3.5. АЛГОРИТМИ МАРКОВА ДЛЯ ПЕРЕТВОРЕННЯ РЯДКІВ

Нормальний алгоритм Маркова задається алфавітом V і нормальною схемою підстановок (продукцій, правил). Загальна стратегія роботи алгоритму Маркова полягає в тому, щоб піддавши деякий рядок x ряду операцій (підстановок) перетворити його у вихідний рядок y. Єдиною структурою, з якою оперує алгоритм Маркова, є вхідний рядок x. І вхідний рядок x, і вихідний рядок y складаються із символів алфавиту V. Отже, рядки x і y належать до рефлексивного замкнення V^*.

Простою марківською продукцією називається запис $u \to w$, де u і w рядки в V^*, при чому алфавіт V не містить символів «→», і «.». Ліва частина (слово) марківської продукції (u) називається *антецедентом*, а права частина (w) – *консеквентом*.

Продукція з антецедентом u і консеквентом w застосовувана до рядка $z \in V^*$, якщо існує хоча б одне входження u в z. В протилежному випадку продукція незастосовувана до рядка z. Якщо продукція застосовувана, то перше (крайнє ліве) входження замінюється на w. Наприклад, якщо продукція ***'ba'→'c'*** застосовувана до вхідного рядка ***'ababab'***, то в результаті буде отримано рядок ***'acbab'***. А в той же час продукція ***'baa'→'c'*** незастосовувана до того ж рядка ***'ababab'***.

Алгоритм Маркова складається з упорядкованої множини продукцій $P_1, P_2, \ldots, P_n$. Алгоритм побудови послідовності слів визначається таким чином. В якості початкового слова виступає рядок z і до нього застосовують продукції за своїм порядком. Послідовність виконання алгоритму залежить від того, застосовувана чи незастосовувана до рядка чергова продукція. Якщо вона застосовувана, то рядок перетворюється у відповідності до продукції. Якщо ж продукція незастосовувана (тобто в рядку не знайдено входження, яке можна замінити), то здійснюється

перехід до наступної продукції. Алгоритм Маркова завершується в одному із двох випадків:

- до рядка незастосовувана жодна з наявних продукцій;
- до рядка застосовувана термінальна (кінцева) продукція.

Термінальна продукція – це запис у вигляді ***'x→y.'***, де *x* і *y* – рядки в ***V****, а символ «**.**» йде одразу за консеквентом.

Приклад

Викреслення з рядка всіх символів 'a', крім останнього в рядку.

V={a, b, c};

P1: 'ab'→'b'; P2: 'ac'→'c'; P3: 'aa'→'a'.

Хай початковим словом *z* є рядок *'bacaabaa'*, то послідовність перетворень буде такою:

(P1) 'bacaabaa' → 'bacabaa'
(P1) 'bacabaa' → 'bacbaa'
(P2) 'bacbaa' → 'bcbaa'
(P3) 'bcbaa' → 'bcba'

Використовуючи наведену вище систему запису алгоритмів, визначимо виконання алгоритму Маркова.

Алгоритм MARKOV. Задано алгоритм Маркова з кінцевою послідовністю продукцій $P_1, P_2, \ldots, P_n$, які треба застосувати до вхідного рядка $z_0 \in V^*$. Введемо індекси *i* та *j*. Позначимо z_i результат *i*-го перетворення рядка z_0. Тоді роботу алгоритму можна представити наступною послідовністю дій.

1. *i←0:* встановлюємо початкове значення індексу поточного значення рядка, що підлягає перетворенню.
2. *j←1:* встановлюємо початкове значення індексу поточної продукції, що застосовується до рядка.
3. Повторюємо крок п. 4 доти, доки *j<=n:* визначаємо цикл для перевірки застосовуваності продукцій.
4. Перевіряємо, чи можна застосувати продукцію P_j до рядка z_i. Якщо «так», то:
   4.1. Продукція P_j застосовується до рядка z_i і встановлюється нове значення індексу рядка *i←i+1.*
   4.2. Якщо продукція P_j є термінальною, то перехід до п. 5, у протилежному випадку встановлюється початкове значення індексу продукції *j←1* і переходимо до п. 3;
   у протилежному випадку, тобто коли продукцію P_j не можна застосувати до рядка z_i, то:

4.3. встановлюється нове значення індексу продукції $j \leftarrow j+1$.

5. Закінчити виконання алгоритму.

В продукціях можуть використовуватися змінні. Так, попередній алгоритм може бути заданий таким чином:

P_1: *'a'x⟶ x, де* $x \in V$.

