УДК 004.91:510.25+519.17

# КОНСТРУКТИВНО-ПРОДУКЦИОННАЯ МОДЕЛЬ ГРАФОВОГО ПРЕДСТАВЛЕНИЯ ТЕКСТА

*В.И. Шинкаренко, Е.С. Куропятник*

У статті розглянута графова модель тексту, що дозволяє прискорити обробку інформації. Дана модель дозволяє виявляти однакові фрагменти в документах зі зміною порядку слідування речень та інших частин. Використання конструктивно-продукційних структур для формалізації даної моделі є перспективним підходом для подальшої автоматизації процесу роботи з моделлю і відповідно з текстом.
Ключові слова: графова модель тексту, стискання графа, конструктивно-продукційна структура, співставлення текстів.

В статье рассмотрена графовая модель текста, позволяющая ускорить обработку информации. Данная модель позволяет выявлять одинаковые фрагменты в документах с изменением порядка следования предложений и других частей. Использование конструктивно-продукционных структур для формализации данной модели является перспективным подходом для дальнейшей автоматизации процесса работы с моделью и соответственно с текстом.
Ключевые слова: графовая модель, сжатие графа, конструктивно-продукционная структура, сопоставление текстов.

The article describes the graph model of the text, allowing speeds up processing. This model allows us to identify the same fragments in the documents with the change in the order of sentences and other parts. Using constructive-synthesizing structure to formalize this model is a promising approach to further automate the process of working with the model and the text accordingly.
Key words: graph model of the text, graph compression, constructive-synthesizing structure, text comparison.

## Введение

Исследования и моделирование в различных отраслях науки и общественного производства, а также образовательные процессы неотрывно связаны с обработкой уже имеющихся научных изданий, разработкой и документированием новых методик, разработкой текстов компьютерных программ и тому подобное. Скорость и интенсивность развития сфер науки, образования и производства, средств их информационной поддержки порождает проблему обработки текстов, которая может быть разделена на несколько составляющих. На сегодня наиболее актуальными являются поиск, сравнение, проверка правописания, выделение знаний. Разработка и модернизация средств автоматизации для решения данных вопросов позволяет улучшить временные показатели решения данных задач.

Существует множество методик для предварительной обработки текстов с целью упрощения их дальнейшей обработки. Среди них приведение символов к нижнему регистру и единой кодировки, удаление стопслов и замена слов их грамматическими основами [1, 2], удаление или игнорирование пунктуационных знаков [3], игнорирование пробелов, пустых строк, комментариев в текстах программ [4].

Особое внимание уделяется обработке текстов с целью выявления плагиата. Для этого разработано множество алгоритмов, моделей и программ на их основе [4]. Среди них графовое представление семантики текста [3, 5], токенизации [6], морфологический анализ и синтаксический разбор [4], латентный семантический анализ [6, 7] и др.

Представление текста в виде графовой модели [8] позволяет ускорить обработку текстовой информации. Использование КПС для формализации данной модели является перспективным подходом для дальнейшей автоматизации процесса работы с моделью и соответственно текстом.

Формализация средствами конструктивно-продукционных структур (КПС) позволяет описать не только структуру объектов, но их свойства, определить допустимые операции над ними, алгоритмические элементы построения конструкций на их основе [9]. Ранее была рассмотрена обобщенная КПС [10]. Данный подход является универсальным для решения многих задач и применяется в разноплановых работах, посвященных адаптации алгоритмов сжатия [11], формированию структур данных на логическом уровне [12], обобщенных грамматик [13].

Особенностью КПС является возможность формирования конструкций различной природы на основе операций подстановки, связывания, вывода и других вспомогательных операций, а также использования атрибутов [10].

## Обобщенная конструктивно-продукционная структура

Обобщённая КПС определяется носителем $M$, сигнатурой $\Sigma$ и аксиоматикой $\Lambda$ $C = \langle M, \Sigma, \Lambda \rangle$.

Носитель включает в себя конструктивные элементы с набором атрибутов. Сигнатура состоит из множества имен операций и отношений. Аксиоматика включает множества определений, аксиом, правил, свойств, правил и пр. Носитель и сигнатура определяются аксиоматикой.

Преобразования КПС позволяют применять их для различных предметных областей. Специализация КПС подразумевает определение атрибутов носителя, семантическую составляющую, а также определения конечного конкретного набора операций, их атрибутов. В ходе интерпретации каждой операции сигнатуры ставится в соответствие алгоритм ее выполнения, являющийся элементом носителя некоторой алгоритмической

структуры [14]. Реализации КПС заключается в построении конструкций из элементов носителя КПС путем выполнения алгоритмов, связанных с операциями сигнатуры.

Подход на основе КПС может быть использован для формализации понятий текста, его составляющих. Особенности определения понятий текста, предложения, слова определяют носитель КПС и суть операций по ее преобразованию. Так, например, если слово рассматривать с точки зрения естественного языка, то оно не может включать символы-цифры, а если с точки зрения языков программирования, слово – лексема, которая может содержать цифры и иные символы (имена переменных, функций, классов и т.п.). Таким образом, определение подобных понятий может как расширять, так и сужать область определения элементов носителя, конструкций, операций над ними.

## Формализованная спецификация текста средствами КПС

Формализованная спецификация текста и его составляющих дает возможность автоматизировать разработку и обновление программ для решения поставленных задач.

Введем некоторые ограничения и дополнения аксиоматики и обозначений [10] в соответствие с рассматриваемой предметной областью.

С данной целью определим КПС и выполним ее специализацию:

$$C = \langle M, \Sigma, \Lambda \rangle \; {}_S\mapsto \; {}_S C_T = \langle M_T, \Sigma_T, \Lambda_T \rangle, \tag{1}$$

где $M_T$ – носитель, включающий все символы электронного представления текста и конструкции, построенные на них, $\Sigma_T$ – операции и отношения на элементах $M_T$, $\Lambda_T$ – аксиоматика, определяющая $M_T$ и $\Sigma_T$.

