УДК 681.3.02 DOI: 10.31114/2078-7707-2021-4-67-72

Формализация структуры программного обеспечения управляющих информационных систем

А.Д. Иванников

Институт проблем проектирования в микроэлектронике РАН, г. Москва

adi@ippm.ru

Аннотация — **На основе алгебраического подхода формализуется понятие оператора над памятью цифровых систем, вводится понятие обобщенной памяти, включающей кроме собственно памяти и регистров состояние активной метки оператора и также значение времени, формализуется функционал оператора над обобщенной памятью. Проводится анализ структуры области определения и области значений оператора. Вводится понятие произведения операторов, формулируется условие существования введенного произведения. Показывается, что программа цифровой системы является произведением операторов, множество которых является полугруппой. Разработанную формальную модель предполагается использовать для создания эффективной системы автоматизированной генерации тестовых примеров для программного обеспечения цифровых систем.**

Ключевые слова — **модель программного обеспечения, программа как произведение операторов, алгебраическая модель, операторы над обобщенной памятью.**

I. Введение

Еще в 90-ые года во многих работах указывалось на перспективность использования информационных технологий в различных областях современной человеческой деятельности: коммуникациях, образовании, науке и технике. В последующие тридцать лет информационные технологии постоянно развивались и находили применение во всех областях человеческой деятельности [1-5].

В любых методах информационных технологий важную роль играет программное обеспечение, в частности, при разработке и использовании управляющих информационных систем. Разработка и отладка управляющего программного и микропрограммного обеспечения требует постоянного совершенствования методов и средств разработки, как автономно, так и совместно с проектом аппаратной части управляющих систем [6-9].

Целью данной работы является разработка формальной модели программно-микропрограммного обеспечения и его фрагментов с учетом особенностей современных цифровых систем. Формальная модель позволит формализовать и автоматизировать составление отладочных тестов для каждой проверяемой функции программно-микропрограммного обеспечения. Основой предлагаемой модели является алгебраическое представление программного обеспечения, предложенное в [10].

При разработке модели необходимо учитывать особенности программно-микропрограммного обеспечения современных цифровых микроэлектронных систем [11-13], а именно:

- циклический характер работы программно-микропрограммного обеспечения, поскольку обычно цифровая система функционирует непрерывно, пока включено питание;

- выполнение определенных функций программно-микропрограммного обеспечения часто инициируется сигналами внешней среды;

- в современных цифровых системах широко используются различные типы прерываний.

II. Формализация понятия оператора

Рассмотрим множество элементов $\mathbf{B} = \{b_1, b_2, \ldots\}$, каждый из которых может находиться в одном из состояний конечного множества $\mathbf{Z}_b^*$. Назовем эти элементы элементами памяти. Элементами памяти в нашем случае являются как собственно ячейки ЗУ, так и программно доступные регистры блоков. Множество состояний памяти есть $\mathbf{P} = \prod_{b \in \mathbf{B}} \mathbf{Z}_b^*$. Элементы p этого множества определяют состояния всех элементов памяти. Отображение $\varepsilon : \mathbf{P}_1 \to \mathbf{P}_2$, где $\mathbf{P}_1 \in \mathbf{P}$, $\mathbf{P}_2 \in \mathbf{P}$, есть оператор над памятью, $\mathbf{P}_1$ – область определения оператора, $\mathbf{P}_2$ – область значений.

Рассмотрим конечное множество операторов E. При выполнении программ и микропрограмм существенными являются не только изменения состояния элементов памяти (ячеек ЗУ и регистров), но также и переходы от одних команд (фрагментов) к другим. Введем в рассмотрение элемент $Сч$ с конечным множеством значений $\mathbf{N}$. Пусть задано отображение $\iota : \mathbf{N} \to \mathbf{E}$. Здесь $\mathbf{N}$ есть множество меток операторов, в частном случае подмножество натуральных чисел. Если элементы множества $\mathbf{E}$ есть отдельные команды, то в качестве элементов множества $\mathbf{N}$ могут рассматриваться адреса команд.

