ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ

Научный журнал Отделения нанотехнологий и информационных технологий Российской академии наук

Издается с 2007 года
Журнал выходит ежеквартально

Учредители:
Российская академия наук
Институт проблем информатики Российской академии наук

РЕДАКЦИОННАЯ КОЛЛЕГИЯ

Адрес редакции:

Москва 119333, ул. Вавилова 44, корп. 2, ИПИ РАН,
редакция журнала «Информатика и её применения»
Тел. 8(499)135-86-92, e-mail rust@ipiran.ru

Журнал «Информатика и её применения» включен в «Перечень ведущих рецензируемых научных журналов и изданий, в которых должны быть опубликованы основные научные результаты диссертации на соискание ученой степени доктора и кандидата наук», утвержденный ВАК

Журнал зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций 31 марта 2009 г.
Свидетельство о регистрации средства массовой информации ПИ № ФС77-35837

Подписной индекс журнала в каталоге «Пресса России» 88018 (годовая подписка)

Информатика и её применения

Том 3 Выпуск 2 Год 2009

СОДЕРЖАНИЕ

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 2–14

ВЕРОЯТНОСТНЫЙ АНАЛИЗ ВРЕМЕНИ ПРОЯВЛЕНИЯ НЕИСПРАВНОСТИ В СЕТИ АВТОМАТОВ

А. В. Печинкин[1], С. Л. Френкель[2]

Аннотация: Предложен способ распространения математической модели оценки распределения вероятностей времени обнаружения неисправности в некоторой цифровой системе, описанной как конечный автомат, на сеть подавтоматов, в которую декомпозирован данный исходный автомат. Под «латентностью» понимается время (число тактов работы автомата), которое проходит с момента проявления неисправности в той или иной переменной, описывающей модель автомата, до проявления ее в выходной последовательности автомата. Латентность вычисляется как число шагов до попадания цепи Маркова (ЦМ) с множеством состояний, представляющим собой произведение множеств состояний исправного и неисправного автоматов, в поглощающее состояние (на случайных входных наборах автомата). Предлагаемый подход к оценке распределения вероятностей латентности основан на расширении пространства состояний описываемой системы, что позволяет строить ЦМ для пар подавтоматов (в общем зависимых) исправного и неисправного автоматов и выразить это распределение в терминах вероятностных характеристик подавтоматов.

Ключевые слова: тестирование цифровых схем; конечные автоматы; цепи Маркова

1 Введение

При проектировании современных высоконадежных цифровых систем часто применяются различные «толерантные к ошибкам» (fault-tolerant) решения, например самопроверяемость (self-checking), самовосстановление (self-recovery), самокоррекция (self-reconfiguration) [1]. Для таких систем исключительно важным является вопрос о времени обнаружения ошибки функционирования. Под «обнаружением» неисправности понимают несовпадение значений выходов исправной системы и системы с неисправностью, что означает, например, некорректное функционирование по сравнению с заданной спецификацией проектируемой системы. В общем случае под «выходом» понимают некоторые переменные, с которыми связывают желаемый результат. Время от момента появления неисправности, приводящей к тому, что при определенном возможном значении входных переменных происходит несовпадение значений отдельных переменных (например, переменных, описывающих внутренние состояния) исправной системы и системы с неисправностью, до момента ее проявления в хотя бы одной из выходных переменных (момента обнаружения неисправности) называют «*латентностью обнаружения* соответствующей *неисправности*», или FDL (Fault Detection Latency, см. рис. 1, на котором FDL $= k$). Как далее будет показано на примерах, несовпадение внутренних состояний исправной системы и системы с неисправностью еще не обязательно ведет к несовпадению выходных переменных.

Рис. 1 Явление латентности обнаружения неисправности, появляющейся при выполнении некоторой операции в момент τ и проявляющейся на выходе в момент $\tau + k$

Будем рассматривать цифровые системы, представленные моделью конечного автомата (FSM — Finite State Machine), в частности автоматом Мили (Mealy) с множеством состояний $\mathcal{A} = \{a_1, \ldots, a_r\}$, входов $\mathcal{X} = \{x_1, \ldots, x_n\}$ и выходов $\mathcal{Y} = \{y_1, \ldots, y_m\}$, где x_i, $i = \overline{1,n}$, и y_j, $j = \overline{1,m}$, — булевы переменные. В этой модели внутреннее состояние $a(t+1) \in \mathcal{A}$ в момент $t+1$ и вектор $\vec{y}(t) = (y_1(t), \ldots, y_m(t))$ выходных переменных в момент t определяются через внутреннее состояние $a(t) \in \mathcal{A}$ и вектор $\vec{x}(t) = (x_1(t), \ldots, x_n(t))$ входных переменных в момент t функциями δ (переходов состояний) и λ (выходов) по формулам

[1]Институт проблем информатики Российской академии наук, apechinkin@ipiran.ru
[2]Институт проблем информатики Российской академии наук, fsergei@mail.ru

$$a(t+1) = \delta(a(t), \vec{x}(t))\,; \qquad \vec{y}(t) = \lambda(a(t), \vec{x}(t))\,. \tag{1}$$

Подробно некоторые модели латентности обнаружения неисправностей (связанные, естественно, с моделями самих неисправностей) будут рассмотрены далее.

Необходимость оценки возможной латентности обнаружения неисправностей на ранних этапах проектирования цифровых устройств (например, на автоматном уровне [2]) диктуется следующими причинами:

- высокой сложностью современных систем, делающей практически невозможным полное исключения имеющихся дефектов при производственном тестировании;
- влиянием внешних помех (космических лучей, излучения бытовой электроники и т. п.) на современные сложные системы, особенно созданные с использованием наноэлектроники [3].

Поэтому современные системы должны включать в себя те или иные средства самотестирования и самокоррекции [1] возможных ошибок, влияние которых на производительность и надежность проектируемых систем зависит в том числе и от времени между проявлением, обнаружением и коррекцией неисправности. При этом в большинстве случаев значительная (относительно тактовых частот работы системы) латентность может рассматриваться как фактор, снижающий надежность [4, 5].

Поскольку латентность зависит от структуры автомата, описывающего проектируемую систему, ее оценку желательно проводить на достаточно ранних этапах проектирования, чтобы эффективно управлять выбором проектных решений на как можно более ранних стадиях.

Будем считать, что адекватной мерой латентности является ее функция распределения вероятностей (ФР) $F_{\mathrm{FDL}}(x)$, т. е. ФР $F_{\mathrm{FDL}}(x) = \mathbf{Pr}(k \le x)$ числа k тактов (см. рис. 1) от появления неисправности до несовпадения хотя бы одной выходной переменной исправной и неисправной систем на входной последовательности двоичных независимых векторов (обоснованность данной меры в настоящей статье не рассматривается, отметим только, что эта мера широко используется в литературе, см., например, [6, 7]). При этом проявление неисправности моделируется как попадание в поглощающее состояние ЦМ, соответствующей паре «исправный – неисправный автоматы» при случайных независимых входах и использовании определенного способа сравнения поведения этих автоматов.

Для одиночного конечного автомата Мили, представленного своей таблицей переходов, данную задачу можно считать принципиально решенной [6], хотя, разумеется, остаются проблемы, связанные с затратами ресурсов по мере роста размерности автомата. Однако в случае, когда конечный автомат представлен как сеть подавтоматов, применение метода [6] для вычисления указанной выше ФР латентности в терминах тех или иных характеристик подавтоматов наталкивается на серьезные трудности, связанные со сложностью описания таких систем в терминах ЦМ.

В предлагаемой статье рассматривается возможный подход к вычислению ФР латентности автомата, декомпозированного на три взаимодействующих подавтомата. Эта ФР вычисляется в терминах вероятностей переходов подавтоматов с учетом специфики конкретного алгоритма взаимодействия.

2 Основной подход к оценке латентности обнаружения неисправности

2.1 Модель неисправности

Сделаем прежде всего несколько необходимых замечаний о понятии «автомат с неисправностью» (модели неисправности). Под неисправностью f будем понимать некоторую трансформацию (относительно проектной спецификации) структуры или функции устройства, вызванную производственными, проектными или индуцированными в процессе эксплуатации факторами.

Наиболее простая классификация неисправностей состоит в разбиении множества возможных неисправностей на перманентные (permanent, см. [1]) неисправности, действующие (в смысле указанных трансформаций) в течение всего времени наблюдения за работой устройства, и переходные (transient, см. [1, 3]) неисправности, представляющие собой лишь кратковременное (по сравнению с интервалом наблюдения) отличие от спецификации.

Наиболее распространенной моделью перманентной неисправности является «одиночная константная неисправность» [8, 9]. Пусть проектируемое устройство должно реализовать булеву функцию $f(x_1, \ldots, x_n)$, где $x_1, \ldots, x_n$ — некоторые булевы переменные. Неисправность устройства может быть задана, например, как $f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \equiv f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$, или, короче, $x_i \equiv 1$, что означает, что неисправное устройство работает так, как если бы переменная x_i прини-

мала значение логической «1» вне зависимости от комбинаций входных наборов (в англоязычной литературе такая неисправность называется "stuck-at-1"). Аналогично можно рассмотреть неисправность $x_i \equiv 0$ ("stuck-at-0").

Что касается автоматных моделей устройств, то данные неисправности можно связать с булевыми аргументами функций δ и λ в (1).

Заметим, что проявление неисправности f в конечном автомате можно задавать еще и в виде неправильных переходов

$$\left\{a_i \xrightarrow{\vec{x}} a_j\right\} \to \left\{a_i \xrightarrow{\vec{x},f} a_k\right\}, \quad k \neq j,$$

на входном векторе $\vec{x}$ из состояния a_i в состояние a_k вместо a_j. Таким образом можно моделировать проявление как константных, так и других неисправностей.

Очевидно, что неисправность может приводить к неправильному функционированию автомата, проявляющемуся непосредственно в момент появления неисправности или через какое-то время после ее появления, а также вообще не проявляться ни при каких наборах входных сигналов. Под «проявлением» неисправности здесь понимается изменение состояния и/или значений одной или нескольких выходных переменных в неисправном устройстве относительно состояния и/или значений тех же переменных в исправном устройстве при одинаковых значениях входных переменных. В частности, напомним, что модель константной неисправности определяется неисправностью $x_i \equiv 0$ ("stuck-at-0") или неисправностью $x_i \equiv 1$ ("stuck-at-1").

Пример 1. В качестве примера рассмотрим неисправность $x_2 \equiv 1$ в простейшем автомате, представленном табл. 1.

В этой таблице a_t и a_s — предыдущее и последующее состояния автомата, $X(a_t, a_s)$ — входной вектор, при котором происходит переход из состояния a_t в состояние a_s, и $Y(a_t, a_s)$ — выходной вектор. Напомним, что, как обычно при использовании кубической формы представления автоматов (см., например, [2, 8]), обозначения x_i и $\overline{x}_i$ в столбце входов "$X(a_t, a_s)$" означают равенство единице и нулю i-й координаты вектора $\vec{x}$, отсутствующие в этом столбце переменные могут принимать любые значения (0 или 1), а в столбце выходов "$Y(a_t, a_s)$" указываются только биты, принимающие единичное значение на соответствующих переходах (остальные биты нулевые), причем полагающиеся при векторной записи скобки опускаются.

Из табл. 1 видно, что входной вектор $\overline{x}_1$ оставляет как исправный, так и неисправный автоматы в состоянии a_1 и при этом выходной вектор равен y_1, y_4, т. е. неисправность не проявляется. Аналогично при входном векторе x_1, x_2 оба автомата из состояния a_1 переходят в состояние a_2 при выходном векторе y_2, y_4. Однако при входном векторе $x_1, \overline{x}_2$ исправный автомат остается в состоянии a_1 при выходном векторе y_2, y_3, в то время как неисправный автомат переходит из состояния a_1 в состояние a_2 при выходном векторе y_2, y_4 (неправильный переход из состояния a_1 при входном наборе $(x_1 = 1, x_2 = 0)$ произойдет по условию $(x_1 = 1, x_2 = 1)$). Это означает, что при данных предыдущем состоянии и входном векторе неисправность проявится как в переходе состояний, так и в выходном векторе.

Далее, при входном векторе $x_2, \overline{x}_3$ из состояния a_2 оба автомата переходят в состояние a_1 при выходном векторе y_1, y_4. Аналогично при входном векторе x_2, x_3 оба автомата из состояния a_2 переходят в состояние a_2 при выходном векторе y_2, y_4. Однако при входном векторе $\overline{x}_2, \overline{x}_3$ из состояния a_2 исправный автомат переходит в состояние a_1 при выходном векторе y_2, y_4, а неисправный автомат при этом же входном векторе также переходит из состояния a_2 в состояние a_1, но при выходном

Таблица 1 Поведение исправного автомата и автомата с неисправностью $x_2 \equiv 1$

a_t	Исправный			С неисправностью		
	a_s	$X(a_t, a_s)$	$Y(a_t, a_s)$	a_s	$X(a_t, a_s)$	$Y(a_t, a_s)$
a_1	a_1	$\overline{x}_1$	y_1, y_4	a_1	$\overline{x}_1$	y_1, y_4
	a_1	$x_1, \overline{x}_2$	y_2, y_3	a_2	$x_1, \overline{x}_2$	y_2, y_4
	a_2	x_1, x_2	y_2, y_4	a_2	x_1, x_2	y_2, y_4
a_2	a_1	$\overline{x}_2$	y_2, y_4	a_1	$\overline{x}_2, \overline{x}_3$	y_1, y_4
				a_2	$\overline{x}_2, x_3$	y_2, y_4
	a_1	$x_2, \overline{x}_3$	y_1, y_4	a_1	$x_2, \overline{x}_3$	y_1, y_4
	a_2	x_2, x_3	y_2, y_4	a_2	x_2, x_3	y_2, y_4

векторе y_1, y_4. Таким образом, неисправность проявляется только в выходном векторе.

Наконец, при входном векторе $\overline{x}_2, x_3$ из состояния a_2 исправный автомат переходит в состояние a_1 при выходном векторе y_2, y_4, а неисправный автомат переходит из состояния a_2 в состояние a_2, правда, при том же выходном векторе y_2, y_4. Теперь неисправность проявится только в переходе состояний, но не в выходном векторе.

В заключение приведем значения функций δ и λ для исправного автомата:

$$\delta(a_1,\vec{x}) = \begin{cases} a_1\,, & \overline{x}_1 \vee (x_1 \wedge \overline{x}_2) = 1\,, \\ a_2\,, & \text{иначе;} \end{cases}$$

$$\delta(a_2,\vec{x}) = \begin{cases} a_1, & \overline{x}_2 \vee (x_2 \wedge \overline{x}_3) = 1\,, \\ a_2, & \text{иначе;} \end{cases}$$

$$\begin{aligned} &\lambda_1(a_1,\vec{x}) = \overline{x}_1\,; && \lambda_2(a_1,\vec{x}) = x_1\,; \\ &\lambda_3(a_1,\vec{x}) = x_1 \wedge \overline{x}_2\,; && \lambda_4(a_1,\vec{x}) = \overline{x}_1 \vee (x_1 \wedge x_2)\,; \\ &\lambda_1(a_2,\vec{x}) = x_2 \wedge \overline{x}_3\,; && \lambda_2(a_2,\vec{x}) = \overline{x}_2 \vee (x_2 \wedge x_3)\,; \\ &\lambda_3(a_2,\vec{x}) \equiv 0\,; && \lambda_4(a_2,\vec{x}) \equiv 1\,, \end{aligned}$$

а также функций δ^f и λ^f для неисправного автомата:

$$\delta^f(a_1,\vec{x}) = \begin{cases} a_1, & \overline{x}_1 = 1\,, \\ a_2, & \text{иначе}\,; \end{cases}$$

$$\delta^f(a_2,\vec{x}) = \begin{cases} a_1, & \overline{x}_3 = 1\,, \\ a_2, & \text{иначе}\,; \end{cases}$$

$$\begin{aligned} &\lambda^f_1(a_1,\vec{x}) = \overline{x}_1\,; && \lambda^f_2(a_1,\vec{x}) = x_1\,; \\ &\lambda^f_3(a_1,\vec{x}) \equiv 0\,; && \lambda^f_4(a_1,\vec{x}) \equiv 1\,; \\ &\lambda^f_1(a_2,\vec{x}) = \overline{x}_3\,; && \lambda^f_2(a_2,\vec{x}) = x_3\,; \\ &\lambda^f_3(a_2,\vec{x}) \equiv 0\,; && \lambda^f_4(a_2,\vec{x}) \equiv 1\,. \end{aligned}$$

2.2 Латентность обнаружения неисправности

Проиллюстрируем простыми примерами явление латентности обнаружения неисправности в автомате для рассмотренной выше модели неисправности.

Поскольку, как отмечалась во введении, представляет интерес некоторая вероятностная мера латентности, то необходимо определиться с источником случайности в поведении автоматов. Будем считать, что им является случайность входных сигналов, характеризуемая вероятностью единичного значения для каждого входа (бита входного вектора).

Пример 2. Явление латентности обнаружения неисправности в автомате покажем на автомате из примера 1.

Обозначим через $p_i = \mathbf{Pr}(x_i = 1)$ вероятность единичного значения для i-го входа (i-го бита входного вектора) независимо от значений остальных входов. Соответственно, вероятность $q_i = \mathbf{Pr}(x_i = 0)$ нулевого значения этого входа равна $q_i = 1 - p_i$.

Пусть в некоторый момент (который будем отождествлять с моментом 0) в реальном автомате, находившемся в состоянии a_1, возникла неисправность $x_2 \equiv 1$. Вычислим значения латентности (FDL) на некоторых последовательностях поступающих сигналов.

(1) Первая последовательность — $\overline{x}_1 \to x_1, x_2 \to x_2, \overline{x}_3 \to x_1, \overline{x}_2$ (вероятность возникновения такой последовательности $q_1 \cdot p_1p_2 \cdot p_2q_3 \times \times p_1q_2$). Тогда на этой последовательности поступающих сигналов последовательность состояний и выходных сигналов (в скобках) в исправном автомате будет $a_1 \to a_1(y_1, y_4) \to a_2(y_2, y_4) \to a_1(y_1, y_4) \to a_1(y_1, y_4)$, а в неисправном автомате — $a_1 \to a_1(y_1, y_4) \to a_2(y_2, y_4) \to a_1(y_1, y_4) \to a_2(y_2, y_4)$, т. е. на четвертом шаге впервые на исправном и неисправном автоматах появились разные значения внутренних переменных, причем одновременно различными стали и состояния, и выходные сигналы. Видно, что на этой последовательности входных наборов $\mathrm{FDL} = 4$.

(2) Следующая последовательность — $\overline{x}_1 \to x_1, x_2 \to x_2, x_3 \to \overline{x}_2, \overline{x}_3$ (вероятность $q_1 \times \times p_1p_2 \cdot p_2p_3 \cdot q_2q_3$). Тогда в исправном автомате $a_1 \to a_1(y_1, y_4) \to a_2(y_2, y_4) \to a_2(y_2, y_4) \to a_1(y_2, y_4)$, а в неисправном — $a_1 \to a_1(y_1, y_4) \to a_2(y_2, y_4) \to a_2(y_2, y_4) \to a_1(y_1, y_4)$. Здесь также $\mathrm{FDL} = 4$, но неисправность проявляется только в выходном сигнале.

(3) Наконец, пусть поступила последовательность $\overline{x}_1 \to x_1, x_2 \to \overline{x}_2, x_3 \to x_1, x_2, x_3 \to x_2, x_3 \to x_2, \overline{x}_3 \to x_1, \overline{x}_2$ (вероятность $q_1 \cdot p_1p_2 \times \times q_2p_3 \cdot p_1p_2p_3 \cdot p_2p_3 \cdot p_2q_3 \cdot p_1q_2$). Тогда переходы исправного и неисправного автоматов $a_1 \to a_1(y_1, y_4) \to a_2(y_2, y_4) \to a_1(y_2, y_4) \to a_2(y_2, y_4) \to a_2(y_2, y_4) \to a_1(y_1, y_4) \to a_1(y_1, y_4)$ и $a_1 \to a_1(y_1, y_4) \to a_2(y_2, y_4) \to a_2(y_2, y_4) \to a_2(y_2, y_4) \to a_2(y_2, y_4) \to a_1(y_1, y_4) \to a_2(y_2, y_4)$ и $\mathrm{FDL} = 7$. Интересно отметить, что в данном случае на третьем шаге на исправном и неисправном автоматах состояния были различными (при одинаковых выходных сигналах), но затем на четвертом

Таблица 2 Пример не обнаруживающей себя неисправности

a_t	$X(a_t, a_s)$	Исправный		С неисправностью	
		a_s	$Y(a_t, a_s)$	a_s	$Y(a_t, a_s)$
a_1	$\overline{x}_1$	a_1	y_1, y_2	a_1	y_1, y_2
	$x_1, \overline{x}_3$	a_1	y_3	a_1	y_3
	$x_1, x_3, \overline{x}_4$	a_1		a_1	
	$x_1, x_3, x_4, \overline{x}_5$	a_2	y_2, y_3	a_3	y_2, y_3
	x_1, x_3, x_4, x_5	a_3	y_2, y_3	a_2	y_2, y_3
a_2	$\overline{x}_1, x_4$	a_1	y_1	a_1	y_1
	$\overline{x}_4$	a_2	y_2	a_3	y_2
	$x_1, \overline{x}_2, x_4$	a_2	y_2	a_2	y_2
	x_1, x_2, x_4	a_3	y_2	a_2	y_2
a_3	$\overline{x}_1, x_4$	a_1	y_1	a_1	y_1
	x_1, x_2, x_4	a_2	y_2	a_3	y_2
	$\overline{x}_4$	a_3	y_2	a_2	y_2
	$x_1, \overline{x}_2, x_4$	a_3	y_2	a_2	y_2

шаге неисправный автомат «восстановился» и работал в точности, как исправный, до седьмого шага.

Пример 3. Приведем пример, когда неисправность вообще не обнаруживается по выходу (табл. 2). Ошибочный переход $a_1 \to a_3$ вместо $a_1 \to a_2$ может произойти, например, в результате некоторой переходной (transient) неисправности. При этом, однако, поскольку выходы исправного и «сбойного» автоматов совпадают, причем и при следующих переходах, рассогласование в выходных переменных автоматов не происходит.

2.3 Модель латентности в терминах произведения исправного и неисправного автоматов

Поскольку латентность определяется по факту несовпадения выходов автомата при наличии и отсутствии неисправности, указанная информация в том или ином виде должна содержаться в математической модели вычисления ФР латентности. Такая модель предложена в [6]. Ниже приводится несколько отличное от [6] описание этой модели, предложенное в [10].

Пусть $\vec{x}(t) = (x_1(t), \ldots, x_n(t)) \in \mathcal{X}$ — вектор двоичных входных переменных в момент t, $a(t) \in \mathcal{A}$ и $a^f(t) \in \mathcal{A}$ — состояния исправного и f-неисправного автоматов Мили в этот момент, $\vec{y}(t) = (y_1(t), \ldots, y_m(t)) \in \mathcal{Y}$ и $\vec{y}^f(t) = (y_1^f(t), \ldots, y_m^f(t)) \in \mathcal{Y}$ — соответствующие векторы двоичных выходных переменных. Предполагается, что входные векторы представляют собой независимые (во времени и побитно) случайные последовательности.

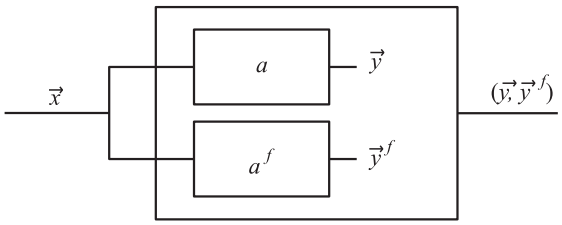


Рис. 2 Произведение исправного и неисправного конечных автоматов

На рис. 2 изображен автомат — произведение исправного и неисправного автоматов. Выходной вектор $(\vec{y}, \vec{y}^f)$ представляет собой вектор размерности $2m$, координатами которого являются выходные векторы $\vec{y}$ и $\vec{y}^f$ исправного и неисправного автоматов. Соответственно, пространство состояний такого автомата (произведения автоматов) — прямое произведение множеств $a(t) \in \mathcal{A}$ и $a(t) \in \mathcal{A}$, т. е. множество пар (a_k, a_l^f), $k, l = \overline{1, r}$.

В [6] вероятность того, что хотя бы на одном входном наборе значение хотя бы одного выходного бита в присутствии неисправности f будет отлично от исправного автомата, выражается как вероятность поглощения ЦМ, соответствующей переходам произведения исправного и неисправного автоматов (рис. 2) на указанной случайной входной последовательности. Поглощающее состояние определяется подмножеством состояний, при которых выходные векторы обоих автоматов не совпадают хотя бы в одном бите. Обозначим указанную цепь как ЦМПА (цепь Маркова произведения автоматов).

Кратко модель [6] может быть описана следующим образом:

– входные переменные исправного и неисправного автоматов представляют собой двоичные сигналы с вероятностями $p_i = \mathbf{Pr}(x_i = 1)$ и $q_i = \mathbf{Pr}(x_i = 0) = 1 - p_i$, $i = \overline{1,n}$, одновременно подаваемые на входы обоих автоматов. Соответственно, вероятности переходов как в исходном автомате, так и в автомате с неисправностью f, определяются формулой вычисления вероятности единичного значения булевой функции по известным вероятностям единиц ее аргументов. Например, переходы из состояния a_1 в состояние a_1 в исправном автомате табл. 1 происходят с вероятностью $\mathbf{Pr}(x_1 = 0) + \mathbf{Pr}(x_1 = 1, x_2 = 0) = q_1 + p_1 q_2$;

– начальные состояния у обоих автоматов одинаковы, поскольку различие в их поведении может проявиться только после первого такта работы;

– неисправности не добавляют новых состояний, и множество состояний исправного и неисправного автоматов совпадают;

– пространство состояний указанной ЦМПА представляет собой множество $\mathcal{W} = \{w_i,\ i = \overline{1, r^2+1}\} = \{(a_k, a_l),\quad k,l = \overline{1,r}\} \cup s_A$ из (r^2+1) элементов, где s_A — поглощающее состояние, а r — число состояний исходного автомата. Под поглощающим состоянием s_A понимается фиктивное состояние пары автоматов, появляющееся при первом несовпадении хотя бы в одном бите выходных векторов обоих автоматов.

Переходы в ЦМПА — пары (w_1, w_2) — отражают возможные изменения пар состояний в исправном и неисправном автоматах под действием случайных входов автомата. Соответственно, матрица $P = (p_{w_1,w_2})$, $w_1, w_2 \in \mathcal{W}$, вероятностей переходов за один такт задает вероятности того, что под влиянием случайно выбранного (на основе распределений p_i, $i = \overline{1,n}$) входного вектора в исправном автомате будет иметь место переход $a_i \to a_j$, тогда как в неисправном — $a_k \to a_l$. Иными словами, эта матрица описывает вероятности всех возможных переходов пар автоматов, причем те из них, при которых есть несовпадение хотя бы на одном из выходов, рассматриваются как переходы в поглощающее состояние s_A. Вероятности переходов, в свою очередь, вычисляются по вероятностям входных сигналов согласно пункту (1).

Пример 4. Таблица 3 соответствует матрице P вероятностей переходов ЦМПА для автоматов из примера 1, где в каждой паре (i,j) левый номер соответствует состоянию a_i в исправном автомате, а правый — состоянию a_j в неисправном автомате. Расчеты проводились с помощью программы, написанной В. В. Чаплыгиным (ИПИРАН). При расчетах были использованы следующие значения вероятностей p_i: $p_1 = 0{,}1$, $p_2 = 0{,}2$, $p_3 = 0{,}3$.

