яскравості стримера, що відображає залежність яскравості стримера від його координат.

Література

1. Коломієць Р.О. Виділення фрактального спектра кірліан-зображень/ Р.О. Коломієць // ВІСНИК ЖДТУ. – 2010. – № 1. – 109–114 С.

2. Коротков К. Основы ГРВ-биоелектрографии / Коротков К. – СПб. : СПбГИТМО (ТУ), 2001. – 360 с.

3. Измерение электропроводимости и солености воды кондуктометрическим методом [Електронний ресурс] / Мосин О.В. – Режим доступу : http://www.o8ode.ru/article/answer/method/izmerenie_elektroprovodimocti_i_colenocti_vody_konduktometri4eckim_metodom.htm.

4. Пономарев О.А. Свойства жидкой воды в электрических и магнитных полях / О.А. Пономарев, Е.Е. Фесенко // Биофизика, 2000. – Т. 45. – № 3. – С. 389–398.

5. Третьяков Ю.Д. Неорганическая химия / Третьяков Ю.Д. М. : Академия, 2005. – 380 с.

6. Грановский В.Л. Электрический ток в газах / Грановский В.Л. – М. : Наука, 1971. – 560 с.

7. Буссонов Л. А.Теоретические основы электротехники. Электрические цепи / Л. А. Буссонов. М: Высшая школа, 1998. – 640 с.

8. Арцимович Л.А.Движение заряженных частиц в электрическом и магнитом полях / Л. А. Арцимович, С. Ю. Лукьянов. М. : Наука, 1978. – 224 с.

Надійшла 22.9.2011 р.

УДК 004.414.3

О.С. САВЕНКО, Ю.П. КЛЬОЦ, В.С. ШЕВЦОВ
Хмельницький національний університет

МЕТОДИ ЛЕКСИЧНОГО АНАЛІЗУ ТЕХНІЧНОГО ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті проведено порівняння методів лексичного аналізу тексту технічного завдання на розробку програмного забезпечення або програмно-технічного комплексу. Показані основні недоліки відомих методів лексичного аналізу, і шляхи їх вирішення.

The article compared the methods of lexical analysis of text specification for software development or software-technical complex. Showing the major shortcomings of existing methods of lexical analysis, and solutions.

Ключові слова: лексичний аналіз, методи лексичного аналізу, технічне завдання на розробку ПЗ, програмне забезпечення, граматики Хомського, link граматика.

Вступ. Щороку кількість програмних продуктів стрімко зростає. Це зумовлено великим попитом на комп'ютерну техніку та інші пристрої, де використовуються спеціалізовані програми та утиліти, також зростає складність самих програм та програмно-технічних пристроїв (комплексів). Чим складніше програмне забезпечення (ПЗ), тим важче перевірити повноту та несуперечливість вимог технічного завдання на його розробку. Процес розробки технічного завдання детально описано в практичних рекомендаціях зі специфікації програмного забезпечення [1]. В процесі розробки технічного завдання часто допускаються помилки при формуванні функційних та нефункційних вимог.

Функційні вимоги – це вимоги, що описують дії, які мають виконувати компоненти системи або система в цілому [2, 4]. Нефункційні вимоги – це всі інші вимоги, що не мають відношення до функційних вимог [3, 4].

Невчасне виявлення помилок технічного завдання на етапі проектування ПЗ може призвести до збоїв ПЗ і як наслідок – до можливого виходу з ладу обладнання та техногенних аварій. Для розв'язання задачі виявлення помилок на етапі складання технічного завдання на розробку ПЗ необхідно перевіряти коректність функційних та нефункційних вимог до розроблюваного ПЗ та відповідність їх стандартам технічної області застосування [5–7]. Перевірку технічного завдання на розробку програмного забезпечення можна проводити групі експертів або лексичним аналізатором.

Аналіз технічного завдання за допомогою групи експертів, під час якого кожен експерт аналізує свою частину технічного завдання і робить висновок. Аналіз малих за об'ємом технічних завдань на повноту та несупереливість виконується швидко та з малими фінансовими затратами, проте для великих за обсягом технічних завдань характерні різке сповільнення праці експертів та великі фінансові витрати на оплату праці. При великих об'ємах тексту людині властиво з часом пропускати (ігнорувати) деякі слова або речення, які можуть бути важливими.