**Частичная аксиоматика носителя.** Носитель включает множества терминалов и нетерминалов. $M_T \supset T_T \bigcup N_T, T_T = T_K \bigcup T_L \bigcup T_D \bigcup T_N \bigcup T_{NP}$, $T_K$, $T_L$ – множество кириллических и латинских символов соответственно, которые входят в текст; $T_D$ – множества символов-разделителей для отображения знаков пунктуации и пробела; $T_N$ – множество цифр; $T_{NP}$ – множество непечатных символов, не включая пробел; $N_T$ – множество нетерминалов (вспомогательных символов).

Элементы носителя ${}_{\overline{w}}m_i$ имеют набор атрибутов $\overline{w}_i = \langle type, code \rangle$. Принадлежность конкретного атрибута элементу носителя будем обозначать $w_i \lrcorner m$. Тип ($type \lrcorner m$) принимает значение терминал/нетерминал ($ter / nter$). Атрибут *code* указывает на машинное представление символа-терминала. В качестве идентифицирующего атрибута выступает код.

**Частичная аксиоматика отношений и операций.** Сигнатура $\Sigma_T$ состоит из множества операций $\Sigma_T = \langle \Xi, \Theta, \Phi, \{\rightarrow\} \rangle \bigcup \Psi$, $\Xi = \{\cdot\}$, где $\Xi$ – множество операций связывания, $\Theta$ – множество операций вывода, $\Phi = \{=, :=\}$ – множество операций над атрибутами; $\Psi$ – множество правил продукций вида $\psi_i :\prec s_i, g_i \succ$, $i$ – номер правила, $s$ – последовательность операций подстановки, $g$ – последовательность операций над атрибутами, «$\rightarrow$» – отношение подстановки. $\Theta = \{\Rightarrow, |\Rightarrow, \|\Rightarrow\}$ – операции подстановки, частичного и полного вывода.

Операция конкатенации $\cdot(m_i, m_j)$ связывает два элемента носителя. Результатом выполнения операции является форма ${}_{w_l} l$. Форма может быть построена таким образом:

- ${}_{w_l} l = {}_{w_0} \cdot ({}_{w_j} m_j, \; {}_{w_k} m_k)$ для $\forall {}_{w_i} m_i \in M_T$;
- ${}_{w_l} l = {}_{w_j} m_j$, если $l = {}_{w_0} \cdot (\varepsilon, {}_{w_j} m_j) = {}_{w_0} \cdot ({}_{w_j} m_j, \; \varepsilon)$, где $\varepsilon$ – пустой элемент;
- ${}_{w_l} l = {}_{w_0} \cdot ({}_{w_1} l_1, \; {}_{w_2} l_2)$ для $\forall {}_{w_i} l_i = {}_{w_j} m_j \,|\cdot({}_{w_j} m_j, \; {}_{w_k} m_k)|\cdot({}_{w_j} l_j, {}_{w_k} m_k), \; \forall \; {}_{w_i} m_i \in M_T$.

Множество значений атрибутов формы определяется совокупностью значений атрибутов ее составляющих.

Сентенциальная форма – форма, полученная в результате вывода из аксиомы (начального нетерминального символа, принадлежащего носителю) согласно правилам вывода конкретизированной КПС.

Формы, в которых отсутствуют нетерминальные элементы – конструкции. Конструкции имеют такие атрибуты: тип языковой конструкции, набор кодов. Атрибут тип ($type \lrcorner l$) может принимать такие значения: к-слово, к-предложение, к-абзац, к-текст.

Отношение подстановки – отношение с атрибутами ${}_{w_i} l_i \, {}_w \rightarrow {}_{w_j} l_j$, где $l_i, l_j$ – сентенциальные формы [11].

Для заданной формы ${}_{w_l}l = {}_{w_0}\cdot({}_{w_1}l_1,\ {}_{w_2}l_2, \ldots,\ {}_{w_h}l_h, \ldots,\ {}_{w_k}l_k)$ и доступного отношения подстановки ${}_{w_p} \to ({}_{w_h}l_h,\ {}_{w_q}l_q)$ такого, что ${}_{w_h}l_h$ – подформа ${}_{w_l}l$ результатом трехместной операции подстановки ${}_{w_l^*}l^* = {}_{w_p}\Rightarrow({}_{w_h}l_h, {}_{w_q}l_q,\ {}_{w_l}l)$ будет форма ${}_{w_l^*}l^* = {}_{w_0}\cdot({}_{w_1}l_1,\ {}_{w_2}l_2, \ldots,\ {}_{w_q}l_q, \ldots,\ {}_{w_k}l_k)$ .

Двухместная операция частичного вывода ${}_{w_l^*}l^* = {}_{w_p}|\Rightarrow (\Psi,\ {}_{w_l}l)$ ($|\Rightarrow \in \Theta$) заключается в:

- выборе одного из доступных правил подстановки $\psi_r : \langle s_r, g_r \rangle$ с отношениями подстановки $s_r$ ;
- выполнении на его основе операций подстановки;
- выполнении операций над атрибутами $g_r$ в соответствующей последовательности.

Операция полного вывода или просто вывода ($\|\Rightarrow \in \Theta$) заключается в пошаговом преобразовании форм, начиная с начального нетерминала и заканчивая конструкцией, удовлетворяющей условию окончания вывода, что подразумевает циклическое выполнение операций частичного вывода. Операция двухместная ${}_{w_l^*}l^* = \|\Rightarrow (\Psi,\ {}_{w_l}l)$.

Операция присваивания $:=(a,b)$ копирует значение операнда $b$ в $a$.

Операция $=(w_i, w_j)$ выполняет сравнения атрибутов. Результатом операции сравнения является значение «истина», если $w_i = w_j$, иначе – «ложь».

Элемент принадлежит форме $m \in l$, если $\exists l_1, l_2, l_3 :\ l = \cdot(l_1, l_2)\,\&\,((l_1 = \cdot(l_3, m) | l_1 = \cdot(m, l_3)) | l_2 = \cdot(l_3, m))$, где $l_1, l_2, l_3$ – формы, образованные с помощью операции конкатенации.

**Расширение аксиоматики носителя.** Конструкция является к-словом $type\text{↵}l = cw$, если $\forall m_i \in l : m_i \in T_K \bigcup T_L \bigcup T_N$.

Конструкция является к-предложением $type\text{↵}l = cs$, если $l = \cdot(l_1, m), m \in T_{SD}$, $T_{SD} = \{"!","?",".","..."\}$, $T_{SD} \subset T_D$.

