УДК 621.391

В. А. Лужецький, д. т. н., доц.;
М. В. Моторний, асп.

МОДЕЛІ УЩІЛЬНЕННЯ ТЕКСТІВ НА ОСНОВІ АБСТРАКТНИХ СИНТАКСИЧНИХ ДЕРЕВ

Вступ

Розповсюдження програмного коду у мережі підвищило інтерес до проблем його компактного представлення. Ущільнення є невід'ємною частиною в разі передавання коду в мережах з низькою пропускною здатністю, зокрема в бездротових (wireless) мережах. Ущільнення також є важливим у тому випадку, коли програмні модулі зберігаються на машині кінцевого споживача [1].

Серед наявних методів ущільнення програмного коду можна виділити три групи:

1) методи, що модифікують об'єктний код з використанням прийомів оптимізувальної компіляції. Новий код еквівалентний старому за функціональністю, але менший за розміром [2];

2) методи, що ущільнюють об'єктний код з урахуванням деяких статистичних характеристик формату інструкцій конкретної машини [3, 4];

3) методи, що ущільнюють абстрактні синтаксичні дерева (АСД), отримані з вихідного коду, за допомогою словникових [1] чи статистичних методів [5, 6].

Найефективніше ущільнення досягається третім сімейством методів, тому що побудова АСД відбувається в суворій відповідності з абстрактною граматикою (АГ), що дозволяє досягти результатів, які перевершують універсальні компресори. Однак істотним недоліком методів даної групи є втрата ними коментарів та інформації про форматування, що унеможливлює їхнє використання в процесі резервного копіювання, а також поширення вихідного коду між розробниками.

У даній статті описується підхід, що усуває зазначений недолік, а також наводяться моделі методів ущільнення вихідного коду.

Метод ущільнення вихідного коду на основі АСД

Вихідні тексти комп'ютерних програм є реченнями формальних мов, представлених як літеральні рядки. Звичайні контекстно-вільні (конкретні) граматики змішують інформацію про суть програми з інформацією, що підвищує читаність тексту для людини (чи машини). Для узагальненого представлення програм, написаних різними мовами програмування, використовуються абстрактні синтаксичні дерева [7]. АСД – це дерева, що абстрагуються від несуттєвих деталей мови програмування, наприклад символів, що обрамляють блок операторів. Кожне АСД відповідає деякій АГ, в точності як вихідний код відповідає граматиці конкретної мови. Наприклад, таким фрагментам коду на мовах C і Pascal

```
if (a == b)
    f ();
else
    g ();
```

```
if a = b then
    f
else
    g;
```

відповідає АСД, зображене на рис. 1.

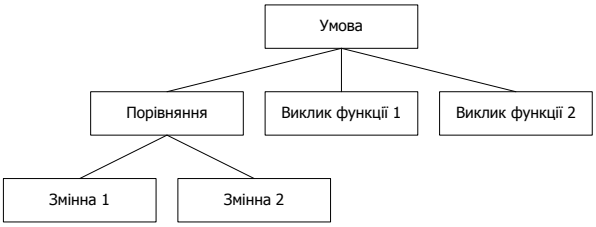

Рис. 1. Приклад абстрактного синтаксичного дерева

Лексеми, якими оперує метод ущільнення на основі АСД, розділяються на такі види.

1) Значущі лексеми. Це лексеми, що входять до складу граматики мови і оброблюються синтаксичним аналізатором.

2) Незначущі лексеми. Це лексеми, що входять до складу лексики мови, але не входять до складу граматики. До них належать роздільники, коментарі та ін.