Аналіз технічного завдання за допомогою лексичного аналізатора дозволяє усунути недоліки першого методу, для нього не є критичним обсяг технічного завдання, але критичною є складність мовних конструкцій технічного завдання. Також лексичні аналізатори проводять лексичний аналіз набагато швидше, ніж експерти.

Постановка задачі. Для розв'язання задачі перевірки технічного завдання на розробку програмного

забезпечення чи програмно-технічного комплексу на повноту, несуперечливість, відповідність вимогам та стандартам предметної галузі необхідно провести аналіз відомих методів лексичного аналізу та розробити метод лексичного аналізу, що дозволяє проводити аналіз технічного завдання на розробку ПЗ з урахуванням функційних та нефункційних вимог.

Методи лексичного аналізу. Для складання технічних завдань використовуються різні мови світу, які групуються за своїм походженням та часом розвитку, наприклад: германська, слов'янська та ін. Мови, що входять до однієї групи не завжди мають однакові граматики. Розглянемо германську групу мов, до якої належать німецька та англійська мови. У граматиці англійської мови значення слів в реченнях визначаються їх розташуванням, тобто залежно від розташування, слово може бути іменником, прикметником чи дієсловом. В англійській мові визначений прямий порядок слів, зміна якого призведе до зміни змісту речення. В німецькій мові є можливість зміни порядку слів у реченні, при цьому речення не втрачає свого інформаційного змісту. Отже, описати групу мов однією граматикою [8] неможливо, тому що англійська мова є контекстнозалежною, німецька мова – контекстновільною.

Існує велика кількість лексичних аналізаторів та методів лексичного аналізу, але доцільно розглядати методи, що дозволяють описати мову якою буде складатися технічне завдання. Виділимо наступні методи лексичного аналізу, які задовольняють нашим вимогам:

1) метод лексичного аналізу, що використовує формальні граматики Хомського [8];

2) метод лексичного аналізу, що базується на link граматиці [9].

Лексичний аналіз з використанням формальних граматик Хомського. Розглянемо формальні граматики та методи їх опису. З класичного визначення граматики слідує, що граматика – це математична система, яка визначає мову [8].

Граматикою називається множина $G=(N,\Sigma,P,S)$, де:

- N – множина не термінальних символів, або не терміналів;
- Σ – множина термінальних символів або терміналів, яка не перетинається з N, $\Sigma \cap N=\varnothing$;
- P – підмножина множини $(N \cup \Sigma)^*N(N \cup \Sigma)^*x(N \cup \Sigma)^*$ елемент (α,β) множини P називається правилом, і записується у вигляді $\alpha \to \beta$;
- S – виділений символ з множини N, який називається початковим символом.

Граматики класифікуються за своїми правилами [8]. Нехай $G=(N,\Sigma,P,S)$ – граматика, тоді граматика G називається:

1) правосторонньою, якщо кожне правило з множини P має вигляд $A \to xB$, або $A \to x$, де A, $B \in N$, $x \in \Sigma^*$, наприклад: G=({<цифра>},{0, … , 9},{<цифра> $\to$ 0|, … , |9},{<цифра>}), під <цифра> слід розуміти, як єдиний не термінальний символ;

2) контекстновільною (безконтекстною) називається граматика, в якій кожне правило з P має вигляд: $A \to \alpha$, де $A \in N, \alpha \in (N \cup \Sigma)$, наприклад : $G=(\{E,T,F\}, \{a,+,*\},P,E)$, де P складається з правил: $E \to E+T|T \quad T \to T*F|F, T \to (E)|a$;

3) контекстнозалежною називається граматика, де кожне правило з P має вигляд: $\alpha \to \beta$, де $|\alpha| \le |\beta|$, в контекстнозалежній граматиці недопустимими є правила типу $A \to e$, оскільки вона має гарантувати рекурсивність створюваної нею мови;

4) граматика, яка не відповідає жодному з вище наведених правил, називається граматикою загального вигляду.

Розглянемо методи проведення лексичного аналізу на основі цих граматик. Існують наступні методи лексичного аналізу: непрямий і прямий лексичний аналіз [8].