В общем случае фрагменты программы или микропрограммы, которые будут рассматриваться как операторы, могут иметь несколько входов. Обозначим конечное множество входов оператора ε через $\mathbf{A}_1^{\varepsilon}$. Отображение $\imath$, ставящее в соответствие каждому входу всех операторов из $\mathbf{E}$ одно или несколько значений $Сч$, представим в виде:

$$\imath : \mathbf{N} \to \bigcup_{\varepsilon \in \mathbf{E}} \mathbf{A}_1^{\varepsilon}.$$

Назовем обобщенной памятью множество $\mathbf{B} \cup \{Сч\} \cup \mathbf{T}$, где $\mathbf{T}$ – множество моментов времени. Состоянием обобщенной памяти назовем $o = (p, n, t)$, где p – состояние памяти перед выполнением оператора; n – есть метка выполняемого оператора; t – время начала выполнения оператора. Различные элементы o образуют множество состояний обобщенной памяти $\mathbf{O}$, где $\mathbf{O} = \mathbf{P} \times \mathbf{N} \times \mathbf{T}$. Будем считать, что каждый оператор ε, кроме преобразования состояния памяти, осуществляет передачу управления на следующий оператор и требует для своего выполнения определенного времени. В соответствии с этим оператором ε будем называть отображение

$$\varepsilon : \mathbf{O}_1 \to \mathbf{O}_2;\ \mathbf{O}_1 \subset \mathbf{O};\ \mathbf{O}_2 \subset \mathbf{O},$$

где $\mathbf{O}_1$ – область определения оператора;

$\mathbf{O}_2$ – область значений оператора.

Сам оператор ε задается в виде команды, микрокоманды или фрагмента программы (макрокоманды).

III. Структура областей определения и значения операторов

Рассмотрим структуру множеств $\mathbf{O}_1$ и $\mathbf{O}_2$.

$$\mathbf{O}_1 = \bigcup_{a_1 \in \mathbf{A}_1} \mathbf{O}_1^{a_1};\ \mathbf{O}_1^{a_1} = \mathbf{P}_1^{a_1} \times \{a_1\} \times \mathbf{T},$$

где $\mathbf{A}_1$ – конечное множество состояний $Сч$, соответствующих входам в оператор ε (множество входных меток);

a_1 – элемент множества $\mathbf{A}_1$;

$\mathbf{O}_1^{a_1}$ – подобласть определения ε, соответствующая входу с меткой a_1;

$\mathbf{P}_1^{a_1}$ - подмножество допустимых состояний памяти перед выполнением оператора ε, соответствующее входу с меткой a_1;

$\mathbf{T}$ – множество моментов времени.

$$\mathbf{O}_2 = \bigcup_{a_2 \in \mathbf{A}_2} \mathbf{O}_2^{a_2};\ \mathbf{O}_2^{a_2} = \mathbf{P}_2^{a_2} \times \{a_2\} \times \mathbf{T};$$

$$\mathbf{A}_2 \cap \mathbf{A}_1 = \varnothing,$$

где $\mathbf{A}_2$ – конечное множество состояний $Сч$, соответствующих выходам оператора ε (множество выходных меток);

a_2 – элемент множества $\mathbf{A}_2$;

$\mathbf{O}_2^{a_2}$ – подобласть определения ε, соответствующая выходу с меткой a_2;

$\mathbf{P}_2^{a_2}$ - подмножество возможных состояний памяти после выполнения оператора ε, соответствующих выходу с меткой a_2;

$\mathbf{T}$ – множество моментов времени.

Оператор ε, заданный указанным образом, определяет новое состояние памяти: $p_2 = \varepsilon_p(p_1, a_1)$, $p_1 \in \mathbf{P}_1^{a_1}$, $a_1 \in \mathbf{A}_1$, где ε_p – отображение, задающее новое состояние памяти при выполнении оператора ε; новое состояние $Сч$: $a_2 = \varepsilon_a(p_1, a_1)$, $p_1 \in \mathbf{P}_1^{a_1}$, $a_1 \in \mathbf{A}_1$, где ε_a – отображение, задающее метку выхода при выполнении оператора ε; новое значение времени: $t_2 = t_1 + t(p_1, a_1)$, $t_1 \in \mathbf{T}$, $t_2 \in \mathbf{T}$, где t_1 – значение времени перед выполнением оператора; t_2 – значение времени после выполнения оператора; $t(p_1, a_1)$ – время выполнения оператора.

