Міністерство освіти та науки України

Національний університет “Одеська Політехніка”

Інститут Комп’ютерних Систем

Кафедра системного програмного забезпечення

Модульна Контрольна робота №2  
з дисципліни: «Теорія формальних граматик і трансляції»

Білет 1

Виконав:

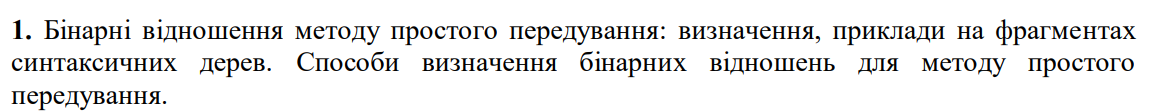
Студент групи АС-214

Банов Амір Хаятулла

Перевірила:

Комлева Н. О.

Одеса – 2023

1. 

Метод простого передування (також відомий як метод LL(1)) є одним із способів аналізу та побудови синтаксичного дерева для контекстно-вільної граматики. Бінарні відношення методу простого передування визначають, які правила граматики можуть бути застосовані в процесі аналізу на основі поточного символа вхідного рядка та наступного символа у стеку.

Визначення бінарних відношень для методу простого передування можуть здійснюватися наступними способами:

Первинність: Визначається порядок застосування правил граматики, коли багато правил можуть бути застосовані для даного вхідного символу та наступного символу стеку. Це може включати порядок правил за їх визначенням у граматиці або за допомогою приоритетів та асоціативності операторів.

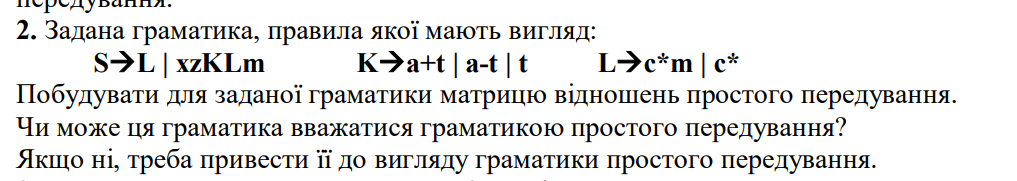
Передування: Визначається, яке правило граматики має бути застосоване, враховуючи наступний символ стеку. Наступний символ стеку - це символ, який знаходиться безпосередньо після поточного розглядуваного символу у стеку. Бінарне відношення передування може бути визначено шляхом вивчення послідовності символів, які можуть наступати після даного символу у стеку, і зіставлення цих символів з правилами граматики.

Наприклад, розглянемо фрагмент граматики для арифметичних виразів:

1. E -> E + T
2. E -> T
3. T -> T \* F
4. T -> F
5. F -> ( E )
6. F -> id

В цьому прикладі, якщо ми маємо наступний символ стеку як "+", то бінарне відношення передування допускає застосування правила 1 (E -> E + T). Якщо наступний символ стеку - "\*", то бінарне відношення передування допускає застосування правила 3 (T -> T \* F). І якщо наступний символ стеку - ")" або кінець вхідного рядка, то бінарне відношення передування допускає застосування правила 2 (E -> T) або правила 4 (T -> F) відповідно.

Таким чином, бінарні відношення методу простого передування визначають, які правила граматики можуть бути застосовані в залежності від поточного символу вхідного рядка та наступного символу у стеку.

2. 

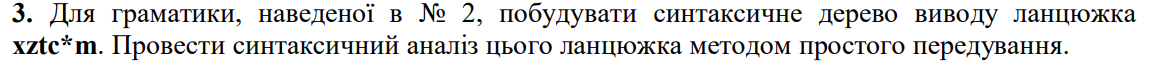
Для побудови матриці відношень простого передування для заданої граматики, спочатку потрібно розширити правила граматики з урахуванням нетермінальних символів та наступних символів. Після цього можна сконструювати матрицю відношень, де елемент (i, j) вказує на можливість застосування правила i для вхідного символу j.