С учетом сказанного ФР $F_{\mathrm{FDL}}(k) = \mathbf{Pr}(\mathrm{FDL} \le k)$, $k \ge 1$, латентности обнаружения неисправности f, т. е. вероятность события «латентность не превосходит k тактов» можно записать в виде

$$F_{\mathrm{FDL}}(k) = \\ = \mathbf{Pr}(\min\{m > \tau : \vec{y}(\tau+m) \ne \vec{y}^f(\tau+m)\} \le k) = \\ = \sum_{w \in \mathcal{W}} p_w(0) p_{w,s_A}(k)\,, \quad (2)$$

где $\vec{y}(t)$, $\vec{y}^f(t)$ — векторы выходов исправного и f-неисправного автоматов; m — число тактов, прошедшее с момента появления неисправности (например, для неисправности $x_i \equiv 1$ — с момента «залипания» переменной x_i в состоянии «1» в неисправном автомате после такта τ); $p_w(0)$, $w \in \mathcal{W}$, — координаты вектора $\vec{p}(0)$ (размерности r^2+1) вероятностей состояний ЦМПА в начальный момент (на такте τ, после которого возникла неисправность, см. рис. 1). Поскольку в начальный момент оба автомата исправны, причем имеют одинаковые состояния (и, соответственно, одинаковые вероятности состояний), то координаты $p_w(0)$ с номерами $(i-1)r + i$, $i = \overline{1,r}$, совпадают с i-ми координатами вектора из вероятностей состояний исправного автомата на такте τ, а остальные координаты $p_w(0)$ равны нулю; $p_{w_1,w_2}(k)$, $k \ge 1$, $w_1, w_2 \in \mathcal{W}$, — элемент матрицы $P(k) = P^k$ вероятностей переходов ЦМПА за k тактов.

Таблица 3 Матрица P вероятностей переходов ЦМПА

Элемент	$(1,1)$	$(1,2)$	$(2,1)$	$(2,2)$	Поглощающее
$(1,1)$	0,900	0,000	0,000	0,020	0,080
$(1,2)$	0,630	0,000	0,000	0,006	0,364
$(2,1)$	0,126	0,080	0,000	0,006	0,788
$(2,2)$	0,140	0,240	0,000	0,060	0,560
Поглощающее	0,000	0,000	0,000	0,000	1,000

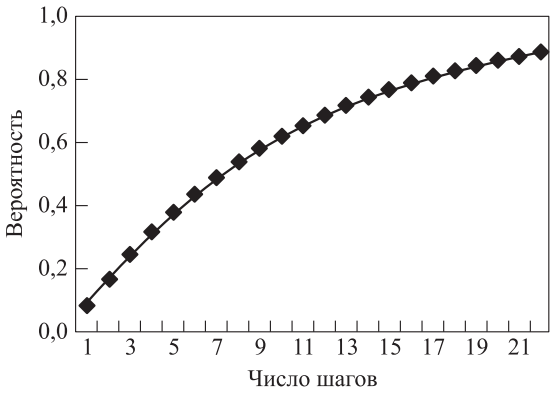


Рис. 3 Вероятность попадания в поглощающее состояние из состояния (a_1, a_1)

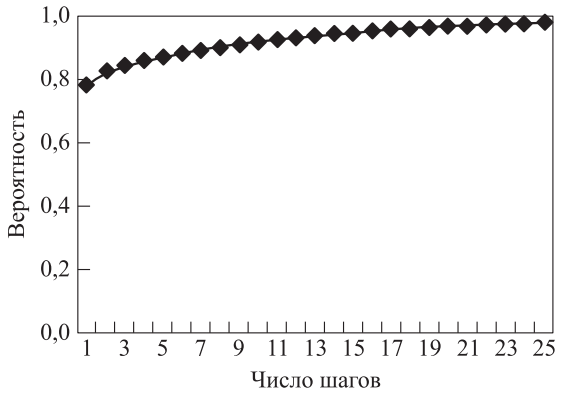


Рис. 4 Вероятность попадания в поглощающее состояние из состояния (a_2, a_1)

Матричная запись равенства (2) имеет вид

$$\mathbf{Pr}(\mathrm{FDL} \le k) = \vec{p}(0) P^k \vec{1}_e \,,$$

где $\vec{1}_e$ — вектор размерности (r^2+1), все координаты которого, кроме последней, равны нулю, а последняя — единице.

Пример 5. На рис. 3 и 4 представлены графики ФР латентности для пары автоматов из примера 1 (см. также пример 4). При этом считалось, что момент τ возникновения неисправности совпадал с начальным моментом 0 функционирования автомата, причем в первом случае (рис. 3) предполагалось, что в этот момент исправный (неисправный также) автомат находился в состоянии a_1 (вектор начальных вероятностей $\vec{p}(0) = (1,0,0,0,0)$). Предположение второго случая (рис. 4) о том, что в момент 0 исправный автомат находился в состоянии a_2, а неисправный — в состоянии a_1 (вектор начальных вероятностей $\vec{p}(0) = (0,0,1,0,0)$), кажется на первый взгляд несколько искусственным, но оказывается весьма полезным при дальнейшем изучении латентности.

3 Декомпозиция конечного автомата в сеть подавтоматов

3.1 Проблема декомпозиции конечного автомата

Поскольку целью настоящей статьи является демонстрация возможности использования модели произведения исправного и неисправного автоматов для вычисления латентности обнаружения неисправности в случае, когда при проектировании системы, специфицированной некоторым автоматом, он должен быть декомпозирован в некоторую сеть подавтоматов, то в этом пункте будет кратко изложена постановка проблемы такой декомпозиции. Подробное описывание алгоритмов декомпозиции автомата по критерию сокращения латентности обнаружения неисправности можно найти в [11].

В настоящей работе для упрощения изложения предлагается декомпозиция исходного автомата в сеть трех взаимодействующих подавтоматов, хотя результаты работы элементарно переносятся на декомпозицию в сеть любого числа подавтоматов.

При декомпозиции в сеть подавтоматов (рис. 5) каждый из подавтоматов реализует свою часть таблицы переходов исходного автомата (все его состояния и переходы между ними представляются в таблице переходов соответствующего подавтомата). Координирует работу подавтоматов управляющий блок, включающий в себя, в частности, автомат-супервизор (SFSM — Supervisor FSM).

В каждый момент времени один из подавтоматов находится в рабочем состоянии, когда его выходы используются согласно функциональному назначению (спецификации), а два других — в тес-

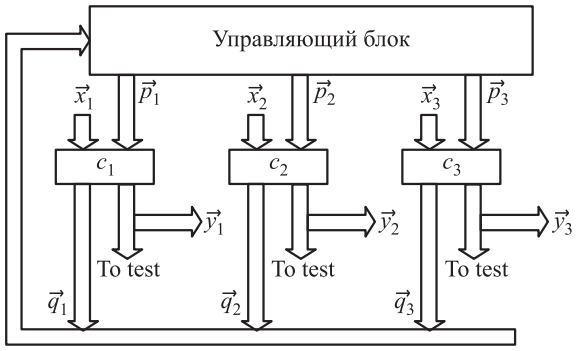


Рис. 5 Декомпозиция конечного автомата в сеть подавтоматов

товом, когда их выходы только проверяются на совпадение с исправными (на соответствие спецификации), но ни в каких действиях с окружением (например, в управлении некоторым операционным блоком, для которого проектируется автомат) они не участвуют.

Переменные выходов супервизора также присутствуют в подавтоматах, увеличивая число входов по сравнению с рабочими. Существенно, что переходы внутри одного подавтомата не изменяют состояния супервизора. При этом вектор $\vec{x}^i$, $i = \overline{1,3}$, входных переменных i-го подавтомата имеет вид $\vec{x}^i = (\vec{x}_i, \vec{p}_i)$, где $\vec{x}_i \in \mathcal{X}_i$, $\vec{p}_i \in \mathcal{P}_i$, $\mathcal{X}_i$ — подмножество входных наборов, существенное в состояниях подавтомата c_i, $\mathcal{P}_i$ — множество сигналов супервизора, поступающих на этот подавтомат. Очевидно, что $\bigcup\limits_{i=1}^{3} \mathcal{X}_i = \mathcal{X}$. Аналогично векторы $\vec{y}^i$, $i = \overline{1,3}$, выходов каждого подавтомата представимы в виде $\vec{y}^i = (\vec{y}_i, \vec{q}_i)$, где $\vec{y}_i \in \mathcal{Y}_i$, $\vec{q}_i \in \mathcal{Q}_i$, $\mathcal{Y}_i$ — множество выходов подавтомата c_i, $\mathcal{Q}_i$ — множество дополнительных выходов подавтомата c_i, передающих информацию о его текущем состоянии супервизору.

Тестирование сети автоматов с целью обнаружения неисправности производится на основе выходных сигналов (“to test” на рис. 5) с подавтоматов.

3.2 Модель декомпозиции исправного автомата

Опишем теперь модель декомпозиции автомата в сеть подавтоматов, для которой в следующем разделе будет построен алгоритм вычисления ФР латентности обнаружения неисправности. С точки зрения вероятностных характеристик латентности эта модель достаточно адекватно отражает функционирование приведенного выше декомпозированного конечного автомата.

Декомпозиция производится в соответствии с таблицей переходов исходного автомата и следующими правилами функционирования подавтоматов и автомата-супервизора (управляющего блока на рис. 5):

- если переходы происходят внутри одного подавтомата (компонента сети), то переходы и выходы этого компонента соответствуют переходам и выходам исходного автомата;
- если начальное и конечное состояния данного перехода принадлежат разным подавтоматам, то подавтомат, содержащий начальное состояние, становится тестируемым и переходит в некоторое (определенное таблицей переходов данного подавтомата в рабочем режиме) начальное состояние, а подавтомат, содержащий конечное состояние перехода, инициируется специальным сигналом автомата-супервизора;
- конечное состояния перехода тестируемого подавтомата (до момента его перехода в рабочий режим) определяется таблицей переходов данного подавтомата в тестовом режиме);
- выходом сети подавтоматов являются вектор $\vec{y} = (\vec{y}_1, \vec{y}_2, \vec{y}_3)$, состоящий из векторов $\vec{y}_1$, $\vec{y}_2$ и $\vec{y}_3$ выходов подавтоматов, и номер k рабочего подавтомата.

Заметим, что в соответствии с приведенными правилами функционирования сети подавтоматов, если в исправном режиме функционирования рассматривать только вектор $\vec{y}_k$ выходов рабочего подавтомата, то он будет совпадать с вектором выходов исходного автомата.

Пример 6. Рассмотрим (исправный) автомат табл. 4.

Предположим, что произведена декомпозиция этого автомата на подавтоматы c_1, c_2 и c_3 с множествами состояний $\mathcal{A}_1 = \{a_1, a_2, a_5\}$, $\mathcal{A}_2 = \{a_6, a_7, a_8\}$ и $\mathcal{A}_3 = \{a_3, a_4, a_9\}$. Декомпозиция произведена в соответствии с правилами функционирования подавтоматов и автомата-супервизора (таблица переходов которого здесь не приводится, см. [11]), определяемыми табл. 5.

Приведем комментарии к табл. 4 и 5.

Если рабочий подавтомат находится в состоянии a_i (столбец “a_t”) и приходит сигнал $\vec{x}$ (столбец “$X(a_t, a_s)$”), переводящий исходный автомат (табл. 4) в состояние a_j (столбец “a_s”) того же подавтомата, то этот подавтомат остается рабочим при новом состоянии a_j (см. табл. 4 и 5). Если же рабочий подавтомат находится в состоянии a_i и приходит сигнал $\vec{x}$, переводящий исходный автомат в состояние a_j другого подавтомата, то рабочим становится новый подавтомат при состоянии a_j (столбец “a_s” табл. 4), а прежний рабочий подавтомат переходит в режим тестирования при состоянии a_k (столбец “a_s” табл. 5). Более того, в это же состояние переходит и тестируемый подавтомат, если меняются ролями два других. Наконец, переходы тестируемого подавтомата (до момента любого изменения рабочего подавтомата) происходят в соответствии с табл. 5).

Например, если работает подавтомат c_1, находящийся в состоянии a_1, и приходит сигнал x_3, x_4, x_5, то подавтомат c_1 остается рабочим при том же состоянии a_1 (см. табл. 4 и 5). Но ес-

Таблица 4 Таблица переходов автомата

a_t	a_s	$X(a_t, a_s)$	$Y(a_t, a_s)$
a_1	a_1	x_3, x_4, x_5	$-$
	a_5	$x_3, x_4, \overline{x}_5$	y_1, y_2
	a_7	$x_3, \overline{x}_4$	y_1, y_3
	a_9	$\overline{x}_3$	y_2
a_2	a_1	x_5, x_6, x_7	y_2, y_3, y_4
	a_7	$x_5, x_6, \overline{x}_7$	y_3
	a_6	$x_5, \overline{x}_6$	y_2, y_3
	a_6	$\overline{x}_5$	y_2
a_3	a_6	x_5, x_6	y_2, y_3
	a_7	$x_5, \overline{x}_6$	y_3, y_4
	a_8	$\overline{x}_5, \overline{x}_6$	y_3, y_4
	a_2	$\overline{x}_5, x_6$	y_2
a_4	a_3	x_6, x_7	y_2
	a_5	$x_6, \overline{x}_7$	y_2, y_3
	a_7	$\overline{x}_6, x_7$	y_3
	a_8	$\overline{x}_6, \overline{x}_7$	y_3, y_4
a_5	a_2	x_1, x_2	y_3, y_4
	a_3	$x_1, \overline{x}_2$	y_4, y_5
	a_4	$\overline{x}_1$	y_3, y_5
a_6	a_2	x_4, x_5	y_1, y_3
	a_5	$x_4, \overline{x}_5$	y_2
	a_7	$\overline{x}_4$	y_1, y_3
a_7	a_4	x_2, x_3	y_2
	a_5	$\overline{x}_2, x_3$	y_1, y_3
	a_7	$\overline{x}_3$	y_3, y_4
a_8	a_8	x_3, x_4	y_2
	a_6	$x_3, \overline{x}_4$	y_1, y_3
	a_7	$\overline{x}_3, x_5$	y_1, y_3
	a_8	$\overline{x}_3, \overline{x}_5$	$-$
a_9	a_3	x_1, x_3	y_3
	a_4	$x_1, \overline{x}_3$	y_3, y_5
	a_5	$\overline{x}_1$	y_3, y_4

Таблица 5 Таблица переходов подавтоматов сети автоматов

a_t	a_s	$X(a_t, a_s)$	$Y(a_t, a_s)$
a_1	a_1	x_3, x_4, x_5	$-$
	a_5	$x_3, x_4, \overline{x}_5$	y_1, y_2
	a_5	$x_3, \overline{x}_4$	y_1, y_3
	a_5	$\overline{x}_3$	y_2
a_2	a_1	x_5, x_6, x_7	y_2, y_3, y_4
	a_5	$x_5, x_6, \overline{x}_7$	y_3
	a_5	$x_5, \overline{x}_6$	y_2, y_3
	a_5	$\overline{x}_5$	y_2
a_3	a_9	x_5, x_6	y_2, y_3
	a_9	$x_5, \overline{x}_6$	y_3, y_4
	a_9	$\overline{x}_5, \overline{x}_6$	y_3, y_4
	a_9	$\overline{x}_5, x_6$	y_2
a_4	a_3	x_6, x_7	y_2
	a_9	$x_6, \overline{x}_7$	y_2, y_3
	a_9	$\overline{x}_6, x_7$	y_3
	a_9	$\overline{x}_6, \overline{x}_7$	y_3, y_4
a_5	a_2	x_1, x_2	y_3, y_4
	a_5	$x_1, \overline{x}_2$	y_4, y_5
	a_5	$\overline{x}_1$	y_3, y_5
a_6	a_8	x_4, x_5	y_1, y_3
	a_8	$x_4, \overline{x}_5$	y_2
	a_7	$\overline{x}_4$	y_1, y_3
a_7	a_8	x_2, x_3	y_2
	a_8	$\overline{x}_2, x_3$	y_1, y_3
	a_7	$\overline{x}_3$	y_3, y_4
a_8	a_8	x_3, x_4	y_2
	a_6	$x_3, \overline{x}_4$	y_1, y_3
	a_7	$\overline{x}_3, x_5$	y_1, y_3
	a_8	$\overline{x}_3, \overline{x}_5$	$-$
a_9	a_3	x_1, x_3	y_3
	a_4	$x_1, \overline{x}_3$	y_3, y_5
	a_9	$\overline{x}_1$	y_3, y_4

ли при том же начальном условии приходит сигнал $\overline{x}_3$, то рабочим становится подавтомат c_3 при состоянии a_9 (см. табл. 4), а подавтомат c_1 переходит в режим тестирования при состоянии a_5 (см. табл. 5). Наконец, если тестируются подавтоматы c_1 и c_2, находящиеся в любых (возможных для них) состояниях, рабочий подавтомат c_3 находится в состоянии a_4 и приходит сигнал $\overline{x}_6, x_7$, то подавтомат c_2 становится рабочим (при состоянии a_7, см. табл. 4), подавтомат c_3 начинает тестироваться (из состояния a_9, см. табл. 5), а подавтомат c_1 переходит в состояние a_5 (см. табл. 5).

Отметим, что в данном примере при переходе подавтомата из рабочего состояния в режим тестирования его тестирование начинается из одного и того же состояния. Более того, в это же состояние переходит и тестируемый подавтомат, если меняются ролями два других. Так, подавтомат c_1 при всех таких изменениях переходит в состояние a_5, подавтомат c_2 — в состояние a_8 и подавтомат c_3 — в состояние a_9.

3.3 Учет неисправности в модели декомпозиции

Обратимся теперь к автомату с некоторой неисправностью f. Он также может быть декомпозирован на три подавтомата с теми же самыми множествами состояний. Однако переходы и выходные векторы как исходного f-неисправного автомата, так и его подавтоматов, будут, вообще говоря, отличаться от переходов и выходных векторов исправного автомата.

Контроль правильности функционирования (например, с использованием механизма самопроверки [1]) проводится как на тестируемом, так и на работающем подавтоматах. Если хотя бы по одному из выходов зафиксирована ошибка, работа системы

останавливается (например, для проведения самовосстановления).

Поскольку порядок тестирования рабочих подавтоматов (фактически совпадающий с тестированием всего неисправного автомата) полностью определяется моделью произведения автоматов пункта 2.3, кратко остановимся на процедуре тестирования неисправного подавтомата.

Пусть в некоторый момент k-й неисправный подавтомат, находившийся в рабочем режиме, из состояния a_j при входном сигнале $\vec{x}$ переходит в режим тестирования, которое начинается из состояния a_j. Поставим ему в соответствие k-й исправный подавтомат в режиме тестирования с состоянием a_j. Теперь, воспользовавшись моделью произведения автоматов пункта 2.3, будем наблюдать за выходами $\vec{y}$ и $\vec{y}^f$ исправного и неисправного k-х подавтоматов. Наблюдение проводится до того момента, когда впервые произойдет несовпадение выходных сигналов или неисправный автомат снова станет рабочим.

В ряде случаев для снижения латентности оказывается более эффективным проводить контроль подавтоматов по схеме, когда в каждый момент времени неработающие подавтоматы тестируются (независимо друг от друга и от работающего подавтомата) на сигналах от некоторого встроенного генератора (для простоты можно считать, что данный генератор находится в управляющем блоке и передает сигналы для тестирования в выходных векторах $\vec{p}_i$, $i = \overline{1,3}$, см. рис. 5). Именно для такой схемы декомпозиции в следующем разделе будет построена математическая модель латентности.

Данный подход может оказаться эффективным, если, например, проектируемое устройство содержит встроенный генератор для других целей [1, 9]. Заметим, что использование независимого тестирования во многих случаях может улучшить качество тестирования, например благодаря выбору вероятностного распределения тестовых сигналов по критерию минимизации латентности обнаружения неисправности [4].

В заключение этого раздела отметим, что в рассмотренной модели декомпозиции переходы каждого подавтомата в общем случае не образуют ЦМ, поскольку изменение состояния на очередном шаге зависит не только от его состояния на предыдущем шаге, но и от значений сигналов супервизора на данном шаге, которое, вообще говоря, зависит от состояний не только данного, но и других двух подавтоматов на предыдущем шаге. Тем не менее, как только что было показано, и в этом случае можно использовать модель произведения исправного и неисправного автоматов пункта 2.3 и выразить ФР латентности обнаружения неисправности в терминах матрицы переходных вероятностей подавтоматов сети, но с учетом режимов (рабочего или тестового) их функционирования.

4 Алгоритм вычисления функции распределения латентности обнаружения неисправности

4.1 Общие положения

В этом разделе предлагается математическая модель для описанной выше декомпозиции автомата в сеть независимо тестируемых подавтоматов и алгоритм вычисления ФР латентности обнаружения неисправности для этой модели, базирующийся на модели произведения исправного и неисправного автоматов, описанной в пункте 2.3.

Пусть первый подавтомат имеет n_1 состояний, второй — n_2 состояний и третий — n_3 состояний. Всего у автомата $n = n_1 + n_2 + n_3$ состояний. Сделаем следующие дополнительные предположения:

- в момент любого выхода каждого подавтомата из рабочего режима его тестирование начинается из одного и того же состояния. Более того, в этот момент так же заново из одного и того же состояния начинается тестирование того подавтомата, который продолжает тестироваться;
- тестирование подавтоматов производится независимыми сигналами, т. е. тестовые сигналы каждого подавтомата статистически независимы и от информационных сигналов (входных сигналов исходного декомпозируемого автомата), и от тестовых сигналов второго тестируемого подавтомата;
- в подавтомате-супервизоре, управляющем переключением режимов подавтоматов, неисправности не возникают.

Сделанные предположения позволяют в качестве математической модели процесса функционирования неисправного декомпозированного автомата с тестированием подавтоматов рассмотреть ЦМ, в которой зависимость ФР латентности обнаружения неисправности от дополнительного тестирования подавтоматов определяется только через ФР времени обнаружения неисправности, которую можно вычислить заранее.

4.2 Вычисление функции распределения времени обнаружения неисправности при тестировании подавтомата

Рассмотрим k-й, $k = \overline{1,3}$, подавтомат неисправного автомата. Предполагая, что в началь-

ный момент он начинает тестироваться (на основе сравнения его выходных сигналов с выходными сигналами k-го исправного подавтомата), причем в этот момент информационные сигналы прекращаются и остальные подавтоматы останавливаются, найдем ФР $F_k^f(m)$ числа шагов до проявления неисправности f в этом подавтомате в тестовом режиме. Для этого воспользуемся моделью пункта 2.3 и рассмотрим ЦМ с множеством состояний $\mathcal{Y}_k = \{(i,j),\ i,j = \overline{1,n_k}\} \cup s_A$, где состояние $y = (i,j)$ означает, что подавтомат k исправного автомата находится в состоянии i, а соответствующий подавтомат неисправного автомата — в состоянии j. Положим $N_k = n_k^2 + 1$. Переход из состояния $y_1 = (i_1, j_1)$ в состояние $y_2 = (i_2, j_2)$ осуществляется при поступлении входного набора (сигнала), переводящего исправный подавтомат из состояния i_1 в состояние i_2, а подавтомат неисправного автомата — из состояния j_1 в состояние j_2, причем выходные сигналы обоих подавтоматов совпадают. Попадание в поглощающее состояние s_A ЦМ на очередном шаге при нахождении на предыдущем шаге в состоянии $y_1 = (i_1, j_2)$ происходит в том случае, когда поступает любой сигнал, переводящий подавтомат исправного автомата из состояния i_1 в состояние i_2, подавтомат автомата с неисправностью — из состояния j_1 в состояние j_2, и выходные сигналы этих подавтоматов при таком переходе не совпадают.

Для определения матрицы P_k переходных вероятностей ЦМ необходимо знать также вероятности входных сигналов при тестировании. Эти вероятности задаются условиями тестирования подавтомата.

Таким образом, находим матрицу P_k переходных вероятностей ЦМ. Отметим, что если вероятность обнаружения неисправности при тестировании k-го подавтомата отлична от нуля, то матрица P_k является полустохастической, т. е. хотя бы для одной строки сумма элементов меньше единицы.

Вектор $\vec{p}(m) = (p_1(m), \ldots, p_{N_k}(m))$ вероятностей состояний ЦМ после m-го шага определяется по формуле

$$\vec{p}(m) = \vec{p}(0) P_k^m \,,$$

а начальное условие $\vec{p}(0)$ определяется состоянием i_0, с которого начинается тестирование подавтомата, и имеет вид

$$p_{(i,j)}(0) = \begin{cases} 1\,, & i = j = i_0\,, \\ 0\,, & \text{иначе}\,. \end{cases}$$

Поэтому

$$F_k(m) = \vec{p}(m)\vec{1}_e = \vec{p}(0) P_k^m \vec{1}_e \,.$$

4.3 Вычисление функции распределения латентности обнаружения неисправности декомпозированного автомата

Построим обрывающуюся ЦМ, описывающую совместное функционирование до момента обнаружения неисправности (момента обрыва ЦМ) как исправного автомата, так и f-неисправного автомата и его декомпозиции на три подавтомата.

Свяжем с работой декомпозированного автомата функционирующий в дискретном времени случайный процесс $\nu(t) = (\nu_1(t), \nu_2(t), \nu_2(t), \nu_4(t))$, где $\nu_2(t)$ — номер состояния, в которое переходит на такте t исправный автомат, $\nu_1(t)$ и $\nu_3(t)$ — номера работающего после этого перехода подавтомата неисправного автомата и его состояния, $\nu_4(t)$ — число тактов, прошедших с момента последней передачи управления от одного подавтомата неисправного автомата к другому. Процесс обрывается в момент обнаружения неисправности.

Учитывая теперь предположения пункта 4.1, получаем, что случайный процесс $\nu(t)$ является ЦМ с множеством состояний $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$, где

$$\mathcal{X}_k = \left\{(i,j,m)\,,\ i = \overline{1,n}\,,\ j = \overline{1,n_k}\,,\ m \geq 0\right\}\,, \qquad k = \overline{1,3}\,.$$

При этом состояние (i,j,m) множества $\mathcal{X}_k$ означает, что исправный автомат находится в состоянии i, работает k-й подавтомат, который пребывает в состоянии j, и с момента последней передачи управления прошло m шагов.

Определим матрицу Q переходных вероятностей построенной ЦМ. Для этого приведем все возможные (ненулевые) переходы этой цепи и вычислим вероятности таких переходов.

Из (непоглощающего) состояния $x_1 = (i_1, j_1, m)$, $x_1 \in \mathcal{X}_k$, $k = \overline{1,3}$, может произойти переход в (непоглощающее) состояние $x_2 = (i_2, j_2, m+1)$, $x_2 \in \mathcal{X}_k$. Такой переход возникает в том случае, когда передача управления от одного подавтомата автомата с неисправностью к другому не происходит, причем выходные сигналы рабочего подавтомата автомата с неисправностью и исправного автомата совпадают, и, кроме того, не происходит обнаружения неисправности ни в одном из двух тестируемых подавтоматов. Для того чтобы определить вероятность $q(x_1, x_2)$ данного перехода, найдем сначала условную вероятность этого перехода при условии, что проявление неисправности в тестируемых подавтоматах не происходит. Условная вероятность перехода будет совпадать с вероятностью $q_{(i_1,j_1),(i_2,j_2)}$ перехода

исправного и неисправного автоматов из состояний i_1 и j_1 в состояния i_2 и j_2 (без тестирования подавтоматов). При тестировании подавтоматов вероятность $q_{(i_1,j_1),(i_2,j_2)}$ необходимо дополнительно умножить на условные вероятности $F_s(m+1)/F_s(m)$ и $F_t(m+1)/F_t(m)$ (где $s \neq k, t \neq k$ и $s \neq t$) того, что неисправность не будет обнаружена при тестировании нерабочих подавтоматов на $(m+1)$-м шаге при условии, что она не была обнаружена на m-м шаге (напомним, что функции $F_k(m)$, $k = \overline{1,3}$, были определены в предыдущем пункте). Таким образом, элемент $q(x_1, x_2)$ матрицы Q_{kk} вероятностей переходов из состояний подмножества $\mathcal{X}_k$ в состояния этого же подмножества имеет вид

$$q(x_1, x_2) = \frac{F_s(m+1)}{F_s(m)} \cdot \frac{F_t(m+1)}{F_t(m)} q_{(i_1,j_1),(i_2,j_2)} \cdot$$

Кроме того, из (непоглощающего) состояния $x_1 = (i_1, j_1, m)$, $x_1 \in \mathcal{X}_k$, $k = \overline{1,3}$, может произойти переход в (непоглощающее) состояние $x_2 = i_2, j_2, 0)$, $x_2 \in \mathcal{X}_s$, $s = \overline{1,3}$, $s \neq k$, (в неисправном автомате происходит передача управления). Переход происходит в том случае, когда при входном сигнале исправный автомат переходит из состояния i_1 в состояние x_2, неисправный автомат из состояния j_1 k-го подавтомата переходит в состояние j_2 s-го подавтомата, выходные сигналы рабочего подавтомата автомата с неисправностью и исправного автомата совпадают и не происходит обнаружения неисправности ни в одном из двух тестируемых подавтоматов. Так же, как и прежде, элемент $q(x_1, x_2)$ матрицы Q_{ks} вероятностей переходов из состояний подмножества $\mathcal{X}_k$ в состояния подмножества $\mathcal{X}_s$ получается умножением вероятности $q_{(i_1,j_1),(i_2,j_2)}$ на условные вероятности $F_s(m+1)/F_s(m)$ и $F_t(m+1)/F_t(m)$ (где $t \neq k$ и $t \neq s$):

$$q(x_1, x_2) = \frac{F_s(m+1)}{F_s(m)} \cdot \frac{F_t(m+1)}{F_t(m)} q_{(i_1,j_1),(i_2,j_2)} \cdot$$

Вероятности остальных переходов (остальные элементы матриц Q_{ks}, $k, s = \overline{1,3}$) равны нулю.