Каждое выполнение оператора ε характеризуется парой (a_1, a_2), $a_1 \in \mathbf{A}_1$, $a_2 \in \mathbf{A}_2$. Структура множеств $\mathbf{O}_1$ и $\mathbf{O}_2$ может быть задана следующим образом:

$$\mathbf{O}_1 = \prod_{\substack{a_1 \in \mathbf{A}_1 \\ a_2 \in \mathbf{A}_2}} \mathbf{O}^{a_1 a_2};\ \mathbf{O}^{a_1 a_2} = \mathbf{P}_1^{a_1 a_2} \times \{a_1\} \times \mathbf{T};$$

$$\mathbf{P}_1^{a_1 a_2} \cap \mathbf{P}_1^{a_1 a_2'} = \varnothing \text{ при } a_2 \neq a_2';$$

$$\mathbf{O}_2 = \bigcup_{\substack{a_1 \in \mathbf{A}_1 \\ a_2 \in \mathbf{A}_2}} \mathbf{O}_2^{a_1 a_2};\ \mathbf{O}_2^{a_1 a_2} = \mathbf{P}_2^{a_1 a_2} \times \{a_2\} \times \mathbf{T},$$

где $\mathbf{O}_1^{a_1 a_2}$, $\mathbf{O}_2^{a_1 a_2}$ - множества состояний обобщенной памяти перед и после выполнения оператора ε при входе в него по метке a_1 и выходе по метке a_2;

$\mathbf{P}_1^{a_1 a_2}$, $\mathbf{P}_2^{a_1 a_2}$ - множества состояний памяти перед и после выполнения оператора ε при входе в него по метке a_1 и выходе по метке a_2.

В случае, если прохождение оператора ε по пути $(a_1 a_2)$ невозможно, $\mathbf{P}_1^{a_1 a_2} = \varnothing$, $\mathbf{P}_2^{a_1 a_2} = \varnothing$.

Оператор ε определяет конечное множество подоператоров $\varepsilon^{a_1 a_2}$, $a_1 \in \mathbf{A}_1$, $a_2 \in \mathbf{A}_2$, соответствующих входу в оператор по метке a_1 и выходу по метке a_2:

$$\varepsilon^{a_1 a_2} : \mathbf{P}_1^{a_1 a_2} \times \{a_1\} \times \mathbf{T} \to \mathbf{P}_2^{a_1 a_2} \times \{a_2\} \times \mathbf{T}.$$

Каждый подоператор $\varepsilon^{a_1 a_2}$ определяет новое состояние памяти:

$$p_2 = \varepsilon_p^{a_1 a_2}(p_1),\ p_1 \in \mathbf{P}_1^{a_1 a_2},\ p_2 \in \mathbf{P}_2^{a_1 a_2},$$

где $\varepsilon_p^{a_1 a_2}$ - отображение, задающее новое состояние памяти, новое состояние $Сч$, равное a_2, и новое значение времени:

$$t_2 = t_1 + t^{a_1 a_2}(p_1),\ t_1 \in \mathbf{T},\ t_2 \in \mathbf{T},\ p_1 \in \mathbf{P}_1^{a_1 a_2},$$

где $t^{a_1 a_2}(p_1)$ – время выполнения оператора ε при входе в него по метке a_1 и выходе по метке a_2.

IV. Программно-микропрограммное обеспечение как полугруппа операторов

Определим произведение операторов $\varepsilon^{\mathbf{I}}$ и $\varepsilon^{\mathbf{II}}$, $\varepsilon^{\mathbf{I}} \in \mathbf{E}$, $\varepsilon^{\mathbf{II}} \in \mathbf{E}$ как оператор $\varepsilon^{\mathbf{III}} = \varepsilon^{\mathbf{I}} \varepsilon^{\mathbf{II}}$. Необходимым условием существования этого произведения является $\mathbf{A}_1^{\mathbf{I}} \cap \mathbf{A}_1^{\mathbf{II}} = \varnothing$, где $\mathbf{A}_1^{\mathbf{I}}$, $\mathbf{A}_1^{\mathbf{II}}$ - множества $\mathbf{A}_1$ операторов $\varepsilon^{\mathbf{I}}$ и $\varepsilon^{\mathbf{II}}$ соответственно.

