[6] Аракелян Э.К. Оптимальное распределение нагрузки между параллельно работающими энергетическими блоками с учетом фактора надежности / Э.К. Аракелян, Н.В. Мань, Н.Ч. Хунг // Вестник МЭИ. 1997. № 3. С. 15–20.
[7] Плетнев Г.П. Автоматическое управление вредными выбросами в переменных режимах ТЭС / Г.П. Плетнев, Т.Е. Щедеркина, А.С. Горбачев //Теплоэнергетика, 1995. № 4. – С. 54–56.
[8] Добровольская Т.С. Определение оптимального алгоритма работы оборудования методом динамического программирования / Т. С. Добровольская // Восточно-Европейский журнал передовых технологий. – 2014. – Т. 5, № 8 (71). – С. 53–58.
[9] Економіка підприємства : Підручник / За заг. ред. С.Ф. Покропивного. – 2-ге вид., перероб. та доп.- К.: КНЕУ, 2000.- 528 с. : іл.
[10] Тепловой расчет котлов. Нормативный метод. Санкт-Петербург: 1998. Изд. 3-е, перераб.

**References**

[1] N. N. Kozhevnikov. *Ekonomika i upravlenie energeticheskimi predpriyatiyami: Uchebnik dlya stud. vyissh. ucheb. zavedeniy*. Moscow Izdatelskiy tsentr «Akademiya», 432 p., 2004.
[2] V.A. Makarchyan, *Programmnyiy kompleks raspredeleniya nagruzok TETs so slozhnyim sostavom oborudovaniya, shemami otpuska tepla i elektroenergii*, Teploenergetika. no. 5, pp. 71-77, 2013.
[3] K.M. Reymov. *"Opredelenie kriteriya optimalnogo raspredeleniya aktivnoy nagruzki mezhdu agregatami TES"*, Avtomatizirovannyie tehnologii i proizvodstva. no. 2 (12), pp. 25-27, 2016.
[4] E. K. Arakelyan i dr. *"Problemyi sovremennyih avtomatizirovannyih sistem upravleniya tehnologicheskim protsessom na baze programmno-tehnicheskih kompleksov i vozmozhnyiy put ih resheniya"* , Vestnik MEI no 1. pp. 15-20, 2014.
[5] I. P. Ozerova *"Obosnovanie neobhodimosti raspredeleniya nagruzok mezhdu agregatami TES na baze kompleksnogo kriteriya"*, Izvestiya Tomskogo politehnicheskogo universiteta [Izvestiya TPU].: trudyi II-go seminara vuzov Sibiri i Dalnego Vostoka po teplofizike i teploenergetike, Tomsk, 24-25 oktyabrya 2001, 2002 vol. 305 pp. 108-114.
[6] E.K. Arakelyan, N.V. Man and N.Ch. Hung *"Optimalnoe raspredelenie nagruzki mezhdu parallelno rabotayuschimi energeticheskimi blokami s uchetom faktora nadezhnosti"*, Vestnik MEI. no. 3, pp. 15-20, 1997.
[7] G.P. Pletnev, T.E. Schederkina and A.C. Gorbachev *"Avtomaticheskoe upravlenie vrednyimi vyibrosami v peremennyih rezhimah TES"*, Teploenergetika, no. 4, pp. 54-56, 1995.
[8] T.S. Dobrovolskaya *"Opredelenie optimalnogo algoritma rabotyi oborudovaniya metodom dinamicheskogo programmirovaniya"*, Vostochno-Evropeyskiy zhurnal peredovyih tehnologiy. vol. 5, no. 8 (71), pp. 53–58, – 2014.
[9] S.F. Pokropivniy. *Ekonomika pidpryemstva* : *Pidruchnik* 2nd ed. Kiev : KNEU, 2000. – 528 p.
[10] *Teplovoy raschet kotlov. Normativnyiy metod*. 3rd ed. Sankt-Peterburg 1998.