Матрица Q переходных вероятностей общей ЦМ, описывающей совместное функционирование как виртуального, так и рабочего автоматов, представима в виде

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \end{pmatrix} .$$

Обозначим через $\vec{q}(n) = (\vec{q}_1(n), \vec{q}_2(n), \vec{q}_2(n))$ вектор вероятностей состояний ЦМ после n-го шага. Здесь $\vec{q}_k(n)$, $k = \overline{1,3}$, $n \geq 0$, — вектор, координатой $q_{k,x}(n) = q_{k,(i,j,m)}(n)$, $x \in \mathcal{X}_k$, которого является вероятность того, что после n-го шага ЦМ будет находиться в состоянии $x = (i, j, m) \in \mathcal{X}_k$. Вектор $\vec{q}(n)$ определяется по формуле

$$\vec{q}(n) = \vec{q}(n-1)Q = \vec{q}(0)Q^n .$$

Вектор $\vec{q}(0)$ начальных вероятностей определяется условиями пуска системы.

Окончательно вероятность того, что неисправность проявится до момента n (ФР латентности обнаружения неисправности декомпозированного автомата), определяется выражением

$$F_{FDL}(n) = \mathbf{Pr}(FDL \leq n) = 1 - \vec{q}(n)\vec{1} = \\ = 1 - \sum_{k=1}^{3} \sum_{(i,j,m) \in \mathcal{X}_k} q_{k,(i,j,m)}(n) .$$

Таким образом, удалось представить ФР латентности декомпозированного автомата в терминах ФР латентности такого же, но не декомпозированного автомата и ФР времен обнаружения неисправности при тестировании подавтоматов, на которые автомат был декомпозирован в процессе проектирования.

4.4 Анализ возможности применения предложенной модели

Скажем несколько слов о сфере возможного применения алгоритма вычисления ФР латентности обнаружения неисправности декомпозированного конечного автомата на основе рассмотренной выше модели.

На первый взгляд кажется, что, поскольку обрывающаяся ЦМ, моделирующая неисправный автомат, имеет бесконечное число состояний (время до обнаружения неисправности тестируемого подавтомата, вообще говоря, может быть сколь угодно большим), то возникают серьезные трудности с практической реализацией алгоритма. Однако здесь нужно учесть следующие обстоятельства.

При декомпозиции автомата на подавтоматы с небольшим числом состояний у каждого подавтомата ФР $F_k^f(m)$ числа шагов до проявления неисправности в k-м подавтомате в тестовом режиме имеет простой аналитический вид (смесь небольшого числа геометрических распределений или, более обще, распределений Паскаля), что фактически сводит трудоемкость расчетов декомпозированного автомата к трудоемкости расчетов недекомпозированного автомата.

Часто встречается такая ситуация, когда времена между передачами управления от одного подавтомата к другому достаточно малы. Это позволяет при

расчетах общей латентности автомата использовать небольшое число значений ФР $F_k^f(m)$.

Наконец, во многих случаях можно существенно уменьшить размерность ЦМ при большом числе состояний подавтоматов, воспользовавшись следующими двумя принципами.

Первый принцип связан с тем, что, как правило, функционирование реального и виртуального автоматов начинается из одного и того же непоглощающего состояния $(\vec{i}, \vec{i})$, а из всех состояний $(\vec{i}, \vec{i})$ достижимы далеко не все состояния $(\vec{i}_1, \vec{i}_2)$. Недостижимые состояния можно исключить из рассмотрения, решая задачу построения дерева достижимых вершин ориентированного графа, порожденного переходами с ненулевыми вероятностями ЦМ.

Второй принцип основан на том, что для многих состояний $(\vec{i}_1, \vec{i}_2)$ ЦМ, имеющих ненулевые вероятности перехода в поглощающее состояние, вероятности таких переходов за один шаг в точности равны 1, а значит, их можно объединить в одно состояние, из которого переход в поглощающее состояние обязательно происходит на первом же шаге. Аналогично можно объединить в одно состояние все такие состояния, переход из которых в поглощающее состояние происходит ровно на втором шаге, и т. д.

5 Заключение

Поскольку декомпозиция некоторого исходного автомата в сеть подавтоматов широко применяется в современном проектировании сложных цифровых устройств (не только для снижения латентности обнаружения неисправностей, но и для минимизации потребления энергии в проектируемых системах — так называемое Low Power Design, см. [12]), а оценка латентности обнаружения неисправности важна для многих приложений [4, 5], распространение методов предсказания ФР латентности [6] на сети подавтоматов, получаемых в результате декомпозиции конечно-автоматной спецификации проектируемого устройства, представляется весьма актуальной.

Предложенный здесь подход к оценке ФР латентности основан на расширении пространства состояний описываемой системы, что позволяет строить ЦМ для пар подавтоматов (в общем зависимых) исправного и неисправного устройств и выражать распределение вероятностей латентности в терминах вероятностных характеристик подавтоматов.

Отметим, что в [7] рассматривается иной подход к вычислению ФР латентности времени обнаружения неисправности. Он основан не на явном использовании произведения двух автоматов, а на отдельном рассмотрении траекторий автомата, на которых неисправность не проявляется, и траекторий, соответствующих наличию искаженных выходов. При этом вероятность обнаружения неисправности на i-м шаге равна вероятности того, что переходы автомата до $(i-1)$-го шага соответствовали первому классу траекторий, а на i-м шаге оказались во втором классе траекторий. Траектории рассматриваются как марковские. Трудность использования такого подхода состоит в том, что подобное разбиение на траектории в общем случае не удается автоматизировать.

Литература

1. *Lala P.* Self-checking and fault-tolerant digital design. — Morgan Kaufmann Publ., 2000.
2. *Baranov S.* Logical synthesis of digital systems. — Tallin: TTU Publs., 2008.
3. *Baumann R.* Soft errors in advanced computer systems // IEEE Design and Test., 2005 (May–June). P. 258–266.
4. *Soh B. C., Dillon T. S.* Incorporation of multiple errors in reliability modeling of fault-tolerant systems // IEEE Fault Tolerant Systems, Pacific Rim International Symposium Proceedings, 1991. P. 148–153.
5. *Hagbae K., Kang G. S., Chuck R.* On reconfiguration latency in fault-tolerant systems // IEEE Aerospace Applications Conference Proceedings, 1995. P. 287–301.
6. *Shedletsky J., McCluskey E.* The error latency of fault in a sequential digital circuit // IEEE Transaction on Computers, 1976. Vol. 25. No. 6. P. 655–659.
7. *Goot R., Levin I., Ostanin S.* Fault latencies of concurrent checking FSMs // Euromicro Symposium on Digital System Design (DSD'02) Proceedings, 2002.
8. *Карибский В. В., Пархоменко П. П., Согомонян Е. С., Халчев В. Ф.* Основы технической диагностики. — М.: Энергоиздат, 1976.
9. *Bennets R. G.* Design of testable logic circuits. — Addison-Wesley Publ. Co., 1984.
10. *Frenkel S., Pechinkin A., Chaplygin V., Levin I.* A mathematical tool for support of fault-tolerant embedded systems design // ERCIM/DECOS Dependable Smart Systems: Research, Industrial Applications, Standardization, Certification and Education. Workshop on Dependable Embedded Systems. Luebeck, Germany, 2007.
11. *Levin I., Abramov B., Ostrovsky B.* Reduction of fault latency in sequential circuits by using decomposition // 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2007. P. 261–272.
12. *Sudnitson A.* Computational kernel extraction for synthesis of power-managed sequential components // IEEE 9th Conference (International) on Electronics, Circuits and Systems (ICECS 2002) Proceedings. Dubrovnik, Croatia, 2002. P. 749–752.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 15–24

РЕЗЕРВНОЕ КОПИРОВАНИЕ, ИСПОЛЬЗУЮЩЕЕ СНИМКИ

В. А. Козмидиади[1]

Аннотация: Рассмотрена сложность резервного копирования (backup), использующего различия между двумя последовательными снимками (snapshots). Предложены формальная модель, в которой можно точно поставить проблему, а также алгоритм, который решает эту проблему за почти линейное время от объема различий.

Ключевые слова: резервное копирование (backup); снимки (snapshots); восстановление файловой системы; журнал файловой системы

1 Введение

Под *резервным копированием* (*backup*) [1–4] понимается создание копии данных, предназначенной для их восстановления в оригинальном месте при повреждении или разрушении. Резервное копирование выполняется периодически, поэтому в случае нарушения данных содержимое сохраненной копии может отставать от оригинальных данных. Резервное копирование предполагает наличие оригинала, который выступает передатчиком, и копии, играющей роль приемника.

Далее будет рассматриваться только резервное копирование файловой системы. Резервное копирование обычно занимает значительное время, поэтому на это время нельзя запретить доступ пользователей к файловой системе. Но тогда при копировании может происходить создание, изменение, удаление файлов и директорий, а это может привести к тому, что созданная копия будет неконсистентной, т. е. данные могут оказаться не согласованными друг с другом. Чтобы избежать этого создают *снимок* (*snapshot*) файловой системы [5, 6]. Снимок можно рассматривать как доступную только для чтения файловую систему, данные которой есть замороженные в определенный момент времени данные исходной системы. Изготовление снимка состоит из двух этапов. Перед изготовлением завершаются все выполняющиеся запросы к файловой системе. Далее на очень небольшой промежуток времени T задерживаются все новые запросы и делается сам снимок. Промежуток времени T для многих файловых систем является константой и имеет очень небольшое значение. Предполагается, что снимок содержит консистентные данные, поэтому при резервном копировании целесообразно использовать снимок. Таким образом, резервирование данных обеспечивается следующим образом. Периодически изготавливаются снимки файловой системы, после чего выполняется резервное копирование, т. е. происходит синхронизация, замена старого снимка на новый. Если на оригинальной системе происходит отказ, приведший к искажению или потере данных, осуществляется восстановление файловой системы по последнему сохраненному ее снимку.

Передача всего снимка фактически означает передачу всех данных файловой системы. Делать это, пусть даже с низкой частотой, недопустимо при сколь-нибудь значительных объемах файловой системы. Если частота синхронизации не слишком низка, объем сделанных изменений обычно невелик: он на много порядков меньше объема всей файловой системы. Поэтому хотелось бы найти способ передавать только изменения, которые отличают предыдущую копию от текущего оригинала. При этом будем предполагать, что созданная копия не изменяется до следующего резервного копирования. Обычно системы порождения снимков дают возможность сохранять несколько снимков, сделанных в разные моменты времени. Поэтому можно считать, что в распоряжении имеется два снимка. Первый снимок соответствует копии, хранящейся на приемнике и сделанной в предыдущем цикле резервного копирования. Второй снимок соответствует копии, которая должна быть создана в следующем цикле. Для передачи изменений нужно определить, в чем эти снимки различаются. Некоторые изменения могут касаться структуры файловой системы, например добавление и удаление файлов и директорий. Такие изменения приводят к необходимости передачи информации о произведенных изменениях структуры и, если это надо, передачи данных файлов. Другие изменения сводятся к модификации данных файлов. Это приводит к передаче информации о сделанных модифи-

[1]Институт проблем информатики Российской академии наук, v.kozmidiady@gmail.com

кациях. В данной работе не будет рассматриваться вопрос об экономной передаче данных или модификации этих данных. Это совершенно отдельная тема; сошлемся здесь на работы по сжатию данных [7] и алгоритмы передачи, которые учитывают наличие на приемнике более старой версии данных [8]. В статье рассматривается только проблема поиска различий между снимками. Понятно, что при отсутствии дополнительной информации для этого нужно рассмотреть оба снимка. Если число файлов и директорий в первом снимке равно N_1, а во втором снимке — N_2, для выяснения различий потребуется никак не меньшая сложность, чем $O(N_1 + N_2)$. Цель настоящей работы — привлечь дополнительную информацию (ниже она названа журналом работы файловой системы), чтобы получить сложность $O(\Delta)$. Здесь Δ — количество измененных, созданных или удаленных файлов либо директорий между двумя снимками. Сложность этой задачи подчеркнута в [9].

При рассмотрении будем ориентироваться на UNIX-подобные файловые системы, например такие, как [10].

2 Определения и формулировка задачи

Приведенные ниже определения дают возможность точно сформулировать решаемую в работе задачу.

2.1 G — класс DAG

Пусть G — класс непустых направленных ациклических графов (directed acyclic graphs, DAG), которые удовлетворяют следующим ограничениям:

- в графе выделена вершина R (корень), из которой доступны все остальные вершины;
- каждой вершине V приписан номер (натуральное число), причем разным вершинам приписаны разные номера. Вершины мы будем отождествлять с их номерами;
- каждой вершине приписаны данные;
- только листья могут иметь несколько входящих дуг;
- каждая вершина имеет тип $t \in \{D, F\}$. Тип удовлетворяет следующим свойствам:
  (1) тип D могут иметь только вершины, имеющие не более одной входящей дуги;
  (2) тип F могут иметь только листья;
  (3) вершина-корень имеет тип D;
- каждая дуга $V_1 \to V_2$ имеет имя. Из вершины не могут исходить две дуги с одинаковыми именами.

2.2 Семантика

Каждый член $g \in G$ — это некоторый граф файловой системы. Вершинами V этого графа являются директории и файлы. Дуги $V_1 \to V_2$ соответствуют отношениям «Директория V_1 содержит V_2 — другую директорию или файл».

1. Номер вершины — это номер ее индексного дескриптора (inode number). Индексный дескриптор является описателем отдельной директории или файла.
2. Тип вершины — указание на то, соответствует ли она директории (D) или файлу (F).
3. Имя дуги $V_1 \to V_2$ — это имя директории или файла V_2.
4. Если вершина имеет тип D (директория), то данные, приписанные вершине, — это метаданные директории[1]. Если вершина имеет тип F (файл), то данные, приписанные вершине, — это метаданные файла и его содержание.

Граф файловой системы может не быть деревом из-за наличия жестких ссылок[2] (hard links) и символических ссылок[3] (symbolic links). Далее предполагается, что в файловой системе отсутствуют символические ссылки, поэтому, как нетрудно видеть, ее граф является DAG.

2.3 Номера индексных дескрипторов и их свойства

Номер индексного дескриптора является уникальным идентификатором элемента файловой

[1] Метаданные директории включают права доступа владельца и других пользователей, время создания, последней модификации и т. д. Метаданные файла дополнительно содержат размер файла и другие параметры.

[2] Жесткая ссылка — это ссылка, содержащаяся в директории и указывающая на принадлежащий этой директории файл, точнее на его индексный дескриптор. Жесткая ссылка включает в себя имя файла. Использование жестких ссылок — стандартный способ организации файловой системы. На индексный дескриптор одного файла может ссылаться несколько жестких ссылок. Одно из применений — введение равноправных синонимичных имен файла. Жесткие ссылки на директории не допускаются.

[3] Символическая ссылка, как и жесткая ссылка, содержит имя файла и находится в директории, но, в отличие от жесткой ссылки, указывает не на индексный дескриптор, а на другую жесткую или символическую ссылку. Поэтому, если в файловой системе не предприняты специальные меры, возможно образование циклов из символических ссылок.

системы, т. е. индексного дескриптора, соответствующего файлу или директории. Свойства номеров индексных дескрипторов:

- номера индексных дескрипторов не используются повторно, т. е. если какой-либо элемент имел некоторый номер, а далее был удален, то этот номер никогда в дальнейшем не будет присвоен новому элементу;
- номер индексного дескриптора элемента присваивается ему при создании и остается неизменным до тех пор, пока элемент не будет удален. Отсюда следует, что:
  (1) изменение данных или метаданных элемента сохраняет номер элемента неизменным;
  (2) изменение состава директории сохраняет номер индексного дескриптора директории неизменным;
  (3) перемещение и переименование элемента с помощью функции rename(...) не изменяет номер элемента. Если перемещаемый элемент является директорией, при перемещении подчиненные элементы также не меняют своих номеров;
- если файл имеет несколько ссылок, номер индексного дескриптора файла не зависит от использованного пути к файлу.

2.4 Операции над $g \in G$

Обозначим через $V \in g$, $g \in G$, отношение «V является вершиной графа g». Ниже рассматривается набор операций над графами $g \in G$. Операция O может быть применена к графу G, если $O(g) \in G$.

1. $R(V_1, V_2)$ — удаление дуги $V_1 \to V_2$. Вершина V_1 должна иметь тип D, а вершина V_2 — тип F или D. Это означает, что удаляемая дуга может начинаться только в директории. Если V_2 имеет тип F (в этом случае в V_2 может вести несколько дуг), а удаляемая дуга — это единственная дуга, ведущая в V_2, то вместе с дугой удаляется и вершина V_2. Таким образом, вместе с удалением последней жесткой ссылки удаляется и сам файл. Если V_2 имеет тип D (в этом случае в V_2 может вести только одна дуга), то V_2 должна быть листом, т. е. несмотря на то, что ей соответствует директория, она должна быть пуста, в ней не может начинаться ни одна дуга.
2. $A_E(V_1, V_2, n)$ — добавление дуги $V_1 \to V_2$. Добавляемой дуге присваивается имя n. Вершина V_1 должна иметь тип D, а вершина V_2 — тип F. Это добавление жесткой ссылки.
3. $A_V(V_1, V_2, n)$ — добавление дуги $V_1 \to V_2$ и вершины V_2, $V_2 \notin g$. Вершина V_2 может иметь любой тип, а вершина V_1 должна иметь тип D. Добавляемой дуге присваивается имя n. Создание новой директории или файла.
4. $M(V_1, V_2, V_3, n)$ — перемещение дуги. Создается дуга $V_3 \to V_2$, ей присваивается имя n. Дуга $V_1 \to V_2$ удаляется. Вершины V_1 и V_3 должны иметь тип D. Это переименование с перемещением. Файл (директория) V_2, который раньше находился в директории V_1, получает новое имя и теперь размещается в директории V_3.
5. $C(V)$ — изменение метаданных и/или содержания вершины.

Операции имеют следующие свойства:

- ни одна из операций не может изменить ни тип вершины, ни ее номер;
- ни одна из операций не может удалить вершину-корень.

2.5 Журнал работы файловой системы

Этот журнал представляет собой линейно упорядоченное (по возрастанию номеров) множество номеров индексных дескрипторов, сопровождаемых набором признаков и номеров (параметров). Набор признаков следующий:

- Create, создание вершины;
- Del, удаление вершины;
- LCount, удаление дуги без удаления вершины (изменение числа жестких ссылок на файл без удаления файла);
- Mod, изменение метаданных (для директории) либо метаданных или содержания (для файла);
- Move, перемещение дуги.

Таким образом, журнал представляет собой последовательность записей; запись содержит номер индексного дескриптора и набор признаков.

Каждая из перечисленных выше операций добавляет в журнал определенный номер или номера, если их там не было, и устанавливает для них некоторые признаки, а некоторые устраняет. Если какой-то признак требуется установить, а он уже установлен, никаких действий не предпринимается. Аналогично — для устранения признаков. Например, если для вершины нужно установить признак Del, а для нее уже установлены признаки Create, Mod, их следует устранить. Ниже перечислены допустимые комбинации установленных признаков:

Таблица 1 Операции, добавляемые номера и признаки

Операция	Добавляемый номер	Устанавливаемый признак
$R(V_1, V_2)$ без удаления вершины V_2	V_2	LCount
$R(V_1, V_2)$ с удалением вершины V_2	V_2	Del
$A_E(V_1, V_2, n)$	V_2	Create
$A_V(V_1, V_2, n)$	V_2	Create
$M(V_1, V_2, V_3, n)$	V_2	Move
$C(V)$	V	Mod

– Create;
– Del;
– [LCount], [Mod], [Move].

Таблица 1 описывает эти добавляемые при выполнении операций номера и признаки.

Рассмотрим S_1 и S_2 — два последовательных снимка файловой системы. Снимок S_2 получается из S_1 с помощью определенной последовательности операций $O_1, O_2, \ldots, O_n$, которые были перечислены выше. Вначале журнал файловой системы пуст. Каждая операция O_i $(1 \le i \le n)$, возможно, добавляет в журнал один или несколько номеров. Для некоторых номеров устанавливаются признаки. В итоге обработки всей последовательности операций достигается снимок S_2 и образуется журнал, который описывает изменения, произошедшие за это время. Если $V(S)$ — множество номеров, присутствующих в снимке S, то множество номеров в журнале есть подмножество $V(S_1) \cup V(S_2)$. Журнал работы файловой системы между двумя последовательными снимками будем обозначать через $J(S_1, S_2)$.

Примечание. Описанная структура журнала работы файловой системы неестественна и очень неудобна для работы. Но эта структура определяется в основном соображениями производительности. Например, был бы естественным журнал, представляющий собой последовательный файл. Хвост этого файла постоянно пополняется описаниями выполненных операций. Такое решение очень неэффективно. Файл может оказаться очень большим, если снимки делаются редко. Только операции чтения не требуют записи в этот файл. Кроме того, следует учитывать, что обычно файловая система может одновременно выполнять несколько операций. Особенно это важно для распределенных файловых систем. Журнал с последовательной организацией оказывается «узким местом», так как является совместно используемым ресурсом, требующим монопольного захвата. В то же время индексные дескрипторы файлов и директорий, находящиеся в работе, размещаются в памяти. Введение в них дополнительных полей не вызывает затруднений. Поэтому разработчику средств резервного копирования трудно предлагать удобную для него структуру журнала, ему приходится соглашаться с тем, что согласны сделать для него разработчики файловой системы.

2.6 Формулировка задачи

Пусть имеются S_1 и S_2 — два последовательных снимка файловой системы — и журнал работы файловой системы $J(S_1, S_2)$. Требуется построить такую последовательность операций (возможно, отличающуюся от исходной), которая S_1 переводит в S_2. Эта работа выполняется на передатчике. Предполагается, что на приемнике есть файловая система, состояние которой совпадает со снимком S_1. Тогда, передавая на приемник построенную последовательность операций и выполняя на приемнике эти операции, можно привести состояние файловой системы преемника в состояние, соответствующее S_2.

Требуется оценить сложность построения нужной последовательности операций. Но чтобы оценить сложность, следует описать средства, которые можно применять. К этому описанию и перейдем.

2.7 Инструментарий и оценка сложности

При построении требуемой последовательности операций допускается применение любых стандартных для UNIX функций, используемых для доступа к файловой системе. Набор таких функций содержится в [10]. Кроме того, можно применять специальные функции доступа к снимкам и журналу. Примерный перечень этих функций приведен ниже.

Принятые при описании функций обозначения

S	снимок
S_1, S_2	два последовательных снимка
V	номер индексного дескриптора (здесь отождествлены индексные дескрипторы, их номера и вершины графа снимка)
$V(S)$	множество номеров индексных дескрипторов, присутствующих в снимке S

n_J количество упомянутых номеров индексных дескрипторов в журнале работы файловой системы $J(S_1, S_2)$

N количество номеров в $V(S_1) \cup V(S_2)$

2.8 Описание функций

$\mathrm{get\_j}(V)$**.** Функция возвращает запись, следующую за записью с номером индексного дескриптора, т. е. с вершиной V. $\mathrm{get\_j}(-1)$ дает первую запись.

$\mathrm{is\_vertex}(V, S)$**.** Функция-предикат; возвращает **true**, если $V \in V(S)$, **false** — в противном случае.

$\mathrm{type}(V, S)$**.** Функция возвращает тип вершины V в снимке S. Если $V \in V(S_1) \cap V(S_2)$, то $\mathrm{type}(V, S_1) = \mathrm{type}(V, S_2)$.

$\mathrm{name}(V_1, V_2, S)$**.** Функция возвращает имя дуги $V_1 \to V_2$ в снимке S.

$\mathrm{path}(V, S)$**.** Функция возвращает некоторый путь от вершины-корня до V в снимке S.

$\mathrm{pred}(V, S)$**.** Функция возвращает множество вершин снимка S, предшествующих вершине S.

$\mathrm{succ}(V, S)$**.** Функция возвращает множество вершин снимка S, которые являются непосредственными потомками вершины V в S. Эта функция напоминает библиотечную функцию readdir(...).

Выполнение всех этих функций, кроме первой, $\mathrm{get\_j}(V)$, есть $O(V(S))$. Функция $\mathrm{get\_j}(V)$ выполняется на константное время.

Примечание. Функции pred( ) и succ( ) следовало бы заменить парами функций. Первая функция возвращает первый элемент списка, а вторая — следующий по указанию предыдущего.

2.9 Оценки сложности

Мерой сложности, используемой для оценки построения последовательности операций, будем считать число вызовов стандартных и специальных функций.

3 Граф операций

Граф операций строится на передатчике по снимкам S_1, S_2 и журналу работы файловой системы $J(S_1, S_2)$. Далее этот граф используется для того, чтобы привести файловую систему приемника в состояние, идентичное снимку S_2 передатчика. Граф операций является DAG, как это будет ясно из дальнейшего.

3.1 Вершины графа операций

Граф операций состоит из вершин, которым сопоставлены номера индексных дескрипторов. Множество его вершин включает:

- все вершины из S_1, которые упомянуты в журнале $J(S_1, S_2)$. Эти вершины соответствуют элементам S_1 (файлам и директориям), которые были изменены, созданы или удалены в промежутке времени от изготовления S_1 до изготовления S_2;
- вершины из S_1, которые не упомянуты в журнале $J(S_1, S_2)$, но находятся на путях, ведущих от корневой вершины R к вершинам, уже включенным в граф операций.

Последнее нужно для того, чтобы можно было построить полный путь от R к каждой вершине графа операций. Полное имя состоит из (простых) имен, разделенных символами «/» (UNIX-подобные системы) или «\» (Windows). Если взять полный путь от корневой вершины R к некоторой вершине, принадлежащей S_1 или S_2, то полное имя образуется из имен, приписанных дугам, составляющим этот путь. Даже если файловые системы приемника и передатчика идентичны, номера индексных дескрипторов их соответствующих элементов могут быть различны, в отличие от имен. Поэтому, когда передатчик (оригинал) требует, чтобы приемник выполнил какие-либо действия над своей файловой системой, указания должны формулироваться с использованием имен, а не номеров индексных дескрипторов.

3.2 Дуги графа операций

Эти дуги соответствуют отношению «родитель — непосредственный потомок» либо в S_1, либо в S_2. Каждой дуге приписано имя дуги и вектор операций, возможно пустой. Элементы вектора операции описаны в разд. 2.4. Некоторые операции сопровождаются дополнительными параметрами.

Граф операций является DAG, так как таковыми являются S_1 и S_2.

3.3 Конфликтные ситуации

Конфликт, который приводит к необходимости введения временного имени, заключается в том, что в какой-либо директории нужно создать элемент (директорию или файл) с некоторым именем, а в этой директории уже содержится элемент с тем же именем. Поскольку в снимке S_2 создаваемый элемент существует, то конфликт имеет временный

характер — мешающий элемент должен быть переименован, удален или перенесен. Поэтому создаваемому элементу можно присвоить временное имя, а после завершения операций над мешающим элементом заменить временное имя настоящим. Заметим, что контекст конфликта — всегда директория.