Значущі лексеми у свою чергу розділяються

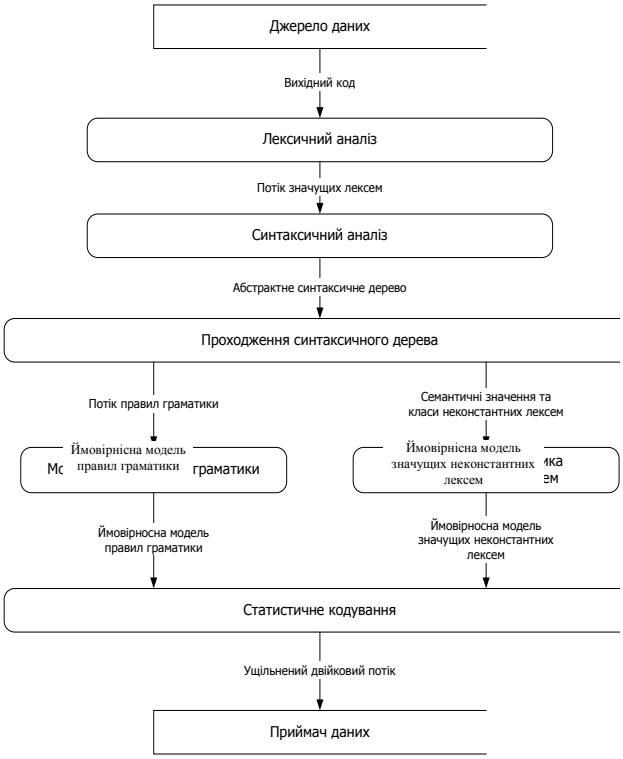

Рис. 2. Діаграма Гейна-Серсона процесу ущільнення вихідного коду на основі АСД

на такі види.

1) Константні лексеми. Це лексеми, яким відповідає одне і тільки одне семантичне значення. До них належать ключові слова мови, позначення базових операцій та ін.

2) Неконстантні лексеми. Це лексеми, яким відповідає кілька семантичних значень. До даного виду належать ідентифікатори, числа та ін. Неконстантні лексеми за граматичним контекстом поділяються на класи (наприклад, ідентифікатори керувальних змінних циклів), що дозволяє досягти кращого коефіцієнта ущільнення.

Процес ущільнення вихідного коду на основі АСД описується діаграмою Гейна-Серсона [8] (рис. 2). У процесі лексичного аналізу послідовність символів вихідного тексту перетворюється у послідовність лексем. Синтаксичний аналіз перетворює отриману послідовність лексем в абстрактне синтаксичне дерево. В результаті проходження дерева формуються ймовірносні моделі правил граматики, семантичних значень і класів лексем. АСД кодується на підставі отриманих моделей деяким статистичним кодером.

Кращі наявні реалізації використовують для моделювання правил граматики метод PPM* (передбачення за частковим збігом з необмеженим контекстом) [9]. В разі ущільнення вузла дерева контекстами вибираються всі батьківські вузли, включаючи кореневий. Стратегія зважування ігнорує передбачення, отримані від контекстів нульового порядку. Кодування здійснюється арифметичним кодером.

Модель ущільнення на основі АСД

Для побудови моделі введемо у розгляд такі поняття.

***Означення* 1.** Пара $\mathbf{m} = \{\mathbf{S}, \mathbf{P}\}$, де **S** – множина символів, **P** – множина предикатів, називається *текстовою системою*.

Означення 2. Підмножина $\mathbf{S}_i$, $i = \overline{0, N}$ множини $\mathbf{S} = \{\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_N\}$ називається *підмножиною символів i-го рівня*.

Множина символів $\mathbf{S}_i$ складається з підмножин символів $\mathbf{S}_i = \{\mathbf{S}_{i,n}\}$, $n = \overline{1, N_i}$ з різними визначальними властивостями. Підмножина $\mathbf{S}_{i,n} = \left\{s_{i,n}^m\right\}$, $m = \overline{1, M_{i.n}}$, містить символи однієї визначальної властивості $P(\mathbf{S}_{i,n})$.

Символи підмножин $\mathbf{S}_{N,n}$ мають деякі характеристики $\chi_n\left(s_{N,n}^m\right)$.

***Означення* 3.** Характеристика, що визначає частоту появи символу в тексті, називається *статистичною характеристикою*.