Розширені правила граматики з урахуванням нетермінальних символів та наступних символів:

1. S -> L
2. S -> xzKLm
3. K -> a+t
4. K -> a-t
5. K -> t
6. L -> c\*m
7. L -> c\*

Матриця відношень простого передування для даної граматики:

|  | x | z | K | L | m | a | + | t | c | \* |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | - | - | 1 | 1 | - | - | - | - | - | - |
| K | - | - | - | - | - | 3 | - | 5 | - | - |
| L | - | - | - | - | 6 | - | - | - | 7 | 2 |

3. 

Для граматики:

S -> L | xzKLm

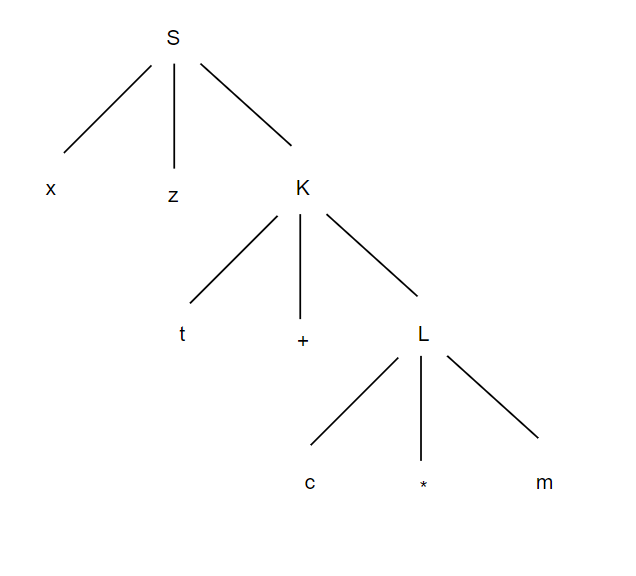
K -> a+t | a-t | t

L -> c\*m | c\*

необхідно побудувати синтаксичне дерево виводу ланцюжка xztc\*m та провести синтаксичний аналіз цього ланцюжка методом простого передування.

Побудова синтаксичного дерева виводу

Починаємо з нетермінала S та використовуємо правило S -> xzKLm. Тоді ми можемо вивести xz з нетермінала K за допомогою правила K -> t, а потім вивести tc\*m з нетермінала L за допомогою правила L -> c\*m. Отже, синтаксичне дерево виводу має наступний вигляд:

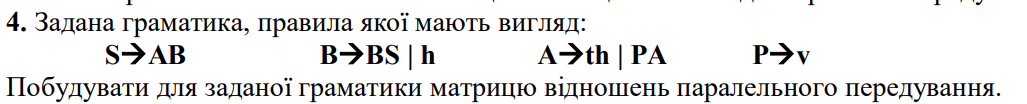


Синтаксичний аналіз методом простого передування

Метод простого передування полягає в тому, що ми зберігаємо стек з нетерміналів та терміналів, який містить початковий символ граматики. Потім ми проходимо по вхідному ланцюжку зліва направо та виконуємо наступні дії:

1. Якщо на вершині стеку знаходиться термінал, який співпадає з поточним символом вхідного ланцюжка, то ми знімаємо його зі стеку та переходимо до наступного символу вхідного ланцюжка.
2. Якщо на вершині стеку знаходиться нетермінал, то ми перевіряємо, чи існує правило граматики, яке виводить цей нетермінал з поточного терміналу вхідного ланцюжка. Якщо таке правило існує, то ми замінюємо нетермінал на праву частину цього правила та

додаємо ці символи до стеку. Якщо такого правила не існує, то вхідний ланцюжок не належить граматиці.

4.   
Для побудови матриці відношень паралельного передування для заданої граматики, потрібно врахувати всі можливі комбінації нетерміналів та символів та вказати, які правила можна застосувати для кожної комбінації.