3.4 Синхронизация приемника и передатчика

Предположим, что на приемнике находится файловая система в состоянии, соответствующем снимку S_1. Для того чтобы привести ее в состояние, идентичное снимку S_2 передатчика (их синхронизацию), требуется обойти вершины графа операций в определенном порядке, перемещаясь по дугам и выполняя операции, приписанные дугам.

4 Построение графа операций

Следующие предположения обычно выполняются для многих файловых систем, кроме того, они нужны, чтобы сделать оценки сложности:

- содержимое журнала работы файловой системы $J(S_1, S_2)$ находится в оперативной памяти;
- все промежуточные деревья и сам граф операций также размещаются в оперативной памяти;
- при оценках сложности алгоритмов предполагается, что для всех номеров-вершин графа операций строится B-дерево. Ключом этого B-дерева является номер, а значением — указатель размещения экземпляра структуры, описывающей вершину. В связи с этим появляются логарифмические оценки;
- при отказе приемника после его восстановления выполняется возврат к содержимому его файловой системы в соответствии со снимком S_1. Это занимает очень мало времени. В файловой системе ZFS (Zettabyte File System) это достигается с помощью команды ‘zfs rollback S’ [11]. Поэтому изменения можно делать непосредственно в файловой системе, не используя временные файлы и их последующее переименование.

4.1 Фазы построения

Построение разбивается на следующие фазы:

Фаза 1. Построение дерева номеров индексных дескрипторов (I-дерева).

Фаза 2. Построение промежуточного дерева M.

Фаза 3. Построение окончательного графа (графа операций).

Примечание. Разбиение на приведенные фазы сделано для упрощения изложения. Выполнение некоторых фаз можно совместить (возможно, что все их можно уложить в одну фазу).

4.1.1 Фаза 1 (построение I-дерева)

На этой фазе строится дерево, вершины которого — только номера из S_1, находящиеся в журнале $J(S_1, S_2)$, т. е. из $V(S_1) \cap V(J(S_1, S_2))$. Это дерево имеет в качестве корневой вершину R и является минимальным поддеревом S_1, содержащим указанные вершины из $V(S_1) \cap V(J(S_1, S_2))$. При построении для каждой вершины из $V(S_1) \cap V(J(S_1, S_2))$ вычисляется полный путь к ней от R и этот путь укладывается в дерево. В I-дереве каждой дуге приписано имя, взятое из S_1. Дугам, ведущим в вершины, присутствующие в журнале, приписывается вектор операций из журнала для этих вершин, а остальным — пустой вектор операций.

Алгоритм построения. Последовательно анализируются номера из журнала. Обрабатываются только те из них, которые принадлежат S_1. Для каждого такого номера выполняются следующие действия: вычисляется полный путь в S_1 от вершины R до вершины с этим номером. Этот путь состоит из номеров, причем первый номер — номер корневой вершины R, а все номера на пути принадлежат S_1. Анализируется I-дерево, уже построенное к этому моменту. Возможны два случая, описанные ниже:

(1) обрабатываемый номер еще отсутствует в I-дереве. В I-дерево включается та часть пути, которая в дереве отсутствует. Каждой добавленной дуге, ведущей в промежуточную вершину, приписывается имя в вычисленном полном пути и пустой вектор операций. Дуге, ведущей в добавленную вершину, приписывается имя в вычисленном полном пути и вектор операций из журнала;

(2) обрабатываемый номер уже присутствует в I-дереве (в этом случае дуге, ведущей в эту вершину, обязательно приписан пустой вектор операций). Этой дуге приписывается вектор операций из журнала.

Векторы операций, которые присвоены дугам в I-дереве, содержат только операции, присутствующие в журнале, а именно:

- Mod (изменение содержимого или метаданных файла);
- LCount (изменение числа ссылок на файл без удаления файла);
- Move (перемещение файла);
- Del (удаление файла);

– Create (создание файла);
– Mod (изменение метаданных директории);
– Move (перемещение директории);
– Del (удаление директории);
– Create (создание директории).

Сложность алгоритма. $O(n_J \log(N))$.

4.1.2 Фаза 2 (построение дерева M)

На этой фазе I-дерево расширяется до M-дерева за счет включения дополнительных дуг, связанных с перемещением файлов и директорий. В I-дереве обрабатываются дуги с операциями Move «Перемещение файла» или Move «Перемещение директории». Если перемещение происходит в пределах S_1, ($V_3 \in V(S_1)$), дуга «до перемещения» удаляется, а вместо нее создается дуга «после перемещения». Если перемещение происходит в пределах $V(S_2)\backslash V(S_1)$, делается то же самое, однако предварительно в дерево включается путь от R до вершины V_3, из которой должна исходить новая дуга. Если перемещение выводит за пределы S_2, операция дуги заменяется операцией Del «Удаление файла» или Del «Удаление директории». Последнее может быть только в случае, когда имеем дело не со всей файловой системой, а лишь с ее фрагментом, т. е. подграфом файловой системы.

Алгоритм построения. Последовательно анализируются номера из журнала. Обрабатываются только номера i, которые принадлежат S_1. В I-дереве рассматриваются дуги, которые ведут в вершину i и у которых векторы операций имеют операцию Move «Перемещение файла» или Move «Перемещение директории». Если $i \notin V(S_2)$, то операция заменяется операцией Del «Удаление файла» или Del «Удаление директории». Если $i \in V(S_2)$, то вычисляется полный путь от R до этого номера i в S_2. Этот путь состоит из номеров, причем первый номер — вершина подграфа S_1, а все номера в пути принадлежат S_1 или S_2. В M-дерево включается та часть пути, которая в M-дереве отсутствует. Для образующих эти части пути вершин j возможны два случая:

– вершина j отсутствует в S_1. Дуге приписывается операция Create «Создание директории»;
– вершина j присутствует в S_1, в частности это вершина i. Вершина k, предшествующая в вычисленном полном пути вершине j, может как принадлежать, так и не принадлежать S_1, однако дуга $k \to j$ отсутствует в уже построенном к этому моменту M-дереве. Поэтому имеется еще одна и только одна дуга, ведущая в вершину j ($l \to j$), причем в векторе операций этой дуги должны быть операции Move «Перемещение файла» или Move «Перемещение директории». Эта дуга удаляется и создается новая дуга $k \to j$. Новой дуге приписывается вектор операций удаленной дуги. В этом векторе операция Move «Перемещение файла» или Move «Перемещение директории» заменяется операцией Move «Перемещение дуги». В качестве параметров этой операции указываются l и имя дуги $l \to j$. Таким образом, у вершины j заменяется родитель l, им становится k.

Если родитель k вершины j отсутствует в S_1, то дуге $k \to j$ приписывается имя в вычисленном полном пути. Если же j присутствует в S_1, анализируется наличие у k исходящей дуги с именем в вычисленном полном пути. Если такой дуги нет, то дуге $k \to j$ приписывается то же простое имя, что и в предыдущем случае. Если же такая дуга есть, то дуге $k \to j$ приписывается временное уникальное для родителя имя. Кроме того, в вектор операций добавляется вспомогательная операция Rename «Переименование временного имени» с параметром — имя из пути.

Примечание. M-граф по-прежнему представляет собой дерево, потому что любая вершина, кроме корня, имеет только одну входящую дугу, а связность потеряться не может.

Сложность алгоритма. $O(n_J \log(N))$.

4.1.3 Фаза 3 (построение графа операций)

На этой фазе M-дерево расширяется до графа операций за счет включения дуг, соответствующих удаленным и добавленным ссылкам. Каждый номер из журнала, если он принадлежит $V(S_1) \cap V(S_2)$ и соответствует файлу, может иметь операцию LCount «Изменение числа ссылок на файл без удаления файла». Для каждого такого номера по спискам ссылок в S_1 и S_2 вычисляется набор удаленных или добавленных ссылок. В M-дерево добавляются дуги, которые отвечают удаленным и добавленным ссылкам. В результате образуется граф операций.

Алгоритм построения. Последовательно анализируются номера из журнала. Обрабатываются только такие номера l, что $l \in V(S_1) \cap V(S_2)$ с операцией LCount «Изменение числа ссылок на файл без удаления файла» или $l \in V(S_1)\backslash V(S_2)$ с операцией Del «Удаление файла».

– Операция LCount «Изменение числа ссылок на файл без удаления файла». Строится два списка:
  (1) L_1 — список ссылок на файл l в S_1. Каждый элемент списка состоит из номера родителя p_i и имени ссылки;

(2) L_2 — аналогичный список ссылок на файл l, но в S_2.

Далее эти два списка сравниваются. Возможны три ситуации, рассмотренные ниже. В любой из них если оказывается, что родитель p_i не принадлежит построенному к этому моменту графу операций, то он сам и полный путь до него включаются в граф операций.

1. Номер $i \in L_1 \backslash L_2$. Добавляется дуга $p_i \to l$, которой приписывается вспомогательная операция «Удаление ссылки». Если дуга $p_i \to l$ уже имеется, из нее удаляется операция LCount «Изменение числа ссылок на файл без удаления файла» и добавляется операция «Удаление ссылки».
2. Номер $i \in L_2 \backslash L_1$. Добавляется дуга $p_i \to l$, которой приписывается операция «Создание ссылки». Если дуга $p_i \to l$ уже имеется, из нее удаляется операция LCount «Изменение числа ссылок на файл без удаления файла» и добавляется операция «Создание ссылки». Дуге приписывается имя способом, описанным в фазе 2.
3. Номер $i \in L_1 \cap L_2$. Если дуга $p_i \to l$ уже имеется, из нее удаляется операция LCount «Изменение числа ссылок на файл без удаления файла». Если в итоге образуется пустой вектор операций, дуга удаляется. В противном случае дуга сохраняется.

– Операция Del «Удаление файла». Строится L — список ссылок на файл l в S_1. Дуга с операцией Del «Удаление файла» удаляется. Для каждого номера $i \in L$ добавляется дуга $p_i \to l$, которой приписывается операция «Удаление ссылки».

Граф операций представляет собой DAG, а не дерево из-за наличия дуг, соответствующих добавленным или удаленным ссылкам. Поскольку эти дуги могут вести только в листья, циклы невозможны.

Сложность алгоритма. $O(n_J \log(N))$.

5 Выполнение синхронизации

5.1 Свойства графа операций

1. Граф операций описывает только различия между снимками S_1 и S_2 и операции, которые нужно выполнить, чтобы перейти от S_1 к S_2.
2. Граф операций полностью описывает структуру тех директорий и файлов S_2, которые отсутствуют в S_1.
3. Граф операций не полностью описывает структуру директорий и файлов S_1. В нем присутствуют только те директории, файлы и пути к ним, которые необходимы для адресации поддиректорий и файлов, которые имелись в S_1 и были изменены, удалены или перенесены в другие директории.
4. Операции, приписанные дугам, соответствуют действиям, которые нужно производить при синхронизации.

5.2 Векторы операций, приписанные дугам в графе операций

Эти векторы могут содержать следующие операции:

– изменение содержимого или метаданных файла;
– создание ссылки;
– удаление ссылки;
– создание файла;
– изменение метаданных директории;
– перемещение дуги. Параметры:
  - номер индексного дескриптора директории, из которой удаляется дуга;
  - имя удаляемой дуги;
– создание директории;
– удаление директории;
– переименование временного имени. Параметр — присваиваемое имя.

5.3 Синхронизация

Выполнение синхронизации состоит из двух фаз:

(1) фазы создания и модификации;
(2) фазы удаления и устранения временных имен.

При выполнении обеих фаз происходит обход графа операций в порядке «от родителей к потомкам», т. е. для некоторой вершины l сначала обрабатываются все ее исходящие дуги, ведущие к файлам, а потом делается рекурсивный спуск по исходящим дугам, ведущим к директориям.

5.3.1 Фаза создания и модификации

На этой фазе выполняются только следующие операции:

– изменение содержимого или метаданных файла;
– создание ссылки;
– создание файла;
– изменение метаданных директории;
– перемещение дуги;
– создание директории.

Сначала происходит обработка текущей директории «в ширину» с выполнением всех перечисленных выше операций. Все дуги, которые соответствуют созданию файла из S_2 или созданию ссылки на него, имеют одну и ту же операцию «Создание файла». На приемнике по этой операции выполняется функция open(. . . , O_CREAT | O_EXCL, . . .). Если ошибка не возникает, файл создан и выполняется его копирование из S_2 передатчика. При ошибке EEXIST вместо создания файла создается ссылка на уже существующий файл. Если файл уже существовал в S_1, то операции для него — «Создание ссылки» и «Удаление ссылки». Первая операция делается на данной фазе, а вторая — на следующей. Далее происходит рекурсивный спуск по каждой из дуг, ведущих к директориям.

Примечания.

1. В процессе обхода из векторов операций дуг, ведущих к файлам, можно удалять все выполненные операции (в векторе могут остаться только операции «Удаление ссылки» и «Переименование временного имени»). Если дуга после удаления имеет пустой вектор операций, то дугу также можно удалить. Это позволяет уменьшить обход на следующей фазе. Удалять дуги, ведущие к директориям, во время обхода в первой фазе не удается, хотя и они могут быть не нужны. Однако после первой фазы можно выполнить такое удаление. *Критерий удаления*: дуга, ведущая к директории, может быть удалена, если и она, и все дуги подграфа имеют пустые векторы операций.
2. Для одного элемента все операции, выполняемые на этой фазе, кроме операции «Перемещение дуги», можно осуществлять в любом порядке. Операция «Перемещение дуги», если она присутствует в векторе, должна выполняться первой. Нужный порядок выполнения операций может быть достигнут за счет соответствующего упорядочения вектора операций при построении графа операций.

5.3.2 Фаза удаления и устранения временных имен

На этой фазе выполняются только следующие операции:

– удаление ссылки;
– удаление директории;
– переименование временного имени.

Сначала происходит обработка текущей директории «в ширину» с выполнением операций «Удаление ссылки» (это может быть удаление ссылки или файла) или «Удаление директории». Отметим, что создание файлов и ссылок на них выполнены на предыдущей фазе, поэтому удаление ссылок не может привести к удалению файлов, если это не требуется. После этих действий все конфликтные ситуации в директории, если они имели место, разрешены — мешающих элементов больше нет. Поэтому только после выполнения операций удаления для директории возможно выполнение операций «Переименование временного имени». Наконец, происходит рекурсивный спуск по каждой из дуг, ведущих к директориям.

Примечание. Порядок — сначала выполнение операций «Удаление ссылки» и «Удаление директории», далее «Переименование временного имени», а потом рекурсивный спуск — может быть достигнут за счет соответствующего порядка дуг, исходящих из вершины (они, конечно, образуют двунаправленный список). Нужный порядок можно обеспечить при построении графа операций.

5.4 Особенности выполнения фаз

5.4.1 Трансляция и удаление директорий

Трансляция номера индексного дескриптора передатчика в номер приемника требуется в двух местах:

(1) перемещении файла или директории, которые существовали в S_1;
(2) создании ссылки на файл, который существовал в S_1.

«Удаление директории» не всегда удаляет пустую директорию. Это может быть по двум причинам:

(1) существовавшая в S_1 директория перемещена за пределы снимка;
(2) на файл, принадлежащий удаляемой директории, сделана ссылка.

Должна быть удалена только ссылка из удаляемой директории. Для того чтобы удаляемая директория была пуста, нужно сделать следующее:

– дополнить граф операций, включив в него все удаляемые вершины и дуги, ведущие к ним;

Таблица 2 Возможности параллельного выполнения операций

Операция		Операция								
		1	2	3	4	5	6	7	8	9
1	Изменение содержимого или метаданных файла	u	y	u	u	y	n	n	u	u
2	Создание ссылки	y	y	u	u	y	n	n	u	u
3	Удаление ссылки	u	u	y	u	u	n	u	y	n
4	Создание файла	u	u	u	y	y	n	n	u	u
5	Изменение метаданных директории	y	y	u	y	u	n	u	u	u
6	Перемещение дуги	n	n	n	n	n	u	n	u	u
7	Создание директории	n	n	u	n	u	n	u	u	u
8	Удаление директории	u	u	y	u	u	u	u	u	n
9	Переименование временного имени	u	u	n	u	u	u	u	n	u

Обозначения:
y — параллельное выполнение возможно;
n — параллельное выполнение невозможно;
u — ситуация невозможна (операции несовместимы или выполняются в разных фазах).

– на фазе удаления и устранения временных имен изменить порядок обхода графа операций: не «от родителей к потомкам», а «от потомков к родителю» (DFO).

Изменение порядка обхода приводит к значительной модификации кода процессов приемника. В частности, появляется необходимость иметь таблицу трансляции «номер индексного дескриптора передатчика → номер приемника».

5.4.2 Возможности параллельного выполнения операций

Таблица 2 показывает, какие операции можно выполнять параллельно. В случае файла параллельность означает возможность одновременного выполнения для него нескольких операций. В случае директории — возможность одновременного выполнения операций над директорией и файлами, которые находятся в подграфе этой директории.

6 Заключение

В итоге не было получено результата, которого добивались: $O(\Delta)$. Однако сама эта постановка вопроса недостаточно точна. Достигнутый результат — $O(n_J \log(N))$. Можно считать, что n_J — это действительно Δ. Но $\log(N)$ обусловлен не описанным алгоритмом, а временем выполнения специальных функций, предоставленных разработчиками файловой системы. Можно ли выбрать другой набор функций, достаточных для поставленных целей, но выполняющихся быстрее? Можно ли разработать такую структуру журнала, которая удовлетворяет разработчиков, но более удобна и позволяет реализовать более простые алгоритмы? Подобные вопросы по-прежнему остаются открытыми.

Литература

1. *Batten C., Barr K., Saraf A., Treptin S.* Abstract pStore: A secure peer-to-peer backup system. MIT Laboratory for Computer Science. Technical Report MIT-LCS-TM-632, December 2001. 12 p.
2. *Suparna B., Mohan C., Brannon K. W., Inderpal N., Hui-I H. I., Mahadevan S.* Coordinating backup/recovery and data consistency between database and file systems // 2002 ACM SIGMOD Conference (International) on Management of Data Proceedings. Madison, June 2002. P. 500–511.
3. *Killijian, M.-O., Courtès L., Powell D.* A survey of cooperative backup mechanisms. Universitès de Toulouse. Laboratoire d'Analyse et d'Architecture des Systèmes. LAAS Technical Report 06472. April 2007. 23 p.
4. *Preston W. C.* Inexpensive backup solutions for open systems. — C.A.: O'REILLY, 2007. 760 p.
5. *Neeta G.* Understanding and exploiting snapshot technology for data protection. Part 1: Snapshot technology overview. April 2006. http://www-128.ibm.com/developerworks/tivoli/library/t-snaptsm1.
6. *Brinkmann A., Effert S.* Snapshots and continuous data replication in cluster storage environments // 4th Workshop (International) on Storage Network Architecture and Parallel I/Os Proceedings. San Diego, 2007. P. 3–10.
7. *Salomon D.* Data compression. — N. Y.: Springer, 1998. 448 p.
8. *Tridgell A.* Efficient algorithms for sorting and synchronization. A thesis submitted for the degree of Doctor of Philosophy at The Australian National University. February 1999. 115 p.
9. Want 'zfs diff' to list files that have changed between snapshots, July 2008. http://bugs.opensolaris.org/bugdatabase/printableBug.do?bug_id=6425091.
10. Solaris ZFS Administration guide, 2008. http://dlc.sun.com/pdf/817-2271/817-2271.pdf.
11. IEEE Std 1003.1,2004 Edition. http://www.unix.org/version3/ieee_std.html.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 25–33

АДАПТАЦИЯ БИОМЕТРИЧЕСКОЙ СИСТЕМЫ К ИСКАЖАЮЩИМ ФАКТОРАМ НА ПРИМЕРЕ ДАКТИЛОСКОПИЧЕСКОЙ ИДЕНТИФИКАЦИИ*

О. С. Ушмаев[1]

Аннотация: Рассмотрены проблемы учета искажающих факторов при биометрической идентификации. На примере дактилоскопической идентификации проанализировано влияние искажений на качество идентификации. Предложены методы учета искажений на основе нормальной аппроксимации факторов. Проведенные эксперименты показали эффективность предложенного подхода к учету искажений. На основе разработанных методов предложена технология адаптации биометрической системы к искажающим факторам.

Ключевые слова: биометрическая идентификация; автоматическая дактилоскопическая идентификация; искажающие факторы

1 Введение

На сегодняшний день технологии биометрической идентификации получают широкое распространение в различных системах гражданской идентификации. Это и паспортно-визовые документы нового поколения, и идентификация получателей социальных услуг, и идентификационные карты работников отдельных категорий (государственные служащие, транспортная безопасность) и пр. [1–5].

Длительное накопление биометрических массивов приводит к ситуации, когда одновременно обрабатывается информация, собранная с использованием разных технических средств, в разное время и при разных условиях. Постепенное отклонение характеристик биометрической информации от эталонных значений приводит к отклонению показателей качества биометрической идентификации от проектных значений.

В качестве примеров постепенной деградации биометрической идентификации можно привести следующие:

- возрастные изменения лицевой биометрии ведут к тому, что качество идентификации при увеличении временного лага между регистрацией в системе и идентификацией падает;
- сезонные колебания качества дактилоскопической идентификации: зима и лето характеризуются принципиально различными условиями получения отпечатков пальцев;
- изменение в качестве идентификации отпечатков пальцев, полученных из разных источников: бумажных дактокарт, «живым» сканированием, следов отпечатков и т. д.

В [6–9] приведены примеры деградации биометрической идентификации для широкого круга технологий и систем.

Производители биометрических технологий постоянно совершенствуют алгоритмы распознавания с целью повышения устойчивости к искажающим факторам. Однако многие проблемы устранить невозможно.

В этой связи актуальной проблемой проектирования и эксплуатации биометрических технологий является учет влияния искажающих факторов. Проблема особенно актуальна в случаях, когда искажения невозможно подавить штатными средствами биометрической системы.

Далее статья организована следующим образом. Раздел 2 посвящен оценке влияния искажающих факторов при дактилоскопической идентификации. В разд. 3 изложен подход к адаптации биометрической системы к искажениям. В разд. 4 приведены результаты экспериментов. В разд. 5 рассмотрены отдельные вопросы практического применения изложенного подхода. Основные выводы представлены в заключении.

*Работа поддержана грантами РФФИ (проект 07-07-00031) и Программой ОНИТ РАН «Информационные технологии и методы анализа сложных систем». Работа выполнена в рамках НОЦ ИПИ РАН — ВМК МГУ «Биометрическая информатика».

[1]Институт проблем информатики Российской академии наук, oushmaev@ipiran.ru

Рис. 1 Процесс создания шаблона

2 Влияние искажающих факторов

В своей методологической основе большинство биометрических технологий являются технологиями распознавания образов. При регистрации человека биометрическая информация оцифровывается в количественные данные, пригодные для дальнейшей идентификации (рис. 1) [6].

На этапе идентификации шаблоны сравниваются. Далее преимущественно используются пороговые методы принятия решения. Если результат сравнения биометрических шаблонов выше некоторого порогового значения, то принимается гипотеза о принадлежности образцов одному человеку. В обратном случае считается, что образцы принадлежат разным людям. Изменяя порог, можно управлять соотношением ошибок 1-го и 2-го рода, FRR (False Rejection Rate) и FAR (False Accptance Rate) (рис. 2).

Соответственно, с точки зрения ошибок принятия решения об идентификации биометрическая система характеризуется двумя распределениями: результаты сравнений в «своих» сравнениях и результаты сравнения в «чужих» сравнениях.

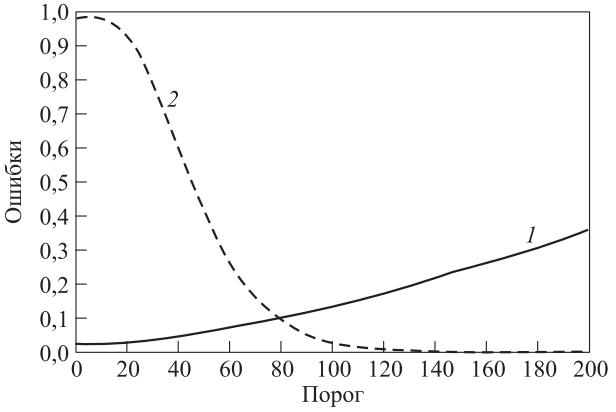


Рис. 2 Пороговое принятие решения: *1* — FRR, *2* — FAR

Под воздействием внешних систематических факторов распределения могут претерпевать определенные изменения. На рис. 3 приведен пример изменения распределений под влиянием эластичных деформаций отпечатков пальцев [7], достаточно распространенного и значимого искажающего фактора в дактилоскопической идентификации.

Как видно из рис. 3, искажающий фактор вносит смещения и изменения дисперсии, при этом общий вид распределений не изменяется. Аналогичная ситуация наблюдается для более широкого класса биометрик и искажающих факторов. Среди публично доступных данных по проблеме следует выделить протоколы тестирований (NIST FRVT, NIST PVT, FpVTE[1]) [10, 11].

В то же время незначительное на первый взгляд различие приводит к существенным изменениям в качестве распознавания (рис. 4) и в зависимостях FAR и FRR от порога (рис. 5). Такие изменения необходимо учитывать при проектировании и эксплуатации биометрической системы.

Необходимость учета изменения ошибок связана с типичной логикой принятия решения о биометрической идентификации. Согласно bioAPI (Biometric Application Programming Interface), основному биометрическому стандарту на API биометрических библиотек, при идентификации предъявляемый образец последовательно сравнивается с хранимыми образцами. Полученный результат — мера сходства биометрических образцов — приводится к единой шкале по таблице ошибок 1-го и 2-го рода (рис. 6). Ответ в таком случае

[1]NIST — National Institute of Standards and Technology; FRVT — Face Recognition Vendor Test; PVT — PHIGS Validation Tests; PHIGS — Programmer's Hierarchical Interactive Graphics System; FpVTE — Fingerprint Vendor Technology Evaluation.

