СПИСОК ЛИТЕРАТУРЫ

1. Гольденберг Л. М. Цифровая обработка сигналов / Л.М. Гольденберг, Б.Д. Матюшкин, М.Н. Поляк. – М.: Радио и связь, 1985.
2. The Scientist and Engineer's Guide to Digital Signal Processing. Second Edition / Ed. by S. W. Smith. – California Technical Publishing, 1999.. http://www.dspguide.com/
3. Blahut R. E. Fast algorithms for digital signal processing / R.E. Blahut. – Addison-Wesley, 1985.
4. Numerical Recipes in C. The Art of Scientific Computing. Second Edition / W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery. – Cambridge University. Press, 1988. http://www.library.cornell.edu/nr/
5. Arndt, J. Algorithms for programmers: ideas and source code. – 2002. http://www.jjj.de/

Статья поступила в редакцию 5.07.2005

УДК 681.3:519.6

Г. С. Иванова

ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ СТРУКТУРИЗАЦИИ АЛГОРИТМОВ

Проанализированы характеристические особенности структурных алгоритмических конструкций, предложены математические модели как самих конструкций, так и структурных алгоритмов в целом. Определено понятие факторизации фрагмента алгоритма и аксиоматика этой операции. Выявленные свойства структурного алгоритма и формальная постановка задачи структуризации неструктурного являются основой для автоматизированного решения задачи структуризации алгоритмов.

Одним из принципов построения технологичных программ считается структурное программирование [1], которое предполагает при разработке программ на универсальных процедурных языках высокого уровня использование только так называемых "структурных" конструкций. К структурным относят три основные конструкции: следование, ветвление и цикл с условием на входе ("цикл-пока"), и три дополнительные: выбор, цикл с заданным количеством повторений ("счетный") и цикл с условием на выходе ("цикл-до"). Из накопленного к настоящему времени опыта структурного программирования следует, что технологичность программ, составленных с использованием только этих структур, гораздо выше, чем в том случае, когда разработчик использует неструктурное программирование. Это связано с тем, что структурные программы легче анализировать, они содержат меньше ошибок, проще тестируются и отлаживаются. Однако

разработка таких программ предполагает некоторый опыт программирования и определенным образом сформированное "программистское" мышление, которым инженеры-математики — разработчики новых алгоритмов решения различных задач — могут и не обладать. В связи с этим большинство универсальных языков допускает использование неструктурных вариантов передачи управления.

Неструктурный вид программ, получаемых при использовании операторов неструктурной передачи управления, порождает целый ряд проблем. Во-первых, в этом случае сложно разобраться в получаемой программе, что уменьшает возможности "ручной" оптимизации на данном уровне. Во-вторых, существенно усложняется, если не становится невозможным, автоматический анализ вычислительной сложности полученных программ. Естественным решением проблемы является создание средств автоматического преобразования программ к структурному виду.

Задача структуризации не нова. Она ставилась как самостоятельно, так и в качестве подзадачи при решении других задач, например задачи декомпиляции, задачи распараллеливания алгоритмов и задачи анализа вычислительной сложности.

Принципиальная возможность структуризации программ была доказана, еще в начале 60-х годов прошлого века академиком В.М. Глушковым [2]. В последующем им и его последователями была разработана система алгоритмических алгебр, содержащая строгую методику структуризации алгоритмов [3], но для очень ограниченного класса схем программ — схем Янова.

Для структуризации реальных программ и алгоритмов в настоящее время предложено много способов [4–9], но при этом не существует общей методики выявления неструктурностей, предлагаемые способы не базируются на анализе моделей алгоритмов или программ, а потому не могут быть автоматизированы. Единственная известная автоматизированная методика выполняет лишь частичную структуризацию программ в процессе декомпиляции [10]. Она основана на выявлении неструктурностей в программах по присутствию оператора GOTO. Однако использование этого оператора может и не нарушать структурности конструкций.

Применительно к рассматриваемому случаю частичной структуризации недостаточно. Следовательно, требуется разработать подход к преобразованию программ к структурному виду, пригодный для автоматизации.

Модель программы для задачи структуризации. При решении частных задач структуризации программ, написанных на языках высокого уровня, на правила преобразования оказывает существенное вли-

яние семантика конкретного языка. Таким образом, результаты, полученные при исследовании программ на одном языке, требуют дополнительной интерпретации для преобразования программ, написанных на другом.