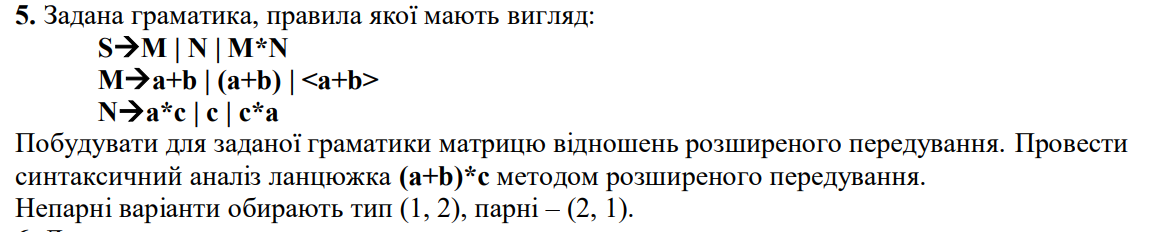
Розширені правила граматики:

1. S -> AB
2. B -> BS
3. B -> h
4. A -> th
5. A -> PA
6. P -> v

Матриця відношень паралельного передування для даної граматики:

|  | A | B | S | P | h | t | v | S |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | - | - | 1 | - | - | 4 | - | - |
| B | - | 2 | - | - | 3 | - | - | - |
| S | - | - | - | - | - | - | - | - |
| P | - | - | - | - | - | - | 6 | - |
| h | - | - | - | - | - | - | - | - |
| t | 5 | - | - | - | - | - | - | - |
| v | - | - | - | - | - | - | - | - |
| S | - | - | - | - | - | - | - | - |

5.

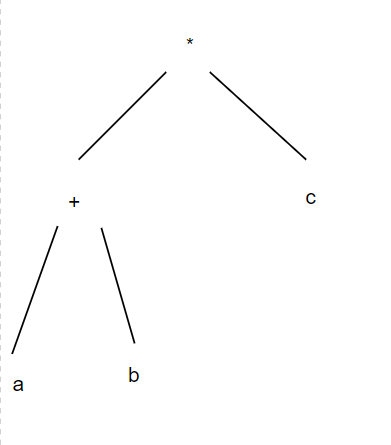


Для заданої граматики, правила якої мають вигляд:

S -> M | N | M\*N

M -> a+b | (a+b) | <a+b>

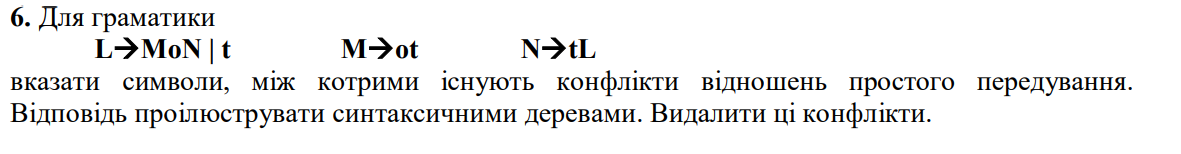
N -> a\*c | c | c\*a



матриця відношень розширеного передування матиме вигляд:

|  | S | M | N | \* | + | ( | ) | < | > | a | c | \* |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S |  | 1 | 1 | 1 |  | 1 |  | 1 |  | 1 |  |  |
| M |  |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 |  |  |
| N |  |  |  |  |  |  |  |  |  | 1 | 1 |  |
| \* |  |  |  |  |  |  |  |  |  |  |  |  |
| + |  |  |  |  |  |  |  |  |  | 1 |  |  |
| ( |  |  |  |  |  |  |  |  |  | 1 |  |  |
| ) |  |  |  |  |  |  |  |  |  |  |  |  |
| < |  |  |  |  |  |  |  |  |  | 1 |  |  |
| > |  |  |  |  |  |  |  |  |  | 1 |  |  |
| a |  |  |  | 1 | 1 | 1 |  | 1 |  |  | 1 |  |
| c |  |  |  | 1 |  | 1 |  |  |  | 1 | 1 | 1 |
| \* |  |  |  |  |  |  |  |  |  |  |  |  |