***Означення* 4.** Підмножина $\mathbf{P}_i$, $i = \overline{1, N}$ множини $\mathbf{P} = \{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_N\}$ називається *підмножиною предикатів i-го рівня*.

Множина предикатів $\mathbf{P}_i$ складається з предикатів *i*-го рівня $\mathbf{P}_i = \{p_{i,k}\}$, $k = \overline{1, K_i}$.

***Означення* 5.** Кортеж довжини L_i, що складається з символів *i*-го рівня $\mathbf{T}_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,L_i}\}$ називається *текстом i-го рівня*.

Довжина тексту *i*-го рівня дорівнює довжині кортежу $\mathbf{T}_i$.

***Означення* 6.** Кортеж вигляду $\mathbf{T}^* = \{k_1, k_2, \ldots, k_H\}$, де k_h — код, що складається з символів множини $\mathbf{S}_0$, називається *ущільненим текстом*.

Довжина ущільненого тексту дорівнює

$$L^* = \sum_{h=1}^{H} l_h,$$

де l_h – довжина *h*-го коду.

***Означення* 7.** Перетворення вигляду

$$\mathbf{T}_0 \xrightarrow{\boldsymbol{\psi}} \mathbf{T}^* \tag{1}$$

у разі виконання умови $L^* < L_0$ називається *ущільненням тексту*.

Перетворення (1) можна представити як композицію перетворень $\boldsymbol{\psi} = = \mathbf{P}_1 \circ \mathbf{P}_2 \circ \ldots \circ \mathbf{P}_N \circ \boldsymbol{\varphi}$.

Перетворення $\boldsymbol{\varphi}$ задає множину функцій кодування $\boldsymbol{\varphi} = \{\varphi_1, \varphi_1, \ldots, \varphi_F\}$, що враховують характеристики χ_n символів тексту $\mathbf{T}_N$: $k_i = \varphi_n(\chi_n(s_{Nn}^m))$, $i = \overline{1, W_N}$, де k_i – код *i*-го символу тексту $\mathbf{T}_N$.

***Означення* 8.** Перетворення вигляду

$$\mathbf{T}^* \xrightarrow{\boldsymbol{\psi}^{-1}} \mathbf{T}_0' \tag{2}$$

називається відновленням вихідного тексту за ущільненим текстом.

Метод ущільнення забезпечує ущільнення без втрат, якщо виконується умова $\mathbf{T}_0' = \mathbf{T}_0$. Коли $\mathbf{T}_0' \sim \mathbf{T}_0$, причому в $\mathbf{T}_0'$ збережені всі істотні характеристики і властивості $\mathbf{T}_0$, то мова йде про ущільнення з втратами.

Перетворення (2) є композицією перетворень $\boldsymbol{\psi}^{-1} = \boldsymbol{\varphi}^{-1} \circ \mathbf{P}_N^{-1} \circ \mathbf{P}_{N-1}^{-1} \circ \ldots \circ \mathbf{P}_1^{-1}$.

В термінах наведеної узагальненої моделі ущільнення на основі АСД описується перетворенням вигляду

$$\mathbf{T}_0 \xrightarrow{\mathbf{P}_1} \mathbf{T}_1 \xrightarrow{\mathbf{P}_2} \mathbf{T}_2 \xrightarrow{\mathbf{P}_3} \mathbf{T}_3 \xrightarrow{\mathbf{P}_4} \mathbf{T}_4 \xrightarrow{\boldsymbol{\varphi}} \mathbf{T}^*$$

і текстовою системою

$$\mathbf{m} = \left\langle \{\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4\}, \{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \boldsymbol{\varphi}\} \right\rangle.$$

Множина предикатів $\mathbf{P}_1$ описує формування байтів з бітів. Множини $\mathbf{P}_2$ і $\mathbf{P}_3$ задають правила лексичного і синтаксичного аналізу відповідно. Множина $\mathbf{P}_4$ задає правила формування символів відповідно до марковської моделі джерела.

