переважно для уточнення поняття алгоритму.

Що це – мова чи проблема?

У теорії автоматів *проблема* (*задача*) – це питання про те, чи є даний ланцюжок елементом певної мови. Виявляється, все, що називають «проблемою» у більш широкому розумінні слова, можна виразити у вигляді проблеми приналежності деякій мові. Точніше, якщо V – алфавіт, і L – мова над алфавітом V, то *проблему* L формулюють так.

- Дано ланцюжок α із V^*, потрібно вияснити, належить ланцюжок α мові L, чи ні.

Приклад 17. Задачу перевірки простоти даного числа можна виразити у термінах приналежності мові L_p, яка складається з усіх двійкових ланцюжків, які зображають прості числа. Отже, відповідь «*так*» відповідає ситуації, у якій ланцюжок з нулів і одиниць є двійковим поданням простого числа. Інакше відповіддю буде «*ні*». Для деяких ланцюжків знайти рішення доволі просто. Наприклад, ланцюжок 0011101 не може подавати просте число з тої причини, що двійкове подання будь-якого числа, за винятком 0, починається з 1. Проте розв'язування даної проблеми для ланцюжка 11101 не настільки очевидно.

У наведеному нами означенні «проблеми» є одне слабке місце. Справа в тім, що зазвичай під проблемами розуміють НЕ питання вияснення чи щось істинне, чи ні, а запити на обробку або перетворення якихось вхідних даних (бажано, у найліпший спосіб). Наприклад, задача аналізатора в компіляторі мови С – визначити, чи належить даний ланцюжок символів ASCII множині L_C усіх правильних програм на С – точно відповідає нашому означенню. Але в завдання аналізатора входять також формування дерева синтаксичного розбору та інші дії. Більше того, компілятор у цілому розв'язує задачу перекладу С-програми в об'єктний код для якоїсь машини, і ця задача дуже далека від простої відповіді «так» або «ні» на запитання про правильність такої програми.

Тим не менше, означення «проблем» як мов витримало перевірку часом і дає змогу успішно розв'язувати задачі, які виникають у теорії складності. У рамках цієї теорії ми шукаємо нижні межі складності певних задач. Особливо важливими є методи доведення того, що певні типи задач не можуть бути розв'язані за час, менший ніж експонента від розміру вхідних даних. Виявляється, що в цьому сенсі задача, сформульована в термінах теорії мов (тобто така, що вимагає відповіді «так» або «ні»), так само важка, як і початкова задача, яка потребує «знайти розв'язок».

Отже, якщо ми можемо довести, що важко визначити, чи належить даний ланцюжок множині L_X усіх правильних програм на мові X, то з цього випливає, що переводити програми з мови X в об'єктний код не легше. Справді, якщо б було легко згенерувати код, то ми могли б просто запустити транслятор і, коли він успішно виробив би об'єктний код, зробити висновок, що отриманий ланцюжок є правильною програмою, яка належить L_X. Оскільки останній крок визначення, чи отриманий об'єктний код, не може бути складним, то за допомогою швидкого алгоритму генерації коду ми могли б ефективно розв'язати задачу про належність ланцюжка множині L_X. Але так ми приходимо до суперечності з припущенням про те, що визначити приналежність ланцюжка мові L_X важко. Отже, методом від протилежного ми довели твердження «якщо перевірка приналежності мові L_X важка, то і компіляція програм, написаних на мові X, також важка».

Даючи відповідь на запитання у заголовку цього пункту, скажемо, що насправді **мова і проблема – це одне й те саме**. Застосування термінів залежить від нашої точки зору. Коли нас цікавлять ланцюжки як такі, наприклад, множина $\{0^m1^m \mid m \geq 1\}$, ми схильні бачити в цій множині ланцюжків певну мову. Але часто важливою є семантика ланцюжків, тобто ланцюжки розглядають як закодовані графи, логічні вирази чи цілі числа. У таких випадках нас більше цікавлять не самі ланцюжки, а об'єкти, які ці ланцюжки подають. І тоді ми схильні розглядати множину ланцюжків як якусь проблему.

