УДК 004.021 **Information Control Systems and Technologies, pp. 145-149**

К.т.н. Пригожев А.С.

МОДЕЛЬ ПРОГРАММНОГО КОДА НА БАЙЕСОВСКОЙ СЕТИ ФУНКЦИОНАЛЬНЫХ ГРАФОВ

Ph.D. Prygozhev O.

SOFTWARE CODE MODEL BASED ON BAYESIAN NETWORK OF FUNCTIONAL GRAPH

Software size increase and time reduction for his development has led to the emergence of a new class of data storage systems - software code repositories. A common drawback of such systems is the storage of solved problem as software source text. This leads to redundant data storage within the repository. Therefore, it is advisable to move from storing multiple language implementations of the same task to storing the problem solution and generating it for different language syntaxes.

Now exists software code generators for programming languages [1,2]. But they require from the developer additional knowledge of their internal languages and notations. They also implement software source code generation according to some formal description of the algorithm, but they do not have the opportunity to solve the inverse problem — the generation of a formal description of the problem solution from the software source code. The solution to this inverse problem in our case is the key one, since fast development technologies are now widespread. When using them, the customer is actively involved in solving the problem and changes are made directly to the software source code.

To solve this problem, it is necessary to solve two main subtasks: developing a unique identification method for operators of different programming languages performing the same actions and developing the software source text generation method.

The method of unique identification of various program operators is based on the use of a finite set S_p,.primes, the which simplicity is proven. The finiteness of this set follows from the fact that $S_p \subset \mathbb{N}$, the fact that the greatest prime number, the simplicity of which is proved, is known [] and the well-known statement from the fact that every finite set of natural numbers contains the greatest number. Also consideration the set C characters of the Unicode table or any other signed alphabet []. The Unicode table or any other alphabet defines a bijection $\varphi\colon \mathrm{C} \to \mathrm{N}$, where $\mathrm{N} \subset \mathbb{N}$ is the set of character positions in the alphabet. Introduce into consideration a bijection defining a table of prime numbers, the simplicity of which is proved, ordered in ascending order: $\omega\colon \mathrm{N} \to \mathrm{S_p}$. The result of the computation of the composition $\psi(c) = \varphi \circ \omega$, where $c \in C$ will be called the semantic number of the symbol. Considering the uniqueness of the decomposition into prime factors, any composite number whose factors are the semantic number of a symbol will be called the semantic number of the set. In this case, the well-known set-theoretic operations on sets of union, intersection, difference can be represented as known arithmetic operations, respectively, of the least common multiple $lcm(m,n)$, the greatest common divisor $gcd(m,n)$ and integer division $\mathrm{s}_{alph}/\mathrm{m}$. Here m, n are the semantic numbers of the sets belonging to a certain universe, and $\mathrm{s_{alph}}$ is the semantic number of the given universe, which in our case is the sign alphabet. The concept of semantic number is defined and makes sense only for finite sets.

The language semantic constant (LSC) is a prime number that is in the next position in the table of prime numbers after the last used symbol semantic number of this language. This number will be denoted by $\mathrm{l_{sn}}$. Then the semantic number of the symbols vector is determined according to the following formula:

$$\begin{pmatrix} \psi(c_1) \\ \psi(c_2) \\ \vdots \\ \psi(c_n) \end{pmatrix} * \begin{pmatrix} 1 \\ l_{sn} \\ \vdots \\ l_{sn}^n \end{pmatrix} \tag{1}$$

The semantic number of the empty element is 1. First component in the second factor of the formula (1) means the zero degree $\mathrm{l_{sn}}$.

Introduce into consideration set of domain attributes vectors Subj: $\mathrm{Subj} = \mathrm{A}_1 \times \mathrm{A}_2 \times \ldots \times \mathrm{A}_\mathrm{n}$. Element $a \in A_i$ may be either a separate numerical value or a numerical vector, or a set of them.

All sets in this Cartesian product are ordered by reducing the degree of use for uniquely identifying the vector $\mathrm{s} \in \mathrm{Subj}$ Attributes used in unique identification cannot be equal to 0.