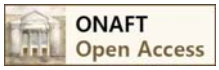
УДК 004.435

# ПАРАДИГМА ПОДАННЯ ЛІНГВІСТИЧНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОПОМОГОЮ ПОРОДЖУВАЛЬНИХ ГРАМАТИК

**С. Великодний**[1], **О. Тимофєєва**[2]

[1]Національний університет «Одеська морська академія», [2]Одеський державний екологічний університет
E-mail: [1]velykodniy@gmail.com; [2]hellena1985th@gmail.com






DOI: 10.15673/atbp.v9i3.720

*Анотація: В статті розглянуто створення системи понять, що формують парадигму реінжинірингу інформаційних технологій, який необхідний у випадку їх еволюційного розвитку. Лінгвістичне забезпечення*

*інформаційних технологій розглядає побудову програмної системи за допомогою однієї або декількох (узгоджених між собою) мов програмування, кожна з яких заснована на правилах конкретної граматики. Математичний апарат породжувальних граматик дозволяє описати процес переведення програмної системи, яка написана однією мовою програмування, на іншу визначену мову. Створена парадигма дозволяє працювати з багаторівневими інформаційними технологіями, складові частини яких написано різними мовами програмування. Сформована у статті парадигма, з наукової точки зору, ляже в основу методології реінжинірингу програмних систем, а з практичної – стане у пригоді системним програмістам, які працюють із мультимовними надбудовами програмних систем, що набувають еволюційного розвитку із плином часу та вдосконалення в процесі використання.*

*Abstract: The article describes the creation of a system's concepts that form the paradigm of reengineering information technologies. Linguistic support of information technology considers the construction of a software system using one or more mutually agreed programming languages. Each programming language is based on the rules of a particular grammar. The mathematical apparatus of generative grammars allows us to describe the process of translating a program system written in one programming language into another specific language. The created paradigm allows you to work with multi-level information technologies, the parts of which are written in different programming languages. The paradigm formed in the article, from the scientific point of view, is laid in the basis of the methodology of information technology reengineering, and from the practical point of view it will be necessary for system programmers working with multilanguage superstructures of software systems that evolve over time and are improved in the process of exploitation.*

Ключові слова: програмна система, генеративна лінгвістика, мова програмування, граматика, продукція, аксіома, алфавіт, ланцюжок, символ.

Keywords: program system, generative linguistics, programming language, grammar, production, axiom, alphabet, chain, symbol.

## Вступ

Основою будь-якого суспільства є інформація. Без обміну та аналізу інформацією неможливий не тільки розвиток живого організму, але й його існування. Існує багато засобів передачі інформації між істотами, здебільшого за допомогою органів відчуттів. Щодо людини, то одним із основних засобів передачі інформації є мова, причому не тільки її звукова компонента (адже інформація може бути передана письмовою мовою, мовою жестів, шрифтом Брайля тощо).

Сучасною основою прискорення обміну інформацією між людством є інформаційні технології, технічне забезпечення яких складається із багатьох різноманітних пристроїв, що, по суті, є модифікаціями комп'ютера. Ці пристрої також обмінюються інформацією у вигляді даних (не тільки між собою, але й з людиною – користувачем або оператором), причому, в залежності від рівнів представлення даних, інформація може подаватися у вигляді: двійковий, вісімковий, десятковий, шістнадцятковий коди; машинний код; низько- та високорівневі мови програмування тощо.

## Аналіз основних наукових досягнень і літератури

Наукову основу будь-якої мови (в тому числі й мови програмування) становить лінгвістика, що вивчає закони, моделі та правила мови. Особливою галуззю лінгвістики, яку варто застосовувати до будови мов програмування є генеративна лінгвістика [1], засновником якої був Аврам Ноам Чомські (Avram Noam Chomsky, у радянські часи зустрічалася інтерпретація «А. Н. Хомський»), що створив революцію у мовознавстві [2, 3].

За способом завдання вірних ланцюжків, формальні граматики розділяють на породжувальні та розпізнавальні. До породжувальних відносяться ті граматики, за допомогою яких можна побудувати будь-який вірний ланцюжок з вказівкою його структури та неможна побудувати жодного невірного ланцюжка. Вперше, поняття породжувальної (генеративної) граматики було запропоновано А. Н. Чомські [4]. Розпізнавальна граматика – це граматика, що дозволяє встановити вірність довільно обраного ланцюжка, та, у разі якщо він вірний, з'ясувати його будову.

Лінгвістичне забезпечення інформаційних технологій розглядає побудову програмної системи за допомогою однієї або декількох (узгоджених між собою) мов програмування, кожна з яких заснована на правилах конкретної граматики [5].

До формальних мов відносяться, зокрема, штучні мови для спілкування оператора з комп'ютером (мови програмування) [6].

Мета дослідження – створити систему понять, що формує парадигму реінжинірингу інформаційних технологій, яка дозволить працювати з багаторівневими програмними системами, складові частини яких написано різними мовами програмування.

## Постановка задачі

Для завдання опису формальної мови необхідно, по-перше, вказати алфавіт, тобто сукупність об'єктів, що називаються символами (або літерами), кожен я яких можна відтворювати у необмеженій кількості примірників, та, по-друге, завдати формальну граматику мови, тобто перерахувати правила, за якими з символів вибудовуються їх послідовності, що належать до визначеної мови.