Множини символів $\mathbf{S}_0$ та $\mathbf{S}_1$ є множинами бітів та байтів відповідно. Множина $\mathbf{S}_2$ — це множина термінальних символів граматики, що складається з двох підмножин $\mathbf{S}_2 = \{\mathbf{S}_{2,1}, \mathbf{S}_{2,2}\}$. Підмножина $\mathbf{S}_{2,1}$ відповідає неконстантним, а $\mathbf{S}_{2,2}$ – константним лексемам. Множина $\mathbf{S}_3$ складається з підмножин $\mathbf{S}_3 = \{\mathbf{S}_{3,1}, \mathbf{S}_{3,2}\}$, де $\mathbf{S}_{3,1} = \mathbf{S}_{2,1}$, а $\mathbf{S}_{3,2}$ – множина нетермінальних символів граматики. Множина $\mathbf{S}_4$ формується на основі змішаної стратегії зважування ймовірностей символів, обумовленої методом ущільнення PPM*. Формування символів, що відповідають *i*-му порядку моделі описується формулою

$$\mathbf{S}_4 = \mathbf{S}_4 \cup \left\{ s_{4,i}^j, s_{4,i}^{j+1} \right\},$$

де $s_{4,i}^j$ – символ, що у термінах PPM називається контекстом (він утворюється приєднанням поточного символу тексту $\mathbf{T}_3$ до $s_{4,i-1}$), $s_{4,i}^{j+1}$ – символ, що відповідає символу переходу до моделі мен-

шого порядку.

Функція $\boldsymbol{\varphi}$ описує арифметичне кодування з урахуванням статистичної характеристики χ_n.

В описаній моделі вираз

$$\mathbf{T}_i \xrightarrow{\mathbf{P}_{i+1}} \mathbf{T}_{i+1} \xrightarrow{\mathbf{P}_{i+1}^{-1}} \mathbf{T}_i \qquad (3)$$

справедливий для всіх *i*, крім *i* = 1. Як було зазначено раніше, вихід лексичного аналізатора складається зі значущих лексем, що подаються на вхід синтаксичного аналізатора, та незначущих лексем, що відкидаються, тому що не впливають на граматичну структуру тексту. Останнє й обумовлює невиконання рівностей $\mathbf{T}_1 = \mathbf{T}_1'$ і $\mathbf{T}_0 = \mathbf{T}_0'$, з чого випливає, що модель забезпечує ущільнення з втратами.

На практиці це обмежує галузь застосування методів ущільнення на основі АСД. Реалізації зводяться до спеціалізованих компресорів байта-коду і внутрішніх модулів (back-end) для компіляторів чи інтерпретаторів [9]. Незначущі лексеми, до яких належать пробіли, символи табуляції і переведення рядка, а також коментарі, хоча і не становлять інтерес для генератора об'єктного коду, але можуть містити керувальну інформацію для інших інструментальних засобів, таких як генератори документації Javadoc і Doxygen. Крім того, багато великих компаній додержуються власного стилю форматування, і, отже, будуть змушені відмовитися від використання методів ущільнення на основі АСД у системах керування вихідним кодом.

Модифікована модель ущільнення на основі АСД

З метою усунення зазначених недоліків, пропонується метод ущільнення з урахуванням незначущих лексем, що забезпечує ущільнення без втрат. Для цього визначимо класи незначущих лексем і введемо відповідні додаткові підмножини і предикати.

За функціональним навантаженням незначущі лексеми розділяються на такі класи.

1) Роздільники, що визначають відступи (indentation). До них належать пробіли і символи табуляції на початку рядків, а також символи переведення рядка.

2) Роздільники, що підвищують читабельність тексту. Це в основному пробіли, що вставляються після ком, крапок, а також до знаків арифметичних дій і іноді дужок.

3) Інформаційні коментарі. Це коментарі, що виконують винятково інформативну роль, містять довільний текст.