Естественным выходом из этого положения является преобразование к структурному виду не программ, а алгоритмов, описанных в некотором операторном базисе, который является допустимым семантикой всех или широкого класса языков высокого уровня. Такой класс операторов определен в [11]. Он включает операторы: изменения данных, вычисления условий, ветвления и слияния потоков управления, позволяя описывать как структурные, так и неструктурные конструкции.

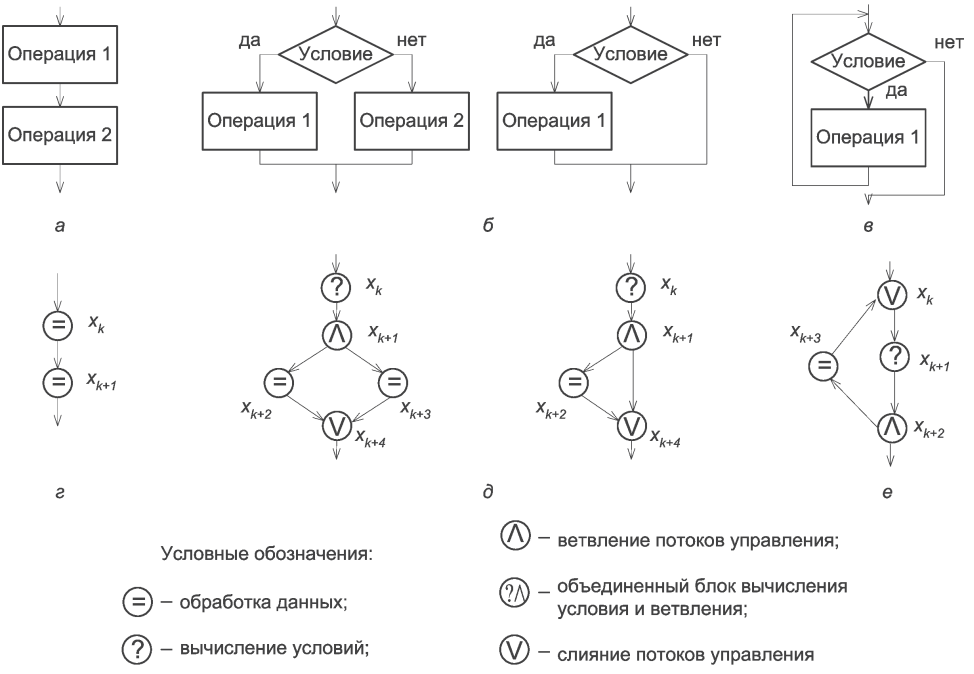
Модель программы для решения задачи структуризации должна отображать как отдельные операторы, так и последовательность их выполнения. Таким свойством обладает управляющий граф алгоритма, однако при переходе от алгоритма к модели линейные последовательности операторов должны быть свернуты, что уменьшит размерность задачи, но не вызовет потери информации о структуре программы. Кроме того, поскольку при решении задачи структуризации может потребоваться изменение порядка выполнения операторов преобразования данных, в модели должно быть отображено отношение “операторы–данные”.

Окончательно, в качестве модели программы будем использовать композицию управляющего графа и двудольного ориентированного графа “операторы-данные” [11], которая отображает не только потоки управления, но и потоки данных.

Основные структурные конструкции, их модели и свойства. На рисунке представлены фрагменты схем алгоритмов и модели основных структурных конструкций: следования, ветвления (полная и усеченная) и цикла-пока.

Перечисленные выше конструкции в свое время были выбраны в качестве основных [1], так как через них можно легко реализовать любую из дополнительных структурных конструкций: выбор, цикл-до и счетный цикл, в то время, как обратное достаточно сложно.

Поскольку в алгоритмических языках программирования высокого уровня, таких как Паскаль и Си, существуют операторы, реализующие как основные, так и дополнительные конструкции, при разработке алгоритмов программ для этих языков обычно используют все шесть конструкций. Однако при разработке модели структурного алгоритма целесообразно использовать минимальное количество конструкций, ограничившись основными, так как это позволит упростить анализ алгоритмов.



Основные структурные конструкции алгоритма и их модели:
а, г — следование; *б, д* — ветвление; *в, е* — цикл-пока

Исследование свойств моделей основных структурных конструкций начнем с модели оператора обработки данных, которая в качестве элемента входит в любую рассматриваемую конструкцию.