Лема про накачування для регулярних мов

Розглянемо інструмент, для доведення **нерегулярності** деяких мов – *лему про накачування* (pumping lemma). Інша назва – *лема про розростання*.

Лема про накачування для регулярних мов. Нехай L – регулярна мова. Існує константа n (залежна від L) така, що кожний ланцюжок $\alpha \in L$, який задовольняє нерівність $|\alpha| \geq n$, можна розбити на три ланцюжки $\alpha = \beta\gamma\omega$ так, що виконуються умови:

1) $\gamma \neq \lambda$;

2) $|\beta\gamma| \leq n$;

3) для довільного $k \geq 0$ ланцюжок $\beta\gamma^k\omega \in L$.

Це означає, що завжди можна знайти такий ланцюжок γ недалеко від початку ланцюжка α, котрий можна «накачати». Отже, якщо ланцюжок γ повторити довільну кількість разів або вилучити (k=0), то одержаний ланцюжок буде належати мові L.

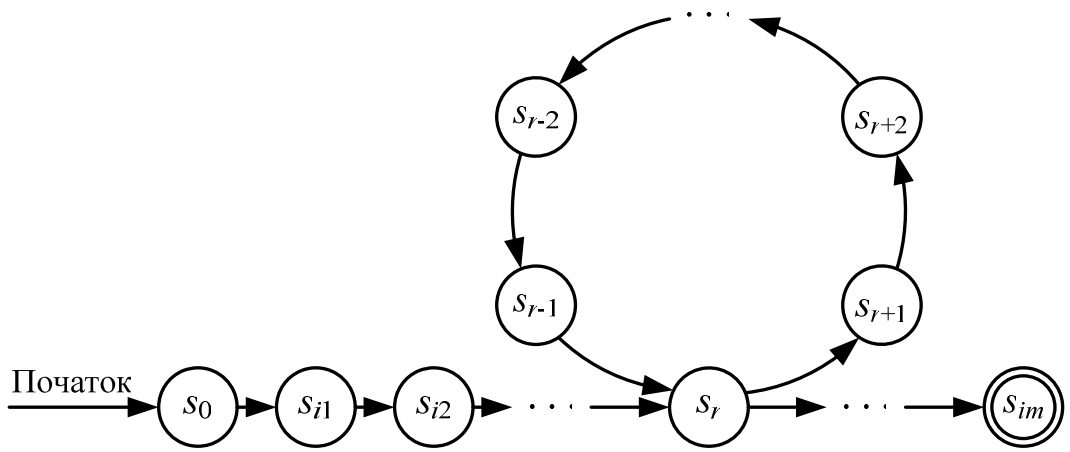

Рис. 8

Доведення. Нехай L – регулярна мова. Тоді існує детермінований скінченний автомат M, для якого $L = L(M)$, тобто що розпізнає цю мову. Нехай кількість станів автомата M дорівнює n (константа, яка фігурує у формулюванні леми). Припустимо, що послідовні стани, у які автомат M переходить під дією вхідного ланцюжка α, такі:

$$s_0, s_{i_1}, s_{i_2}, \ldots, s_{i_m},$$

де $m = |\alpha|$ – довжина ланцюжка α. За умовою леми, $m \geq n$, тому в списку s_0, s_{i_1}, s_{i_2}, …, s_{i_m}, який складається принаймні з n+1 стану, обов'язково є повторення (це випливає з принципу коробок Діріхле). Нехай s_r – перший повторюваний стан. Позначимо як γ ту частину ланцюжка α, що переводить автомат зі стану s_r, коли він зустрічається вперше, до стану s_r, коли він зустрічається вдруге. Позначимо як β частину ланцюжка α перед γ, а як ω – частину ланцюжка α після γ. Очевидно, що $|\gamma| \geq 1$ (отже, $\gamma \neq \lambda$) та $|\beta\gamma| \leq n$ (оскільки всі стани до другої появи s_r, різні). Більше того, ланцюжок $\beta\gamma^k\omega$ для будь-якого цілого невід'ємного числа k має переводити автомат у той самий заключний стан, що й ланцюжок βγω. Справді, частина γ^k ланцюжка $\beta\gamma^k\omega$ просто переводить стани автомата вздовж петлі, яка починається та закінчується в стані s_r. Ця петля проходиться k разів (рис. 8). Звідси випливає, що ланцюжки $\beta\gamma^k\omega$ допускаються автоматом тому, що ланцюжок βγω допускається; отже, усі вони належать мові $L(M)$. Доведення леми завершено.