4) Керувальні коментарі. Такі коментарі містять команди, що керують компілятором чи іншими засобами оброблення вихідного коду (препроцесорами, генераторами документації). Даний клас відрізняється від попереднього наявністю граматики.

Клас роздільників, що визначають відступи, добре передбачується терміналами «{» (початок блоку операторів у мові С), «}» (кінець блоку операторів) і «;» (кінець оператора). Для ефективного ущільнення цього класу потрібно провести попередній аналіз вихідного коду з метою з'ясувати такі параметри відступів у тексті.

1) *C* – вид символу, що використовується для оформлення відступів. Таким видами символів звичайно є пробіли чи знаки табуляції.

2) *N* – кількість символів, що використовується для оформлення одного вкладеного блоку. Типові значення – 1, 4 чи 8.

3) *T* – розмір табуляції, прийнятий у даному тексті. Звичайно один знак табуляції еквівалентний 4 чи 8 пробілам.

Якщо отриманим параметрам строго відповідають усі відступи тексту, що ущільнюється (що практично завжди справедливо для такої мови, як Python), то у вихідний текст записуються тільки дані параметри. В іншому випадку параметри вибираються так, щоб вони відповідали більшості відступів тексту, що ущільнюється. Для відступів, що не збігаються, створюється словник, що ущільнюється будь-яким універсальним методом (виходячи з лексичної структури роздільників,

слід припустити, що найкращий коефіцієнт ущільнення буде досягнутий застосуванням методу RLE).

Алгоритм відновлення відступів за параметрами складається з трьох альтернатив.

1) Лексема, що починає вкладений блок, збільшує значення поточного відступу на *N* символів.

2) Лексема, що закінчує вкладений блок, зменшує значення поточного відступу на *N* символів.

3) Лексема, що визначає кінець оператора, формує символ переведення рядка, а потім кількість знаків *C* в поточному відступі.

Параметр *T* використовується для усунення ефекту, що створюється деякими інструментальними засобами редагування. Даний ефект полягає в наявності більше одного символу, що претендує на параметр *C*, наприклад, пробілу і символу табуляції, що погіршує коефіцієнт ущільнення. Заміна *T* пробілів на один символ табуляції чи навпаки дозволяє збільшити кількість відступів з однаковими параметрами.

В разі ущільнення роздільників, що підвищують читаність тексту, слід враховувати, що в одних випадках вони краще передбачаються за попереднім контекстом (пробіл після крапки чи коми), а в інших – за наступним (пробіл до тире чи знака додавання). Тому для ефективного кодування роздільника підтримується дві ймовірносні моделі першого порядку за типами двох навколишніх значущих лексем – для попереднього і наступного контексту відповідно. Початково, одна з цих моделей, наприклад, перша, вибирається за активну. Статистичний кодер оновлює обидві моделі, але для ущільнення використовує тільки активну. Якщо в процесі кодування з'ясовується, що пасивна модель краще передбачує роздільники, то відбувається взаємна заміна активної і пасивної моделей.

Сукупність усіх роздільників визначає стиль кодування. Задавши наперед визначені стилі, можна прискорити сходження до оптимального коду. Крім того, в разі ущільнення набору текстів, що належать до одного програмного продукту, слід очікувати однакового стилю кодування, властивого усім текстам набору.

Вмістом інформаційних коментарів є текст природною мовою. Оскільки конструювання граматик для природних мов значною мірою затруднено наявністю великої кількості винятків [10], то для ущільнення інформаційних коментарів можна використовувати будь-який універсальний метод. На теперішній час краще ущільнення текстів забезпечується сімейством методів PPM п'ятого-восьмого порядку [11]. Покращення коефіцієнта ущільнення можна досягти, з огляду на спосіб обрамлення інформаційних коментарів. Нижче подані способи обрамлення коментарів, що використовуються Sun Microsystems для мови Java і Oracle для мови XML.

```
/*                                          <!—| Текст XML-коментаря.
 * Текст Java-коментаря.                    +—>
   */
```