Множества входных $\mathbf{A}_1^{\mathbf{III}}$ и выходных $\mathbf{A}_2^{\mathbf{III}}$ меток оператора $\varepsilon^{\mathbf{III}}$ есть $\mathbf{A}_1^{\mathbf{III}} = \mathbf{A}_1^{\mathbf{I}} \cup \mathbf{A}_1^{\mathbf{II}}$; $\mathbf{A}_2^{\mathbf{III}} = (\mathbf{A}_2^{\mathbf{I}} \cup \mathbf{A}_2^{\mathbf{II}}) / ((\mathbf{A}_1^{\mathbf{I}} \cap \mathbf{A}_2^{\mathbf{II}}) \cup (\mathbf{A}_1^{\mathbf{II}} \cap \mathbf{A}_2^{\mathbf{I}}))$.

На рис. 1 показаны возможные четыре случая связей между операторами $\varepsilon^{\mathbf{I}}$ и $\varepsilon^{\mathbf{II}}$.

Соответственно, при выполнении операторов $\varepsilon^{\mathbf{I}}$, $\varepsilon^{\mathbf{II}}$ существует определенная последовательность их работы, зависящая от начального состояния памяти p и входа $a_1^{\mathbf{III}}$. Любое выполнение оператора $\varepsilon^{\mathbf{III}}$ характеризуется следом s, который есть слово в алфавите

$$\left\{ (a_1^{\mathbf{I}}, a_2^{\mathbf{I}}) \middle| a_1^{\mathbf{I}} \in \mathbf{A}_1^{\mathbf{I}}, a_2^{\mathbf{I}} \in \mathbf{A}_2^{\mathbf{I}} \right\} \cup \left\{ (a_1^{\mathbf{II}}, a_2^{\mathbf{II}}) \middle| a_1^{\mathbf{II}} \in \mathbf{A}_1^{\mathbf{II}}, a_2^{\mathbf{II}} \in \mathbf{A}_2^{\mathbf{II}} \right\}.$$

То есть

$$s = (a_{1j_1}^{i_1}, a_{2k_1}^{i_1})(a_{1j_2}^{i_2}, a_{2k_2}^{i_2}) .. (a_{1j_m}^{i_m}, a_{2k_m}^{i_m}), \quad (1)$$

где $i_1, i_2, ..., i_m$ принимают значения из множества $\{\mathbf{I}, \mathbf{II}\}$;

$i_1 = i_3 = i_5 = ...$; $i_2 = i_4 = i_6 = ...$; $i_1 \neq i_2$;

$a_{1j_1}^{i_1}, a_{1j_3}^{i_3}, ...$ принадлежат множеству $\mathbf{A}_1^{i_1}$;

$a_{2k_2}^{i_1}, a_{2k_2}^{i_3}, ...$ принадлежат множеству $\mathbf{A}_2^{i_1}$;

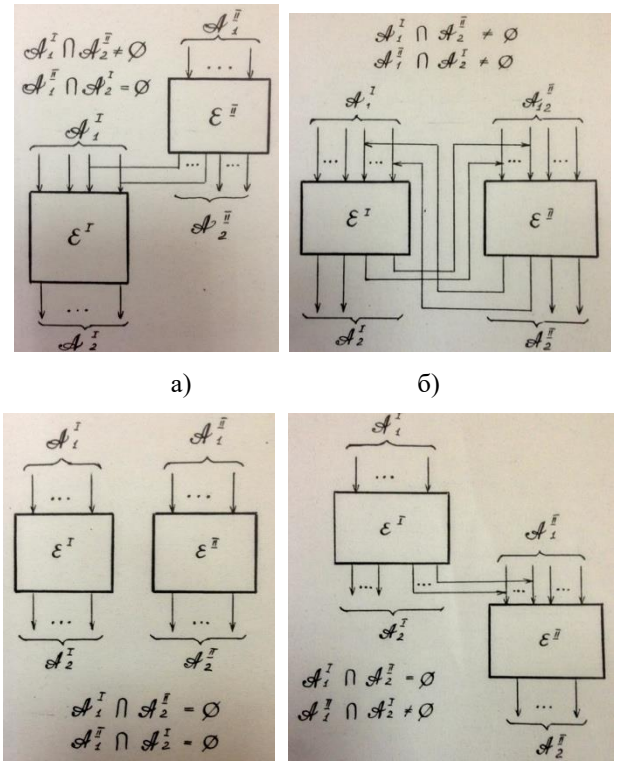
$a_{1j_2}^{i_2}, a_{1j_4}^{i_4}, ...$ принадлежат множеству $\mathbf{A}_1^{i_2}$;