Непрямий лексичний аналіз або лексичний аналіз з поверненнями полягає в послідовній перевірці версій про класи лексем [8]. Якщо перевірка поточної версії не підтверджується, то відбувається повернення назад по ланцюжку символів і здійснюється перевірка наступної версії. Аналіз чергової лексеми починається з запуску автомата, розташованого першим. Він читає перший символ a_i оброблюваного підланцюжка. Далі йде розпізнавання лексеми, у результаті чого читаються n символів. Поточним символом стає a_{i+n}. Якщо версія про лексему підтверджується, то сканер видає її і завершує роботу. Його наступний запуск починається з читання символу a_{i+n}, що знову передається першому автомату. Якщо ж автомат не може розпізнати лексему, то він видає сигнал відмови у блок управління та активізує наступний за пріоритетом автомат. Блок управління здійснює повернення до символу a_i і передає його новому активному автомату. Процес повернення здійснюється доти, поки лексема не буде розпізнана. Якщо жоден з автоматів не розпізнає лексему, то управління передається обробнику помилки, який видає лексему «помилка» і здійснює її нейтралізацію. У найпростішому випадку нейтралізація полягає у переході на наступний символ a_{i+1} перед повторним запуском лексичного аналізатора.

Прямий лексичний аналіз дозволяє визначити значення лексеми без повернень назад по ланцюжку символів [8]. Він будується на основі множини автоматів, поєднаних в один детермінований автомат, що розпізнає окремі лексеми. Такий автомат на кожному кроці читає один вхідний символ і переходить у наступний стан, що наближає його до розпізнавання поточної лексеми чи формування інформації про помилку. Для лексем, що мають однакові підланцюжки, автомат має спільні фрагменти, які реалізують

єдину множину станів.

На основі вище описаних методів лексичного аналізу працюють лексичні аналізатори Lex, Flex, Bison.

Лексичний аналіз з використанням link граматик. Link граматики належать до класу контекстновільних граматик [8–9]. Для опису мови граматикою використовуються зв'язки (link). На рис. 1 наведено приклад формування зв'язків між словами.

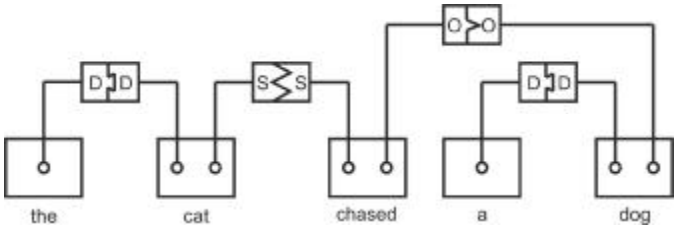

Рис. 1. Сполучення слів за змістовними зв'язками

Для link граматики характерні властивості [9]:

1) для кожного слова описуються зв'язки, за допомогою яких слова будуть сполучатися в речення;

2) після лексичного аналізу для кожного слова проставляються зв'язки з іншими словами з дотриманням лексичної та семантичної структури речення;

3) немає явного поділу на складові чи категорії.

В Link граматиці використовується словник, де для кожного слова визначенні всі можливі зв'язки, також вводиться поняття "з'єднувача" (connector), за допомогою з'єднувачів слова сполучаються за змістовними характеристиками.

Нижче наведенні основні типи змістовних зв'язків [9]:

1) S використовується для сполучення іменника з дієсловом;

2) використовується для сполучення дієслова з об'єктом, який його характеризує;

3) A використовується для сполучення прикметника з іменником;

Важливим є порядок розміщення зв'язків між словами у реченні – рис. 2

Рис. 2. Порядок сполучення слів у реченні

Метод лексичного аналізу з використанням link граматик використовує метод динамічного програмування, а саме знаходження оптимальної тріангуляції опуклого многокутника [11, с. 320].

Нижче представлений псевдокод алгоритму лексичного аналізу з використанням link граматики [12]:

```
/*Функція PARSE виконує лексичний аналіз тексту з використанням link граматик.*/
PARSE
1. t ← 0
2. for each disjunct d of word 0
3.         do if left[d] = NIL
4.                 then t ← t + COUNT(0, N, right[d],NIL)
5. return t
```