Будь-яка мова програмування являє собою безліч ланцюжків у деякому кінцевому алфавіті. У лінгвістиці замість

терміну «алфавіт» використовується термін «словник» тому, що елементи, з яких він складений, являють собою словоформи [7]. У той же час, ланцюжок над словником розглядається як словосполучення або речення.

Зауважимо, що кожен символ алфавіту розглядається як нероздільний у тому сенсі, що при побудові ланцюжків ніколи не використовуються його графічні елементи (частини символів) та будь-яка послідовність символів однозначно являє деякий ланцюжок.

Практично ця вимога досягається, наприклад, шляхом встановлення «пробілу» (проміжку стандартної довжини) між символами. Цей «пробіл» перевищує довжину будь-якого з проміжків, що зустрічається всередині символів алфавіту.

Правила формальної граматики необхідно розглядати як «продукції» (правила виходу) – елементарні операції, які у разі застосування у визначеній послідовності до вихідного ланцюжка (аксіоми), породжують лише вірні ланцюжки. Сама ж послідовність правил, що використовується у процесі породження деякого ланцюжка, є її викладом. Визначена таким чином мова – являє собою формальну систему. Відомими прикладами формальних систем служать логічні обчислення (висловлювання, предикати), що відносяться до розділів математичної логіки.

**Матеріали та результати досліджень**

Породжувальною граматикою або, коротко, граматикою зветься впорядкований набір:

$$G = \langle \mathrm{A}, \Psi, \varepsilon, \mathrm{Z} \rangle, \tag{1}$$

де $\mathrm{A} = \{a_1, a_2, \ldots, a_m\}$ – основний термінальний алфавіт;

$\Psi$ – кінцевий допоміжний (позатермінальний) алфавіт, символи якого позначуються рядковими грецькими літерами;

$\varepsilon$ $(\varepsilon \in \Psi)$ – початковий (позатермінальний) символ;

$\mathrm{Z}$ – кінцева система підстановок:

$$\mathrm{Z} = \{u_i \to v_i \,|\, i = 1, 2, \ldots, k\}, \tag{2}$$

де $u_i$ – ланцюжок;

$v_i \in \delta(v)$, де $\delta(v)$ – вільна напівгрупа над об'єднаним алфавітом $\Theta$:

$$\Theta = (\mathrm{A} \cup \Psi). \tag{3}$$

Інакше кажучи, символи основного алфавіту $\mathrm{A}$ є елементарними одиницями мови, що визначається; символи алфавіту $\Psi$ – метазмінні, що використовуються при викладі вірних ланцюжків (у природних мовах такими мета змінними є граматичні класи: іменник, дієслово тощо); $\varepsilon$ – метазмінна аксіома з якої вибудовуються усі вірні ланцюжки (у природних мовах аксіомі відповідає граматичний клас «речення»); $\mathrm{Z}$ – схема граматики, що складається з продукцій (правила викладу – граматичні правила мови, що визначається).

Наприклад, породжувальною граматикою є граматика:

$$G_0 = \langle \{a, b, c\}, \{\alpha, \beta, \chi\}, \varepsilon, \mathrm{Z}_0 \rangle, \tag{4}$$

причому $\mathrm{Z}_0$ має систему правил:

$$\mathrm{Z}_0 = \begin{cases} \varepsilon \to abc, \\ \varepsilon \to b, \\ \varepsilon \to \alpha\alpha, \\ abc \to c. \end{cases} \tag{5}$$

Визначення мови $L(G)$, що завдана породжувальною граматикою $G$, пов'язане з поняттям «виклад».

Нехай $x, y$ – ланцюжки, що належать до вільної напівтрупи $\delta(v)$. Ланцюжок $y$ безпосередньо виводиться з ланцюжка $x$ у граматиці $G$:

$$x \underset{G}{\Rightarrow} y \text{ або } x \Rightarrow y \text{ (коли } G \text{ мається на увазі)}, \tag{6}$$

якщо у схемі $\mathrm{Z}$ поданої граматики знайдеться продукція $u \to v$ така, що

$$\begin{cases} x = x_1 u x_2, \\ y = x_1 v x_2; \end{cases} \tag{7}$$

де $x_1, x_2 \in \delta(v)$. Тобто ланцюжок $y$ отримуємо як результат застосування до ланцюжка $x$ продукції $u \to v \in \mathrm{Z}$, що значить заміну у ланцюжку $x$ виділеного входження лівої частини $u$ поданої продукції її правою частиною $v$. Наприклад, у граматиці $G_0$:

$$b\varepsilon c \Rightarrow b\alpha\alpha c, \quad abcba \Rightarrow cba, \quad \ldots \tag{8}$$