$a_{2k_2}^{i_2}, a_{2k_4}^{i_4}, ...$ принадлежат множеству $\mathbf{A}_2^{i_2}$;

$a_{1j_1}^{i_1} = a_1^{\mathbf{III}}$; $a_{2k_1}^{i_1} = a_{1j_2}^{i_2}$;

$a_{2k_2}^{i_2} = a_{1j_3}^{i_3}$; …; $a_{2k_m}^{i_m} = a_2^{\mathbf{III}}$;

m – количество букв в следе s, то есть количество прохождений через операторы $\varepsilon^{\mathbf{I}}$, $\varepsilon^{\mathbf{II}}$.



а) б) в) г)

Рис. 1. Различные случаи взаимодействия операторов $\varepsilon^{\mathbf{I}}$ и $\varepsilon^{\mathbf{II}}$

Каждой паре $(a_1^{\mathbf{III}}, a_2^{\mathbf{III}})$ оператора $\varepsilon^{\mathbf{III}}$ соответствует множество следов вида (1). Необходимым условием того, что $p \in \mathbf{P}_1^{\mathbf{III} a_1 a_2}$, где $\mathbf{P}_1^{\mathbf{III} a_1 a_2}$ - множество состояний памяти, на котором определен оператор $\varepsilon^{\mathbf{III} a_1 a_2}$, то есть оператор $\varepsilon^{\mathbf{III}}$ при входе в него по метке a_1 и выходе по метке a_2 является конечность следа s

. Для каждой пары (a_1, a_2), $a_1 \in \mathbf{A}_1^{\mathbf{III}}$, $a_2 \in \mathbf{A}_2^{\mathbf{III}}$ рассмотрим множество $\mathbf{S}^{a_1 a_2}$ конечных следов. Каждый след $s \in \mathbf{S}^{a_1 a_2}$ определяет выполнение последовательности операторов и задает преобразование состояния памяти, элемента $Сч$ и времени.

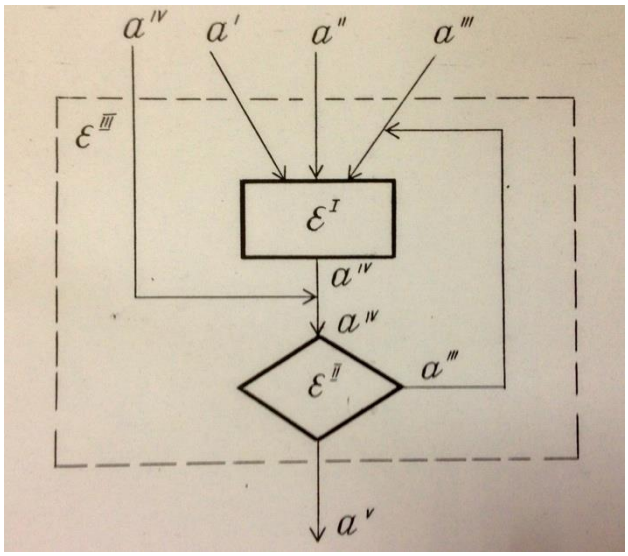


Рис. 2. Пары операторов, произведения которых существуют

На рис. 2 показаны пары операторов $\varepsilon^{\mathbf{I}}$ и $\varepsilon^{\mathbf{II}}$, произведения которых существуют. Для рисунка 3, а $\mathbf{A}_1^{\mathbf{I}} = \{a^{\mathbf{I}}, a^{\mathbf{II}}, a^{\mathbf{III}}\}$, $\mathbf{A}_2^{\mathbf{I}} = \{a^{\mathbf{IV}}\}$, $\mathbf{A}_1^{\mathbf{II}} = \{a^{\mathbf{IV}}\}$, $\mathbf{A}_2^{\mathbf{II}} = \{a^{\mathbf{III}}, a^{\mathbf{V}}\}$. Входными и выходными множествами меток оператора $\varepsilon^{\mathbf{III}}$ являются $\mathbf{A}_1^{\mathbf{III}} = \{a^{\mathbf{I}}, a^{\mathbf{II}}, a^{\mathbf{III}}, a^{\mathbf{IV}}\}$, $\mathbf{A}_2^{\mathbf{III}} = \{a^{\mathbf{V}}\}$.