1. Модель оператора обработки данных представляет собой одновершинный кусок $G^0(X^0,U^0)$, такой что:

$$X^0 = \{x_k\}, \tag{1а}$$

$$t(x_k) = \text{“обработка данных”}, \tag{1б}$$

$$U^0 = U^0_{i,j} = \{\vec{u}(\varnothing,\, x_k),\ \vec{u}(x_k,\, \varnothing)\}, \tag{1в}$$

где $t(x_k)$ — тип k-ой вершины; $U^0_{i,j}$ — подмножество внешних ребер куска $G^0(X^0, U^0)$.

Из соотношений (1а)–(1в) следует, что

$$|X^0| = 1, \tag{1г}$$

$$|U^0| = 2, \tag{1д}$$

$$P(X^0) = S(X^0) = 1, \tag{1е}$$

где $P(X^0)$, $S(X^0)$ — полустепени исхода и захода куска $G^0(X^0,\, U^0)$.

Нетрудно убедиться, что характеристики (1б), (1г–1е) составляют полный набор формальных признаков (инвариантов) данной модели.

2. Конструкция “следование” подразумевает последовательное выполнение операций обработки данных. Моделью конструкции является кусок управляющего графа $G^1(X^1, U^1) = Ch(x_k, x_{k+n})$, где Ch — простая цепь, в которой x_k — вершина начала конструкции следования, а x_{k+n} — вершина конца конструкции следования (см. рисунок, *г*), т. е. подмножество внешних ребер куска $G^1(X^1, U^1) - U^1_{i,j} = \{\vec{u}(\varnothing, x_k), \vec{u}(x_{k+n}, \varnothing)\}$.

Формальными признаками данной конструкции являются

$$P(X^1) = S(X^1) = 1, \tag{2а}$$

$$(\forall x_l \in X^1)(p(x_l) = s(x_l) = 1 \& t(x_l) = \text{“обработка данных”}), \tag{2б}$$

где $P(X^1)$, $S(X^1)$ — полустепени исхода и захода куска $G^1(X^1, U^1)$; $p(x_l)$, $s(x_l)$ — полустепени исхода и захода вершины x_l, $t(x_k)$ — тип k-й вершины.

3. Конструкция “ветвление” подразумевает выполнение операций вычисления условия, ветвления потоков управления, обработки данных и слияния потоков управления. Моделью конструкции является кусок управляющего графа $G^2(X^2, U^2)$ (см. рисунок, *д*).

Подмножество внешних ребер куска $G^2(X^2, U^2) - U^2_{i,j} = \{\vec{u}(\varnothing, x_k), \vec{u}(x_{k+4}, \varnothing)\}$, где x_k — вершина начала конструкции ветвления, x_{k+4} — вершина конца конструкции ветвления.

Формальные признаки данной конструкции:

$$P(X^2) = S(X^2) = 1, \tag{3а}$$

$$t(x_k) = \text{“вычисление условия”}, \tag{3б}$$

$$t(x_{k+4}) = \text{“слияние”}, \tag{3в}$$

$$(F^{+1}(x_k) = x_{k+1}) \& (t(x_{k+1}) = \text{“разветвление”}), \tag{3г}$$

$$\begin{aligned}(F^{+1}(x_{k+1}) = \{x_{k+2}, x_{k+3}\}) \& ((x_{k+2} = x_f \& x_{k+3} = x_h) \vee \\ \vee (x_{k+2} = x_{k+4} \& x_{k+3} = x_h) \vee (x_{k+2} = x_f \& x_{k+3} = x_{k+4})) \& \\ \& (t(x_h) = t(x_f) = \text{“обработка данных”}).\end{aligned} \tag{3д}$$

где $P(X^2)$, $S(X^2)$ — полустепени исхода и захода куска $G^2(X^2, U^2)$, $t(x_k)$ — тип k-й вершины.

Характеристики (3а)–(3д) образуют полный набор формальных признаков (инвариантов) данной модели.

4. Конструкция “цикл-пока” подразумевает также выполнение операций вычисления условия, ветвления потоков управления, обработки данных и слияния потоков управления, но в другой последовательности. Моделью конструкции является кусок управляющего графа $G^3(X^3, U^3)$ (см. рисунок, *е*). Подмножество внешних ребер куска

$G^3(X^3, U^3) - U^3_{i,j} = \{\vec{u}(\varnothing, x_k), \vec{u}(x_{k+2}, \varnothing)\}$, где x_k — вершина начала цикла, x_{k+2} — вершина конца цикла.