Конструкция является к-абзацем $type\text{↵}l = cp$, если $l = \cdot(l_1, l_2)\ \ \&\, l_2 = \cdot(m_1, m_2)$, $code\text{↵}m_1 = 13$, $code\text{↵}m_2 = 10$ (переход на новую строку, возврат каретки).

К-абзац может включать в себя несколько предложений, к-предложение – несколько абзацев.

Конструкция является к-текстом $type\text{↵}l = ct$, если $l = \cdot(l_1, l_2)\ \ \&\ type\text{↵}l_1 = cp\ \ \&\ (type\text{↵}l_2 = cp \mid l_2 = \varepsilon)$ и $l_1, l_2$ имеют смысловую связь.

**Интерпретация КПС текста.** Частично интерпретируем структуру $C_T$ (1) с помощью алгоритмической структуры $C_A$ :

$$\left\langle C_T, C_{A,T} = \left\langle M_{A,T}, V_{A,T}, \Sigma_{A,T}, \Lambda_{A,T} \right\rangle \right\rangle_I \mapsto \left\langle {}_A C_T,\ {}_A C_T = \left\langle M_1, \Sigma_1, \Lambda_1 \right\rangle \right\rangle, \tag{2}$$

где $\Lambda_1 \supset \Lambda_T$, $V_{A,T} = \{A_i^0 \mid_{X_i}^{Y_i}\}$ – множество базовых алгоритмов [10, 13], $X_i, Y_i$ – множества определений и значений алгоритма $A_i^0 \mid_{X_i}^{Y_i}$. $\Lambda_{A,T} = \{M_{A,T} = \bigcup\limits_{A_i^o \in V_{A,T}} (X(A_i^0) \bigcup Y(A_i^0)) \bigcup \Omega(C_T)\}$ – неоднородный носитель, $\Omega(C_T)$ – множество языковых конструкций, которые удовлетворяют $C_T$; $\Lambda_1 = \{(A_3 \mid_{l_1, l_2}^{l} \text{↵}"\cdot");\ \ (A_4 \mid_{l_h, l_q, f_i}^{f_i} \text{↵}"\Rightarrow");$ $(A_5 \mid_{f_i, \Psi}^{f_j} \text{↵}"|\Rightarrow");\ (A_6 \mid_{\sigma, \Psi}^{\overline{\Omega}} \text{↵}"\|\Rightarrow");\ \ (A_7 \mid_{a,b}^{a} \text{↵}":=");\ \ (A_8 \mid_{a,b}^{c} \text{↵}"=")\}$.

Структура ${}_A C_T$ включает алгоритмы выполнения операций:

- $A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j}$ – композиция алгоритмов, $A_i \cdot A_j$ – последовательное выполнение алгоритма $A_j$ после алгоритма $A_i$ ;
- $A_2^0 \mid_{Z}^{A_1}$ – условное выполнение: алгоритм $A_i$ выполняется, если условие $Z$ истинно;
- $A_3 \mid_{l_1, l_2}^{l}$ –конкатенация, $l, l_1, l_2$ – формы;

– $A_4 |^{f_i}_{l_h, l_q, f_i}$ – подстановка, $l_h, l_q, f_i$ – формы;

– $A_5 |^{f_j}_{f_i, \Psi}$, $A_6 |^{\overline{\Omega}}_{\sigma, \Psi}$ – частичный и полный вывод, где $f_i, f_j$ – формы, $\sigma$ – аксиома, $\overline{\Omega}$ – множество сформированных конструкций;

– $A_7 |^{a}_{a,b}$ – присвоение операнду $a$ значение операнда $b$;

– $A_8 |^{c}_{a,b}$ – сравнение атрибутов $a$ и $b$, если $a = b$, то $c = true$, в противном случае $c = false$.

**Конкретизация КПС текста.** Выполним конкретизацию структуры $C_T$:

$$C_{T\,K} \mapsto {}_K C_T = \langle M_2, \Sigma_2, \Lambda_2 \rangle,$$

где $\Lambda_2 \supset \Lambda_1$, $\Lambda_2 \supset \{M_2 = T_T \bigcup N;\ N = \{{}_{w_i} n_i\} = \{\alpha, \gamma, \delta, \kappa, \pi, \rho, \sigma, \tau\}\}$. Атрибутом нетерминала ($w_i$) является тип конструкции ($kind\text{↵}w_i$), для построения которой он используется.

Далее рассмотрим правила для построения к-текста, а также других конструкций, которые могут в него входить.

Правило $s_1$ позволяет начать выполнение построения текста, определив значение соответствующего атрибута как текст:

$$s_1 = \langle \sigma \to \tau \rangle,\ g_1 = \langle kind\text{↵}\tau := ct \rangle.$$

Правило $s_2 - s_4$ позволяют определить составляющие текста – абзацы (одного или много), определив значение соответствующего атрибута как абзац

$$s_2 = \langle \tau \to \alpha\pi \rangle,\ g_2 = \langle kind\text{↵}\alpha := \mathrm{c}p \rangle,\ s_3 = \langle \pi \to \alpha\pi \rangle,\ s_4 = \langle \pi \to m_1 m_2 \rangle,$$

где $m_1, m_2$ – символы возврата каретки и перехода на новую строку.

Правила $s_2 - s_4$ позволяют определить составляющие абзаца – предложение (много или одно), определив значение соответствующего атрибута как предложение

$$s_5 = \langle \alpha \to \rho\gamma\alpha \rangle,\ s_6 = \langle \alpha \to \rho\gamma \rangle,\ g_6 = \langle kind\text{↵}\rho := \mathrm{c}s \rangle,\ s_7 = \langle \gamma \to m_{end} \rangle,$$

где $m_{end} \in T_{SD}$ – символ признак окончания предложения.

Правила $s_8 - s_9$ позволяют определить составляющие предложения: одно или более слов

$$s_8 = \langle \rho \to \kappa \rangle,\ g_8 = \langle kind\text{↵}\kappa := \mathrm{c}w \rangle,\ s_9 = \langle \rho \to \kappa\delta\rho \rangle.$$

Правила $s_{10} - s_{11}$ позволяют определить разделитель между словами

$$s_{10} = \langle \delta \to m_{sep} \rangle,\ s_{11} = \langle \delta \to m_{sep}\delta \rangle,$$

где $m_{sep} \in T_{WD}, T_{WD} = T_D \setminus T_{SD}$.