Для рис. 2б $\mathbf{A}_1^{\mathbf{I}} = \{a^{\mathbf{I}}\}$, $\mathbf{A}_2^{\mathbf{I}} = \{a^{\mathbf{II}}\}$, $\mathbf{A}_1^{\mathbf{II}} = \{a^{\mathbf{III}}\}$, $\mathbf{A}_2^{\mathbf{II}} = \{a^{\mathbf{IV}}, a^{\mathbf{V}}\}$, $\mathbf{A}_1^{\mathbf{III}} = \{a^{\mathbf{I}}, a^{\mathbf{III}}\}$, $\mathbf{A}_2^{\mathbf{III}} = \{a^{\mathbf{II}}, a^{\mathbf{IV}}, a^{\mathbf{V}}\}$.

Таким образом, на множестве операторов $\mathbf{E}$ определена полугруппа, причем произведение любой последовательности операторов, если $\mathbf{A}_2^i \cap \mathbf{A}_1^j \neq \varnothing$ при $i \neq j$, для всех операторов существует и является оператором. Отметим, что при использовании введенной операции умножения результат произведения набора операторов не зависит от последовательности умножения.

В качестве одного из операторов можно рассматривать команду останова. Эта команда в программах реального времени используется для останова в режиме ожидания запроса на прерывание. В этом случае команду останова можно рассматривать как оператор, выполняющийся за один такт, при отсутствии запроса на прерывание передающий управление на свою входную метку и, как и все остальные команды при немаскированном прерывании, в качестве одного из аргументов проверяющий состояние регистра запроса на прерывание. Отметим, что для вычислительных программ, в которых прерывания могут не использоваться, команда останова может рассматриваться как оператор с $\mathbf{A}_2 = \varnothing$.

Программой является оператор, у которого $\mathbf{A}_2 = \varnothing$. Программа может содержать или не содержать оператор останова.

После того, как программа или микропрограмма сформирована в виде произведения некоторого множества операторов, выделим в множестве входных меток некоторое множество внешних входов программы или микропрограммы. Хотя в общем случае возможно наличие нескольких входов в программу или микропрограмму, в программе или микропрограмме цифровой системы в целом, как правило, имеется одна точка, с которой начинается ее выполнение при включении питания.

Учет действия прерываний осуществляется выделением в множестве $\mathbf{B}$ регистров запроса на прерывание и регистров векторов прерывания.

V. Заключение

Если отладка программного или микропрограммного обеспечения осуществляется с использованием моделирования технических средств [14-16], она может быть выполнена на машинных моделях как функционально-логического уровня, так и уровня архитектуры или микроархитектуры.

Составление набора тестов в этом случае должно осуществляться, прежде всего, исходя из выполняемых функций, а затем - исходя из структуры программы или микропрограммы [17, 18]. В последнем случае могут быть использованы различные методы: метод покрытия путей, линейных участков или переходов, тестирование по особым значениям, иерархическое тестирование и др. [19, 20]. Использование предложенной модели позволит учесть особенности программно-микропрограммного обеспечения современных цифровых систем и разработать систему его автоматизированного тестирования с высокой эффективностью. При этом возможна реализация автоматизированной генерации тестовых входных взаимодействий как на основе

структуры программно-микропрограммного обеспечения, так и на основе анализа заданных в спецификации алфавита выполняемых функций и их возможных последовательностей.

ЛИТЕРАТУРА

[1] Тихонов А.Н., Иванников А.Д. Информатизация российского образования и общества в целом // Международное сотрудничество. 1997. № 4. С. 1.

[2] Климов А.В., Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Методы адаптации параллельной потоковой вычислительной системы под задачи отдельных классов // Информационные технологии и вычислительные системы. 2009. №3. С. 12-21.

[3] Иванников А.Д., Тихонов А.Н., Цветков В.Я. Критерии готовности к использованию информационных технологий // Международный журнал прикладных и фундаментальных исследований. 2009. № 3. С. 84-85.