Коментарі керування або цілком визначаються деякою граматикою (директиви компілятора Delphi), або містять суміш формальних фраз і тексту природною мовою (документувальні коментарі Javadoc і Doxygen). Для ущільнення частини коментарів, що відповідають граматиці, можна використовувати методи на основі АСД. У деяких випадках для збільшення коефіцієнта ущільнення слід зробити граматику коментарів частиною граматики основної мови, оскільки це дозволить об'єднати словники неконстантних лексем. Наступний приклад мовою Java демонструє таку можливість (імена і порядок параметрів у коментарі та оголошенні методу збігаються).

```
/**
* Пересуває об’єкт у вказану точку.
* @param x нова координата X
* @param y нова координата Y
*/
public abstract void setLocation (int x, int y);
```

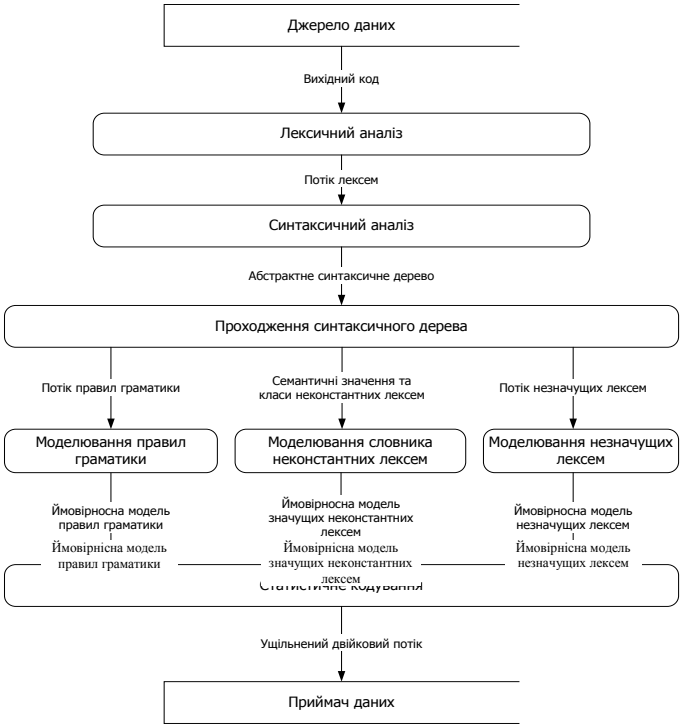

Рис. 3. Діаграма Гейна-Сересона модифікованого процесу ущільнення вихідного коду на основі АСД

Частина коментарів, що є текстом природною мовою, ущільнюється аналогічно інформаційним коментарям.

Діаграма модифікованого процесу ущільнення показана на рис. 3. Тут введений новий процес для моделювання незначущих лексем і відповідні потоки даних.

З позицій узагальненої моделі ущільнення текстів описані вище модифікації представляються введенням нових предикатів у множини $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{P}_4$, і нових підмножин символів у множини $\mathbf{S}_2$, $\mathbf{S}_3$, $\mathbf{S}_4$.

До множини предикатів $\mathbf{P}_2$ будуть входити предикати, що задають формування символів, які відповідають не тільки значущим, але і незначущим лексемам. Множина $\mathbf{P}_3$ буде містити предикати, згідно яким підмножини незначущих лексем множини $\mathbf{S}_2$ входять до множини $\mathbf{S}_3$. Множина $\mathbf{P}_4$ доповниться предикатами відповідних моделей ущільнення, використовуваних для кожного класу незначущих лексем.