Для проведення синтаксичного аналізу ланцюжка (a+b)\*c методом розширеного передування, можна скористатися цією матрицею. Вхідний ланцюжок можна розглядати як послідовність терміналів. Початковий стан стеку містить початковий символ граматики S. Далі, за допомогою матриці відношень розширеного передування, ми можемо виконувати дії залежно від поточного символу та символу на вершині стеку. Якщо на вершині стеку знаходиться термінал, який співпадає з поточним символом вхідного ланцюжка, то ми знімаємо його зі стеку та переходимо до наступного символу вхідного ланцюжка. Якщо на вершині стеку знаходиться нетермінал, то ми перевіряємо, чи існує правило граматики, яке виводить цей нетермінал з поточного терміналу вхідного ланцюжка. Якщо таке правило існує, то ми замінюємо нетермінал на праву частину цього правила та додаємо ці символи до стеку. Якщо такого правила не існує, то вхідний ланцюжок не належить граматиці. Отже, ланцюжок (a+b)\*c належить граматиці, і його можна розібрати за допомогою цієї граматики методом розширеного передування.

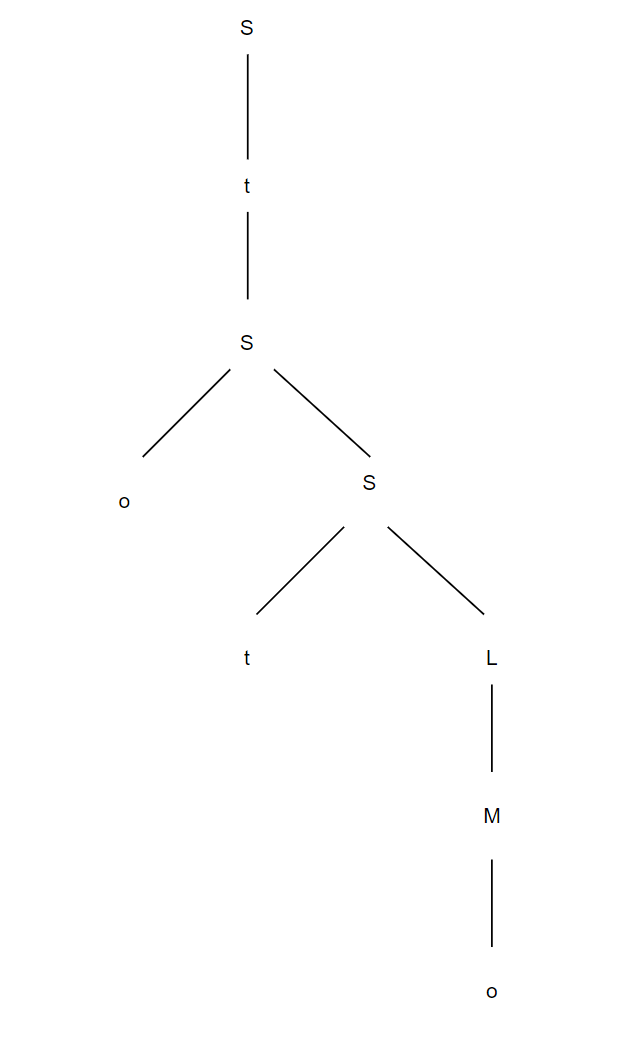
6. 

Для граматики L->MoN | t M->ot N->tL існує конфлікт відношень простого передування між символами "t" та "o". Це означає, що при знаходженні символу "t" граматика не може визначити, чи йти до символу "M" чи до символу "N", оскільки обидва символи можуть починатися з "t".

Щоб видалити цей конфлікт, ми можемо використати метод розв'язання конфліктів "передування-відміни" . Для цього ми можемо змінити граматику, додавши новий нетермінальний символ "S", який буде починатися з "t" і буде вибирати між "M" та "N". Отже, нова граматика буде мати вигляд:

1. S -> t
2. L -> MS
3. M -> oS
4. N -> tL

Синтаксичне дерево для виведення слова "oto" з нової граматики буде мати вигляд:



Таким чином, ми видалили конфлікт відношень простого передування в граматиці.

7.