[4] Тихонов А.Н., Иванников А.Д., Цветков В.Я. Образовательные услуги как инструмент качества образования // Международный журнал прикладных и фундаментальных исследований. 2009. № 3. С. 94-96.

[5] Иванников А.Д. Тематические интернет-порталы как средство агрегации электронного контента в заданной предметной области // Информационные технологии. 2014. №3. С. 43-48.

[6] Иванников А.Д., Стемпковский А.Л. Математическая модель отладки проектов сложных цифровых схем и микросистем на основе представления последних в виде семейства стационарных динамических систем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2014. № 2. С. 123-128.

[7] Romanov A.Yu., Ivannikov A.D., Romanova I.I. Simulation and synthesis of network-on-chip by using NOCSIPM HDL library. 2016 IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO 2016 – Conference Proceedings. 36. 2016. Pp. 300-303.

[8] Гаврилов С.В., Глебов А.Л., Стемпковский А.Л. Анализ помехоустойчивости цифровых схем на основе логических импликаций // Известия высших учебных заведений. Электроника. 2002. № 5. С. 60.

[9] Гаврилов С.В., Иванова Г.А., Рыжова Д.И., Соловьев А.Н., Стемпковский А.Л. Методы синтеза помехозащищенных комбинационных блоков // Информационные технологии. 2015. Т. 21. № 11. С. 821-826.

[10] Ляпунов А.А. К алгебраической трактовке программирования // В кн.: Проблемы кибернетики. – М.: Наука, 1962. Вып. 2. С. 235-241.

[11] Gavrilov S.V., Ivannikov A.D., Stempkovsky A.L. Method of mathematical description for digital system blocks logical models // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2019. № 2. С. 8-11.

[12] Ivannikov A., Kulagin V., Romanov A., Pozdneev B. Algebraic models of digital system design debugging decomposition. Proceedings of 2016 IEEE East-West Design and Test Symposium, EWDTS 2016. 2016. P. 7807712.

[13] Кулагин В.П., Логинов А.А. Анализ программных средств для работы с сетями Петри // Информационные технологии. 2021. Т. 27. № 2. С. 89-96.

[14] Иванников А.Д. Теоретические основы выбора множества отладочных тестов цифровых систем на основе алфавита выполняемых функций // Информационные технологии. 2019. Т.25. № 11. С. 657-662.

[15] Bryce R.C., Sampath S., Memon A.M. Developing a Single Model and Test Prioritization Strategies for Event-Driven Software // IEEE Transactions on Software Engineering. 2011. Vol. 37. № 1. Pp. 48-64.

[16] Иванников А.Д., Стемпковский А.Л. Анализ итерационных методов решения систем логических уравнений и их использование при моделировании цифровых систем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2020. № 3. С. 2-8.

[17] Иванников А.Д. Формирование отладочного набора тестов для проверки функций цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2017. Т. 18. No. 12. С. 795-801.

[18] Lee J., Kang S., Lee D. Survey on Software Testing Practices // Software, IET. 2012. Vol. 6. № 3. Pp.275-282.

[19] Кулямин В.В., Петухов А.А. Обзор методов построения покрывающих наборов // Программирование. 2011. Т. 37. № 3. С. 3-41.

[20] Шилов Н.В. Обзор теории алгебраических моделей программ // В кн.: Мальцевские чтения 2016. Тезисы докладов. 2016. С. 42.

Structure Formalization of Management Information System Software

A.D. Ivannikov

Institute for design problems in microelectronics of RAS, Moscow
adi@ippm.ru

***Abstract* — On the algebraic approach basis, the concept of an operator over the memory of digital systems is formalized, the concept of generalized memory is introduced, which includes, in addition to the memory itself and registers, the state of the active label of the operator and also the time value; the functional of the operator over the generalized memory is formalized. The analysis of the structure of the domain of definition and the range of values of the operator is carried out. The concept of a product of operators is introduced; the condition for the existence of the introduced product is formulated. It is shown that the program of a digital system is a product of operators, the set of which is a semi-group. The developed formal model is supposed to be used to create an effective system for the automated generation of test examples for software of digital systems.**

***Keywords* —software model, program as a product of operators, algebraic model, generalized memory operators.**

References

[1] Tikhonov A.N., Ivannikov A.D. Informatization of Russian education and society as a whole // International cooperation. 1997. No. 4. P. 1.