/*Функція на вхід приймає два слова L і R, та два вказівника l та r. Вказівник l вказує на список слів праворуч від слова L, вказівник r вказує на список слів ліворуч від слова R. Функція COUNT повертає число зв'язків, які можна проставити між словами L та R. */

```
COUNT (L,R,l,r)
1. if L = R + l
2.         then if l = NIL and r = NIL
3.                 then return 1
4.                 else return 0
5.         else total ← 0
6.                 for W ← L + 1 to R − 1
7.                         do for each disjunct d of word W
8.                                 do if ≠ NIL and left[d] ≠ NIL MATCH(l, left[d])
9.                                         then leftcount ← COUNT(L, W, next[l], next[left[d]])
10.                                        else leftcount ← 0
```

```
11.                     if right[d] ≠ NIL and r ≠ NIL and Match(right[d],r)
12.                         then rightcount ← COUNT(W, R, next[right[d]], next[r])
13.                         else rightcount ← 0
14.                     total ← total + leftcount * rightcount
15.                     if leftcount > 0
16.                         then total ← total + leftcount * COUNT(W, R, right[d],r)
17.                     if rightcount > 0 and L = NIL
18.                         then total ← total + rightcount * COUNT(L, W, l, left[d])
19.             return total
```

Функція PARCE – виконує лексичний аналіз з використанням link граматики.

Функція COUNT – виконує підрахунок кількості зв'язків між словами, що знаходяться в межах [L,…,R]. В якості параметрів передаються індекси слів L та R (початок та кінець речення), та два списки із вказівниками l та r, де розміщенні з'єднувачі для кожного слова, які повинні відповідати наступним вимогам [11]:

1) зв'язки не повинні перетинатися і мають задовольняти правилам, які визначені у словнику;

2) для кожного слова [L,…,R] мають бути жорстко визначенні зв'язки;

3) всі зв'язки та з'єднувачі для всіх слів з діапазону [L,…,R] або [L,…,M] і [M+1,…,R] мають відповідати правилу $L \le M < R$.

Порівняння методів лексичного аналізу. Проведемо порівняння методів лексичного аналізу на предмет вирішення задачі перевірки технічного завдання на розробку програмного забезпечення чи програмно-технічного комплексу на повноту, несуперечливість, відповідність вимогам та стандартам предметної галузь. Лексичний аналізатор технічних завдань повинен відповідати наступним вимогам:

- проводити морфологічний аналіз слів;
- виділяти термінальні символи;
- проводити синтаксичний аналіз;
- ідентифікувати функційні та нефункційні вимоги.

Виділимо основні переваги та недоліки кожного з методів, які представимо у таблиці 1.

Таблиця 1

Порівняння методів лексичного аналізу

№	Характеристики	Лексичний аналіз з використанням граматик Хомського	Лексичний аналіз з використанням Link граматики
1	Морфологічний аналіз слів	Не підтримується	Підтримується
2	Поділ на термінальні та нетермінальні символи	Підтримується	Не підтримується
3	Синтаксичний аналіз	Підтримується	Підтримується
4	Ідентифікація функційних та нефункційних вимог	Не підтримується	Не підтримується
5	Правосторонні граматики	Підтримується	Підтримується
6	Контекстновільні граматики	Підтримується	Підтримується
7	Контекстнозалежні граматики	Підтримується	Не підтримується
8	Граматики загального виду	Підтримується	Не підтримується
9	Задання граматики регулярними виразами	Підтримується	Немає засобів
10	Використання детермінованих та недетермінованих автоматів з магазинною пам'яттю	Підтримується	Не підтримується
11	Дерева виводу	Підтримується	Немає засобів
12	Визначення зв'язків між словами	Немає засобів	Підтримується
13	Збір статистичної інформації по зв'язках між словами у тексті	Немає засобів	Підтримується
14	Словник з правилами сполучення слів	Не підтримується	Підтримується

Набір характеристик, представлений в табл. 1, не є вичерпним.

Описані вище методи лексичного аналізу дозволяють лише частково розв'язати задачу лексичного аналізу технічного завдання на розробку програмного забезпечення. Задача ідентифікації функційних та нефункційних вимог з подальшою перевіркою на відповідність нормативній базі залишається не вирішеною.