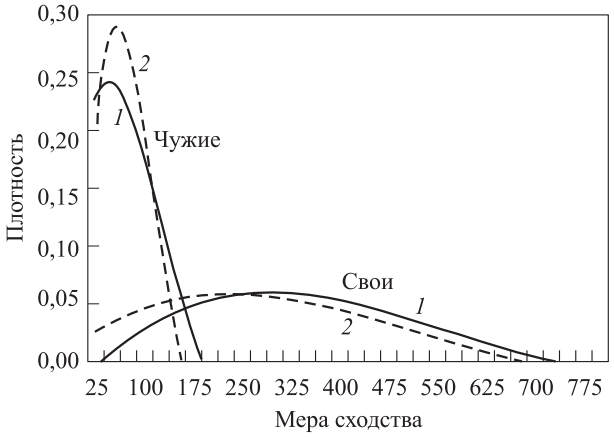


Рис. 3 Изменение распределений: *1* — без компенсации деформаций, *2* — с компенсацией деформаций

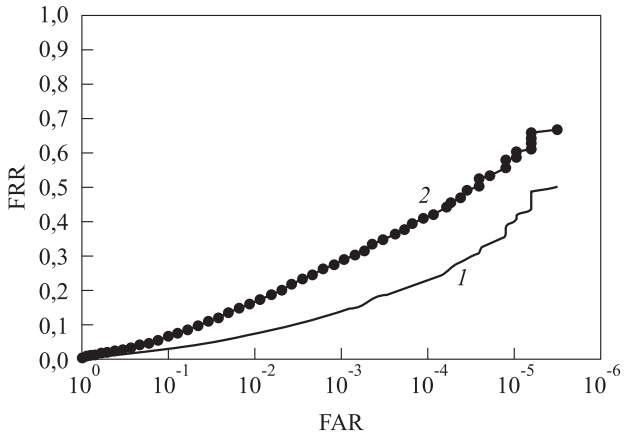
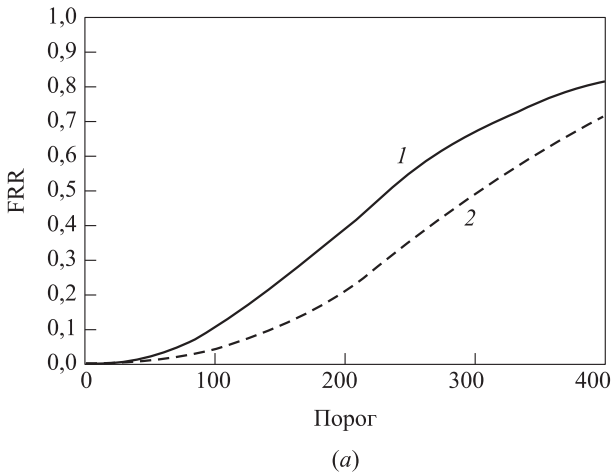


Рис. 4 Изменение качества распознавания: *1* — с учетом деформаций, *2* — без учета деформаций



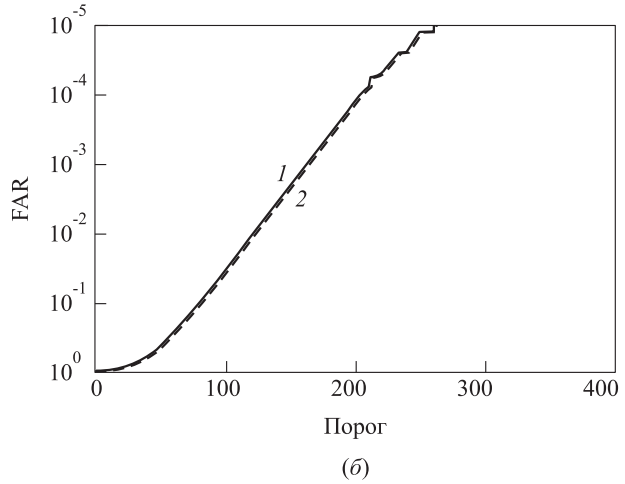


Рис. 5 Изменение ошибок распознавания в зависимости от порога: *1* — без компенсации деформаций, *2* — с компенсацией деформаций (исходная шкала)

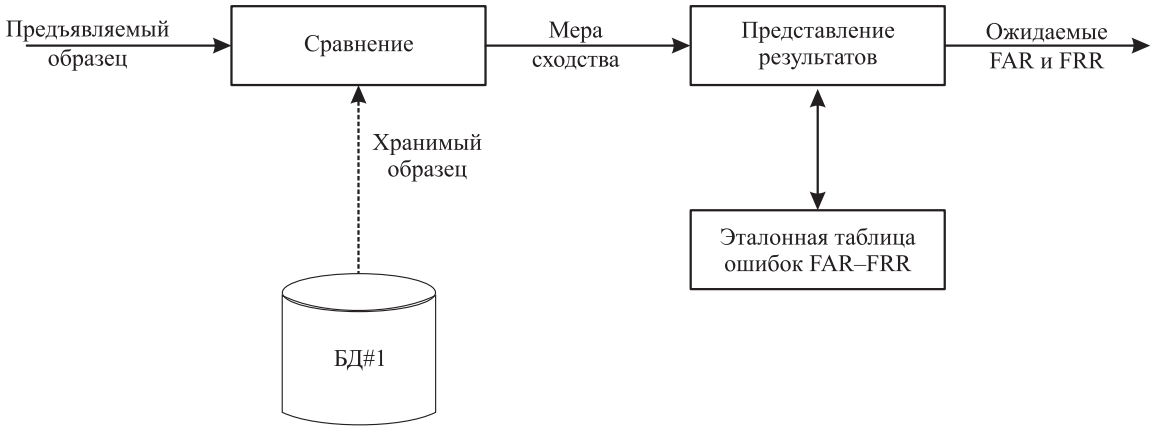


Рис. 6 Процесс биометрической идентификации

может быть сформулирован как точные значения ошибок идентификации.

Под влиянием сильных внешних искажений однородность биометрической базы нарушается, что приводит к невозможности адекватно оценить ожидаемые ошибки распознавания. Как видно из рис. 5, различия в ожидаемых уровнях ошибок могут быть очень значительными.

3 Оценка искажающих факторов

Для количественной оценки влияния искажающего фактора изучим моменты распределений результатов сравнения: математическое ожидание, дисперсию, моменты больших порядков. Чтобы изолировать влияние математического ожидания и дисперсии, заменим моменты больших порядков на производные статистики

$$\gamma_k = \mathbf{E}\left[\left(\frac{s-\mu}{\sigma}\right)^k\right], \qquad (1)$$

где μ — математическое ожидание, σ — среднеквадратическое отклонение.

Как видно из (1), статистики γ_k инварианты относительно линейной замены аргументов.

Данные по изменению статистик искажающего фактора приведены в табл. 1.

Анализ данных табл. 1 показывает, что при грубой оценке искажающего фактора можно ограничиться моментами первых двух порядков. Слабая вариация статистик (1) высших порядков подтверждает предположение о том, что характер распределения не претерпевает принципиальных изменений.

Обозначим первые моменты распределений в своих и чужих сравнениях через (μ_g, σ_g^2) и (μ_i, σ_i^2). Смещения моментов, сопряженные с воздействием искажающего фактора, обозначим через $(\Delta m_g, \Delta s_g^2)$ и $(\Delta m_i, \Delta s_i^2)$ соответственно.

Такая оценка искажений на уровне смещений моментов первых двух порядков имеет явное практическое преимущество. Пусть имеется эталонная таблица соответствий ошибок 1-го рода $\mathrm{FRR}(x)$ и 2-го рода $\mathrm{FAR}(x)$ в зависимости от порога x принятия решения. Тогда скорректированные с учетом искажающего фактора ошибки распознавания определяются следующим образом:

$$\mathrm{FRR}^c(x) = \mathrm{FRR}\left(\alpha_{\mathrm{FRR}} x + \beta_{\mathrm{FRR}}\right);$$
$$\mathrm{FAR}^c(x) = \mathrm{FAR}\left(\alpha_{\mathrm{FAR}} x + \beta_{\mathrm{FAR}}\right),$$

где коэффициенты α и β являются коэффициентами линейного преобразования

$$\frac{x - \Delta m - \mu}{\sqrt{\sigma^2 + \Delta s^2}} \to \frac{x-\mu}{\sigma},$$

которые прямо вычисляются через моменты по следующим формулам:

$$\alpha_{\mathrm{FAR}} = \frac{\sigma_i}{\sqrt{\sigma_i^2 + \Delta s_i^2}};$$

$$\beta_{\mathrm{FAR}} = \mu_i - \frac{\sigma_i}{\sqrt{\sigma_i^2 + \Delta s_i^2}}\left(\mu_i + \Delta m_i\right);$$

$$\alpha_{\mathrm{FRR}} = \frac{\sigma_g}{\sqrt{\sigma_g^2 + \Delta s_g^2}};$$

$$\beta_{\mathrm{FRR}} = \mu_g - \frac{\sigma_i}{\sqrt{\sigma_g^2 + \Delta s_g^2}}\left(\mu_g + \Delta m_g\right).$$

В условиях воздействия нескольких факторов с параметрами $(\Delta m_{jg}, \Delta s_{jg}^2)$, $(\Delta m_{ji}, \Delta s_{ji}^2)$ итоговое воздействие $(\Delta m_g, \Delta s_g^2)$ и $(\Delta m_i, \Delta s_i^2)$ получается суммой отдельных факторов:

$$\Delta m_g = \sum_j \Delta m_{jg}; \quad \Delta s_g^2 = \sum_j \Delta s_{jg}^2;$$
$$\Delta m_i = \sum_j \Delta m_{ji}; \quad \Delta s_i^2 = \sum_j \Delta s_{ji}^2.$$

Сдвиги ошибок $\mathrm{FRR}(x)$ и $\mathrm{FAR}(x)$ вычисляются по всей совокупности факторов. Формально можно ожидать, что некоторые факторы уменьшат дисперсию, в таком случае величина Δs^2 окажется отрицательной. Примером подобного фактора является, например, уменьшение окна сканирования. Уменьшение дисперсии в таком случае обусловлено общим снижением информативности получаемых биометрических образцов.

Таблица 1 Статистики влияния деформаций отпечатков пальцев

Фактор	μ	σ	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
С компенсацией деформации (свои)	280	143	0,41	2,73	3,33	14,41	35,71	147,52
Без компенсации деформации (свои)	265	156	0,84	3,12	4,87	18,36	49,45	208,34
С компенсацией деформации (чужие)	48	24	1,07	4,72	15,80	77,22	430,55	2869,27
Без компенсации деформации (чужие)	46	24	1,14	4,96	16,35	83,01	470,90	3027,34

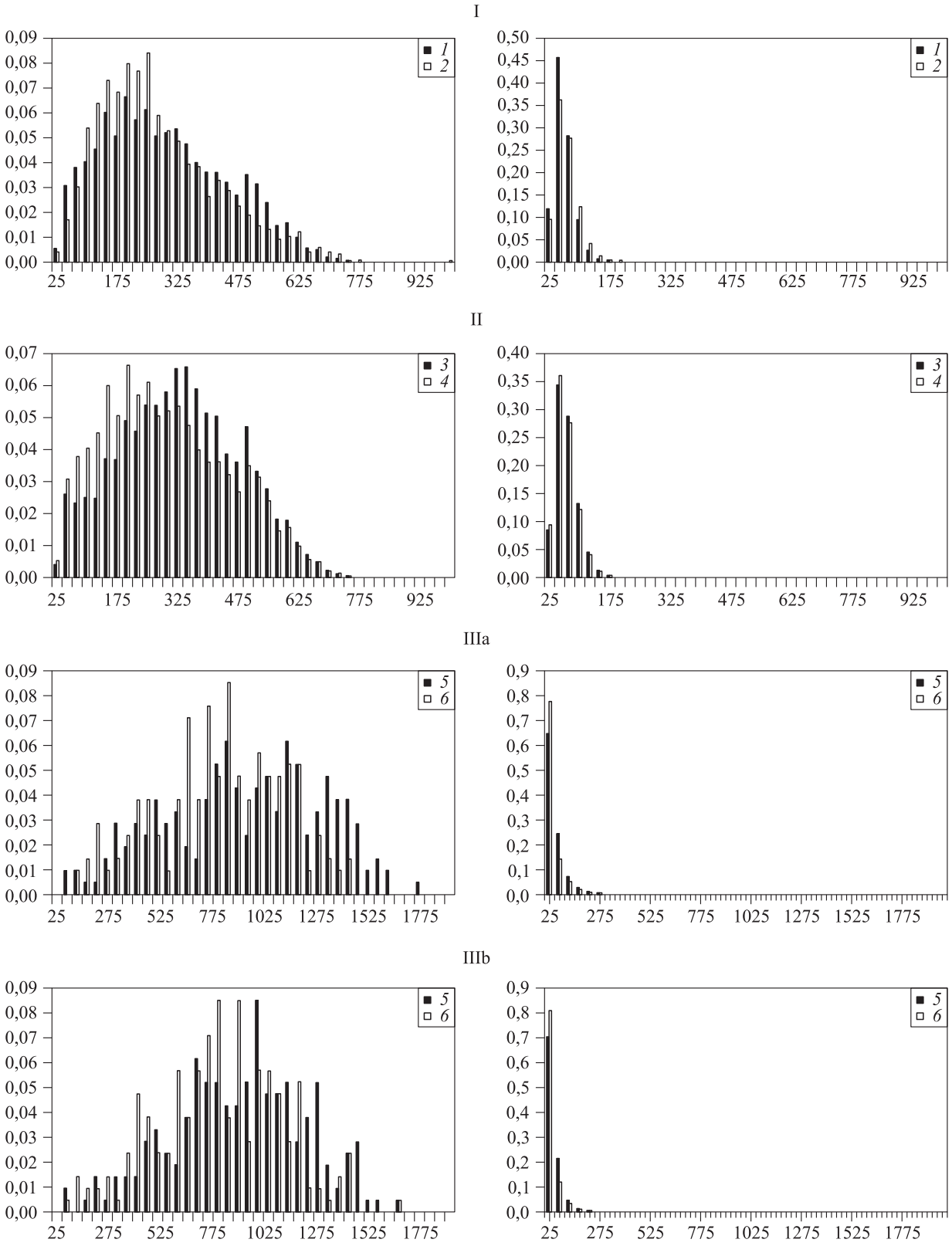


Рис. 7 Гистограммы распределений (слева — распределения в «своих» сравнениях, справа — распределения в «чужих» сравнениях): оптический (*1*) и емкостной (*2*) сканеры; с компенсацией (*3*) и без компенсации (*4*) деформаций; контрольные (*5*) и откатанные (*6*) отпечатки

Таблица 2 Численные оценки моментов

Эксперименты	μ_g	σ_g^2	μ_i	σ_i^2	Δm_g	Δs_g^2	Δm_i	Δs_i^2
I	265	24 366	46	580	−16	4 039	−2	309
II	265	24 366	46	580	15	−3 887	2	21
IIIa	934	168 434	40	3362	−126	−56 192	−14	−139
IIIb	902	134 658	32	2765	−110	−47 744	−11	−560

4 Эксперименты

В данном разделе рассмотрим факторы, влияющие на качество распознавания отпечатков пальцев. Выделим следующие:

- шумы;
- внешние условия (температура, влажность);
- временной лаг между регистрацией и идентификацией;
- деформации отпечатков пальцев;
- характеристики типичного пользователя системы;
- различия в способе получения отпечатков пальцев (например, след отпечатка, «живое» сканирование и оцифровка бумажных носителей).

Можно предположить, что перечисленные факторы по своему воздействию независимы. Поэтому можно проводить их оценку по отдельности.

Базовой технологией распознавания пальцев выберем технологию AMIS (Automated Multibiometric Information System) [12], в качестве основного источника биометрических данных — оптический сканер. Приведем результаты следующих экспериментов, которые могут быть повторены на публично доступных биометрических базах:

I. Изменение характера шумов: переход от оптического к емкостному.

II. Учет деформаций: сравнение качества распознавания AMIS с качеством распознавания AMIS, усиленной методами устранения деформаций отпечатков пальцев [12, 13].

III. Различия в способе получения отпечатков: использование для идентификации откатанных отпечатков.

Эксперимент III проведен раздельно для левого и правого больших пальцев (IIIa и IIIb).

В качестве тестовых баз использовались: FVC2002 (эксперименты I и II) [14] и база SD 29 (эксперименты III) [15]. На рис. 7 приведены гистограммы распределений с учетом влияния искажающих факторов.

Таблица 3 Поправочные коэффициенты

Эксперименты	α_{FRR}	β_{FRR}	α_{FAR}	β_{FAR}
I	0,926	34,402	0,807	10,460
II	1,091	−40,454	0,982	−1,154
IIIa	1,225	−55,803	1,021	13,445
IIIb	1,244	−83,817	1,030	10,365

Численные результаты оценки моментов искажающих факторов и поправочных коэффициентов приведены в табл. 2 и 3. На рис. 8–10 приведены графики нормальной аппроксимации искажающих факторов.

В частности, из таблиц видно, что влияние искажающих факторов на распределения в чужих сравнениях значительно меньше влияния на распределения в своих сравнениях. Поэтому больший интерес представляет оценка ошибки 1-го рода с учетом искажающего фактора. На рис. 11–13 приведены примеры точности оценки ошибок 1-го рода на основе нормальной аппроксимации искажающего фактора.

Как видно из рис. 11–13, предложенный метод позволяет производить достаточно точное приведение шкалы ошибок к эталонному варианту.

5 Влияние на качество идентификации

Основным практическим применением разработанного подхода к учету искажающих факторов является адаптация процедуры идентификации в биометрической системе при неоднородных исходных данных.

В качестве примера рассмотрим адаптацию дактилоскопической системы при использовании двух источников биометрических данных: откатанных и контрольных отпечатков. Предположим, что на этапе идентификации используются только контрольные отпечатки, а база данных собрана с использованием двух типов сканирования в равных пропорциях. В качестве данных возьмем эксперименты IIIa и IIIb.

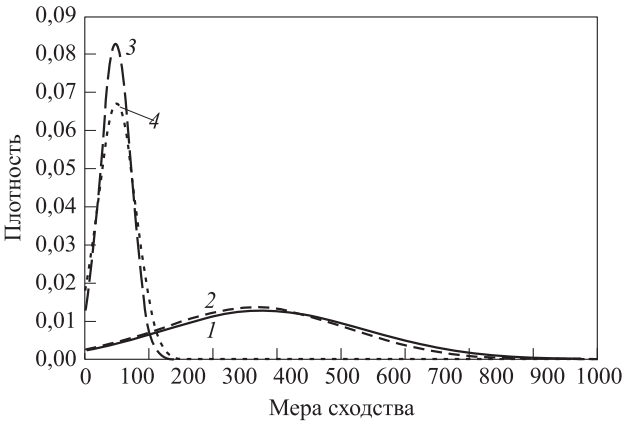


Рис. 8 Нормальная аппроксимация факторов I (изменение типа сканирования): *1* — оптический сканер (свои), *2* — емкостной сканер (свои), *3* — оптический сканер (чужие), *4* — емкостной сканер (чужие)

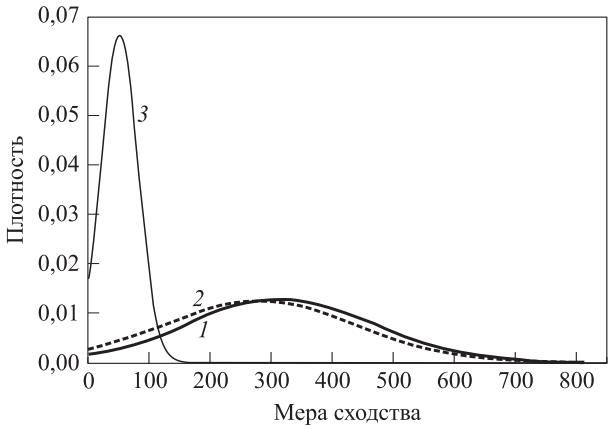


Рис. 9 Нормальная аппроксимация факторов II (учет деформаций): *1* — с компенсацией деформаций, *2* — без компенсации деформаций, *3* — чужие сравнения

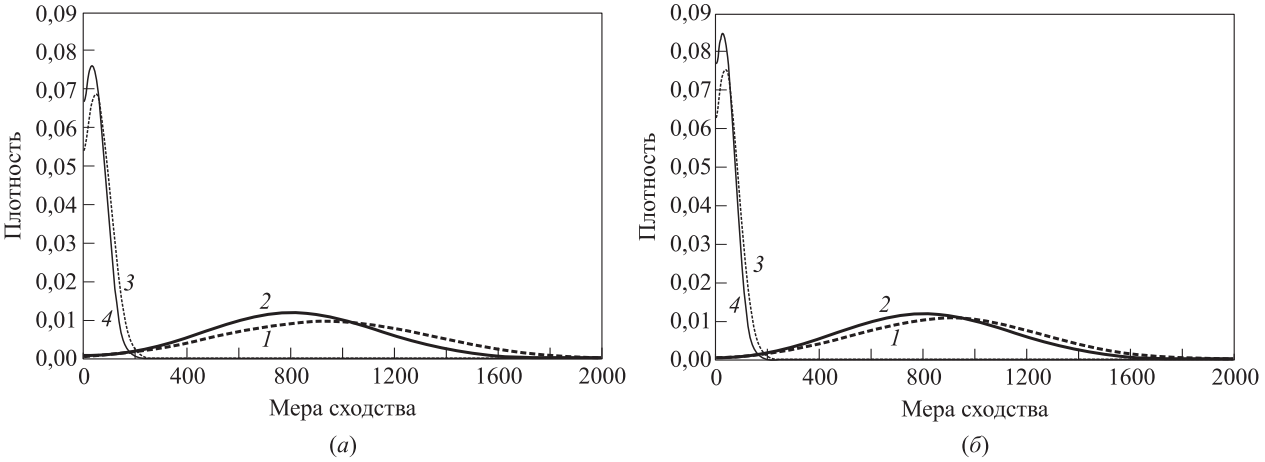


Рис. 10 Нормальная аппроксимация факторов IIIa (переход от сравнения контрольных отпечатков к сравнению контрольных с откатанными, левый большой палец) (*а*) и IIIb (переход от сравнения контрольных отпечатков к сравнению контрольных с откатанными, правый большой палец) (*б*): *1* — контрольные отпечатки (свои), *2* — откатанные отпечатки (свои), *3* — контрольные отпечатки (чужие), *4* — откатанные отпечатки (чужие)

Возможны два основных сценария эксплуатации системы с неоднородностями такого сорта.

Первый заключается в наличии двух функций сравнения: «контрольные с контрольными» и «контрольные с откатанными». Соответственно, учет различного характера биометрической информации будет осуществлен силами производителя алгоритмов распознавания. Но для этого необходима нестандартная реализация программного интерфейса биометрических библиотек. К тому же такой подход требует постоянной модернизации библиотек при каждом новом искажающем факторе.

Второй сценарий заключается в работе с одной функцией сравнения (рис. 6), но с учетом предложенного подхода к учету искажений.

Рассмотрим два варианта принятия решения:

(1) смешивание двух баз без учета сдвигов в ошибках распознавания;

(2) приведение к единой шкале по ошибке 1-го рода или по ошибке 2-го рода.

Основным критерием качества разработанной методики приведения к единой шкале является точность оценки ошибок 1-го и 2-го рода, которая продемонстрирована в разд. 3.

Дополнительным эффектом от учета искажений является улучшение качества идентификации за счет адекватного устранения расслоения базы по неоднородности искажающих воздействий.

На рис. 14 приведены сравнительные графики ошибок 1-го и 2-го рода (для простого смешива-

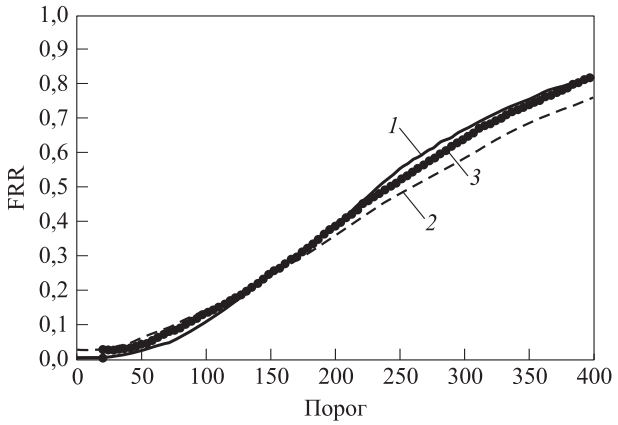


Рис. 11 Пример приведения шкалы ошибки 1-го рода в эксперименте I: *1* — оптический сканер, *2* — емкостной сканер (исходная шкала), *3* — емкостной сканер (приведенная шкала)

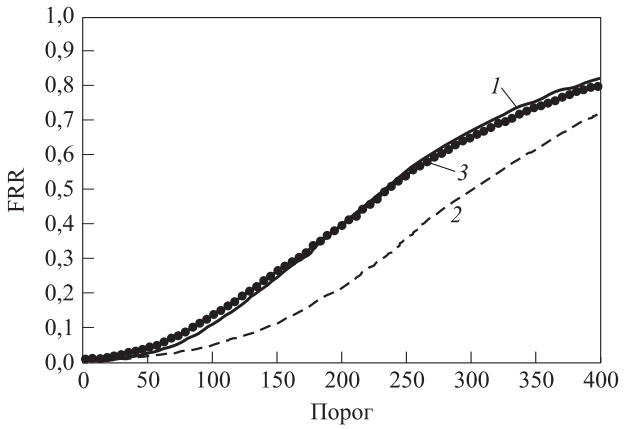


Рис. 12 Пример приведения шкалы ошибки 1-го рода в эксперименте II: *1* — без компенсации деформаций, *2* — с компенсацией деформаций (исходная шкала), *3* — с компенсацией деформаций (приведенная шкала)

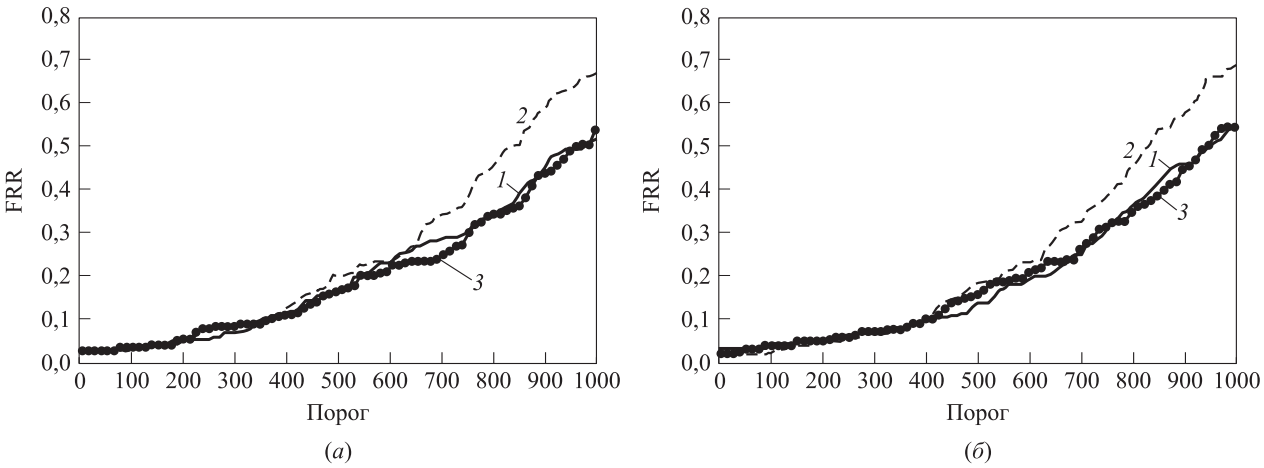


Рис. 13 Пример приведения шкалы ошибки 1-го рода в экспериментах IIIa (*а*) и IIIb (*б*): *1* — контрольные, *2* — откатанные (исходная шкала), *3* — откатанные (приведенная шкала) отпечатки

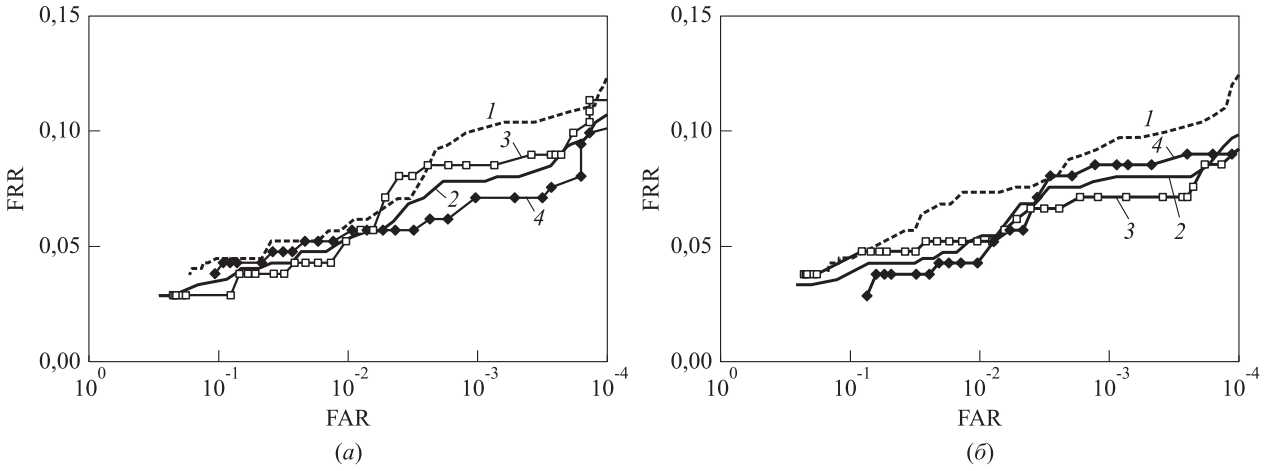


Рис. 14 Изменение качества идентификации в экспериментах IIIa (*а*) и IIIb (*б*): *1* — исходная шкала, *2* — скорректированная шкала, *3* — контрольные отпечатки, *4* — откатанные отпечатки

ния результатов сравнения и корректировки шкал по ошибке 2-го рода). Оценки идентификации на смешанном массиве зависят от доли контрольных и откатанных отпечатков в общем потоке обращений к биометрической системе. В данном модельном примере были взяты значения 50% для каждого класса.