Правила $s_{12} - s_{13}$ позволяют построить слово из одной и более букв:

$$s_{12} = \langle \kappa \to m_c\kappa \rangle,\ s_{13} = \langle \kappa \to m_c \rangle,$$

где $m_c \in T_K \bigcup T_L \bigcup T_N$.

Тут под записью типа $\kappa \to m_c, m_c \in T_K \bigcup T_L \bigcup T_N$ следует понимать множество альтернативных правил, что можно также записать в виде $\kappa \to a \mid б \mid в \ldots$.

Операции над атрибутами выполняются после операции частичного вывода.

**Реализация КПС текста.** Реализация структуры (2) заключается в формировании языковых конструкций из элементов ее носителя путем выполнения алгоритмов, связанных с операциями сигнатуры, по правилам аксиоматики:

$$ {}_{A}C_{T}\ {}_{R}\mapsto \overline{\Omega}({}_{A}C_{T}), $$

где $\overline{\Omega}({}_{A}C_{T}) \subset \Omega({}_{A}C_{T})$.

Рассмотри пример построения конструкции «Колпак под колпаком.». Далее показан вывод текста, состоящего из одного абзаца и одного предложения, заканчивающегося точкой:

$$ \sigma \overset{1}{\Rightarrow} \tau \overset{2}{\Rightarrow} \alpha\pi \overset{4}{\Rightarrow} \alpha'13''10' \overset{6}{\Rightarrow} \rho\gamma'13''10' \overset{7}{\Rightarrow} \rho'.''13''10' $$

Определим количество слов и структуру предложения:

$$ \rho'.''13''10 \overset{9}{\Rightarrow} \kappa\delta\rho'.''13''10 \overset{9}{\Rightarrow} \kappa\delta\kappa\delta\rho'.''13''10' \overset{8}{\Rightarrow} \kappa\delta\kappa\delta\kappa'.''13''10' \overset{10}{\Rightarrow} \kappa'\ \ 'к'\ '\kappa.''13''10'. $$

Далее используя правила 12, 13 получаем фразу «Колпак'32'под'32'колпаком.'13''10'». Здесь в кавычки взяты символы пробела и перехода к новому абзацу.

Для обработки конструкций рассмотрим задачи синтеза и анализа графовой модели текста [8].

## Синтез графа конструктивно-продукционными структурами

Пусть есть множество языковых конструкций $\overline{\Omega}({}_{A}C_{T})$, порожденное структурой ${}_{A}C_{T}$ (2). Задача состоит в определенные структуры $C_g$, порождающей множество конструкций-графов $\overline{\Omega}(C_g)$ такое, что существует биективное отображение $f : \overline{\Omega}({}_{A}C_{T}) \to \overline{\Omega}_g(C_g)$.

Для решения поставленной задачи определим структуру и специализируем ее соответствующим образом:

$$ C = \langle M, \Sigma, \Lambda \rangle\ {}_{S}\mapsto C_g = \langle M_g, \Sigma_g, \Lambda_g \rangle, $$

где $M_g$ – расширяемый носитель, включающий множества конструкций-графов, языковых конструкций и их элементов, $\Sigma_g$ – множество операций и отношений на элементах $M_g$, $\Lambda_g$ – аксиоматика.

**Частичная аксиоматика носителя.** Носитель включает множества терминальных и нетерминальных элементов $M_g = T_g \cup N_g$. Терминалами являются языковые конструкции, построенные КПС (2) и их составляющие ($T_T$), а также конструкции графов и их составляющих $T_g = \overline{\Omega} \cup \Omega_g \cup T_T \cup V \cup E$, $\Omega_g$ – множество конструкций-графов, $V$, $E$ – множества вершин и дуг с их атрибутами.

Вершина имеет атрибуты $\overline{w}_v = \langle id, content, tokens \rangle$, $id$ – идентификатор, принимает целочисленные значения, $content$ – часть текстовой конструкции, $tokens$ – список, содержащий признаки начала языковых конструкций. Атрибуты дуги $\overline{w}_e = \langle id, routes, start, end \rangle$, $id$ – идентификатор, принимает целочисленные значения, $routes$ – множество номеров путей, в которые входит дуга (указывает на порядок обхода графа), $start$, $end$ – вершин, которые являются началом и концом дуги.

Нагруженный граф будем обозначать ${}_{\overline{w}_g}G = \langle V, E \rangle$, $V = \{{}_{\overline{w}_{vi}} v_i\}, E = \{{}_{\overline{w}_{ej}} e_j\}$ – множества вершин и дуг, нагруженых атрибутами. Каждое множество содержит пустой элемент.

Граф имеет такие атрибуты $\overline{w}_g = \langle start\_v, last\_v, current\_v, amount\_l \rangle$, где $start\_v$– стартовая вершина графа, $last\_v$ – последняя добавленная вершина, $current\_v$ – текущая вершина при формировании графа, $amount\_l$ – количество циклов, в которые входит стартовая вершина.

**Частичная аксиоматика операций.** Рассмотрим сигнатуру $\Sigma_g$:

$$ \Sigma_g = \langle \Xi_g, \Theta_g, \Phi, \{\to\} \rangle \cup \Psi_g, $$

где $\Xi_g = \{\cdot, :\widehat{=}, :\doteq, \widetilde{\cup}, \cup\}$ – множество операций преобразования и связывания, $\Theta_g = \{\Rightarrow, |\Rightarrow, \|\widetilde{\Rightarrow}\}$ – множество операций вывода, $\Phi_g = \{\div, :=, \#, +\}$ – множество операций над атрибутами, $\Psi_g$ – множество правил продукций вида $\psi_i :< s_i, g_i >$, $i$ – номер правила, $s$ – последовательность операций подстановки, $g$ – последовательность операций над атрибутами, «$\to$» – отношение подстановки.