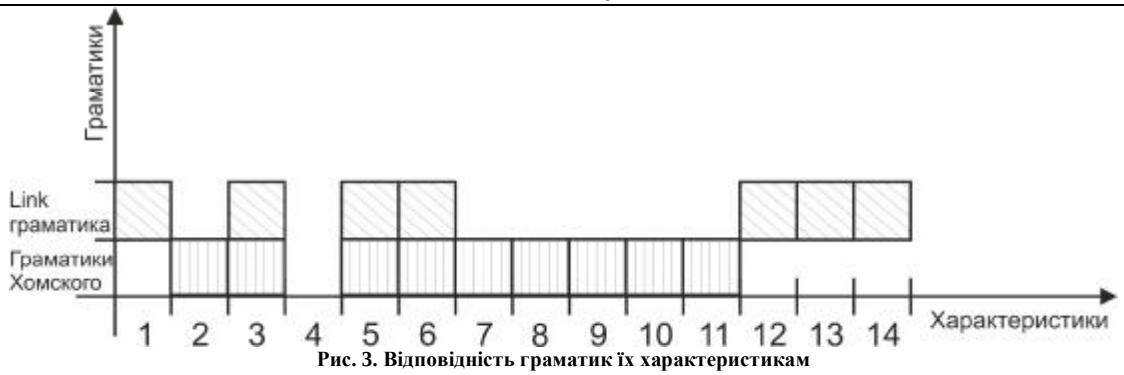

Рис. 3. Відповідність граматик їх характеристикам

Модифікований метод лексичного аналізу на основі правосторонньої граматики Хомського. Для розв'язання задачі перевірки технічного завдання на розробку програмного забезпечення на повноту, несуперечливість, відповідність вимогам предметної області та стандартам необхідно розробити комбінований метод лексичного аналізу, який буде в себе включати Граматики Хомського, Link-граматику, оскільки кожен окремо взятий метод лексичного аналізу не дозволяє в повній мірі вирішити цю задачу. Використання граматик Хомського дозволяє чітко розмежувати слова на термінальні та нетермінальні символи, виконувати синтаксичний аналіз слів. Використання Link-граматики дозволяє відстежити звязки між словами, виконувати морфологічний аналіз слів. Тому доцільним є розроблення комбінованого методу лексичного аналізу з додаванням модуля ідентифікації функційних та нефункційних вимог до програмного забезпечення (рис.4). Такий підхід дозволить підвищити ефективність розв'язання задачі лексичного аналізу технічного завдання на розробку програманого забезпечення.

Рис. 4. Модифікований метод лексичного аналізу

Висновок. Аналіз відомих методів лексичного аналізу показав, що вони базуються на використанні граматики Хомського або Link-граматики. Використання окремо кожної з граматик не дозволяє проводити лексичний аналіз технічних завдань на розробку програмного забезпечення, оскільки кожна з цих граматик окремо не має повного набору відповідних характеристик.

З метою підвищення ефективності розв'язання задачі оцінки повноти, несуперечливості та відповідності технічного завдання стандартам, визначення дотримання вимог та правил з предметної галузі проектування ПЗ, чи в повній мірі та несуперечливо визначені функційні та нефункційні вимоги, доцільним є розроблення модифікованого методу лексичного аналізу, що базується на використанні граматик Хомського та Link-граматики.

Література

1. IEEE Std 830 – 1998 IEEE Recommended Practice for Software Requirements Specifications (Практичні рекомендації по специфікації програмного забезпечення).

2. IEEE Std. 610.12 – 1990 IEEE Standard Glossary of Software Engineering Terminology

3. Petter L. Quantification and Traceability of Requirements / Petter L. , H. Eide – TDT4735 Software Engineering Depth Study – Fall – 2005

4. Кльоц Ю.П. Лексичний аналізатор технічних завдань на розробку критичного програмного забезпечення / Ю.П. Кльоц, В.С. Шевцов. – Труди XII МНПК "Сучасні інформаційні та електронні технології" – 2011 – С. 109.

5. IEC 60880 Software for computers in the safety systems of nuclear power stations (Програмне забезпечення для ЕОМ систем забезпечення безпеки АЕС).

6. ECSS–Q–80–03 Software Dependability and Safety Methods and Techniques (Методи і методики безпеки та надійності програмного забезпечення).

7. IEC 61513 Nuclear power plants – Instrumentation and control systems important for safety – General requirements for systems (АЕС – Інформаційно-керуючі системи, важливі для безпеки – Загальні вимоги до систем).

8. Ахо А. Теория синтаксическоо анализа, перевода, компиляции / А. Ахо, Дж. Ульман. – М., 1978. – Т. 1. – 613 с.