Формальные признаки данной конструкции:

$$P(X^3) = S(X^3) = 1, \tag{4а}$$

$$(t(x_k) = \text{“слияние”}), \tag{4б}$$

$$(t(x_{k+2}) = \text{“разветвление”}), \tag{4в}$$

$$(F^{+1}(x_{k+2}) \cap F^{-1}(x_k) = x_{k+3}) \& (t(x_{k+3}) = \text{“обработка данных”}), \tag{4г}$$

$$(F^{+1}(x_k) = x_{k+1}) \& (t(x_{k+1}) = \text{“вычисление условия”}). \tag{4д}$$

где $P(X^3)$, $S(X^3)$ — полустепени исхода и захода куска $G^3(X^3, U^3)$; $t(x_k)$ — тип k-й вершины.

Характеристики (4а)–(4д) образуют полный набор формальных признаков (инвариантов) данной модели.

Таким образом, множество базовых моделей структурного алгоритма G^B включает четыре модели:

$$G^B = \{G^0, G^1, G^2, G^3\},$$

где G^0 — модель оператора обработки данных; G^1 — модель конструкции следования; G^2 — модель конструкции ветвления; G^3 — модель конструкции цикла-пока.

Анализ инвариантов моделей перечисленных конструкций показывает, что необходимым (но не достаточным) формальным признаком того, что некоторый кусок графа G_j представляет собой модель структурной конструкции, может служить свойство (1е), (2а), (3а), (4а). Недостаточность этого признака обусловлена тем, что некоторый фрагмент с одним входом и одним выходом может включать неструктурные конструкции.

Различать модели конструкции можно, используя свойства (2б), (3б) и (4б).

Структурный алгоритм, его модель и свойства. В соответствии с правилами структурного программирования в качестве элементов обработки данных в основных структурных конструкциях могут выступать не только отдельные операторы обработки данных, но и сами структурные конструкции. В этом случае мы получаем иерархически вложенные (сложные) структурные конструкции с некоторым количеством уровней вложения.

Распознавание сложных конструкций при анализе алгоритмов подразумевает некоторое их отождествление с базовыми. Такое распознавание может быть выполнено путем установления гомеоморфизма

между куском графа, представляющим базовую конструкцию, и куском, являющимся моделью сложной, иерархически вложенной структурной конструкции, с точностью до внутренних вложенных конструкций. Доказательство гомеоморфизма кусков можно выполнить, свернув (факторизовав) вложенные конструкции и установив изоморфизм кусков.

Под операцией свертки (факторизации) структурной конструкции, вложенной в данную, будем понимать свертку в одну всех вершин X_j куска графа $G_j(X_j,\, U_j)$, являющегося моделью этой конструкции, таким образом, что

$$factor(G_j(X_j,\, U_j).) = \begin{cases} G_j(X_j, U_j), \text{ если } |X_j| = 1, \\ G_j^F(X_j^F, U_j^F), \text{ если } |X_j| > 1, \end{cases}$$

где $X_j^F = \{x_n\}$, x_n — единственная вершина, заменяющая свертываемый кусок графа; $t(x_n)$ ="обработка данных", $t(x_n)$ — тип вершины x_n; $U_{j,j}^F = \varnothing$, $U_{j,j}^F$ — множество внутренних ребер (петель) одновершинного куска; $U_j^F = U_{j,k}^F = \{\vec{u}(\varnothing,\, x_n),\, \vec{u}(x_n,\, \varnothing)\} = U_j \backslash U_{j,j} = U_{j,k}$, $U_{j,k}^F$, $U_{j,k}$ — множества внешних ребер одновершинного и свертываемого кусков; $U_{j,j}$ — множество внутренних ребер свертываемого куска.

При таком преобразовании сохраняется основное свойство моделей структурных алгоритмов: $S(X_j) = P(X_j) = S(X_j^F) = P(X_j^F) = 1$.