[2] Klimov A.V., Levchenko N.N., Okunev A.S., Stemokovskiy A.L. Parallel dataflow computing system adaptation for specific task classes // Journal of Information Technologies and Computing Systems. 2009. No. 3. Pp. 12-21.

[3] Ivannikov A.D., Tikhonov A.N., Cvetkov V.Ya. Readiness criteria for information technologies usage //International Journal of Applied and Fundamental Research. 2009. No. 3. Pp. 84-85.

[4] Tikhonov A.N., Ivannikov A.D., Cvetkov V.Ja. Obrazovatel'nye uslugi kak instrument kachestva obrazovanija // Mezhdunarodnyj zhurnal prikladnyh i fundamental'nyh issledovanij. 2009. No 3. Pp. 94-96.

[5] Ivannikov A.D. Subject Internet portals as the means of aggregating electronic content in a given subject area // Information technologies. 2014. No. 3. Pp. 43-48.

[6] Ivannikov A.D., Stempkovsky A.L. Complex Digital Systems and Microsystems Design Debugging Mathematic Model on the Basis of Stationary Dynamic System Family Presentation // Problems of Perspective Micro- and Nanoelectronic Systems Development. 2014. Issue 2. Pp. 123-128.

[7] Romanov A.Yu., Ivannikov A.D., Romanova I.I. Simulation and synthesis of network-on-chip by using NOCSIPM HDL library. 2016 IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO 2016 – Conference Proceedings. 36. 2016. Pp. 300-303.

[8] Gavrilov S.V., Glebov A.L., Stempkovskiy A.L. Digital circuits noise immunity analysis on logical implication base // Izvestiya Vysshikh Uchebnykh Zavedenii. Elektronika. 2002. No. 5. P. 60.

[9] Gavrilov S.V., Ivanova G.A., Ryzhova D.I., Soloviev A.N., Stempkovskiy A.L. Metody sinteza pomekhozashchishchennykh kombinatsionnykh blokov // Informatsionnye tekhnologii, 2015. V. 21. No.11. Pp. 821-826.

[10] Lyapunov A.A. Algebraic approach to programming // In the book: Cybernetic Problems. – Moscow: Nauka. 1962. Issue 2. Pp. 235-241.

[11] Gavrilov S.V., Ivannikov A.D., Stempkovsky A.L. Method of mathematical description for digital system blocks logical models // Problems of Perspective Micro- and Nanoelectronic Systems Development. 2019. Issue 2. P. 8-11.

[12] Ivannikov A., Kulagin V., Romanov A., Pozdneev B. Algebraic models of digital system design debugging decomposition. Proceedings of 2016 IEEE East-West Design and Test Symposium, EWDTS 2016. 2016. P. 7807712.

[13] Kulagin V.P., Loginov A.A. Analisys of software means for Petri network operations //Information Technologies. 2021. V. 27. No. 2. Pp. 89-96.

[14] Ivannikov A.D. Theoretical Basis for the Selection of Design Debugging Tests Set for Digital Systems Based on the Alphabet of Functions Performed // Information Technologies. 2019. Vol. 25. No. 11. Pp. 657-662.

[15] Bryce R.C., Sampath S., Memon A.M. Developing a Single Model and Test Prioritization Strategies for Event-Driven Software // IEEE Transactions on Software Engineering. 2011. Vol. 37. № 1. Pp. 48-64.

[16] Ivannikov A.D., Stempkovsky A.L. Analysis of Iterative Methods for Solving Logical Equation Systems and their Use in Digital System Simulation // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2020. Issue 3. P. 2-8.

[17] Ivannikov A.D. Debugging Input Set Generation for Testing of Control Digital Systems Functions // Mekhatronika, Avtomatizatciya, Upravlenie. 2017. Vol. 18. No.12. Pp. 795-801.

[18] Lee J., Kang S., Lee D. Survey on Software Testing Practices // Software, IET. 2012. Vol. 6. № 3. Pp.275-282.

[19] Kulyamin V.V., Petuhov A.A. Overview of methods for constructing covering sets // Prpgramming. 2011. V. 37. No. 3. Pp. 3-41.

[20] Shilov N.V. Review of the theory of algebraic program models // In the book: Maltcevskie Chteniya 2016. Abstracts. 2016. P. 42.