Ланцюжок $y$ виводиться з ланцюжка $x$ у граматиці $G$, $x \overset{*}{\underset{G}{\Rightarrow}} y$ або $x \overset{*}{\Rightarrow} y$, схоже з (6), якщо ланцюжки $x, y$ співпадають або існує послідовність ланцюжків $z_0, z_1, \ldots, z_k$ така, що:

$$z_0 = x, \quad z_k = y \quad \wedge \quad \forall i \left(1 \le i \le k\right) \quad z_{i-1} \Rightarrow z_i . \tag{9}$$

Послідовність ланцюжків $Q = (z_0, z_1, \ldots, z_k)$ має назву «викладу» ланцюжка $y$ з ланцюжка $x$ у граматиці $G$. Наприклад, у граматиці $G_0$ $\varepsilon \overset{*}{\Rightarrow} acc$, причому послідовність є викладом ланцюжка $acc$ з ланцюжка $\varepsilon$:

$$\left\langle \varepsilon \Rightarrow abc; abc \Rightarrow a\varepsilon c \,|b \to \varepsilon; a\varepsilon c \Rightarrow aabcc \,|\varepsilon \to abc; \right.$$
$$\left. aabcc \Rightarrow acc \,|abc \to c \right\rangle . \tag{10}$$

Слід додати, що на кожному кроці викладу, можна обрати будь-яку з продукцій, що може бути застосована у поточний момент. А це означає, що послідовність застосування продукцій у граматиці довільна та будь-яку продукцію дозволено застосовувати після іншої, але у межах системи правил.

Таким чином, поняття породжувальної граматики докорінно відрізняється від поняття «нормальний алгоритм», у якому підстановки носять визначений характер та суворо виконуються у завчасно зазначеній послідовності.

Виклад $x \overset{*}{\underset{G}{\Rightarrow}} y$ зветься повним, якщо $y \in \delta(\mathrm{A})$, тобто ланцюжок $y$ складається з термінальних символів. Будь-який повний виклад закінчується застосуванням продукцій, якщо їхні праві частини являють собою термінальні ланцюжки. Вказані продукції назвемо «кінцевими продукціями» поданої граматики.

Якщо $x \overset{*}{\underset{G}{\Rightarrow}} y$ та $y \notin \delta(\mathrm{A})$, причому у системі Z не існує правил, що застосовуються до ланцюжка $y$, то виклад ланцюжка $y$ з ланцюжка $x$ у граматиці $G$ називається тупіковим. Наприклад: виклад, наведений у (10) є повним викладом у граматиці $G_0$, $acc$ – кінцева продукція граматики $G_0$. А, наприклад, виклад:

$$\left\langle aabcabc \Rightarrow a\varepsilon abc \,|abc \to \varepsilon; a\varepsilon abc \Rightarrow a\varepsilon c \,|abc \to c; \right.$$
$$\left. a\varepsilon c \Rightarrow a\alpha\alpha c \,|\varepsilon \to \alpha\alpha \right\rangle \tag{11}$$

є тупіковим викладом ланцюжка $a\alpha\alpha c$ з ланцюжка $aabcabc$ у граматиці $G_0$.

Далі розглянемо як саме породжувальна граматика $G = \langle \mathrm{A}, \Psi, \varepsilon, \mathrm{Z} \rangle$ визначає мову, що відповідає їй. Ланцюжок $x \in \delta(\mathrm{A})$ буде вірним, якщо існує принаймні один повний виклад ланцюжка з аксіоми $\varepsilon$ в граматиці $G$. Інакше кажучи, ланцюжок $x$ вірний, якщо:

$x \in \delta(\mathrm{A})$ – ланцюжок $x$ складається з термінальних символів;

$\varepsilon \overset{*}{\underset{G}{\Rightarrow}} x$ – існує виклад ланцюжка $x$ з аксіоми $\varepsilon$.

Безліч усіх вірних ланцюжків у граматиці $G$ створює мову $L(G)$, що породжується граматикою $G$. Наприклад, граматика $G_0$ породжує мову:

$$L(G_0) = \left\{ x^n y x^n \cup y^m x y^m \,|n, m = 0, 1, 2, \ldots \right\}. \tag{12}$$

Отже, кожній граматиці $G = \langle \mathrm{A}, \Psi, \varepsilon, \mathrm{Z} \rangle$ однозначно відповідає мова $L(G)$, що породжена цією граматикою. Проте, ця відповідність не ізоморфна: та ж сама мова може породжуватись різноманітними граматиками. Це дозволяє внести відношення еквівалентності на безліч граматик.