У наведеному правилі введена змінна x, вона позначає будь-який символ із V. В продукціях ще може використовуватися символ Λ, який позначає відсутність символу або «порожній» символ. При цьому мається на увазі, що він завжди є в рядку.

Приклад 1

Хай V – деякий алфавіт, y – символ цього алфавіту. Наведемо алгоритм Маркова (MA: YZ, $y \in V$), що перетворює будь-який рядок $z \in V^*$ в рядок yz, при цьому мається на увазі, що символ y і рядок z з'єднуються за допомогою операції конкатенації. Такий алгоритм можна задати однією продукцією:

$$P_1: \Lambda \longrightarrow y.$$

Алгоритм діє таким чином: на початку будь-якого рядка стоїть порожній символ, значить правило таке, що можна застосувати, тому на початку рядка вставиться символ, який є значенням змінної y. Оскільки це правило є термінальним, то алгоритм закінчує роботу.

Приклад 2

Задача: записати МА, що перетворює будь-який рядок $z \in V^*$ у рядок zy.

Алгоритм вставки будь-якого символу y наприкінці будь-якого рядка не такий тривіальний. Дійсно, якщо застосовувати продукцію $\Lambda \longrightarrow y$, то символ y, як наводилося вище, буде вставлений на початку рядка z, якщо продукція термінальна; а у випадку нетермінальної продукції символ y буде вставлятися циклічно, при чому виходу з циклу не буде. Продукція $z \longrightarrow zy$ є некоректною, тому що z – це будь-який рядок, що належить рефлексивному замкненню алфавіту V. Дійсно, $z \in V^*$, $V^* = \Lambda \cup V \cup V^2 \cup V^3 \cup \ldots$, тобто довжина може бути нескінченною.

Для того, щоб організувати «цикл» в рядку z, уведемо набір *допоміжних символів,* які будемо використовувати у якості *маркерів.* Вони дозволяють відмітити певну позицію у рядку, надаючи таким чином, можливість застосувати до цієї позиції певну продукцію. Для останньої задачі введемо маркер a, який не належить до алфавіту V. Тоді:

МА: $ZY(x \in V, y \in V)$

P_1: $ax \longrightarrow xa$

P_2: $\alpha \longrightarrow y$.
P_3: $A \longrightarrow a$
Тоді, якщо z_0='abc', послідовність перетворень буде такою:
P_3: 'abc'⟶'aabc'
P_1: 'aabc'⟶'aabc'
P_1: 'aabc'⟶'abac'
P_1: 'abac'⟶'abca'
P_2: 'abca'⟶'abcy'
Даний алгоритм можна описати так.
1. Встановлення маркера на початок вхідного рядка $z \longrightarrow az$.
2. Циклічне пересування маркера a по рядку z (продукція P_1) до тих пір, поки не досягнемо $z\alpha$.
3. Закінчується алгоритм застосуванням продукції P_2.

Приклад 3
Даний рядок $z \in V^*$, такий, що $z=z_1z_2z_3\ldots z_{n-1}z_n$. Потрібно сформувати $z^1=z_nz_{n-1}\ldots z_3z_2z_1$, тобто здійснити операцію реверсу (*reverse*) символів рядку (ця операція згадувалася у п. 1.3). У якості маркерів будемо використовувати α і β.
МА: $REVERSE(x, y \in V)$
P_1: $\alpha a \longrightarrow \beta$
P_2: $\beta a \longrightarrow \beta$
P_3: $\beta x \longrightarrow x\beta$
P_4: $\beta \longrightarrow A$
P_5: $\alpha xy \longrightarrow yax$
P_6: $A \longrightarrow \alpha$
Нехай z='abc'

$abc \Rightarrow \alpha'abc'$	(P_6)
$\alpha'abc' \Rightarrow 'b'\alpha'ac'$	(P_5)
$\Rightarrow 'bc'\alpha'a'$	(P_5)
$\Rightarrow \alpha'bc'\alpha'a'$	(P_6)
$\Rightarrow 'c'\alpha'b'\alpha'a$	(P_5)
$\Rightarrow \alpha'c'\alpha'b'\alpha'a$	(P_6)
$\Rightarrow \alpha\alpha'c'\alpha'b'\alpha'a'$	(P_6)
$\Rightarrow \beta'c'\alpha'b'\alpha'a'$	(P_1)
$\Rightarrow 'c'\beta\alpha'b'\alpha'a'$	(P_3)
$\Rightarrow 'c'\beta'b'\alpha'a'$	(P_2)
$\Rightarrow 'cb'\beta\alpha'a$	(P_3)
$\Rightarrow 'cb'\beta'a$	(P_2)
$\Rightarrow 'cba'\beta$	(P_3)
$\Rightarrow 'cba'$	(P_4)

Всього 57 перевірок і 24 підстановки.