Приклад 11. Доведемо, що мова $L = \left\{0^m1^m \middle| m = 0,1,2,\ldots\right\}$ нерегулярна.

Від протилежного. Припустимо, що мова L регулярна. Нехай ланцюжок $\alpha = 0^m1^m$ і $|\alpha| = 2m \geq n$. Тоді за лемою про накачування ланцюжки $\alpha = 0^m1^m = \beta\gamma\omega$ та $\beta\gamma^k\omega$ належать мові L, причому $\gamma \neq \lambda$. Тепер розглянемо такі міркування. Ланцюжок γ не може містити водночас нулі й одиниці, бо ланцюжок γ^2 тоді містив би 10, і ланцюжок $\beta\gamma^2\omega$ не належав би мові L. Отже, ланцюжок γ складається або тільки з нулів, або тільки з одиниць. Але тоді ланцюжок $\beta\gamma^2\omega$ містить або забагато нулів, або забагато одиниць. Звідси випливає, що ланцюжок $\beta\gamma^2\omega$ не належить L. Ця суперечність доводить, що мова $L = \left\{0^m1^m \middle| m = 0,1,2,\ldots\right\}$ нерегулярна.

Лема про накачування для контекстно вільних мов

Лема про накачування для контекстно вільних мов. Нехай L – контекстно вільна мова. Тоді існує така константа n (залежна від L), що довільний ланцюжок $\alpha \in L$, довжина якого $|\alpha| \geq n$, можна так розбити на п'ять ланцюжків $\alpha = \beta\gamma\omega\delta\eta$, що виконуються умови:

1) $|\gamma\omega\delta| \leq n$ (отже, середня частина не дуже довга);
2) $\gamma\delta \neq \lambda$ (позаяк γ та δ ланцюжки, які потрібно „накачувати", ця умова означає, що хоча б один із них непорожній);
3) $\beta\gamma^k\omega\delta^k\eta \in L$ для всіх $k \geq 0$ (два ланцюжки γ та δ можна „накачати" довільну кількість разів або вилучити, якщо $k=0$, і отриманий ланцюжок $\beta\gamma^k\omega\delta^k\eta$ також буде належати мові L).

Доведення цієї леми ми не наводимо.

Приклад 19. Доведемо, що мова $L = \left\{0^m1^m2^m \middle| m = 0,1,2,\ldots\right\}$ не контекстно вільна.

Від протилежного. Припустимо, що мова L контекстно вільна. Тоді можна знайти такі ланцюжки β, γ, ω, δ й η, що хоча б один з ланцюжків γ чи δ непорожній і $\beta\gamma^k\omega\delta^k\eta$ для всіх $k \geq 0$ можна подати у вигляді $0^m1^m2^m$. Спочатку зазначимо, що кожний із ланцюжків γ й δ може бути утворений тільки символом одного типу. Справді, якщо хоча б один із цих двох ланцюжків утворено символами двох або трьох типів, то в ланцюжку $\beta\gamma^2\omega\delta^2\eta$ порушено потрібний порядок символів, тобто він не має вигляду $0^m1^m2^m$. Отже, у ланцюжку $\gamma\delta$ немає принаймні одного символу (наприклад, 0). З іншого боку, оскільки $\gamma\delta \neq \lambda$, то принаймні один символ (наприклад, 1) входить у ланцюжок $\gamma\delta$. Тоді для великих k ланцюжок $\beta\gamma^k\omega\delta^k\eta$ буде містити більше одиниць, ніж нулів, і, отже, не належить мові L. Це суперечить лемі про накачування для контекстно вільних мов. Звідси випливає, що мова $L = \left\{0^m1^m2^m \middle| m = 0,1,2,\ldots\right\}$ не контекстно вільна.

© Щербина Ю.М., 2022