Операция $e \coloneqq (v_1, v_2, G)$ заключается в определении дуги $e$, соединяющей вершины $v_1$ и $v_2$ графа $G$. Если дуга со значениями соответствующих атрибутов отсутствует, то возвращается пустой элемент множества дуг графа $G$, его идентификатор равен нулю.

Операция $v \coloneqq (x, V)$ заключается в нахождении вершины $v$ с атрибутом веса, равным $x$, из множества вершин $V$. Если вершина со значениями соответствующих атрибутов отсутствует, то возвращается пустой элемент множества вершин $V$, его идентификатор равен нулю.

Операция $\div(c, n, L)$ заключается в выполнении $n$ операций из списка $L$, если $c = true$.

Операция вычисления мощности множества $\#Q$ определяет число, равное количеству элементов в $Q$.

Операция сложения двух чисел $+(a,b)$ предполагает нахождение третьего числа, являющегося их суммой.

Операция объединения графов ${}_{\overline{w}_g} G = \widetilde{\bigcup}({}_{\overline{w}_1} G_1, {}_{\overline{w}_2} G_2)$ предполагает формирование нового графа ${}_{\overline{w}_g} G$, включающего объединенные множества вершин и дуг исходных графов ${}_{\overline{w}_g} G = \langle V, E \rangle$, $V = V_1 \bigcup V_2$, $E = E_1 \bigcup E_2$, ${}_{\overline{w}_1} G_1 =< V_1, E_1 >$, ${}_{\overline{w}_2} G_2 =< V_2, E_2 >$, при этом $\bigcup$ – традиционная операция объединения множеств.

Отношение подстановки имеет вид

$$\psi_i = \langle s_i, g_i \rangle,\ s_i = \langle \overline{s}_i, \widetilde{s}_i \rangle,\ g_i = \left\langle \overline{g}_i\ , \widetilde{g}_i\ \right\rangle,$$

где $\overline{s}_i$, $\widetilde{s}_i$ – отношение подстановки для распознавания языковой конструкции и построения конструкции графа соответственно, $\overline{g}_i$, $\widetilde{g}_i$ – операции над атрибутами языковой конструкции и графа, его вершин и дуг соответственно. В случае если операции над атрибутами не выполняются, отношение подстановки имеет вид $\psi = \langle s, \varepsilon \rangle$.

Операция полного вывода $\|\widetilde{\Rightarrow}(\Psi, {}_{w_l} l)$ состоит в:

- определении входной конструкции, набора отношений подстановки и доступных из них;
- выполнении операции частичного вывода, пока языковая конструкция $\omega \in \overline{\Omega}$ полностью не распознана, то есть для каждого ее элемента $\omega_i$ в графе нет соответствующего элемента-вершины или форма графа содержит нетерминалы.

Результатом операции вывода является конструкция-граф.

**Конкретизация КПС графа.** Выполним конкретизацию структуры $C_g$:

$$C_{g\,K} \mapsto {}_K C_g = \langle M_3, \Sigma_3, \Lambda_3 \rangle,$$

где $\Lambda_3 \supset \Lambda_g$, $\Lambda_3 \supset \{ M_3 \supset M_g$, $c \in T_T$, $N_g = \{\alpha, \delta\}$, $T_g \supset \{G, G^*, G^{**}\}$, $G = \langle V, E \rangle$, $V = \{v\}$, $E = \varnothing$, $G^* =< V^*, G^* >$, $V^* = \{v_1^*, v_2^*\}$, $E^* = \{e_1^*\}$, $G^{**} =< V^{**}, G^{**} >$, $V^{**} = \{v_1^{**}, v_2^{**}\}$, $E^{**} = \{e_1^{**}\}\}$.

Для распознавания языковых конструкций определим такие правила:

$$\overline{s}_1 = \langle \sigma_{d_1} \to c\sigma \rangle,\ \overline{g}_1 = \langle \div(code\lrcorner c \neq EOF, 1, d_1 \coloneqq true) \rangle,\ \overline{s}_2 = \langle \sigma_{d_2} \to c \rangle,\ \overline{g}_2 = \langle \div(code\lrcorner c = EOF, 1, d_2 \coloneqq true) \rangle,$$

где $c$ – символ текста, кроме $EOF$ – признак конца текста в его электронном представлении.

Следующие правила описывают добавление первой вершины в пустой граф:

$$\widetilde{s}_1 = \langle \sigma \widetilde{\to} G\alpha \rangle,\ \widetilde{g}_1 = \langle id\lrcorner v \coloneqq \#V, content\lrcorner v \coloneqq c, tokens \coloneqq < cw, cs, cp, ct >$$

$$start\_v\lrcorner G \coloneqq v,\ current\_v\lrcorner G \coloneqq v, last\_v\lrcorner G \coloneqq v, amount\_l\lrcorner G \coloneqq 0 \rangle.$$

Правило $\widetilde{s}_2$ позволяет добавить к графу новую вершину и дугу, связывающую новую вершину с текущей в графе

$$\widetilde{s}_2 \coloneqq \left\langle G\alpha\ {}_{d_1^*}\widetilde{\to} \widetilde{\bigcup}(G, G^*)\alpha, \widetilde{\bigcup}(G, G^*)\alpha \to G\alpha \right\rangle,\ \widetilde{g}_2 = \langle v_1 \coloneqq (c, V), e_1 \coloneqq (current\_v\lrcorner G, v, G) \rangle,$$

$$\div(id\text{↵}v_1 = 0 \,\&\, id\text{↵}e_1 = 0{,}14, d_1^* := true,\ \ id\text{↵}v_1^* := id\text{↵}current\_v\text{↵}G,\ \ content\text{↵}v_1^* := content\text{↵}current\_v\text{↵}G,$$

$$id\text{↵}v_2^* := \#V + 1,\ \ content\text{↵}v_2^* := c, id\text{↵}e_1^* := \#E + 1,\ start\text{↵}e_1^* := v_1^*,\ end\text{↵}e_1^* := v_2^*,\ \ routes\text{↵}e_1^* := \{amount\_l\text{↵}G\},$$

$$\div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{WD}, tokens\text{↵}v_2^* := < cw >),$$

$$\div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{SD}, tokens\text{↵}v_2^* := < cs >)\ ,$$

$$\div(content\text{↵}current\_v\text{↵}G = '10', tokens\text{↵}v_2^* := < cp >),\ last\_v\text{↵}G := v_2,\ current\_v\text{↵}G := v_2)\rangle.$$