Формальное описание структуры "структурного" алгоритма и правила разбора таких алгоритмов определяет аксиоматика операций свертки над сложными структурными конструкциями:

$$\begin{aligned} &factor(G^0) = G^{0F} = G^0, \\ &factor(G_i \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}) = G^{0F}, \\ &factor(\{G_k\} \subset G^{1C} \& G_k \in \{G^{2F}, G^{3F}\}) = G^{1F}, \\ &factor(G_{01}, G_{02} \subset G^{2C} \& G_{01}, G_{02} \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}) = \\ &\qquad\qquad = G^{2F}, \\ &factor(G_0 \subset G^{3C} \& G_0 \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}) = G^{3F}, \end{aligned}$$

где G^{1C} — кусок управляющего графа, соответствующий линейной последовательности операторов обработки данных, операторов ветвления и операторов цикла-пока (в свою очередь, возможно, включающих внутренние структурные конструкции), т. е. $G^{1C} = Ch(G_i, G_j)$, где Ch — последовательность кусков, являющихся моделями указанных выше последовательных операторов, причем

$$(\forall k = \overline{i, j}) G_k \in \{G^0, G^2, G^{2F}, G^3, G^{3F}\};$$

G^{2C} — кусок управляющего графа, соответствующий оператору ветвления (сложной структурной конструкции типа G^2), ветви которого

$G_{01}, G_{01} \subset G^{2C}$ могут содержать вложенные структурные конструкции, т. е.

$$G_{01},\ G_{02} \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\};$$

G^{3C} — кусок управляющего графа, соответствующий оператору цикла-пока (структурной конструкции типа G^3), тело которого $G_0 \subset G^{3C}$ может содержать вложенные структурные конструкции, т. е. $G_0 \in \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}$.

Таким образом, получаем множество моделей конструкций, изоморфных базовым G^B и получаемых из сложных посредством свертки:

$$G^{BF} = \{G^{0F}, G^{1F}, G^{2F}, G^{3F}\}.$$

Формальная постановка задачи структуризации алгоритма. Формально задачу структуризации алгоритмов можно поставить следующим образом.

Выполнить преобразование

$$G^{УН}(< X_1, T >, < \vec{U}_1, P >) \xrightarrow{F_1} G^{УС}(< X_2, T >, < \vec{U}_2, P >) \text{ и}$$

$$G^{ДН}(\{< X_1, T >, < Y_1, < W_1, C_1 >>\}, \{< \overrightarrow{V}^1_1, T^1_1 >, < \overrightarrow{V}^2_1, < T^2_1, D^3_1 >>\}) \xrightarrow{F_1} \xrightarrow{F_1} G^{ДС}(\{< X_2, T >, < Y_2, < W_2, C_2 >>\}, \{< \overrightarrow{V}^1_2, T^1_2 >, < \overrightarrow{V}^2_2, < T^2_2, D^3_2 >>\}),$$

такое что

$$(\forall i \in I) \quad A^H(S_i) = A^C(S_i),$$

где $G^{УН}$ и $G^{УС}$ — куски управляющих графов неструктурной и структурной программ без вершин, сопоставленных началу и концу программы; $S = \{S_i / i \in I\}$ — множество всех допустимых наборов входных данных задачи, для решения которых предназначен данный алгоритм; A^H — неструктурная программа, управляющим графом которого является граф $G^{УН}$; A^C — структурная программа, соответствующая преобразованным графам $G^{УС}$ и $G^{ДС}$; при выполнении следующих условий, вытекающих из свойств алгоритмов:

$$|X_1| \leqslant |X_2|, \quad X_1 \cap X_2 \neq \varnothing,$$

$$|\vec{U}_1| \leqslant |\vec{U}_2|, \quad \vec{U}_1 \cap \vec{U}_2 \neq \varnothing;$$

и существует разрезание B куска графа $G^{УС}$ на совокупность кусков $B(G^{УС})$, удовлетворяющих условиям

$$(\forall G_i^{УС} \in B(G^{УС}))\ G_i^{УС} \neq \varnothing,\ i \in I,$$

$$(\forall G_i^{УС} \in B(G^{УС}))\ (G_i^{УС} \in G^{B} \vee\ G_i^{УС} \in G^{BF}),$$

$$\bigcup_{i \in I} G_i^{УС} = G^{УС},$$

$$(\forall G_i^{УС}, G_{i+1}^{УС} \in B(G^{УС}))(X_i^{УС} \cap X_{i+1}^{УС} = \varnothing\ \&\ U_i^{УС} \cap U_{i+1}^{УС} = \\ = U_{i,i+1} \varnothing |U_{i,i+1}| = 1),\quad i, i+1 \in I,$$
$$(\forall G_i^{УС}, G_j^{УС} \in B(G^{УС}))(X_i^{УС} \cap X_j^{УС} = \varnothing\ \&\ U_i^{УС} \cap U_{i+1}^{УС} = \varnothing), \\ i, j \in I,\ \ j \neq i \pm 1,$$