Consider a directed graph $G = (V; E)$. The set of vertices of a graph is defined by the following relation $\mathrm{V} \subset \mathrm{N} \times \mathcal{P}(\mathrm{Subj})$, where $\mathrm{N} \subset \mathbb{N}$ is a finite subset of natural numbers, $\mathcal{P}(\mathrm{Subj})$ - Boolean vectors Subj. Each vertex of this graph is numbered and contains set of vectors of attributes of the domain. Each edge of the graph is defined as follows: $\mathrm{E} \subset \mathrm{V}^2 \times \mathrm{F}$, where V is the set of vertices, the set F is the set of functions defined as the arithmetic product of two functions: $\mathrm{k_e(s) = p_l(s) * f(s), s \in Subj}$. The function $p(s)$ is defined as follows:

$$p_l(s) = \begin{cases} p_l(s) = 0, \text{если } P(s) = 0 \\ p_l(s) = 1, \text{если } P(s) = 1 \end{cases}$$

In this definition, $\mathrm{s \in Subj, P(s)}$ is an arbitrary one-place predicate on the set Subj. $f(s)$ is of type $\mathrm{f\colon Subj \to Subj}$ and defines a unary operation on the vector $\mathrm{s} \in \mathrm{Subj}$, the result of which is always a number. Note that the function $\mathrm{f(s)}$ can be a non-zero constant function. Graph G is internal presentation for computer of P-schema[2].

The second projection of the vertex v_2 into which the edge enters is connected with the vertex v_1 from which the edge proceeds, by the relation:

$$\text{пр}_2 v_2 = \{v : v = \text{k}_e(\text{пр}_2 v_1), v \neq 0\}$$

In the set of vertices of this graph, existing two special vertices - the starting one and the final one. The starting vertex meets the condition $\pi_1 v = min\, N$, the final vertex meets the condition $\pi_1 v = max\, N$. We denote these vertices as the starting v_s and the final v_f. The set of vectors $\text{s} \in \text{Subj}$, which are the second projection of the element of the initial vertex, means the initial processing data, and the set of vectors, which are the second projection of the final vertex, symbolize the result of the processing.

We now construct the graph $\text{G}_{\text{sem}} = (\text{V}_{\text{sem}}; \text{E}_{\text{sem}})$ of the set of oriented graphs $G = (V; E)$, considered earlier, guided by the following definitions

1. $V_{sem} = \bigcup_{i=1}^{m} V$; m is the number of considered graphs G
2. E_{sem} contains two classes of edges: starting and returning
3. The return edge of the graph G_{sem} always comes from the vertex $v_f$of the graph G
4. The starting edge of the graph G_{sem} always enters the vertex v_s of one graph G

The graph constructed in this way implements the well-known operation of superposition (convolution) of the algorithms [3]. Let us compare to each starting edge the probability of transition along this edge. In this case, created a Bayesian network consisting of graphs G. A group of starting edges emanating from one vertex always have a total probability equal to 1 and each edge is an operation of a fuzzy superposition of schemes.

It was shown earlier that the function f (s) located on the edge of the graph G can be a constant function. If assume that this constant function is the semantic number of a word, and the word denotes an action, then the operation of fuzzy superposition gives the opportunity to get all the options for realizing this action at the same time always generating a unique identifier for this action as a whole. The construction of a program model for a given graph can be carried out both in the process of translation, and using a syntactically correct finished program with the special designed software generators. The model is adapted not only for presentation, but also for the execution of the program code, which is achieved by circulating vectors in the graph.

Reference

1. Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ / Е.А. Яценко, А.С. Мохница // Проблеми програмування, 2004. – N 2,3. – С. 444 – 450. (In Russian)
2. Новая графическая концепция программирования // URL: http://glushkov.org/wp-content/210518%20Lambert%20Academic%20Publishing.pdf (In Russian)
3. Ф.И. Андон, А.Е. Дорошенко, Г.Е. Цейтлин, Е.А. Яценко Алгеброалгоритмические методы параллельного программирования – К.: Академпериодика, 2007.

УДК 004.93’11 **Information Control Systems and Technologies, pp. 149-152**

К.т.н. Гришин С.И., д.т.н. Шинкевич Е.С.

ВЛИЯНИЕ ОСОБЕННОСТЕЙ АНАЛИЗА СТРУКТУРЫ И СВОЙСТВ СИЛИКАТНЫХ КОМПОЗИТОВ НА ПРОЕКТИРОВАНИЕ OLAP-СИСТЕМЫ ЭКСПЕРИМЕНТАЛЬНО-СТАТИСТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Ph.D. Grishin S.I., Dr.Sci. Shinkevich E.S.

SILICATE COMPOSITES STRUCTURE AND PROPERTIES ANALYSIS FEATURES INFLUENCE ON DESIGN OF AN OLAP SYSTEM FOR EXPERIMENTAL STATISTICAL MODELING

Решение задачи обеспечения конкурентоспособности производства строительных материалов приводит к возрастанию роли и значения методов компьютерного материаловедения с использованием