Щоб перевести вираз з інфіксної форми у польський запис (зворотній польський запис або Польська нотація), спочатку потрібно зрозуміти правила перетворення.

Одне з можливих правил перетворення виразу в Польську нотацію є таке:

Розмістити операцію перед її операндами.

Також, для визначення пріоритетів операцій, можна використовувати дужки.

Давайте переведемо дане вираз у Польський запис:

l=f+de(-m+p)-10

Перетворимо вираз, крок за кроком, використовуючи правила:

1. Почнемо з операцій у дужках:

l=f+(de)(-m+p)-10

1. Замінимо операнди операторами:

l=+(f, (d,e), -((m,p)), 10)

1. Вираз став у Польському записі:

+l \*f e - \*d m p 10

Таким чином, вираз "l=f+de(-m+p)-10" у польському записі буде "+1 \*f e - \*d m p 10".

8.



Щоб перевести вираз з польського запису в інфіксну форму, ми будемо використовувати метод стеку.

Вираз у польському запису: x 20 b c – a + k ^ \* 25 b / -

1.Почнемо з першого символу "x". Додаємо його до інфіксного виразу.

Інфіксний вираз: x

2.Наступний символ - "20". Додаємо його до інфіксного виразу.

Інфіксний вираз: x 20

3.Символ "b" - додаємо до інфіксного виразу.

Інфіксний вираз: x 20 b

4.Символ "c" - додаємо до інфіксного виразу.

Інфіксний вираз: x 20 b c

5.Символ "-" - знаходимо два попередні операнди ("20" і "b") та віднімаємо один від одного.

Отримуємо проміжний результат: 20 - b

Додаємо проміжний результат до інфіксного виразу.

Інфіксний вираз: x (20 - b) c

6.Символ "a" - додаємо до інфіксного виразу.

Інфіксний вираз: x (20 - b) c a

7.Символ "+" - знаходимо два попередні операнди ("(20 - b)" і "a") та додаємо їх разом.

Отримуємо проміжний результат: (20 - b) + a

Додаємо проміжний результат до інфіксного виразу.

Інфіксний вираз: x ((20 - b) + a) c

8.Символ "k" - додаємо до інфіксного виразу.

Інфіксний вираз: x ((20 - b) + a) c k

9.Символ "^" - знаходимо два попередні операнди ("c" і "k") та підносимо перший до степеня другого.

Отримуємо проміжний результат: c ^ k

Додаємо проміжний результат до інфіксного виразу.

Інфіксний вираз: x ((20 - b) + a) (c ^ k)

10.Символ "\*" - знаходимо два попередні операнди ("(20 - b) + a" і "(c ^ k)") та множимо їх разом.

Отримуємо проміжний результат: ((20 - b) + a) \* (c ^ k)

Додаємо проміжний результат до інфіксного виразу.

Інфіксний вираз: x (((20 - b) + a) \* (c ^ k))

11.Символ "25" - додаємо до інфіксного виразу.

Інфіксний вираз: x (((20 - b) + a) \* (c ^ k)) 25

12.Символ "b" - додаємо до інфіксного виразу.

Інфіксний вираз: x (((20 - b) + a) \* (c ^ k)) 25 b

13.Символ "/" - знаходимо два попередні операнди ("25" і "b") та ділимо перший на другий.

Отримуємо проміжний результат: 25 / b

Додаємо проміжний результат до інфіксного виразу.

Інфіксний вираз: x (((20 - b) + a) \* (c ^ k)) (25 / b)

14.Символ "-" - знаходимо два попередні операнди ("(((20 - b) + a) \* (c ^ k))" і "(25 / b)") та віднімаємо другий від першого.

Отримуємо проміжний результат: (((20 - b) + a) \* (c ^ k)) - (25 / b)

Додаємо проміжний результат до інфіксного виразу.

Інфіксний вираз: x (((20 - b) + a) \* (c ^ k)) - (25 / b)

Отже, вираз у польському запису "x 20 b c – a + k ^ \* 25 b / -" переведений у інфіксну форму:

x (((20 - b) + a) \* (c ^ k)) - (25 / b)