Правило $\widetilde{s}_3$ позволяет добавить к графу новую дугу, связывающую текущую вершину со стартовой.

$$\widetilde{s}_3 = \left\langle \alpha \ {}_{d_2^*}\!\!\rightarrow \widetilde{\bigcup}(G, G^{**})\alpha, \widetilde{\bigcup}(G, G^{**})\alpha \rightarrow G\alpha \right\rangle\ ,\ \widetilde{g}_3 = \langle v_1 \doteq (c, V), e_1 \hat{=} (current\_v\text{↵}G, v_1, G),$$

$$\div(v_1 = start\_v\text{↵}G \& id\text{↵}e_1 = 0{,}14, d_2^* := true, id\text{↵}v_1^{**} := id\text{↵}curret\_v\text{↵}G, content\text{↵}v_1^{**} :=$$

$$:= content\text{↵}curret\_v\text{↵}G, id\text{↵}v_2^{**} := id\text{↵}start\_v\text{↵}G,\ content\text{↵}v_2^{**} := c,\ id\text{↵}e_1^{**} := \#E + 1, start\text{↵}e_1^{**} :=$$

$$:= v_1^{**}, end\text{↵}e_1^{**} := v_2^{**},\ routes\text{↵}e_1^{**} := \{amount\_l\text{↵}G\},\ amount\_l\text{↵}G := amount\_l\text{↵}G + 1,$$

$$\div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{WD}, 1, tokens\text{↵}v_2^{**} := \cdot(tokens\text{↵}start\_v\text{↵}G, < cw >)),$$

$$\div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{SD}, 1, tokens\text{↵}v_2^{**} := \cdot(tokens\text{↵}start\_v\text{↵}G, < cs >)),$$

$$\div(content\text{↵}current\_v\text{↵}G = '10', 1, tokens\text{↵}v_2^{**} := \cdot(tokens\text{↵}start\_v\text{↵}G, < cp >)),\ current\_v\text{↵}G = start\_v\text{↵}G)\rangle\ .$$

Правило $\widetilde{s}_4$ позволяет изменить нагрузку имеющейся дуги:

$$\widetilde{s}_4 = \left\langle G\alpha \ {}_{d_3^*}\!\!\rightarrow G\alpha \right\rangle,\ \widetilde{g}_4 = \langle v_1 \doteq (c, V\text{↵}G), e_1 \hat{=} (current\_v\text{↵}G, v_1, G)\ ,\ \div(id\text{↵}v_1 \neq 0 \& id\text{↵}e_1 \neq 0{,}3, d_3^* := true,$$

$$\div(start\_v\text{↵}G \neq v_1, 5, routes\text{↵}e_1 := routes\text{↵}e_1 \cup \{amount\_l\text{↵}G\}, \div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{WD}, 1,$$

$$tokens\text{↵}v_1 := \cdot(tokens\text{↵}v_1, < cw >)), \div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{SD}, 1, tokens\text{↵}v_1 := \cdot(tokens\text{↵}v_1, < cs >)),$$

$$\div(content\text{↵}current\_v\text{↵}G = '10', 1, tokens\text{↵}v_1 := \cdot(tokens\text{↵}v_1, < cp >)),\ current\text{↵}v := v_1), \div(start\_v\text{↵}G = v, 6,$$

$$routes\text{↵}e_1 := routes\text{↵}e_1 \cup \{amount\_l\text{↵}G\}, \div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{WD}, 1, tokens\text{↵}v_1 := \cdot(tokens\text{↵}v_1, < cw >)),$$

$$\div(content\text{↵}current\_v\text{↵}G = x \ \&\ x \in T_{SD}, 1, tokens\text{↵}v_1 := \widetilde{\bigcup}(tokens\text{↵}v_1, < cs >)), \div(content\text{↵}current\_v\text{↵}G = '10',$$

$$tokens\text{↵}v_1 := \cdot(tokens\text{↵}v_1, < cp >)),\ current\text{↵}v := start\_v\text{↵}G,\ amount\_l\text{↵}G := amount\_l\text{↵}G + 1))\rangle.$$

Следующее правило позволяют завершить процесс построения конструкции-графа:

$$\widetilde{s}_5 = \langle \alpha \widetilde{\rightarrow} \varepsilon \rangle\ .$$

**Интерпретация КПС графа.** Интерпретируем графовую структуру:

$$\left\langle C_g, C_{A,G} = \langle M_A, V_A, \Sigma_A, \Lambda_A \rangle \right\rangle_I \mapsto \left\langle {}_A C_g,\ {}_A C_g = \left\langle M_g, \Sigma_g, \Lambda_4 \right\rangle \right\rangle,$$

где $\Lambda_4 \supset \Lambda_3$, $V_A = \{A_i^0 \,|_{X_i}^{Y_i}\}$ – множество базовых алгоритмов [10, 13], $X_i, Y_i$ – множества определений та значений алгоритма $A_i^0 \,|_{X_i}^{Y_i}$. $\Lambda_A = \{M_A = \bigcup\limits_{A_i^0 \in V_A} (X(A_i^0) \cup Y(A_i^0)) \bigcup \Omega(C_l) \bigcup \Omega(C_g)\}$ – неоднородный носитель,

$\Omega(C_l), \Omega(C_g)$ – множество языковых и графовых конструкций, которые удовлетворяют $C_l$ и $C_g$ соответственно; $\Lambda_4 \supset \{(A_6 \text{↵}"||\widetilde{\Rightarrow}"); \ (A_9 \text{↵}":\widehat{=}"); \ (A_{10} \text{↵}":\doteq"); \ (A_{11} \text{↵}"\div"); \ (A_{12} \text{↵}"\#"); \ (A_{13} \text{↵}"+"), (A_{14} \text{↵}"\widetilde{\cup}"), (A_{15} \text{↵}"\cup")\}$.