Граматики $G$ та $G'$ – еквівалентні $(G \Leftrightarrow G')$, якщо $L(G) = L(G')$, тобто граматики $G$ та $G'$ породжують одну мову. Наприклад, граматика:

$$G_1 = \langle \{a, b, c\}, \{\varepsilon\}, \varepsilon, \mathrm{Z}_1 \rangle, \tag{13}$$

схема якої має набір правил:

$$\mathrm{Z}_1 = \begin{cases} \varepsilon \to abc, \\ \varepsilon \to b, \\ \varepsilon \to c, \end{cases} \tag{14}$$

породжує мову $L(G_1)$, яка співпадає із мовою $L(G_0)$ та, відповідно, означає, що $G_0 \Leftrightarrow G_1$.

## Висновки

Підсумовуючи викладений матеріал, зазначимо, що, породжувальні граматики можуть знаходити широке застосовуються щодо розгляду лінгвістичного забезпечення програмних систем. Особливої важливості інструмент породжувальних граматик набирає у разі виконання необхідного реінжинірингу програмного коду, який написано різними мовами програмування.

Сформована у статті парадигма, з наукової точки зору, ляже в основу методології реінжинірингу програмних систем, а з практичної – стане у пригоді системним програмістам, які працюють із мультимовними надбудовами програмних систем, що набувають еволюційного розвитку із плином часу та вдосконалення в процесі використання.

## Література

[1] Генеративная лингвистика [Електронний ресурс] / В. П. Руднев // Словарь культуры XX века. – Режим доступу: http://www.gumer.info/bibliotek_Buks/Culture/Rudnev/Dict/_04.php.
[2] Chomsky, N. Logical Syntax and Semantics: Their Linguistic Relevance / Noam Chomsky // Language. – Vol. 31. – No. 1. – P. 36–45.
[3] Chomsky, N. Three Models for the Description of Language / Noam Chomsky // IRE Transactions on Information Theory. – Sep., 1956. – P. 113–124.
[4] What is Generative Grammar? [Електронний ресурс] // wiseGEEK. – Режим доступу: http://www.wisegeek.com/what-is-generative-grammar.htm.
[5] Евристичний аналіз граматики [Електронний ресурс] / Сергій Горелов // Евристичний аналіз будь-якої мови. – Режим доступу: http://www.grammcheck.org/.
[6] Глушков, В.М. Алгебра. Языки. Программирование [Текст] / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – К.: Наук. думка, 1974. – 328 с.
[7] Федоренко, О. Ф. Основи лінгвістичних досліджень = Fundamentals of Linguistic Research [Текст]: підруч. для вузів / О. Ф. Федоренко, С. М. Сухорольська, О. В. Руда. – Л.: «Центр» Львів. нац. ун-ту ім. І. Франка, 2009. – 296 с.

## References

[1] V. P. Rudnev. *Heneratyvnaya lynhvystyka (Slovar' kul'tury XX veka)* [Online]. Available: http://www.gumer.info/bibliotek_Buks/Culture/Rudnev/Dict/_04.php.
[2] Noam Chomsky, “Logical Syntax and Semantics: Their Linguistic Relevance”, Language. vol. 31, no. 1, pp. 36–45.
[3] Noam Chomsky, “Three Models for the Description of Language”, IRE Transactions on Information Theory. Sep., 1956, pp. 113–124.
[4] What is Generative Grammar? (wiseGEEK) [Online]. Available: http://www.wisegeek.com/what-is-generative-grammar.htm.
[5] Serhiy Horelov. *Evrystychnyy analiz hramatyky (Evrystychnyy analiz bud'-yakoyi movy)* [Online]. Available: http://www.grammcheck.org/.
[6] V. M. Hlushkov, H. E. Tseytlyn, E. L. Yushchenko, *Alhebra. Yazyky. Prohrammyrovanye*, Kyiv, USSR.: Nauk. dumka, 328 p., 1974.
[7] O. F. Fedorenko, S. M. Sukhorol's'ka, O. V. Ruda, *Osnovy linhvistychnykh doslidzhen* [Fundamentals of Linguistic Research], Lviv, Ukraine: Tsentr, I. Franko’s LNU, 296 p., 2009.