Как видно из рисунков, корректировка шкалы позволяет прийти к оптимальному решению (примерно среднее арифметическое по ошибкам). В то же время использование ненормированной шкалы ухудшает качество распознавания ниже уровня худшего из класса.

6 Заключение

В статье изложены подходы к адаптации биометрических систем к воздействию искажающих факторов. Предложен подход к декомпозиции и учету искажающих факторов на основе метода нормальной аппроксимации. Преимуществами подхода являются:

- учет каждого фактора в отдельности;
- аддитивность характеристик искажающего фактора.

Разработанные на основе предложенного подхода методы и технологии адаптации к искажающим факторам позволяют в значительной степени компенсировать влияние искажений в задаче дактилоскопической идентификации.

Направлением дальнейших исследований является анализ искажающих факторов для других биометрических методов, таких как изображение лица, изображение радужной оболочки глаза, рукописный почерк и т. д.

Литература

1. *Ушмаев О. С.* Применение биометрии в аэропортах // Biometrics TTS 2007. 22 ноября 2007 г. http://www.dancom.ru/rus/AIA/Archive/RUVII_BioLinkSolutions_BiometricsInAirports.pdf.
2. *Ушмаев О. С.* Реализация концепции многофакторной биометрической идентификации в правоохранительных системах // Интерполитех-2007. http://www.dancom.ru/rus/AIA/Archive/RUVI_BioLinkSolutions_MultimodalBiometricsConcept.pdf.
3. *Синицын И. Н., Губин А. В., Ушмаев О. С.* Метрологические и биометрические технологии и системы // История науки и техники, 2008. № 7. С. 41–44.
4. *Ушмаев О. С.* Сервисно-ориентированный подход к разработке мультибиометрических технологий // Информатика и её применения, 2008. Т. 2. Вып. 3. С. 41–53.
5. *Ушмаев О. С.* Концепция мультибиометрической идентификации в информационно-аналитических системах // Паспортные и правоохранительные системы-2008. Интерполитех-2008. http://www.dancom.ru/rus/AIA/Archive/RUXIX-IPIRAN-Ushmaev-MultimodalBiometricsFramework.ppt.
6. *Bolle R. M., Connell J. H., Pankanti S., Ratha N. K., Senior A. W.* Guide to biometrics. — New-York: Springer-Verlag, 2003.
7. *Novikov S. O., Ushmaev O. S.* Efficiency of elastic deformation registration for fingerprint identification // 7th Conference (International) on Pattern Recognition and Image Analysis: New Information Technologies (PRIA-7-2004) Proceedings. St. Petersburg, October 18–23, 2004. Vol. III. — St. Petersburg: SPbETU, 2004. P. 833–836.
8. Wayman J., Jain A., Maltoni D., Maio D., eds. Biometric systems: Technology, design and performance evaluation. — London: Springer-Verlag, 2004.
9. *Dessimoz D., Champod C., Richiadi J., Drygajlo A.* Multimodal biometrics for identity documents. Research Report, PFS 314-08.05. UNIL, June 2006.
10. Face recognition vendor test. http://www.frvt.org.
11. Fingerprint vendor technology evaluation. http://fpvte.nist.gov.
12. *Ушмаев О. С., Босов А. В.* Реализация концепции многофакторной биометрической идентификации в интегрированных аналитических системах // Системы высокой доступности, 2007. Т. 3. Вып. 4. С. 13–23.
13. *Ushmaev O. S., Novikov S. O.* Integral criteria for large-scale multiple fingerprint solutions / Biometric Technology for Human Identification // Eds. A. K. Jain, and N. K. Ratha. Proceedings of SPIE. Vol. 5404. — SPIE, Bellingham, WA, 2004. P. 534–543.
14. Second fingerprint verification competition. FVC 2002. http://bias.csr.unibo.it/fvc2002/
15. NIST SD 29. NIST Special Database 29 "Plain and Rolled Images from Paired Fingerprint Cards."

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 34–42

ПРИБЛИЖЕННЫЙ МЕТОД ВЫЧИСЛЕНИЯ ХАРАКТЕРИСТИК УЗЛА ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ С ПОВТОРНЫМИ ПЕРЕДАЧАМИ

Я. М. Агаларов[1]

Аннотация: Рассмотрена модель узла коммутации пакетов с повторными передачами для двух схем распределения буферной памяти: полнодоступной и полного разделения. Предложен приближенный метод вычисления интенсивностей потоков и вероятностей блокировок узла. Получены необходимые и достаточные условия существования и единственности решения уравнения для потоков в узле при установившемся режиме работы и доказана сходимость итерационного метода решения указанного уравнения.

Ключевые слова: узел коммутации пакетов; буферная память; повторные передачи; вероятности блокировок; итерационный метод

1 Введение

Одной из основных задач предварительного анализа телекоммуникационных сетей коммутации пакетов с ограниченной буферной памятью является расчет характеристик потоков и вероятностей блокировок в узлах связи. Важность указанных характеристик определяется тем, что от их значений существенным образом зависят другие основные показатели сети (пропускная способность, задержки пакетов и др.).

Существует множество различных моделей узлов коммутации пакетов и методов их расчета (см., например, [1–6]). Для моделей, рассматривающих узел с ограниченной буферной памятью как систему массового обслуживания (СМО) типа $\begin{matrix} M \\ \lambda \end{matrix} \left| \begin{matrix} M \\ \lambda \end{matrix} \right| \overline{m} | N$ или $|PH|PH|1|r$, в предположении отсутствия повторных передач пакетов получены точные методы вычисления характеристик узлов [1, 3, 4, 6]. Приближенные методы расчета узлов, учитывающие повторные попытки передачи, используют модели типа $|PH|PH|1|r$ или $\begin{matrix} M \\ \lambda \end{matrix} \left| \begin{matrix} M \\ \lambda \end{matrix} \right| 1 | N$ и являются итерационными [2, 3, 5, 7]. Для моделей типа $BMAP|PH|1$, $M|G|1|r$ и $MAP|(PH,PH)|1$ с повторными заявками получены точные методы вычисления характеристик (например, в работах [8–10]), которые также могут быть использованы при расчете узлов.

Ниже будут рассмотрены модели узла коммутации пакетов с повторными передачами для двух схем распределения буферной памяти: с полнодоступными буферами и с полным разделением буферной памяти. Предлагается приближенный метод расчета характеристик, который в качестве модели узла использует СМО типа $\begin{matrix} M \\ \lambda \end{matrix} \left| \begin{matrix} M \\ \lambda \end{matrix} \right| \overline{m} | N$ с повторными заявками. Доказаны утверждения о достаточных и необходимых условиях существования и единственности решения уравнения для вероятности блокировки в установившемся режиме работы и сходимости предлагаемого итерационного метода.

2 Модель узла

Математическая модель узла представляется в виде СМО с ограниченной буферной памятью и различными потоками заявок, каждая из которых требует обслуживания только на одной из многоканальных линий связи.

Пусть $0 < N < \infty$ — число мест хранения в буферной памяти, u — узел связи, v — линия связи, Ω_u^+ — множество исходящих из узла u линий, c_v — канальная емкость линии v. Поток заявок, требующих обслуживания на линии v, назовем v-потоком, заявки этого потока — v-заявками.

Пусть выполняются следующие предположения:

1. Места в буферной памяти распределяются согласно одной из двух схем:

   (i) полнодоступная схема — каждое свободное место хранения доступно любой заявке;

[1]Институт проблем информатики Российской академии наук, agglar@yandex.ru

(ii) схема полного разделения памяти — v-заявкам доступны всего N_v мест, где $\sum\limits_{v\in\Omega_u^+} N_v = N$.

2. Если в момент поступления v-заявки в буферной памяти есть доступное свободное место, то она сразу занимает это место. Если в момент поступления v-заявки в системе нет свободного доступного места хранения, то поступившая заявка через некоторое время повторно поступает на систему, оставаясь v-заявкой.
3. Интенсивности первичных потоков v-заявок — заданные величины $0 < \Lambda_v < \infty$, $v \in \Omega_u^+$. Суммарные потоки первичных и повторных v-заявок являются независимыми в совокупности пуассоновскими потоками. Для обслуживания v-заявки требуется одновременно одно место хранения и один канал типа v, $v \in \Omega_u^+$.
4. Первичные нагрузки — реализуемые, т. е. в данном случае интенсивности входных первичных потоков равны интенсивностям выходных потоков выполненных заявок.
5. Принятые в СМО v-заявки обслуживаются линией v в порядке поступления.
6. Время занятия канала v-заявкой — экспоненциально распределенная случайная величина с параметром $0 < \mu_v < \infty$, $v \in \Omega_u^+$, независимая от других случайных событий в узле.
7. Выполненная v-заявка с вероятностью B_v повторяется через заданное время τ_v (тайм-аут) и с вероятностью $1 - B_v$ покидает систему через время t_v навсегда, сразу освободив занятый канал и место буферной памяти.

Будем говорить, что узел блокирован для v-заявки, если в буферной памяти отсутствует доступное место хранения. Ставится задача вычисления вероятностей блокировок и интенсивностей потоков в узле.

3 Вычисление вероятности блокировки и интенсивностей потоков

Пусть Λ_v^* — интенсивность суммарного потока внешних заявок, требующих передачи по линии v, π_v — вероятность блокировки узла для заявок, требующих передачи по исходящей из узла линии v.

Пусть в узле используется полнодоступная схема распределения буферной памяти. Тогда, как следует из описания модели, $\pi_v = \pi_{v'}$, $v, v' \in \Omega_u^+$, и для интенсивностей Λ_v^*, $v \in \Omega_u^+$, справедливы соотношения:

$$\Lambda_v^* = \frac{\Lambda_v}{1-\pi}\,,$$

где $\pi = \pi_v, v \in \Omega_u^+$.

Пусть $\overline{k} = \{\overline{k}_v,\ v \in \Omega_u^+\}$ — состояние буферной памяти узла, $\overline{k}_v = (k_v, k_v', k_v'')$; k_v — число v-заявок в буферной памяти, ожидающих выполнения линией v; k_v' — число v-заявок в буферной памяти, ожидающих тайм-аут и неуспешно переданных в последующий узел; k_v'' — число v-заявок в буферной памяти, успешно переданных в последующий узел и ожидающих потверждения; $A_m = \left\{\overline{k} : \sum\limits_{v\in\Omega_u^+}(k_v + k_v' + k_v'') = m\right\}$ — множество различных состояний, при которых в памяти узла занято ровно m буферов. Тогда с учетом введенных выше обозначений и предположений для вероятности блокировки узла можно написать формулу [1, 6]:

$$\pi = \frac{1}{G_N}\sum_{\overline{k}\in A_N} p\left(\overline{k}, \overline{\rho}^*\right)\,, \tag{1}$$

где

$$p(\overline{k}, \overline{\rho}^*) = \prod_{v\in\Omega_u^+} z_v(\pi, \rho_v, k_v, k_v', k_v'')\,; \tag{2}$$

$$z_v(\pi, \rho_v, k_v, k_v', k_v'') = \begin{cases} \dfrac{\rho_v'^{*k_v'}}{k_v'!}\,\dfrac{\rho_v''^{*k_v''}}{k_v''!}\,\dfrac{\rho_v^{*k_v}}{k_v!} & \text{при } k_v < c_v\,, \\ \dfrac{\rho_v'^{*k_v'}}{k_v'!}\,\dfrac{\rho_v''^{*k_v''}}{k_v''!}\,\dfrac{\rho_v^{*k_v}}{c_v!c_v^{k_v-c_v}} & \text{при } k_v \geq c_v\,; \end{cases} \tag{3}$$

$$G_N = \sum_{m=0}^{N}\sum_{\overline{k}\in A_m} p(\overline{k}, \overline{\rho}^*)\,; \tag{4}$$

$$\overline{\rho}^* = \{\rho_v^*,\ v \in \Omega_u^+\}\,; \tag{5}$$

$$\rho_v^* = \frac{\rho_v}{1-\pi}\,; \quad \rho_v = \frac{\Lambda_v}{\mu_v(1-B_v)}\,; \tag{6}$$

$$\rho_v'^* = \rho_v^*\mu_v\tau_v B_v\,; \quad \rho_v''^* = p_v^*\mu_v t_v, \quad v \in \Omega_u^+\,. \tag{7}$$

Переобозначив $1 - \pi$ через y, выражение в правой части равенства (2) — через $p_{\overline{k}}(\overline{\rho}, y)$, выражение в правой части равенства (4) — через $g_N(\overline{\rho}, y)$, а выражение в правой части равенства (1) — через $1 - q_N(\overline{\rho}, y)$, где $\overline{\rho} = (\rho_v,\ v \in \Omega_u^+)$, $\rho_v = \rho_v^* y = \Lambda_v/(\mu_v(1-B_v))$, $v \in \Omega_u^+$, получим нелинейное уравнение относительно неизвестной переменной y:

$$y = q_N(\overline{\rho}, y)\,. \tag{8}$$

Решим уравнение (8). Как следует из (2)–(7), верно равенство

$$q_N(\overline{\rho}, y) = \frac{g_{N-1}(\overline{\rho}, y)}{g_N(\overline{\rho}, y)}\,. \tag{9}$$

Введем функцию $d_n(\overline{\rho}, y)$ среднего числа заявок в узле с буферной памятью емкости $n \geq 0$:

$$d_n(\overline{\rho}, y) = \frac{1}{g_n(\overline{\rho}, y)} \sum_{m=0}^{n} m \sum_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)\,.$$

Заметим, что g_n, d_n и q_n, $n \geq 0$, — непрерывно-дифференцируемые функции по $y \in (0,\, 1]$. Взяв производную функции g_n по y, из (2)–(7) получим

$$\frac{\partial g_n(\overline{\rho}, y)}{\partial y} = \\ = -\sum_{m=0}^{n} m \sum_{\overline{k} \in A_m} \frac{\prod\limits_{v \in \Omega_u^+} z_n(0, \rho_v, k_v, k_v', k_v'')}{y^{m+1}} = \\ = -\frac{1}{y}\, g_n(\overline{\rho}, y) d_n(\overline{\rho}, y)\,. \quad (10)$$

Взяв производную функции q_N по y, из (9) и (10) получим

$$\frac{\partial q_N(\overline{\rho}, y)}{\partial y} = \frac{q_N(\overline{\rho}, y)}{y} \left[d_N(\overline{\rho}, y) - d_{N-1}(\overline{\rho}, y)\right]\,. \quad (11)$$

Докажем несколько утверждений о свойствах функции $q_N(\overline{\rho}, y)$.

Утверждение 1. *Справедливы неравенства*

$$0 < d_{n+1}(\overline{\rho}, y) - d_n(\overline{\rho}, y) < 1\,, \\ y \in (0,\, 1]\,,\ n \geq 0\,. \quad (12)$$

Доказательство. Подставив выражение для функции $d_n(\overline{\rho}, y)$ и проведя преобразования, получим

$$d_{n+1}(\overline{\rho}, y) - d_n(\overline{\rho}, y) = \frac{\sum\limits_{m=0}^{n+1} m \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)}{\sum\limits_{m=0}^{n+1} \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)} - \\ - \frac{\sum\limits_{m=0}^{n} m \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)}{\sum\limits_{m=0}^{n} \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)} = \\ = \frac{\sum\limits_{m=1}^{n} m \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y) + (n+1) \sum\limits_{\overline{k} \in A_{n+1}} p_{\overline{k}}(\overline{\rho}, y)}{\sum\limits_{m=0}^{n} \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y) + \sum\limits_{\overline{k} \in A_{n+1}} p_{\overline{k}}(\overline{\rho}, y)} - \\ - \frac{\sum\limits_{m=0}^{n} m \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)}{\sum\limits_{m=0}^{n} \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)} = \\ = \frac{(n+1) \sum\limits_{\overline{k} \in A_{n+1}} p_{\overline{k}}(\overline{\rho}, y) g_n(\overline{\rho}, y)}{g_{n+1}(\overline{\rho}, y) g_n(\overline{\rho}, y)} - \\ - \frac{\sum\limits_{\overline{k} \in A_{n+1}} p_{\overline{k}}(\overline{\rho}, y) \sum\limits_{m=0}^{n} m \sum\limits_{\overline{k} \in A_m} p_{\overline{k}}(\overline{\rho}, y)}{g_{n+1}(\overline{\rho}, y) g_n(\overline{\rho}, y)} = \\ = \left[1 - q_{n+1}(\overline{\rho}, y)\right] \left[n + 1 - d_n(\overline{\rho}, y)\right]\,. \quad (13)$$

Докажем утверждение 1 методом индукции. При $n = 0$, как следует из (13), имеем

$$d_2(\overline{\rho}, y) - d_1(\overline{\rho}, y) = 1 - q_1(\overline{\rho}, y)\,,$$

т. е. утверждение 1 при $n = 0$ справедливо.

Пусть неравенства (12) справедливы для некоторого $n > 0$. Докажем, что они справедливы и для $n + 1$. Из (13) получаем

$$d_{n+1}(\overline{\rho}, y) - d_n(\overline{\rho}, y) = \\ = \left[1 - q_{n+1}(\overline{\rho}, y)\right] \left[n + 1 - d_n(\overline{\rho}, y)\right] = \\ = \left[1 - 1 - q_{n+1}(\overline{\rho}, y)\right] \left[n - \right. \\ \left. - d_{n-1}(\overline{\rho}, y) + d_{n-1}(\overline{\rho}, y) - d_n(\overline{\rho}, y) + 1\right] = \\ = \left[1 - q_{n+1}(\overline{\rho}, y)\right] \left[n - d_{n-1}(\overline{\rho}, y) - \right. \\ \left. - \left(d_n(\overline{\rho}, y) - d_{n-1}(\overline{\rho}, y)\right) + 1\right] = \\ = \left[1 - q_{n+1}(\overline{\rho}, y)\right] \left[\frac{d_n(\overline{\rho}, y) - d_{n-1}(\overline{\rho}, y)}{1 - q_n(\overline{\rho}, y)} - \right. \\ \left. - \left(d_n(\overline{\rho}, y) - d_{n-1}(\overline{\rho}, y)\right) + 1\right] = \\ = \left[1 - q_{n+1}(\overline{\rho}, y)\right] \left[\left(d_n(\overline{\rho}, y) - \right.\right. \\ \left.\left. - d_{n-1}\left(\overline{\rho}, y\right)\right) \frac{q_n(\overline{\rho}, y)}{1 - q_n(\overline{\rho}, y)} + 1\right]\,.$$

Так как по предположению $d_n(\overline{\rho}, y) - d_{n-1}(\overline{\rho}, y) > 0$, то правая часть последнего равенства больше нуля; следовательно, $d_{n+1}(\overline{\rho}, y) - d_n(\overline{\rho}, y) > 0$.

Продолжив преобразование правой части последнего равенства и учитывая предположение $d_n(\overline{\rho}, y) - d_{n-1}(\overline{\rho}, y) < 1$, получим

$$d_{n+1}((\overline{\rho}, y) - d_n(\overline{\rho}, y) < \\ < \left[1 - q_{n+1}(\overline{\rho}, y)\right] \left(\frac{q_n(\overline{\rho}, y)}{1 - q_n(\overline{\rho}, y)} + 1\right) = \\ = \frac{1 - q_{n+1}(\overline{\rho}, y)}{1 - q_n(\overline{\rho}, y)} < 1\,,$$

так как $0 < q_n(\overline{\rho}, y) < q_{n+1}(\overline{\rho}, y) < 1$, $n > 0$, $y \in (0, 1]$.

Следовательно, утверждение 1 доказано.

Утверждение 2. *$q_N(\overline{\rho}, y)$ — монотонно-возрастающая функция по $y \in (0, 1]$. При этом $0 < q_N(\overline{\rho}, y) \le q_N(\overline{\rho}, 1) < 1$, $y \in (0, 1]$, и $\lim\limits_{y\to 0} q_N(\overline{\rho}, y) = 0$.*

Доказательство. Возрастание функции $q_N(\overline{\rho}, y)$ следует непосредственно из (11) и утверждения 1. Доказательство неравенств в условии утверждения очевидно следует из (9) и вида функции $g_n(\overline{\rho}, y)$, $n \ge 0$. Для предела имеем:

$$\lim_{y\to 0} q_N(\overline{\rho}, y) = \lim_{y\to 0} \frac{g_{N-1}(\overline{\rho}, y)}{g_N(\overline{\rho}, y)} =$$

$$= \lim_{y\to 0} \left(g_{N-1}(\overline{\rho}, y) \Big/ \left(g_{N-1}(\overline{\rho}, y) + \right.\right.$$

$$\left.\left. + \sum_{\overline{k}\in A_N} \prod_{v\in\Omega_u^+} \frac{z_v(0, \rho_v, k_v, k_v', k_v'')}{y^N} \right)\right) =$$

$$= \lim_{y\to 0} \left(y^N g_{N-1}(\overline{\rho}, y) \Big/ \left(y^N g_{N-1}(\overline{\rho}, y) + \right.\right.$$

$$\left.\left. + \sum_{\overline{k}\in A_N} \prod_{v\in\Omega_u^+} z_v(0, \rho_v, k_v, k_v', k_v'') \right)\right) = 0\,.$$

Утверждение 3. *Производная функции $q_N(\overline{\rho}, y)$ по $y \in (0, 1]$ удовлетворяет следующим соотношениям*:

$$\lim_{y\to 0} \frac{\partial q_N(\overline{\rho}, y)}{\partial y} = \frac{\sum\limits_{\overline{k}\in A_{N-1}} p_{\overline{k}}(\overline{\rho}, 1)}{\sum\limits_{\overline{k}\in A_N} p_{\overline{k}}(\overline{\rho}, 1)}\,; \qquad (14)$$

$$\frac{\partial q_N(\overline{\rho}, y)}{\partial y}\Big|_{y=1} < 1\,. \qquad (15)$$

Доказательство. Проведя преобразования функции $q_N(\overline{\rho}, y)$, получим:

$$\lim_{y\to 0} \frac{q_N(\overline{\rho}, y)}{y} =$$

$$= \lim_{y\to 0} \frac{\sum\limits_{m=0}^{N-1} \sum\limits_{\overline{k}\in A_m} \left(\prod\limits_{v\in\Omega_u^+} z_v(0, \rho_v, k_v, k_v', k_v'') \right) \Big/ y^m}{y \sum\limits_{m=0}^{N} \sum\limits_{\overline{k}\in A_m} \left(\prod\limits_{v\in\Omega_u^+} z_v\,(0, \rho_v, k_v, k_v', k_v'') \right) \Big/ y^m} =$$

$$= \lim_{y\to 0} \frac{\sum\limits_{m=0}^{N-1} \sum\limits_{\overline{k}\in A_m} y^{N-1-m} \prod\limits_{v\in\Omega_u^+} z_v(0, \rho_v, k_v, k_v', k_v'')}{\sum\limits_{m=0}^{N} \sum\limits_{\overline{k}\in A_m} y^{N-m} \prod\limits_{v\in\Omega_u^+} z_v(0, \rho_v, k_v, k_v', k_v'')} =$$

$$= \frac{\sum\limits_{\overline{k}\in A_{N-1}} p_{\overline{k}}(\overline{\rho}, 1)}{\sum\limits_{\overline{k}\in A_N} p_{\overline{k}}(\overline{\rho}, 1)}\,.$$

Очевидно, $\lim\limits_{y\to 0} [d_N(\overline{\rho}, y) - d_{N-1}(\overline{\rho}, y)] = 1$, так как $\lim\limits_{y\to 0} d_n(\overline{\rho}, y) = n$, $n > 0$.

Следовательно, учитывая (11), получаем (14). Справедливость (15) непосредственно следует из (11) и утверждения 1.

Утверждение 4. *Пусть $y^* \in (0, 1]$ — решение уравнения* (8). *Тогда*

$$\frac{\partial q_N(\overline{\rho}, y)}{\partial y}\Big|_{y=y^*} < 1\,.$$

Доказательство. Доказательство следует из (11), так как $q_N(\overline{\rho}, y^*)/y^* = 1$.

Утверждение 5. *Уравнение* (8) *имеет решение $y^* \in (0, 1)$ тогда и только тогда, когда*

$$\frac{\sum\limits_{\overline{k}\in A_{N-1}} p_{\overline{k}}(\overline{\rho}, 1)}{\sum\limits_{\overline{k}\in A_N} p_{\overline{k}}(\overline{\rho}, 1)} > 1\,. \qquad (16)$$

Если уравнение (8) *имеет решение $y^* \in (0, 1)$, то оно единственное положительное решение.*

Доказательство. Пусть выполняется неравенство (16). Тогда, как следует из утверждения 3, $\lim\limits_{y\to 0}(\partial q_N(\overline{\rho}, y)/\partial y) > 1$. Кроме того, как следует из утверждения 2, $\lim\limits_{y\to 0} q_N(\overline{\rho}, y) = 0$. Тогда, так как $q_N(\overline{\rho}, y)$ — непрерывно-дифференцируемая функция по $y \in (0, 1]$, существует значение $y' \in (0, 1)$ такое, что $q_N(\overline{\rho}, y) > y$ для всех $y \in (0, y']$ (следует из теоремы о конечном приращении [11]). В то же время, согласно утверждению 2, $q_N(\overline{\rho}, y) < y$ в окрестности точки $y = 1$ (рис. 1, *а*). Следовательно, кривая $x = q_N(\overline{\rho}, y)$ пересекает прямую $x = y$ хотя бы в одной точке $y = y^* \in (0, 1)$, т. е. уравнение (8) имеет хотя бы одно решение $y^* \in (0, 1)$.

Пусть уравнение (8) имеет решение $y^* \in (0, 1)$ и

$$\frac{\sum\limits_{\overline{k}\in A_{N-1}} p_{\overline{k}}(\overline{\rho}, 1)}{\sum\limits_{\overline{k}\in A_N} p_{\overline{k}}(\overline{\rho}, 1)} \le 1\,. \qquad (17)$$

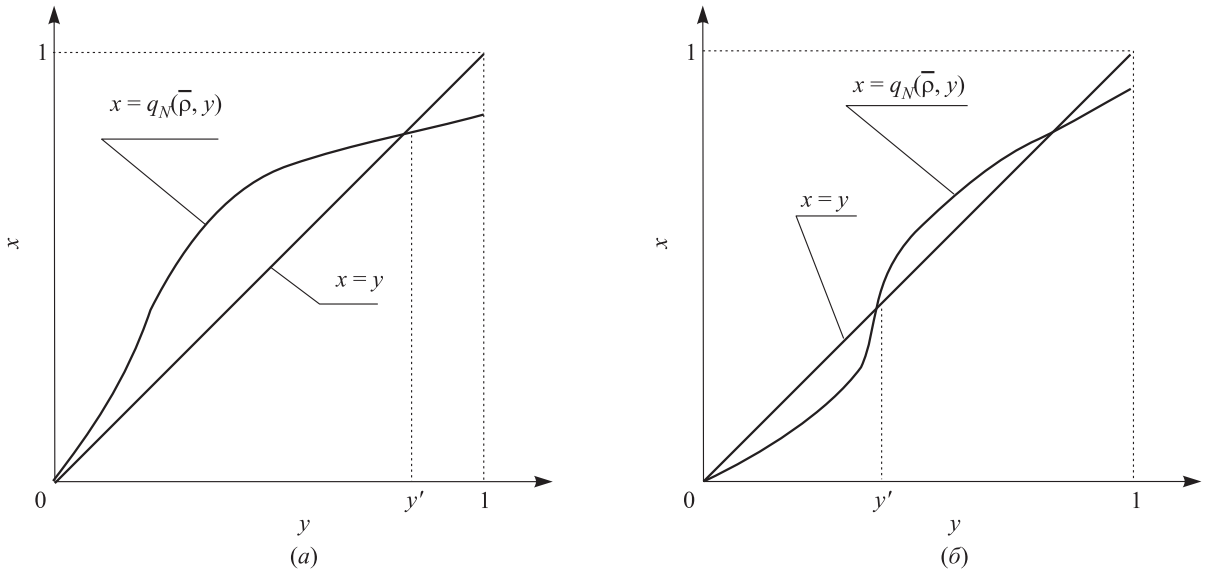


Рис. 1 Примеры кривых $x = q_N(\overline{\rho}, y)$ и $x = y$ (*а*) при существовании решения уравнения (9) и (*б*) при выполнении условий (17)

Тогда из условий утверждений 2 и 3 следует, что уравнение (8) в интервале $(0, 1)$ имеет более одного решения, что может быть только при существовании решения $y' \in (0, 1)$ такого, что в окрестности точки $y = y'$ выполняются неравенства: $q_N(\overline{\rho}, y) < y$ при $y < y'$ и $q_N(\overline{\rho}, y) > y$ при $y > y'$, где y принадлежит указанной окрестности точки y' (рис. 1, *б*). Тогда в точке $y = y'$ производная функции $q_N(\overline{\rho}, y)$ по y больше 1, что противоречит утверждению 4. Следовательно, неравенство (16) справедливо.