Множина $\mathbf{S}_2$ набуде вигляду $\mathbf{S}_2 = \{\mathbf{S}_{2,1}, \mathbf{S}_{2,2}, \mathbf{S}_{2,3}, \mathbf{S}_{2,4}, \mathbf{S}_{2,5}, \mathbf{S}_{2,6}\}$, де $\mathbf{S}_{2,1}$, $\mathbf{S}_{2,2}$ – підмножини значущих лексем, а $\mathbf{S}_{2,3}$, $\mathbf{S}_{2,4}$, $\mathbf{S}_{2,5}$, $\mathbf{S}_{2,6}$ – підмножини незначущих лексем. Підмножина $\mathbf{S}_{2,1}$ відповідає неконстантним, а $\mathbf{S}_{2,2}$ – константним лексемам. Підмножина $\mathbf{S}_{2,3}$ є множиною роздільників, що визначають відступи, $\mathbf{S}_{2,4}$ — множиною роздільників, що підвищують читаність тексту, $\mathbf{S}_{2,5}$ — множиною інформаційних коментарів, $\mathbf{S}_{2,6}$ – множиною керуючих коментарів. Множина $\mathbf{S}_3$ також доповниться відповідними множинами незначущих лексем і набуде вигляду $\mathbf{S}_3 = \{\mathbf{S}_{3,1}, \mathbf{S}_{3,2}, \mathbf{S}_{3,3}, \mathbf{S}_{3,4}, \mathbf{S}_{3,5}, \mathbf{S}_{3,6}\}$, де $\mathbf{S}_{3,i} = \mathbf{S}_{2,I}$, $i = \overline{3,6}$. Множина $\mathbf{S}_4$ доповниться підмножинами відповідних моделей ущільнення, використовуваних для кожного класу незначущих лексем.

Оскільки в модифікованій моделі вираз (3) справедливий для всіх $i = \overline{0,3}$, то модель описує ущільнення без втрат.

Висновки

1. Наявні методи ущільнення на основі АСД забезпечують ущільнення з втратами, що істотно обмежує галузь їхнього застосування.

2. Запропонована модифікація методу ущільнення на основі АСД, що зберігає незначущі лексеми, і тим самим забезпечує ущільнення без втрат.

СПИСОК ЛІТЕРАТУРИ

1. Franz M., Kistler T. Slim Binaries // Communications of the ACM. — 1997. — V. 40(12). — P. 87—94.

2. Debray S., Evans W., Muth R. Compiler techniques for code compression // Workshop on Compiler Support for System Software. — 1999. — P. 117—123.

3. Pugh W. Compressing java classfiles // ACM SIGPLAN Conference on Programming Language Design and Implementation. — 1999. — P. 247—258.

4. Lucco S. Split stream dictionary program compression // Proceedings of the ACM Conference on Programming Language Design and Implementation. — 2000. — P. 56—59.

5. Cameron R. Source encoding using syntactic information source models // IEEE Transactions on Information Theory. — 1988. — V. 34(4). — P. 843—850.

6. Eck P., Changsong X., Matzner R. A new compression scheme for syntactically structured messages (programs) and its applications to Java and the Internet // Data Compression Conference. — 1998. — P. 542.

7. Aho A., Sethi R., Ullman J. Compilers: principles, techniques and tools. — N. Y.: Addison-Wesley, 1986. — 347 p.

8. Gane C., Sarson T. Structured Systems Analysis: Tools and Techniques. — N. Y.: IST, Inc., 1977. — 276 p.

9. Stork C. H., Haldar V., Franz M. Generic Adaptive Syntax-Directed Compression for Mobile Code: Technical Report 00-42 / Department of Information and Computer Science, University of California, Irvine, 2000. — 17 p.

10. Bell T., Witten I. H., Cleary J. G. Modeling for text compression // ACM Computing Surveys. — 1989. — V. 21, 4. — P. 557—591.

11. Cleary J. G., Witten I. H. Data compression using adaptive coding and partial string matching // IEEE Trans. Commun. — 1984. — V. COM-32, 4. — P. 396—402.

Рекомендована кафедрою захисту інформації

Надійшла до редакції 16.09.03
Рекомендована до опублікування 3.06.04

Лужецький Володимир Андрійович — завідувач кафедри; ***Моторний Максим Володимирович*** — аспірант.

Кафедра захисту інформації, Вінницький національний технічний університет.