9. Daniel D. Sleator Parsing English with a Link Grammar / Daniel D. Sleator, Davy Temperley

10. Maggie Johnson Lexical Analysis / Maggie Johnson, Julie Zelenski – CS143 – Handout03 – 2008

11. Thomas H. Cormen Introduction to Algorithms / Thomas H. Cormen, Charles E. Leiserson & Ronald

L. Rivest – 1990

12. Sleator, D. D., D. Temperlay, “Parsing English with a Link Grammar”, Technical report CMU-91-196, Carnegie Mellon University, School of Computer Science, October 1991.

Надійшла 19.9.2011 р.

УДК 004.056.5: 518

А.А. КОБОЗЄВА, В.А. МОКРИЦЬКИЙ
Одеський національний політехнічний університет
Р.Г.МЕЛКУМЯН
Київський національний університет

ОСНОВИ МЕХАНІЧНОЇ МОДЕЛІ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ

В роботі запропоновано основи принципово нової моделі захищеної інформаційної системи, заснованої на механічній інтерпретації системи лінійних алгебраїчних рівнянь із залученням теорії збурень, що ніколи не робилося раніше.

The paper presents fundamentals of a new model of secure information system based on the interpretation of the mechanical system of linear algebraic equations involving the perturbation that has never been done before.

Ключові слова: модель системи, захист інформації, система лінійних алгебраїчних рівнянь, теорія збурень.

Вступ. Широкомасштабне використання обчислювальної техніки й телекомунікаційних систем, перехід до безпаперової технології, збільшення об’ємів оброблюваної інформації й розширення кола користувачів сьогодні призводять до якісно нових можливостей несанкціонованого доступу до ресурсів і даних інформаційних систем, що спричиняє підвищення актуальності вимоги захищеності будь-якої інформаційної системи.

Проблеми побудови систем захисту інформації (СЗІ) дуже широко обговорюються в сучасних відкритих джерелах [1– 5]. Висновок про необхідність створення системного комплексного підходу до захисту інформації на основі єдиного наукового базису [1, 2] ні в кого не викликає сумнівів. Побудова такого наукового базису неможлива без наявності адекватної математичної моделі СЗІ. Існуючі проблеми при створенні таких моделей добре відомі [1– 8]. Основні з них – це складність і різнорідність інформаційних систем, більшість із яких погано формалізуються, вимагають адаптації безпосередньо в процесі функціонування й управління.

Метою роботи є створення основ принципово нової механічної моделі захищеної інформаційної системи.

Як основні математичні інструменти виступають обчислювальна лінійна алгебра, теорія матриць, теорія збурень.

Для досягнення поставленої мети необхідно розв’язати наступні *задачі*:

- обрати механічну інтерпретацію для СЗІ; встановити відповідність між елементами інформаційної системи й параметрами її механічної моделі;
- провести математичну формалізацію механічної моделі;
- формалізувати вимогу стійкості СЗІ стосовно передбачуваних атак у рамках механічної моделі;
- отримати формальні умови стійкості СЗІ до передбачуваного супротивника.

Формалізація системи захисту інформації. Як механічну модель захищеної інформаційної системи завдяки аналогіям, що наведені на рис. 1, логічно розглянути пружну статичну систему S, закріплену на краях [9], наприклад, стрижень.

Оберемо на пружному стрижні скінченну множину точок, кожна з яких буде відповідати конкретному засобу захисту, наявному в розпорядженні інформаційної системи, які для зручності занумеруємо в порядку зліва направо: $1,2,...,n$. Будемо розглядати прогини $b_1, b_2, ..., b_n$ точок $1,2,...,n$ системи S під впливом сил $x_1, x_2, ..., x_n$, прикладених у цих точках. Тоді $x_1, x_2, ..., x_n$ інтерпретуються як атаки, спрямовані безпосередньо на засоби захисту $1,2,...,n$, а $b_1, b_2, ..., b_n$ – результати впливу атак на засоби захисту.

Припустимо, що

сили, що впливають на стрижень, і переміщення (прогини) його складових перпендикулярні вхідному (недеформованому) положенню стрижня, тобто паралельні між собою, а тому повністю визначаються своїми алгебраїчними величинами (рис. 2);

має місце принцип лінійного накладення сил [9].

Нехай a_{ik} – прогин стрижня в точці i під впливом одиничної сили, прикладеної в точці k, або коефіцієнт впливу точки k на i, $i,k = \overline{1,n}$. При спільній дії довільних сил $x_1, x_2, ..., x_n$ на стрижень S