Пусть уравнение (8) имеет более одного положительного решения. Рассуждая точно так же, как и выше (в случае выполнения условий (17)), получим противоречие утверждению 4. Следовательно, утверждение 5 справедливо.

Следствие. *Неравенства*

$$\frac{\mu_v c_v(1-B_v)}{\Lambda_v} > 1\,, \quad \frac{1-B_v}{\Lambda_v \tau_v B_v} > 1\,,$$
$$\frac{1-B_v}{\Lambda_v t_v} > 1\,,\ v \in \Omega_u^+\,,$$

являются необходимым условием существования решения уравнения (8).

Доказательство. Пусть $\overline{k}_v$ — это набор $\overline{k}$, у которого $k_v = 0$. Преобразовав левую часть (16), получим

$$\frac{\sum\limits_{\overline{k}\in A_{N-1}} p_{\overline{k}}(\overline{\rho}, 1)}{\sum\limits_{\overline{k}\in A_N} p_{\overline{k}}(\overline{\rho}, 1)} =$$
$$= \frac{\sum\limits_{\overline{k}\in A_{N-1}} \prod\limits_{v\in\Omega_u^+} z_v\,(0, \rho_v, k_v, k'_v, k''_v)}{\sum\limits_{\overline{k}\in A_N} \prod\limits_{v\in\Omega_u^+} z_v\,(0, \rho_v, k_v, k'_v, k''_v)} \le$$
$$\le \left(\frac{\mu_v c_v(1-B_v)}{\Lambda_v} \times \right.$$
$$\times \sum_{k_v=0}^{N-1} \sum_{\overline{k}_v \in A_{N-1-k_v}} z_v\,(0, \rho_v, k_v+1, k'_v, k''_v) \times$$
$$\left. \times \prod_{v'\in\Omega_u^+\setminus v} z'_v\,(0, \rho_v, k_v, k'_v, k''_v) \right) \Bigg/$$
$$\Bigg/ \left(\sum_{k_v=0}^{N-1} \sum_{\overline{k}_v \in A_{N-1-k_v}} z_v\,(0, \rho_v, k_v+1, k'_v, k''_v) \times \right.$$
$$\times \prod_{v'\in\Omega_u^+\setminus v} z_{v'}\,(0, \rho_v, k_v, k', k''_v) +$$
$$+ \sum_{\overline{k}_v \in A_N} z_v\,(0, \rho_v, 0, k'_v, k''_v) \times$$
$$\left. \times \prod_{v'\in\Omega_u^+\setminus v} z_{v'}\,(0, \rho_v, k_v, k'_v, k''_v) \right)\,.$$

Как следует из правой части последнего неравенства, если $\mu_v c_v(1-B_v)/\Lambda_v \le 1$, то она меньше 1. Поэтому, чтобы выполнилось условие (16), необходимо выполнение первого неравенства в условии следствия для каждого $v \in \Omega_u^+$. Точно так же доказывается необходимость выполнения второго и третьего неравенств в условии следствия.

Пусть $y[n]$, $n \ge 0$, последовательность, полученная по формуле $y[n+1] = q_N(\overline{\rho}, y[n])$, $y[0] = 1$.

Утверждение 6. *Пусть $y^* \in (0,\,1)$ — решение уравнения* (8). *Тогда последовательность $y[n]$, $n \ge 0$, сходится к решению y^*.*

Доказательство. Отметим, что $y[1] < y[0]$ (это следует из утверждения 2, так как $y[0] = 1$). Пусть для некоторого $n > 1$ выполняется условие $y[n] < y[n-1]$. Тогда, как следует из утверждения 2, указанное условие выполняется и для $n+1$, т. е. по индукции следует, что последовательность $y[n]$, $n \ge 0$, монотонно убывает.

Пусть для некоторого $n > 0$ $y[n] > y^*$ (существование такого n следует из равенства $y[0] = 1$). Тогда, как следует из утверждения 2, $y[n+1] = q_N(\overline{\rho}, y[n]) > q_N(\overline{\rho}, y^*) = y^*$, т. е. последовательность ограничена снизу величиной y^*. Значит, существует $\lim\limits_{n\to\infty} y[n] = y^0 \ge y^*$. Так как $q_n(\overline{\rho}, y)$ — непрерывная по y функция, то можно написать $\lim\limits_{n\to\infty} q_N(\overline{\rho}, y[n]) = q_N(\overline{\rho}, y^0) = y^0$, т. е. y^0 — решение уравнения (8). Из единственности положительного решения уравнения (8) получаем $y^0 = y^*$.

Пусть в узле используется схема полного разделения буферной памяти. Тогда для интенсивностей Λ_v^*, $v \in \Omega_u^+$, справедливы соотношения:

$$\Lambda_v^* = \frac{\Lambda_v}{1-\pi_v}\,,$$

где $v \in \Omega_u^+$.

Фиксируем произвольную линию сети v. Пусть $\overline{k}_v = (k_v, k_v', k_v'')$ — состояние буферной памяти линии v; k_v, k_v', k_v'' определены выше. Тогда с учетом введенных ранее предположений и обозначений для вероятности блокировки линии справедлива формула [4]:

$$\pi_v = \frac{1}{G_{N_v}} \sum_{k_v = N_v} z_v(\pi_v, \rho_v, \overline{k}_v)\,, \qquad (18)$$

где

$$z_v(\pi_v, \rho_v, \overline{k}_v) = \begin{cases} \dfrac{\rho_v'^{*k_v'}}{k_v'!}\,\dfrac{\rho_v''^{*k_v''}}{k_v''!}\,\dfrac{\rho_v^{*k_v}}{k_v!} & \text{при } k_v < c_v\,, \\ \dfrac{\rho_v'^{*k_v'}}{k_v'!}\,\dfrac{\rho_v''^{*k_v''}}{k_v''!}\,\dfrac{\rho_v^{*k_v}}{c_v!\,c_v^{k_v-c_v}} & \text{при } k_v \ge c_v\,; \end{cases}$$

$$G_{N_v} = \sum_{m=0}^{N_v} z_v(\pi_v, \rho_v, \overline{k}_v)\,;$$

$$\rho_v^* = \frac{\rho_v}{1-\pi_v}\,;$$

ρ_v, $\rho_v'^*$, $\rho_v''^*$, $v \in \Omega_u^+$ определены выше.

Пусть $y_v = 1 - \pi_v$, а $q_{N_v}(\rho_v, y_v)$ — выражение в правой части (18). Тогда из равенств (18), взяв y_v в качестве неизвестной переменной, получим систему независимых уравнений:

$$y_v = q_{N_v}(\rho_v, y_v)\,, \quad v \in \Omega_u^+\,. \qquad (19)$$

Заметим, что для фиксированной v и заданных параметров Λ_v, μ_v, τ_v, t_v, N_v, $v \in \Omega_u^+$, уравнение в (19) является частным случаем уравнения (8) и совпадает с последним, когда число исходящих линий из узла равно 1. Следовательно, для схемы полного разделения памяти также справедливы все приведенные выше утверждения 1–6 и следствие. Заметим, что неравенство (16) в условии утверждения 5 при $B_v = 0$ и $t_v = 0$ преобразуется в неравенство $\Lambda_v/(\mu_v c_v) > 1$, $v \in \Omega_u^+$. Последовательность $\overline{y}[n]$, $n \ge 0$, в утверждении 6 определяется по формуле:

$$\overline{y}[n] = \{y_v[n],\ v \in \Omega_u^+\}\,,\ y_v[n+1] = q_{N_v}(\rho_v,\, y_v[n])\,,$$
$$y_v[0] = 1\,, \quad n \ge 0\,, \quad v \in \Omega_u^+\,.$$

4 Алгоритм расчета

Для вычисления интенсивностей потоков и вероятностей блокировок в узле предлагается следующий алгоритм, описывающий изложенную выше итерационную процедуру. Введем обозначения: y_u^v — вероятность блокировки узла для заявок, направляемых на линию v,

$$y_u^v = \begin{cases} y_u & \text{для } v \in \Omega_u^+ \text{ при} \\ & \text{полнодоступной схеме,} \\ y_v & \text{при схеме полного распределения} \\ & \text{памяти;} \end{cases}$$

$$q_N^v(\overline{\rho}_u^{-v}, y_u^v) = \begin{cases} q_N(\overline{\rho}, y) & \text{для } v \in \Omega_u^+ \text{ при полнодоступной схеме,} \\ q_{N_v}(\rho_v, y_v) & \text{при схеме полного распределения памяти, } v \in \Omega_u^+\,. \end{cases}$$

Тогда уравнения (8) и (19) записываются в виде:

$$y_u^v = q_N^v(\overline{\rho}_u^v, y_u^v)\,, \quad v \in \Omega_u^+\,.$$

Для значений, вычисляемых на k-м шаге алгоритма, к обозначениям соответствующих параметров приписывается знак $[k]$.

Шаг 0.

1. *Инициализация*. Вычисление начальных значений параметров ρ_v, $v \in \Omega_u^+$: $\Lambda_v[0] = \Lambda_v$, $\rho_v[0] = \Lambda_v[0]/(\mu_v(1-B_v))$, $y_u^v[0] = 1$.
2. *Проверка условий существования решения*. Если для некоторой линии $v \in \Omega_u^+$ выполняется хотя бы одно неравенство $(c_v\mu_v(1-B_v))/\Lambda_v[0] \leq \leq 1$, или $(1-B_v)/(\Lambda_v\tau_vB_v) \leq 1$, или $(t_v(1-B_v))/\Lambda_v[0] \leq 1$, то алгоритм заканчивает работу с результатом «нагрузка не реализуема». Если в узле используется полнодоступная схема и $(c_v\mu_v(1-B_v))/\Lambda_v[0] > 1$, $(1-B_v)/(\Lambda_v\tau_vB_v) > > 1$, $(t_v(1-B_v))/\Lambda_v[0] > 1$ для всех $v \in \Omega_u^+$, то проверяется условие (16) для $\Lambda_v = \Lambda_v[0]$, $v \in \Omega_u^+$, и при невыполнении этого условия алгоритм заканчивает работу с результатом «нагрузка не реализуема».

При вычислении левой части неравенства (16) рекомендуется использовать метод свертки Базена (см. [12]), позволяющий производить рекуррентные вычисления (подробно этот метод описан также в [1, 3–6]).

Шаг k ($k > 0$):

1. *Вычисление вероятностей блокировок*. Используя значения параметров $\overline{p}_u^v[k-1]$, $y_u^v[k-1]$, $v \in \Omega_u^+$, вычисление с помощью формул (1)–(7) значений вероятностей $y[k] = 1-\pi[k]$ — в случае полнодоступной памяти, или $y_v[k] = 1-\pi_v[k]$, $v \in \Omega_u^+$, с помощью формул (18) — в случае полного разделения памяти. При вычислении этих значений рекомендуется использовать метод свертки Базена.
2. *Проверка условий останова алгоритма*. Если хотя бы для одной $v \in \Omega_u^+$ для заданного значения точности выполняется условие

$$\frac{|\Lambda_v^*[k] - \Lambda_v^*[k-1]|}{\Lambda_v^*[k]} > \varepsilon\,,$$

то вычисление параметров $\overline{p}_u^v[k]$, $v \in \Omega_u^+$, и переход к шагу k, положив k равным $k+1$, иначе алгоритм завершает работу.

По завершении алгоритма либо выявится, что нагрузка в системе не реализуема, либо будут вычислены интенсивности потоков, поступающих на линии узла, и стационарные вероятности блокировок для заявок каждого типа.

5 Примеры расчета

Для проверки точности вычисления результатов с помощью предложенного выше алгоритма и приемлемости введенных предположений были проведены вычислительные эксперименты с использованием аналитических и имитационных моделей. Во всех рассмотренных ниже примерах потоки внешних заявок считаются пуассоновскими. В табл. 1 приведены значения вероятности блокировок вновь поступивших извне заявок, полученные на основании точной формулы, приведенной в [4] для СМО типа $M|M|1|0$ с повторными заявками при экспоненциальном распределении интервала времени между повторными попытками (первая строка таблицы), алгоритма из подраздела 5 настоящей статьи (вторая строка) и имитационной модели при постоянном интервале времени между повторными попытками, равном 10 (третья строка). Расчет табл. 1 проведен для узла с одной исходящей одноканальной линией при интенсивности первичного потока $\Lambda = 1$ и емкости накопителя $N_v = 1$. Таблицы 2 и 3 вычислены с помощью алгоритма из подраздела 5 и имитационной модели соответственно при одной исходящей линии с числом каналов 10.

В табл. 4 и 5 приведены значения вероятности блокировки узла с тремя исходящими линиями канальной емкости 10 каждая при $\mu_v = 0{,}2$, $v \in \Omega_u^+$, вычисленные с помощью алгоритма из подраздела 5 и имитационной модели с интервалом повторной попытки, равным 10, соответственно. В табл. 4 и 5 знак «—» в ячейках означает, что предложенная нагрузка Λ_v, $v \in \Omega_u^+$, не реализуема.

В табл. 6 отражены вероятности блокировки такого же узла с накопителем $N = 35$ при экспоненциальном распределении интервала времени между повторными попытками со средним значением τ.

Результаты вычислительного эксперимента показывают, что с увеличением длины интервала между повторными попытками вероятность блокировки увеличивается и приближается к значению, вычисленному с помощью алгоритма из подраздела 5 (см. табл. 4 и 6), т. е. при пуассоновском внешнем потоке заявок предположение, что суммарный входной поток заявок является пуассоновским, вполне приемлемо для предварительного анализа характеристик узла (например, при $\tau c_v\mu_v > 10$). Как показывают табл. 1–3, вероятность блокировки узла существенно зависит от

Таблица 1 Вероятности блокировок при одной исходящей одноканальной линии

№	μ				
	1,1	1,2	2	3	4
1	0,9091	0,8333	0,5000	0,3333	0,2500
2	0,9091	0,8333	0,5000	0,3333	0,2500
3	0,8867	0,8452	0,4944	0,3167	0,2396

Таблица 2 Вероятности блокировок при одной исходящей многоканальной линии ($\varepsilon = 0{,}0001$)

N	μ				
	0,11	0,12	0,2	0,3	0,4
10	0,4845	0,2935	0,0204	0,0017	0,0002
15	0,1181	0,0545	0,0006	0,0000	0,0000
20	0,0489	0,0167	0,0000	0,0000	0,0000

Таблица 3 Вероятности блокировок при одной исходящей линии

N	μ_v				
	0,11	0,12	0,2	0,3	0,4
10	0,5247	0,3238	0,0219	0,0019	0,0001
15	0,1726	0,0912	0,0004	0,0001	0,0000
20	0,1180	0,0371	0,0000	0,0000	0,0000

Таблица 4 Вероятности блокировок при трех исходящих линиях, вычисленные алгоритмом из подраздела 5 ($\varepsilon = 0{,}0001$)

N	Λ_v								
	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
20	0,0677	0,1423	0,2975	0,7653	—	—	—	—	—
25	0,0065	0,0173	0,0394	0,0827	0.1690	0.3827	—	—	—
30	0,0005	0,0019	0,0059	0,0155	0.0361	0.0790	0.1792	0,7259	—
35	0,0000	0,0002	0,0008	0,0030	0,0089	0,0234	0,0574	0,1505	—
40	0,0000	0,0000	0,0001	0,0005	0,0022	0,0075	0,0220	0,0617	0,2449

Таблица 5 Вероятности блокировок при трех исходящих линиях, вычисленные с помощью имитационной модели

N	Λ_v								
	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
20	0,0786	0,1695	0,3549	0,7056	—	—	—	—	—
25	0,0069	0,0190	0,0441	0,0998	0,2266	0,4583	—	—	—
30	0,0007	0,0024	0,0075	0,0184	0,0462	0,1025	0,2380	0,6931	—
35	0,0000	0,0003	0,0007	0,0040	0,0129	0,0307	0,0890	0,2981	—
40	0,0000	0,0000	0,0000	0,0011	0,0041	0,0095	0,0346	0,0790	0,3179

Таблица 6 Зависимость вероятности блокировки при трех исходящих линиях, вычисленные с помощью имитационной модели со случайным интервалом между повторными попытками

τ	Λ_v							
	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7
1	0.0001	0,0001	0,0017	0,0063	0,0210	0,0733	0,1996	0,4222
5	0.0000	0,0002	0,0016	0,0036	0,0446	0,0159	0,1360	0,3273
10	0.0000	0,0002	0,0011	0,0036	0,0101	0,0430	0,0818	0,2774
20	0.0000	0,0003	0,0007	0,0029	0,0089	0,0257	0,0863	0,2045

числа каналов в линии при равной суммарной производительности. Кроме того, как видно из табл. 5 и 6, вероятность блокировки в большей степени зависит от среднего значения длины интервала между повторными попытками передачи, чем от закона распределения длины интервала. Таким образом, предложенный в работе алгоритм позволяет вычислить с достаточной точностью вероятность блокировки узла, интенсивности повторных передач и предельную величину реализуемой нагрузки. Отметим, что полученные в данной статье результаты могут быть использованы для расчета нагрузок в телекоммуникационной сети с повторами заявок в предыдущем узле или из источника.

Литература

1. *Kamoun F., Kleinrock L.* Analysis of shared finite storage in a computer networks node environment under general traffic conditions // IEEE Trans. on Commun., 1980. Vol. 28. No. 7. P. 992–1003.

2. *Агаларов Я. М., Шоргин С. Я.* Рекуррентный метод вычисления параметров сетей связи // Техника средств связи, 1986. Сер. «Системы связи». Вып. 6. С. 42–46.
3. *Башарин Г. П., Бочаров П. П., Коган Я. А.* Анализ очередей в вычислительных сетях. — М.: Наука, 1989.
4. *Бочаров П. П., Печинкин А. В.* Теория массового обслуживания. — М.: Изд-во РУДН, 1995.
5. *Вишневский В. М.* Теоретические основы проектирования компьютерных сетей. — М.: Техносфера, 2003.
6. *Башарин Г. П.* Лекции по математической теории телетрафика. — М.: Изд-во РУДН, 2007.
7. *Таранцев А. А.* Инженерные методы теории массового обслуживания. — М.: Наука, 2007.
8. *D'Apice C., De Simone T., Manzo R., Rizelian G.* $M|G|1|r$ retrial queueing system with priority service of primary customers and a customers-searching server // Distributed Computer and Communication Networks. Stochastic Modelling and Optimization. — М.: Техносфера, 2003. P. 106–117.
9. *Klimenok V. I., Kim C. S.* $BMAP/PH/1$ retrial system operating in random environment // Proceedings of the 5th St.-Petersburg Workshop on Simulation, St.-Petersburg, June 26 – July 2, 2005. — St.-Petersburg: NII Chemistry St.-Petersburg University Publs., 2005. P. 367–372.
10. *Krishnamoorthy A., Babu S.* $MAP|(PH, PH)/c$ retrial queue with selegeneration of priorities and non-preemptive service // Proceedings of the 14th International Conference on Analytical and Stochastic Modeling Techniques and Applications, June 4–6, 2007. Prague, Czech Republic. — Sbr.-Dudweiler: Digitaldruck Pirrot GmbH, 2007. P. 70–74.
11. *Корн Г., Корн Т.* Справочник по математике. — М.: Наука, 1974.
12. *Buzen J. P.* Computational algorithm for closed queuing networks with exponential servers // Communications ACM, 1973. Vol. 16. No. 9. P. 527–531.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 43–52

ВОПРОСЫ РЕАЛИЗАЦИИ ОБЪЕДИНЯЮЩЕЙ СРЕДЫ В АРХИТЕКТУРЕ ДЕЦЕНТРАЛИЗОВАННОЙ ПАКЕТНОЙ КОММУТАЦИИ

В. Б. Егоров[1]

Аннотация: Сформулированы требования к объединяющей среде архитектуры децентрализованной пакетной коммутации, рассмотрены способы ее реализации, особое внимание уделено использованию в качестве объединяющей среды высокоскоростных последовательных интерфейсов, предложен редуцированный вариант такого интерфейса, специализированный для пакетной коммутации.

Ключевые слова: пакетный коммутатор; децентрализованная коммутация; интегрированный коммуникационный микроконтроллер; сверхлокальная сеть; высокоскоростной последовательный интерфейс

1 Концепция децентрализованной пакетной коммутации

Одним из эффективных и доступных отечественным разработчикам средств, облегчающих создание оригинальных устройств пакетной коммутации и маршрутизации, может стать широкое использование интегрированных коммуникационных микроконтроллеров (ИКМ) [1]. Главное достоинство ИКМ заключается в удачном сочетании аппаратных коммуникационных адаптеров и высокопроизводительного программируемого процессорного ядра, интегрированных на одном кристалле. Разнообразные коммуникационные адаптеры, универсальные и специализированные, предоставляют пользователю аппаратную поддержку практически всех используемых в современной телекоммуникации интерфейсов и протоколов канального уровня. В то же время наличие свободно программируемого ядра дает возможность реализации программными средствами на одном и том же приборе, по существу «системе на кристалле», как телекоммуникационных протоколов более высоких уровней, так и других сопряженных задач, например маршрутизации и обеспечения сетевой безопасности. Таким образом, ИКМ оказывается привлекательным универсальным средством для разработки разнообразных пакетных коммутаторов, маршрутизаторов, шлюзов и устройств сетевой защиты, целевая ориентация и функциональные возможности которых могут варьироваться и наращиваться соответствующим программным обеспечением [2].

К сожалению, у ИКМ как «системы на кристалле» имеется и обратная сторона. Однокристальным приборам органически присущи принципиальные ограничения по числу встроенных телекоммуникационных портов, суммарной производительности и пропускной способности. Эти ограничения накладываются числом выводов корпуса, полосой пропускания внутренних трактов данных и производительностью встроенных процессоров.

На основе ИКМ, с учетом как их многочисленных достоинств, так и отмеченных недостатков, удобно строить интеллектуальные малопортовые устройства средней производительности, например маршрутизаторы и устройства сетевой безопасности. Такие устройства востребованы на обширном рынке средств подключения локальных сетей — домашних, офисных и корпоративных — к сетям провайдеров. В настоящее время ИКМ применяются для целей пакетной коммутации и маршрутизации в основном в разнообразных ADSL (Asymmetric Digital Subscriber Line) модемах и недорогих малопортовых маршрутизаторах, часто со встроенными средствами сетевой защиты.

Между тем свойственные ИКМ принципиальные ограничения по числу поддерживаемых коммуникационных портов и суммарной пропускной способности могут быть преодолены специальными архитектурными решениями, в частности в архитектуре децентрализованной пакетной коммутации [3, 4], что открывает ИКМ дорогу в область

[1]Институт проблем информатики Российской академии наук, vegorov@ipiran.ru

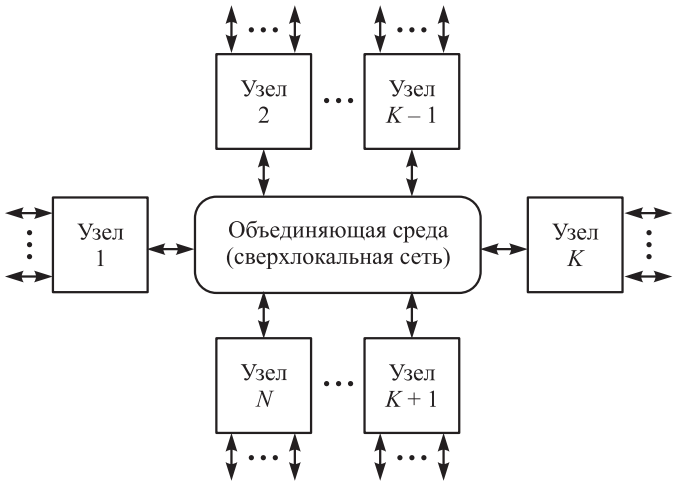


Рис. 1 Концепция децентрализованной пакетной коммутации

многопортовых высокоскоростных пакетных коммутаторов. Обобщенно концепция децентрализованной коммутации предполагает некоторое множество коммутирующих узлов и некую объединяющую их среду, функционирующую на принципах сверхлокальной сети (рис. 1). При этом концепция допускает как гомогенную, так и гетерогенную организацию системы [4]. В гомогенной системе множество функционально идентичных равноправных универсальных узлов, обладающих достаточной производительностью, совокупно выполняют все функции по коммутации, маршрутизации и сетевой защите. При гетерогенной организации выполняемые системой функции распределяются между специализированными узлами. Например, K узлов (узлы $1 \ldots K$ на рис. 1) поддерживают основные внешние коммуникационные порты, выполняют фильтрацию и коммутацию входящих в них пакетов (функции data plane). Из множества остальных узлов (узлы $K+1 \ldots N$ на рис. 1) одни выполняют функции маршрутизации и управления трафиком (control plane), а другие — общего управления системой, тарификации и учета (management plane).

Архитектура децентрализованной пакетной коммутации, ориентированная на ИКМ и в максимальной степени использующая их преимущества, предполагает, что узлы децентрализованного коммутатора реализуются на основе ИКМ. Гомогенная организация системы предполагает во всех узлах однотипные высокопроизводительные ИКМ с широким набором функциональных возможностей. При гетерогенной организации функционально различные узлы могут реализовываться на ИКМ разных типов. В частности, в маршрутизирующих узлах требуется, как правило, ИКМ с высокопроизводительным программируемым ядром и поддержкой больших объемов оперативной памяти, в то время как узлы сбора статистики и тарификации предполагают не столько высокопроизводительное ядро ИКМ, сколько достаточные объемы полупостоянной памяти. В любом случае использование ИКМ «автоматически» обеспечивает всем узлам децентрализованного пакетного коммутатора необходимые внешние порты: основные коммуникационные в коммутирующих узлах и служебные для целей настройки и управления — в маршрутизирующих и вспомогательных.

2 Требования к объединяющей среде

Ключевым элементом архитектуры децентрализованной коммутации, в том числе ориентированной на использование ИКМ, является объединяющая среда. Хотя количественно ее параметры зависят от организации системы — гомогенной или гетерогенной — и характеристик узлов, на качественном уровне можно сформулировать общие требования:

- функционирование на сетевых принципах и реализация в виде сверхлокальной сети;
- достаточная суммарная пропускная способность;
- неблокируемость отдельных трафиков внутри среды;
- удобная практическая реализуемость.

Функционирование объединяющей среды на сетевых принципах в широком плане предполагает:

- обмен информацией между узлами системы в виде пакетов (кадров, ячеек);

– сетевые методы адресации узлов;
– наличие средств подтверждения целостности и достоверности передаваемых данных;
– предоставление заданного уровня качества обслуживания, quality of service (QoS).

Благодаря сетевой организации системы упрощается и унифицируется программное обеспечение узлов, поскольку все коммуникационные порты узлов, как внешние, так и внутренний к объединяющей среде, функционируют на унифицированных принципах и обслуживаются одинаковым образом. В случае реализации узлов на ИКМ такая унификация дает дополнительный эффект, позволяя использовать для подключения узлов к объединяющей среде стандартные встроенные коммуникационные интерфейсы ИКМ. Важным требованием к объединяющей среде, легко реализуемым при сетевой организации, является предоставление заданного уровня QoS и возможность организации между узлами системы динамических виртуальных соединений с гарантированным временем доставки критических данных.