Структура ${}_{A,G}C_g$ включает алгоритмы, реализующие такие операции:

- $A_1^0$, $A_2^0$, $A_3 - A_5, A_7 - A_8$ – аналогичны алгоритмам структуры $C_A$;
- $A_6 \mid_{\sigma,\Psi_g}^{\overline{\Omega}}$ – полный вывод, где $\sigma$ – аксиома, $\overline{\Omega}$ – множество сформированных конструкций;
- $A_9 \mid_{v_1,v_2,G}^{e}$ – определение дуги $e$, соединяющей две заданные вершины $v_1, v_2$ в графе $G$;
- $A_{10} \mid_{x,V}^{v}$ – нахождение вершины $v$ с заданным значением атрибута веса $x$ в множестве вершин $V$;
- $A_{11} \mid_{c,n,L}^{L}$ – выполнение $n$ операций из списка $L$, если $c = true$, $L$ – список из $n$ операций;
- $A_{12} \mid_{Q}^{x}$ –вычисление мощности $x$ множества $Q$;
- $A_{13} \mid_{a,b}^{c}$ – сложения двух чисел $a, b$, $c$ – результат;
- $A_{14} \mid_{G_1,G_2}^{G}$ – объединение графов $G_1, G_2$ в $G$;
- $A_{15} \mid_{Q_1,Q_2}^{Q}$ – объединение множеств $Q_1, Q_2$ в $Q$.

**Реализация КПС графа.** Реализация структуры ${}_AC_g$ заключается в формировании графовых конструкций, которые имеют однозначное соответствие конструкциям $\overline{\Omega}({}_AC_T)$ путем выполнения алгоритмов, связанных с операциями сигнатуры, по правилам аксиоматики:

$$ {}_AC_g \ {}_R\mapsto \overline{\Omega}({}_AC_g), $$

где $\overline{\Omega}({}_AC_g) \subset \Omega({}_AC_g)$.

## Сжатие графа

Данная графовая модель может быть использована для сравнения текстов, поиска подстроки в строке. Для ускорения процесса сравнения можно применить сжатия графа. Для этого рассмотрим такую структуру:

$$ C = \langle M, \Sigma, \Lambda \rangle_S \mapsto C_C = \langle M_C, \Sigma_C, \Lambda_C \rangle, $$

где $\Lambda_C = \{M_C = V \cup E \cup \Omega_g, \ \Sigma_C = \{\{\cdot, :=\}, \{\rightarrow, |\Rightarrow, ||\widetilde{\Rightarrow}\} \cup \Psi_C\}, \Psi_C = \{\psi_C = \langle s_i, g_i \rangle\}\}$.

**Частичная аксиоматика операций.**

Отношение подстановки и операции подстановки, частичного и полного вывода, конкатенации, присваивания аналогичны одноименным операциям КПС графа.

**Интерпретация КПС сжатия графа.** Алгоритмы операций, использованы в рассматриваемой структуре, аналогичны алгоритмам одноимённых операций структуры для построения графа.

**Конкретизация КПС сжатия графа.** Выполним конкретизацию структуры для сжатия графа:

$$ C_{C\ K}\mapsto C_{CK} = \langle M_{CK}, \Sigma_{CK}, \Lambda_{CK} \rangle, $$

где $M_{CK} = M_C \bigcup N_{CK}, N_{CK} = \{\sigma\}$.

Рассмотрим правила, позволяющие сжать граф.

$$ s_1 = \langle \sigma \rightarrow G\sigma \rangle, \ s_2 = \langle G\sigma_d \rightarrow G\sigma \rangle, $$

$$ g_1 = g_2 = \langle e_1 := (v_i, v_j, G), e_2 := (v_j, v_k, G), \div (routes \text{↵} e_1 = routes \text{↵} e_2, 4, d := true, content \text{↵} v_i := \cdot (content \text{↵} v_i, content \text{↵} v_j), $$

$$ tokens \text{↵} v_i := \cdot (tokens \text{↵} v_i, tokens \text{↵} v_j), end \text{↵} e_1 := v_k) \rangle, \ s_3 = \langle \sigma \rightarrow \varepsilon \rangle. $$

**Реализация КПС сжатия графа.** Реализация интерпретируемой структуры ${}_AC_C$ заключается в формировании графовых конструкций, которые имеют однозначное соответствие конструкциям $\overline{\Omega}({}_AC_T)$ и $\overline{\Omega}({}_AC_g)$ путем выполнения алгоритмов, связанных с операциями сигнатуры, по правилам аксиоматики:

$$ {}_AC_C \ {}_R\mapsto \overline{\Omega}({}_AC_C) ,$$

где $\overline{\Omega}({}_AC_C) \subset \Omega({}_AC_C)$ .

## Выводы

Проблема поиска информации, ее сравнении требует автоматизированных средств решения. С данной целью целесообразной является разработка средств формализации для представления текстов.

Формальные грамматические и алгоритмические структуры использованы как средство формализации текста в виде конструкций с атрибутами. Конструктивно-продукционный подход позволяет определить порядок конструирования и обработки текста, представить его в виде графовых структур с целью снижения алгоритмической сложности.

В рамках рассмотренного атрибутивного подхода могут быть решены задачи по таким направлениям, как семантический поиск и сопоставление текстов для выявления заимствований, что является ключевыми моментами в решении проблем плагиата.