Алгоритми Маркова піддавались вдосконаленням. Одне з них – алгоритми *LMA* (*Labeled Markov Algorithm* – помічений алгоритм Маркова).

Введення міток змінює початкову марківську модель у двох аспектах:

1. Якщо продукція застосовується та має мітку переходу, то наступною повинна перевірятися продукція, вказана цією міткою.

2. Якщо продукція не застосовується або не містить мітки переходу, то перевіряється наступна за нею продукція (якщо така є) або алгоритм завершується (якщо продукція була останньою).

Опишемо алгоритм *REVERSE* з допомогою *LMA* для того ж самого вхідного рядка *'abc'*.

LMA: *REVERSE* ($x, y \in V$)

P_1: $\alpha x y \longrightarrow y \alpha x$	(P_1)
P_2: $\alpha x \beta \longrightarrow \beta x$	(P_4)
P_3: $\alpha x \longrightarrow \beta x$	(P_6)
P_4: $\alpha \beta \longrightarrow \Lambda$	
P_5: $\alpha \alpha \longrightarrow \Lambda$	
P_6: $\Lambda \longrightarrow \alpha$	(P_1)

Тоді

$'abc' \Rightarrow \alpha'abc'$	(P_6)
$\Rightarrow 'b'\alpha'ac'$	(P_1)
$\Rightarrow 'bc'\alpha'a'$	(P_1)
$\Rightarrow 'bc'\beta'a'$	(P_3)
$\Rightarrow \alpha'bc'\beta'a'$	(P_6)
$\Rightarrow 'c'\alpha'b'\beta'a'$	(P_1)
$\Rightarrow 'c'\beta'ba'$	(P_2)
$\Rightarrow \alpha'c'\beta'ba'$	(P_6)
$\Rightarrow \beta'cba'$	(P_2)
$\Rightarrow \alpha\beta'cba'$	(P_6)
$\Rightarrow 'cba'$	(P_4)

Всього 27 перевірок і 11 підстановок. Можна ще вдосконалити алгоритм аж до 18 перевірок.

До сих пір розглядалися алгоритми Маркова, які оперували з єдиним вхідним або основним рядком. Припустимо тепер, що потрібний алгоритм, який, наприклад, викреслює із рядка w підрядок u, якщо такий підрядок існує. В цьому випадку існує два вхідних параметри u і w. Написання алгоритму типу *LMA* для виділення підрядка є довгою та стомлюючою справою. Щоб полегшити цю роботу і не зменшити при цьому обчислювальну потужність алгоритму Маркова, можна ввести ще

один програмний атрибут, який називається *коміркою пам'яті*, або *адресною коміркою зберігання*. Такі комірки пам'яті призначені для зберігання рядків, що використовуються лише у проміжних операціях. Є один вхідний і один вихідний рядок, що являє собою основний рядок (*w* та *u*). В алгоритмі можна використовувати кінцеву множину комірок пам'яті, що позначаються $[M_1]$, $[M_2]$, …, $[M_n]$. Доведено, що застосування цієї концепції спрощує складання алгоритму Маркова, однак не дає ніяких додаткових переваг з обчислювальної точки зору.

Висновок. Загальна стратегія роботи алгоритму Маркова полягає в тому, щоб, піддавши деякий рядок x ряду операцій (продукцій), перетворити його на вихідний рядок y. Цей процес перетворення є звичайним у таких галузях застосування ЕОМ, як редагування тексту або компіляція програми. Компіляцію можна представити як процес перетворення рядків вихідної мови програмування (наприклад, Pascal) в рядки об'єктного коду.

Крім функцій обробки рядків, алгоритми Маркова можна з деякими припущеннями застосовувати і для арифметичних обчислень, але це не ефективно.

Марківська модель хоча і виглядає обмеженою, достатньо ефективна для обробки будь-якої обчислювальної функції, тобто будь-якої функції, значення якої можна обчислити. Однак марківська модель незручна, особливо при арифметичних обчисленнях. В той же час алгоритми Маркова дають можливість показати маніпулювання рядками на елементарному та зрозумілому рівні. Мови обробки рядків (наприклад, СНОБОЛ), оператори і функції обробки рядків у сучасних мовах програмування, системах керування базами даних, редактори текстів використовували поняття та ідеї основних примітивних і базових функцій, які «вийшли» із алгоритмів Маркова.