или, другими словами

$$factor^k(G^{УС}) = G^{0F}.$$

Сформулированные полные наборы инвариантов базовых структурных конструкций алгоритмов в виде характеристик их моделей — кусков ориентированного графа — обеспечивают возможность анализа свойств и характеристик алгоритма, в том числе и обнаружения неструктурностей и их структуризации. Выявленные свойства структурного алгоритма и формальная постановка задачи структуризации неструктурного являются основой для автоматизированного решения задачи структуризации алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. Дал У., Дейкстра Э., Хоор К. Структурное программирование. – М.: Мир, 1975.
2. Глушков В. М. Теория автоматов и формальные преобразования программ // Кибернетика. – 1965. – № 5. – С. 1–9.
3. Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра. Языки. Программирование. – Киев: Наукова думка, 1978. – С. 157–173.
4. Ashcroft E., Manna Z. The translation of GOTO Programs to WHILE Programs // Proc. IFIP Congress 71, Lubljana, Yugoslavia, August 23–28, 1971. – Amsterdam: North-Holland Publ. Co. – 1972. – V. 1. – № 1. – P. 250–255.
5. Kosaraju S. R. Analysis of Structured Programs // J. Computer and System Sci. – 1974. – V. 9. – № 2. – P. 232–255.
6. Knuth D. E., Floyd R. W. Notes on avoiding GOTO statements // Inform. Processing Letters. – 1971. – № 1. – P. 23–31.
7. Martin J. L. Generalized structured programming // AFIPS Conf. Proc. – Chicago, 1974. – V. 43. – P. 665–669.
8. Mills H. D. Mathematical foundation for structured programming // IBM Tech. Rep. 1972. FSC-72-6012. – P. 34.
9. Peterson W. W., Kasami T., Tokura N. On the capabilities of the WHILE, REPEAT, and EXIT statements // Comm. ACM. – 1973. – V. 16. – № 8. – P. 503–512.

10. Erosa A. M., Henron L. J. Taming Control Flow: A structured Approach to Elimination GOTO Statements // Proc. IEEE International Conf. on Computer Languages. May 1994. – P. 229–240.

11. Овчинников В. А., Иванова Г. С. Информационно-логическая модель алгоритма // Вестник МГТУ. Серия "Приборостроение". – 2005. – № 1. – С. 109–121.

Статья поступила в редакцию 27.10.2004

Иванова Галина Сергеевна родилась в 1954 г., окончила МВТУ им. Н.Э. Баумана в 1978 г. Канд. техн. наук, доцент кафедры "Компьютерные системы и сети". Автор 25 научных работ в области вычислительной техники и проектирования программных систем.

G.S. Ivanova (b. 1954) graduated from the Bauman Moscow Higher Technical School in 1978. Ph. D. (Eng.), assoc. professor of "Computer Systems and Networks" department of the Bauman Moscow State Technical University. Author of 25 publications in the field of computing technology and design of software systems.

УДК 004.89; 004.912; 004.5

В. А. Фомичев

КЛАСС ФОРМАЛЬНЫХ ЯЗЫКОВ И АЛГОРИТМ ДЛЯ ПОСТРОЕНИЯ СЕМАНТИЧЕСКИХ АННОТАЦИЙ ВЕБ-ДОКУМЕНТОВ

Предложена широко применимая и гибкая теория формального описания структурированных значений текстов на естественном языке (предложений и дискурсов) — теория стандартных К-языков (СК-языков). Анализ выразительной силы класса СК-языков дает возможность предположить, что СК-языки позволяют строить семантические аннотации произвольных Веб-документов и удобны для построения таких аннотаций. Теория СК-языков была использована при разработке широко применимой математической модели лингвистической базы данных (ЛБД) и сложного структурированного алгоритма семантико-синтаксического анализа текстов из представляющих практический интерес подъязыков естественного (русского) языка, базирующегося на построенной модели ЛБД. Алгоритм реализован в системе программирования Visual C++ и может быть широко использован для построения семантических аннотаций Веб-документов.

Благодаря бурному прогрессу компьютерной сети Всемирная Паутина (the World Wide Web, WWW, W3) пользователи сети во всем мире получили быстрый доступ к огромному количеству ЕЯ-текстов, относящихся к различным областям деятельности человека. С середины 1990-х годов специалисты в самых разных предметных областях работают не только с публикациями и базами данных (БД) своих орга-