1. *Efstathios* Stamatatos Plagiarism Detection Based on Structural Information – CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
2. *Leilei Kong, Zhimao Lu, Haoliang Qi and Zhongyuan Han.* Detecting High Obfuscation Plagiarism: Exploring Multi-Features Fusion via Machine Learning // International Journal of u-and e-Service, Science and Technology – 2014. – Vol. 7, N 4. – P. 385–396.
3. *Ahmed Hamza Osman, Naomie Salim, Mohammed Salem Binwahlan.* Plagiarism Detection Using Graph-Based Representation // JOURNAL OF COMPUTING. – 2010. – Vol. 2, N 4. – P. 36 – 41.
4. *Bin-Habtoor A.S., Zaher M.A.* A Survey on Plagiarism Detection Systems [Text] // International Journal of Computer Theory and Engineering. – 2012. – Vol. 4, N 2. – P. 185–188.
5. *Ahmed Hamza Osman, Naomie Salim, Mohammed Salem Binwahlan, Hanza Hentably, Albaraa M. Ali.* Conceptual Similarity and graph-based method for plagiarism detection // Journal of Theoretical and Applied Information Technology. – 31st October 2011. – Vol. 32, N 2. – P. 135–145.
6. *Mozgovoy M.* Maxim Mozgovoy Desktop Tools for Offline Plagiarism Detection in Computer Programs // Informatics in Education. – 2006. – Vol. 5, N 1. – P. 97–112.
7. *Mozgovoy M., Kakkonen T., Cosma G.* Automatic Student Plagiarism Detection: Future Perspectives // Journal of Educational Computing Research. – 2010. – Vol. 43(4). – P. 507–527.
8. *Шинкаренко В.І., Куроп'ятник О.С.* Система контролю плагіату в студентських роботах // Восточно-Европейский журнал передовых технологий. – Харьков: Технологический центр, 2012. – № 4/2 (58). – С. 32–36.
9. *Ільман В.М., Скалозуб В.В., Шинкаренко В.І.* Формальні структури та їх застосування: мононографія. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. – 205 с.
10. *Шинкаренко В.И., Ильман В.М.* Конструктивно-продукционные структуры и их грамматические интерпретации. I. Обобщенная формальная конструктивно-продукционная структура // Кибернетика и системный анализ. – Киев, 2014. – Т. 50, № 5 – С. 8–16.
11. *Шинкаренко В.И. , Васецкая Т.Н.* Моделирования процесса адаптации алгоритмов сжатия средствами конструктивно-продукционных структур // Кибернетика и системный анализ. – 2015. – Т. 51. – № 6. – С. 19–34.
12. *Шинкаренко В.И., Ильман В.М., Забула Г.В.* Конструктивно-продукционная модель структур данных на логическом уровне // Проблемы программирования. – 2014. – № 2–3. – С. 10–16.
13. *Шинкаренко В.И., Ильман В.М.* Конструктивно-продукционные структуры и их грамматические интерпретации. II. Уточняющие преобразования // Кибернетика и системный анализ. – 2014. – № 6. – С. 15–28.
14. *Шинкаренко В.И., Ильман В.М., Скалозуб В.В.* Структурные модели алгоритмов в задачах прикладного программирования. Часть I. Формальные алгоритмические структуры // Кибернетика и системный анализ. – 2009. – № 3. – С. 3–14.


## References


1. EFSTATHIOS, STAMATATOS (2011) Plagiarism Detection Based on Structural Information in CIKM'11. Glasgow, Scotland, UK October 24–28, 2011.
2. LEILEI KONG & ZHIMAO LU & HAOLIANG QI & ZHONGYUAN HAN (2014) Detecting High Obfuscation Plagiarism: Exploring Multi-Features Fusion via Machine Learning. International Journal of u-and e-Service, Science and Technology. 7 (4). P. 385–396.
3. AHMED HAMZA OSMAN & NAOMIE SALIM & MOHAMMED SALEM BINWAHLAN (2010) Plagiarism Detection Using Graph-Based Representation Journal of Computing. 2 (4). P. 36 – 41.
4. A. S. BIN-HABTOOR & M. A. ZAHER (2012) A Survey on Plagiarism Detection Systems. International Journal of Computer Theory and Engineering. 4 (2). P. 185 – 188.
5. AHMED HAMZA OSMAN & NAOMIE SALIM & MOHAMMED SALEM BINWAHLAN & HANZA HENTABLY & ALBARAA M. ALI (2011) Conceptual Similarity and graph-based method for plagiarism detection. Journal of Theoretical and Applied Information Technology. 32 (2). P. 135–145.
6. MOZGOVOY, M. (2006) Desktop Tools for Offline Plagiarism Detection in Computer Programs. Informatics in Education. 5 (1). pp. 97–112
7. M. MOZGOVOY & T. KAKKONEN & G. COSMA (2010) Automatic Student Plagiarism Detection: Future Perspectives. Journal of Educational Computing Research. 43 (4). P. 507-527.

8. V. SHYNKARENKO & E. KUROPYATNICK (2012) Monitoring system of plagiarism in student works. East Europe Journal of Enterprise Technologies. 4/2 (58). p. 32 – 36.
9. V. ILMAN, V. SKALOZUB & V. SHYNKARENKO Formal structures and their applications: monograph. Dnipropetrovsk: DNURT named after academician V. Lazaryan.
10. SHYNKARENKO V. & ILMAN V. (2014) Constructive-synthesizing structures and their grammatical interpretations. I. Generalized formal constructive-synthesizing structure. Cybernetics and Systems Analysis. 50 (5). P. 8–16.
11. SHYNKARENKO V. & VASETSKA T. (2015) Modeling the Adaptation of Compression Algorithms by Means of Constructive-Synthesizing Structures. Cybernetics and Systems Analysis. 51(6). P. 19 – 34.
12. SHYNKARENKO V. I., ILMAN V. M., ZABULA H. V. (2014) Logical view for construction-synthesis model of data structeres. Problems in programming. 2-3 Special issue.
13. SHYNKARENKO V. & ILMAN V (2014). Constructive-production structures and their grammatical interpretations. II. Clarifying conversions. Cybernetics and Systems Analysis. 6. P. 15 – 28.
14. SHYNKARENKO V. & ILMAN V., SKALOZUB V. (2009) Structural models of algorithms in problems of applied programming. i. formal algorithmic structures. Cybernetics and Systems Analysis. 3. P. 3 – 14.

## *Об авторах:*

*Шинкаренко Виктор Иванович,*
доктор технических наук, профессор,
заведующий кафедрой Компьютерные информационные технологии.
Количество научных публикаций в украинских изданиях – 200.
Индекс Гирша – 5.
http://orcid.org/0000-0001-8738-7225,

*Куропятник Елена Сергеевна,*
аспирант кафедры Компьютерные информационные технологии.
Количество научных публикаций в украинских изданиях – 14.
Индекс Гирша – 1.
http://orcid.org/0000-0003-2286-884X.

## *Место работы авторов:*

Днепропетровский национальный университет
железнодорожного транспорта имени академика В. Лазаряна.
49010, Днепропетровск, ул. Лазаряна, 2, ДНУЖТ, кафедра КИТ.
Тел.: (056) 373 1535.
E-mail: shinkarenko_vi@ua.fm, elenadiit@rambler.ru