Сверхлокальность объединяющей среды, организованной на сетевых принципах, может проявляться в нескольких аспектах:

– физической и конструктивной ограниченности среды в пределах одного шасси, а в идеальном варианте — одной платы;
– очень высоких скоростях передачи данных благодаря конструктивной компактности и высококачественным линиям связи;
– повышенной достоверности передачи данных вследствие коротких расстояний, качественной среды передачи данных и возможности сведения к минимуму внешних помех;
– постоянстве конфигурации системы и неизменности параметров сети.

Требование достаточности пропускной способности объединяющей среды само по себе очевидно, но неочевидны критерии этой достаточности. В качестве грубой верхней оценки можно принять, что объединяющая среда должна быть способной одномоментно пропустить через себя сумму всех входных потоков на коммуникационных портах всех узлов системы. На самом деле в худшем случае пиковые нагрузки могут оказаться еще выше, поскольку помимо коммутируемых системой потоков данных в объединяющей среде циркулирует служебная информация, обеспечивающая взаимодействие узлов. С другой стороны, в общем случае некоторые входящие пакеты могут коммутироваться локально в пределах одного узла и не попадать в объединяющую среду, а некоторые вообще не подлежат коммутации. Таким образом, в пакетных коммутаторах с большим числом внешних коммуникационных портов возможно некое статистическое усреднение суммарного потока через объединяющую среду на более низких уровнях. Оценка уровня минимальной достаточности пропускной способности для объединяющей среды представляет собой самостоятельную задачу, которая должна учитывать различные факторы:

– гомогенную или гетерогенную организацию системы;
– скорости передачи данных и временно́е распределение потоков на входных коммуникационных портах;
– возможности буферирования данных в узлах;
– жесткость требований QoS для тех или иных трафиков.

Основная задача объединяющей среды — коммутация потоков данных и сообщений между узлами. С этой точки зрения к ней в принципе применимо требование неблокируемости, характерное для классических коммутационных структур. Однако с практической точки зрения требовать от объединяющей среды стопроцентной неблокируемости трафиков внутри нее вряд ли целесообразно. В современных коммутаторах с обязательными развитыми средствами обеспечения QoS и, как следствие, большими объемами буферов, в типичном случае секционированных по приоритетам, порядок прохождения пакетов через коммутационную структуру слабо коррелирован как с порядком их поступления в коммутатор, так и с порядком их отправки из коммутатора. С этой точки зрения более важным представляется обеспечение достаточной суммарной пропускной способности объединяющей среды, а не достижение в ней неблокируемости как таковой. Наглядно этот тезис подтверждается при шинной реализации объединяющей среды. Хотя любая параллельная шина, разделяемая множеством абонентов, являет собой яркий пример целиком и полностью блокирующей среды, различные шины с успехом применялись в коммутаторах первых поколений до тех пор, пока их пропускной способности хватало для суммарного коммутируемого системой трафика. И если сегодня классическая параллельная шина в качестве объединяющей среды архитектуры децентрализованной коммутации представляет лишь академический интерес, то причина этого отнюдь не в ее блокируемости, а в возросших скоростях передачи данных на внешних коммуникационных портах, при кото-

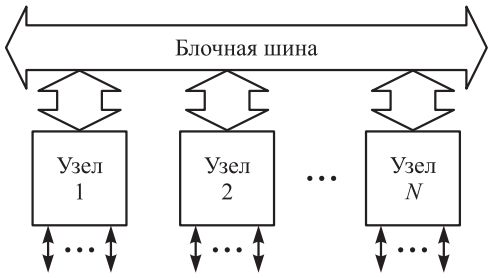


Рис. 2 Блочная шина в качестве объединяющей среды

рых параллельная шина не в состоянии обеспечить необходимые пропускные способности.

Вряд ли требует особых комментариев требование удобной реализуемости объединяющей среды. Однако оно заставляет уделить внимание возможным вариантам ее практической реализации.

Первая и самая простая реализация объединяющей среды может представлять собой, как уже было замечено выше, некую параллельную шину. Правда, требование функционирования на сетевых принципах сразу исключает в качестве объединяющей среды типичные компьютерные шины — от ISA (Industry Standard Architecture) до PCI-X (Peripheral Component Interconnect Extended). Тем не менее параллельная шина не только может функционировать на сетевых принципах, но и оказывается при этом проще компьютерных шин, выполняя функции объединяющей среды более эффективно. Рисунок 2 иллюстрирует использование в качестве концептуальной объединяющей среды архитектуры децентрализованной коммутации предложенной автором блочной шины [5]. Как и всякая другая, блочная шина не обеспечивает неблокируемость трафиков, однако она гарантирует своевременную доставку данных из узла в узел в пределах своей пропускной способности и с учетом QoS. К сожалению, суммарная пропускная способность блочной шины не превышает 2–4 Гбит/с, поэтому она применима лишь в многопортовых децентрализованных коммутаторах со скоростями передачи данных на коммуникационных портах, не превышающих 100 Мбит/с, но практически непригодна для современных коммутаторов с гигабитными скоростями.

Более производительную альтернативу шинам представляют собой коммутационные структуры. Одной из первых появилась технология коммутации Raceway, возникшая как расширение шины VME (VersaModule Eurocard) и унаследовавшая ряд ее особенностей, в частности 32-разрядные параллельные порты коммутационных структур. Эта технология была реализована в 6-портовым интегральном коммутаторе Raceway crossbar компании Cypress Semiconductor [6]. При максимальной частоте 40 МГц каждый 32-разрядный порт коммутатора Raceway crossbar обеспечивал пропускную способность 1,28 Гбит/с, а суммарная пропускная способность коммутатора с учетом неблокируемости коммутируемых трафиков составляла 7,68 Гбит/с. Очевидно, что один такой прибор в качестве объединяющей среды архитектуры децентрализованной коммутации мог обслуживать максимум шесть узлов (рис. 3) и обеспечить пропускную способность, лишь ненамного превышающую, например, возможности блочной шины. Кроме того, в отличие от блочной шины, технология Raceway, унаследовавшая логическую организацию компьютерной шины VME, базировалась отнюдь не на сетевых принципах и, следовательно, не отвечала основному требованию к объединяющей среде архитектуры децентрализованной коммутации. Зато, в отличие от параллельной шины, она позволяла строить сложные коммутационные структуры с использованием множества интегральных коммутаторов, что в принципе открывало возможность неограниченного наращивания числа подключаемых узлов и суммарной пропускной способности объединяющей среды. В частности, интегральные коммутаторы Raceway crossbar позволяли строить коммутирующие структуры с самомаршрутизирующими и неблокирующими топологиями, в частности баньян-сети, сети Клоза, Бенеша [7] и другие. К сожалению, спроектированные на таких интегральных коммутаторах топологически сложные коммутационные структуры, такие как сети

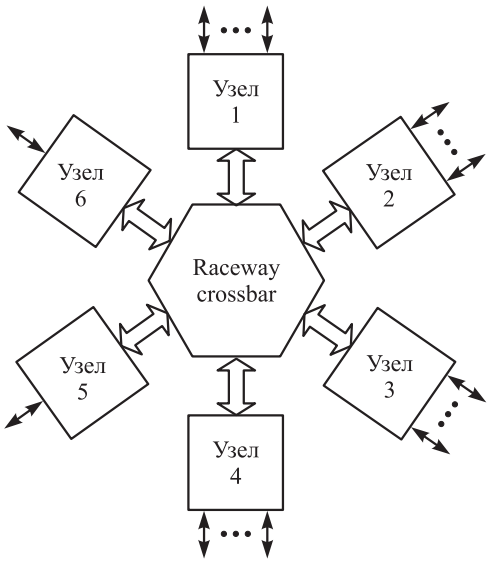


Рис. 3 Параллельный коммутатор Raceway crossbar в качестве объединяющей среды

Клоза или Бенеша, с большим числом коммутаторов и, следовательно, еще бóльшим числом параллельных шин, не только не отвечают требованию удобства реализуемости объединяющей среды, но и вообще вряд ли реализуемы на практике.

3 Высокоскоростные последовательные интерфейсы в качестве объединяющей среды

Существенное упрощение реализации топологически сложных коммутационных структур стало возможным с появлением локальных высокоскоростных последовательных интерфейсов, в которых переход к последовательным методам передачи данных не только резко сократил число сигнальных линий интерфейса, но и заставил сблизить принципы их функционирования с традиционно сетевыми.

Современные методы последовательной передачи данных позволяют при относительно небольших длинах линий связи достигать скоростей в несколько гигабит в секунду на типовой низковольтной дифференциальной паре проводников. Переход к последовательной передаче данных с резким сокращением числа сигнальных линий коммуникационного порта предоставляет возможность существенного увеличения числа портов в одном отдельно взятом интегральном коммутаторе, а общее сокращение сигнальных линий упрощает реализацию топологически сложных коммутационных структур из множества интегральных коммутаторов. Все это в совокупности открывает практически неограниченные перспективы наращивания числа объединяемых абонентов и пропускной способности коммутационных структур, в частности используемых для создания объединяющей среды в архитектуре децентрализованной коммутации. В табл. 1 приведено сравнение различных средств реализации объединяющей среды: блочной шины и интегральных коммутаторов с параллельными и последовательными высокоскоростными интерфейсами портов. Сравнение демонстрирует очевидные преимущества последовательных высокоскоростных интерфейсов при реализации сложных коммутационных структур. Однако для использования их в объединяющих средах архитектуры децентрализованной коммутации этого недостаточно. Как было подчеркнуто выше, необходимо, чтобы такие интерфейсы функционировали в объединяющей среде на сетевых принципах.

Первые высокоскоростные последовательные интерфейсы появились как замена компьютерных параллельных шин, пропускная способность которых перестала отвечать возросшим скоростям передачи данных внутри компьютеров, и поэтому в целом сохранили логическую организацию компьютерных шин. Так, появившиеся одними из первых интерфейсы VXS (VMEbus Switched Serial) и StarFabric унаследовали общую организацию, адресацию абонентов и принципы взаимодействия между ними от своих предшественников, шин VME и PCI соответственно. Тем не менее, уже в технологии StarFabric появились такие чисто сетевые понятия, как пакетная передача данных, разделение сервисов по уровням (физический, звеньевой и транспортный), классы трафика, избыточное кодирование и контроль данных циклическими кодами. Дальнейшее развитие с улучшением всех количественных характеристик этот подход получил в интерфейсах InfiniBand и PCI Express (PCIe).

Однако все эти интерфейсы по-прежнему оставались компьютерными в главном: изначально предназначенные для использования в качестве системного интерфейса компьютера, они были ориентированы на древовидную топологию с единственным центральным процессором («корень» древа), централизованно управляющим самим интерфейсом и подключаемыми к нему периферийными устройствами («стволом» и «кроной»).

Принципиально новое качество в этом плане привнес в новом тысячелетии высокоскоростной

Таблица 1 Сравнение различных средств реализации объединяющей среды

Характеристика	Параллельная блочная шина	Коммутатор с параллельными портами	Коммутатор с последовательными портами
Число объединяемых узлов	8 . . . 12	4 . . . 6	8 . . . 64
Ширина интерфейса к узлу	до 64 разрядов	до 32 разрядов	1 . . . 4 разряда
Скорость обмена данными с узлом	до 4 Гбит/с	до 2 Гбит/с	до 20 Гбит/с (дуплекс)
Общая пропускная способность	до 4 Гбит/с	до 12 Гбит/с	до 1280 Гбит/с
Масштабируемость	плохая	средняя	хорошая
Стоимость подключения узла	средняя	высокая	низкая

интерфейс ввода-вывода RapidIO. Хотя он специфицирован в двух вариантах, параллельном и последовательном, широкое практическое применение нашла именно последовательная его реализация — serial RapidIO (SRIO). Хотя интерфейс SRIO все еще несет в себе некоторые «атавизмы» компьютерных интерфейсов, в частности возможность адресного обращения к памяти по чтению или записи, в нем появились новые чисто сетевые черты, которые, в частности, позволяют эффективно использовать его в качестве объединяющей среды. Главные среди них:

- возможность объединения равноправных (peer-to-peer) абонентов;
- сетевая адресация конечных абонентов;
- сервис инкапсуляции сквозных трафиков со сторонними протоколами;
- развитые механизмы обмена сообщениями между абонентами;
- наличие средств поддержки QoS;
- сегментация и сборка пакетов, segmentation and reassembly (SAR).

Возможность объединения равноправных абонентов, средства инкапсуляции сквозных трафиков и обмена сообщениями в совокупности с поддержкой QoS позволяют естественным образом строить с помощью интерфейса SRIO сверхлокальные сети и использовать их в качестве объединяющей среды архитектуры децентрализованной коммутации, как это было показано на рис. 1. Механизм SAR дает возможность объединяющей среде работать с короткими, длиной не более 256 байт, пакетами, что облегчает достижение требуемого качества обслуживания. Кроме того, сегментация пакетов упрощает реализацию интегральных коммутаторов и, как следствие, объединяющей среды в целом.

Достоинством SRIO является использование на физическом уровне стандарта XAUI (X-Attachment Unit Interface), давно применяемого в таком широко распространенном интерфейсе, как 10 Gigabit Ethernet, что способствует упрощению и удешевлению реализации абонентских адаптеров SRIO. Но это достоинство может обернуться недостатком. Предельная полезная скорость передачи данных на одной дифференциальной паре XAUI равна 2,5 Гбит/с, а на максимально широком звене из четырех пар составляет 10 Гбит/с. Для сравнения, спецификация PCIe-II допускает вдвое большие скорости на дифференциальной паре и ширину звена до 32 пар. Впрочем, скорости передачи данных на интерфейсе SRIO также будут расти в дальнейшем, в частности уже имеется стандарт CEI (Common Electrical Interface) со скоростями 6 и 11 Гбит/с на одной паре при сигнализации, совместимой с XAUI.

В 2007 г. SRIO обрел опасного конкурента — интерфейс ASI (Advanced Switching Interconnect), который, по существу, представляет собой надстройку на стандартном интерфейсе PCIe. Эта надстройка призвана компенсировать как раз те недостатки PCIe, которые препятствуют его использованию в сверхлокальных сетях и тем самым в качестве объединяющей среды. Используя физический уровень и организацию звена PCIe, ASI добавляет такие важные сетевые качества, как:

- равноправие подключаемых абонентов;
- возможность обмена сообщениями между абонентами;
- туннелирование трафиков со сторонними протоколами;
- многоуровневая поддержка QoS.

При этом ASI воспринимает все возможности и преимущества интерфейса PCIe, в частности хорошо отработанную технологию физического уровня и высокие скорости передачи данных на звене. Сохраняя совместимость «сверху вниз», ASI может взаимодействовать со стандартными сегментами PCIe через прозрачные мосты. В результате технология ASI, совмещая «два в одном», смотрится очень привлекательно в качестве объединяющей среды архитектуры децентрализованной коммутации. Однако оборотная сторона такой всеобъемлющей универсальности — большая избыточность в каждом конкретном применении и, главное, сложность реализации с неизбежно высокой ценой результата. На сегодняшний день имелись объявления о планах выпуска коммутаторов ASI от компаний StarGen[1] и Xyratex, однако на рынке подобных продуктов так и не появилось. Поддержку технологии ASI рекламирует компания Xilinx, но реально эта поддержка сводится к введению в серию программируемых микросхем этой компании Virtex-II, трансиверов PCIe и процессорного ядра IBM PowerPC 405 для программной реализации всех «надстроек» ASI на стандартном интерфейсе PCIe. Таким образом, перспективы технологии ASI пока довольно туманны. И даже если в будущем интегральные коммутаторы ASI все же появятся на рынке, вряд ли они будут конкурентоспособны по ценам с коммутаторами SRIO.

[1] StarGen прекратила свое существование, ее преемницей стала компания Dolphinics, в планах которой коммутаторы ASI вообще не числятся.

Но и коммутаторы SRIO отнюдь не дешевы: цена интегрального коммутатора SRIO сопоставима, например, с ценой ИКМ высокой производительности. Конечно, если узлы децентрализованного пакетного коммутатора строятся на традиционных ИКМ, то применение в объединяющей среде интерфейса существующих интегральных коммутаторов SRIO выглядит вполне технически сбалансированным и экономически оправданным. Но если основой узлов коммутатора будут специальные ориентированные на пакетную коммутацию и потому упрощенные и более дешевые ИКМ [8], то логично дополнить упрощенные ИКМ в узлах более дешевыми интегральными коммутаторами в объединяющей среде. Удешевления интегральных коммутаторов можно было бы достичь за счет некоего более простого последовательного интерфейса, специализированного, подобно упрощенным ИКМ, под пакетную коммутацию.

4 Упрощение последовательного интерфейса для пакетной коммутации

Интерфейс SRIO специфицирован на трех уровнях: логическом, транспортном и звеньевом[1]. Каждый из этих уровней обладает избыточностью при использовании интерфейса SRIO в качестве объединяющей среды архитектуры децентрализованной коммутации.

Логический уровень SRIO предлагает пользователю три сервиса:

(1) адресный доступ к памяти и вводу-выводу (input/output);

(2) передачу сообщений (message passing);

(3) инкапсуляцию сквозных потоков данных (data streaming).

Исходя из требований к объединяющей среде, адресный доступ вообще следует признать ненужным и даже вредным сервисом. Он явно противоречит сетевым принципам взаимодействия узлов, поскольку не только заставляет программное обеспечение узлов децентрализованного коммутатора «опускаться» до уровня чтения/записи ячеек памяти и адресуемых регистров, но и предполагает неконтролируемое вторжение в память других узлов. Два других сервиса — передача сообщений и инкапсуляция сквозных потоков данных — релевантны сверхлокальной сети и вполне отвечают требованиям к объединяющей среде. Оба эти сервиса включают SAR, но, к сожалению, в силу непонятных причин реализуются на интерфейсе SRIO различно. В частности, при передаче сегментированного сообщения все его сегменты нумеруются, что дает возможность передавать и транспортировать их в произвольном порядке, а в конечном абоненте-приемнике восстанавливать сообщение независимо от порядка поступления сегментов. Однако при инкапсуляции сквозных потоков данных такая возможность не предусмотрена, и сегменты инкапсулированных блоков данных должны транспортироваться и доставляться получателю в естественном порядке.

С точки зрения объединяющей среды усложнения в конечном абоненте, связанные с произвольным порядком передачи сегментов относительно коротких сообщений, при жестком порядке транспорта сегментов относительно длинных инкапсулированных блоков данных выглядят совершенно неоправданными. Логично было бы пожертвовать свободным порядком транспорта сегментов сообщений, которые зачастую и состоят-то из одного-единственного сегмента, во имя унификации с механизмом транспорта инкапсулированных блоков данных и тем самым существенного упрощения всего механизма SAR у конечного абонента.

На транспортном уровне интерфейс SRIO предоставляет выбор 8- или 16-разрядных адресов конечных абонентов, но адресация должна быть единой в пределах всей сети, объединяемой интерфейсом SRIO. Возможно, в каких-то приложениях, например связанных с массовой цифровой обработкой сигналов, где конечным абонентом интерфейса может быть одна-единственная микросхема аналого-цифрового (АЦП) или цифро-аналогового преобразователя (ЦАП), такие абоненты могут исчисляться сотнями. Но ИКМ, хотя и позиционируемый как «система на кристалле», будучи основой узла децентрализованного пакетного коммутатора, «обрастает» внешними памятями, трансиверами и другим вспомогательным оборудованием, вследствие чего узел представляет собой отнюдь не одинокую микросхему, а, скорее, плату или модуль. Число таких модулей в границах физической реализуемости высокоскоростных интерфейсов класса SRIO вряд ли перевалит за несколько десятков. Соответственно, для целей децентрализованной пакетной коммутации можно было бы ограничиться 8-разрядным адресом конечного абонента.

На звеньевом уровне интерфейса SRIO каждый сегмент сообщения или инкапсулированного бло-

[1] Приблизительные параллели в семиуровневой эталонной модели взаимодействия открытых систем ISO: транспортный, сетевой и канальный уровни соответственно.

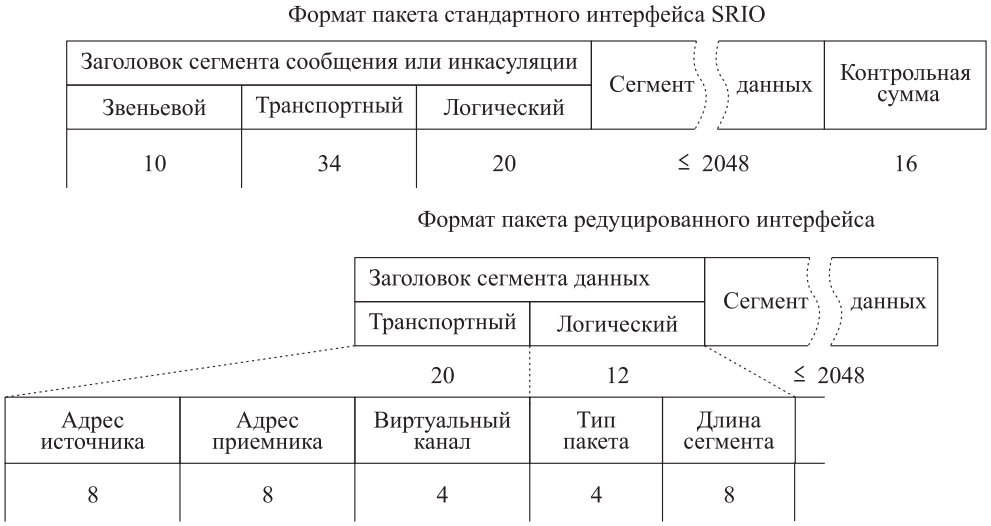


Рис. 4 Форматы пакетов для стандартного SRIO и примерного редуцированного интерфейсов

ка данных оформляется отдельным пакетом и сопровождается своей контрольной суммой. По ней все звеньевые контроллеры, как в конечных абонентах, так и в элементах коммутационных структур, контролируют и подтверждают друг другу правильность передачи на звене отдельного сегмента и самостоятельно выполняют повтор передачи в случае ошибки. Конечно, в целом такой механизм повышает достоверность передачи сообщения или инкапсулированного блока данных, но вряд ли сопровождающие его издержки оправданы в объединяющей среде децентрализованного пакетного коммутатора. Сверхлокальность объединяющей среды предполагает, в частности, короткие и высококачественные линии связи на звене с высокой достоверностью передачи данных. С другой стороны, сетевая организация объединяющей среды гарантирует надежный контроль целостности и достоверности доставленных сообщений и инкапсулированных блоков данных у конечных абонентов. Поэтому отказ от контроля и повтора передач сегментов на звене позволил бы без ощутимых потерь не только упростить контроллер звена интерфейса, но и, главное, ускорить передачу данных за счет исключения контрольных сумм сегментов и обмена подтверждениями на уровне звена.

Таким образом, высокоскоростной последовательный интерфейс, упрощенный для целей пакетной коммутации, который в дальнейшем будем называть **редуцированным интерфейсом**, мог бы отличаться от стандартного интерфейса SRIO:

- исключением сервиса адресного доступа;
- унификацией сервисов сообщений и инкапсуляции;
- ограничением сетевых адресов конечных абонентов одним байтом;
- отказом от контроля передач и подтверждений на уровне звена и, соответственно, исключением контрольных сумм сегментов сообщений и инкапсулированных блоков данных.

Совокупность предложенных упрощений должна обеспечить редуцированному интерфейсу по сравнению со стандартным SRIO бóльшую эффективность за счет повышения пропускной способности и упрощения интерфейсных адаптеров.

Выигрыш в пропускной способности редуцированного интерфейса за счет укорочения заголовков пакетов и отказа от контрольных сумм иллюстрирует рис. 4, где показаны форматы пакетов с сегментами сообщений и инкапсулированных блоков данных для стандартного интерфейса SRIO и примерного редуцированного интерфейса, в котором упрощения коснулись заголовков всех трех уровней, но не затронули дисциплину SAR.

В примерном редуцированном интерфейсе уровень звена настолько упрощается, что звеньевой заголовок оказывается вовсе ненужным. Из двух основных полей звеньевого заголовка стандартного SRIO одно — идентификатор подтверждения — исчезает вследствие отказа от подтверждений на уровне звена, а второе — приоритет пакета — перемещается в транспортный заголовок. В транспортном заголовке, поскольку узлы децентрализованного коммутатора могут иметь по нескольку коммуникационных интерфейсов, для каждой пары узлов предусматривается возможность организации между ними нескольких виртуальных каналов, например, для различных сочетаний их входных и

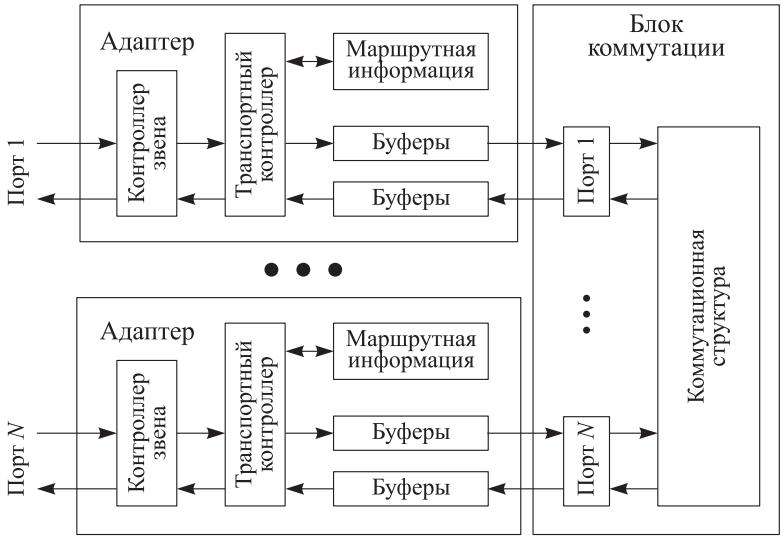


Рис. 5 Обобщенная структура интегрального коммутатора SRIO

выходных интерфейсов. С этой целью в транспортный заголовок редуцированного интерфейса вводится дополнительное поле номера виртуального канала, которое параллельно выполняет функции поля класса обслуживания в логическом заголовке пакета стандартного SRIO и в этом качестве попутно задает приоритет пакета. Заголовок логического уровня редуцированного интерфейса ограничивается двумя тривиальными полями: тип пакета (сообщение/инкапсулированные данные, запрос/подтверждение и т. п.) и фактическая длина сегмента данных.

Таким образом, пакет редуцированного интерфейса может стать на 48 разрядов, т. е. 14,3% исходной длины, короче пакета стандартного SRIO при одной и той же полезной нагрузке. Правда, на редуцированном интерфейсе ошибка передачи данных в любом сегменте потребует повтора передачи всего сообщения или инкапсулируемого блока данных, поэтому упрощение интерфейса оправдано только в сетях с высокой достоверностью передачи данных, что как раз характерно для сверхлокальных сетей объединяющих сред.

Интерфейсные адаптеры редуцированного интерфейса упростятся как у конечных абонентов, так и в элементах коммутационных структур.

У конечных абонентов интерфейсный адаптер редуцированного интерфейса получится более компактным и дешевым за счет сокращения числа и унификации поддерживаемых сервисов, а также вследствие упрощения функций, поддерживаемых контроллером звена интерфейса. Это особенно важно для упрощенных специализированных под пакетную коммутацию ИКМ [8], в которых интерфейсный адаптер объединяющей среды может составлять заметную долю аппаратуры.

Еще бóльшим окажется влияние вносимых в интерфейс упрощений на интегральные коммутаторы. Коммутаторы SRIO в общем случае структурно включают адаптеры порта — по адаптеру на каждый внешний порт коммутатора — и некий блок коммутации для обмена пакетами между портами [9]. Адаптер порта, в свою очередь, состоит из звеньевого и транспортного контроллеров, входных и выходных буферов коммутируемых пакетов и некоего хранилища маршрутной информации со средствами доступа к ней (рис. 5).

Редукция интерфейса упрощает адаптер порта интегрального коммутатора сразу по нескольким направлениям. Во-первых, ограничение адресов конечных абонентов восемью разрядами позволяет контроллеру однозначно и быстро маршрутизировать все входящие пакеты с помощью простой полноразмерной маршрутной таблицы объемом всего на 256 входов без вовлечения сложных алгоритмов и механизмов преобразования сетевых адресов (хеширование адресов, многоуровневые маршрутные таблицы и т. п.). Маршрутная информация может устанавливаться статически вследствие относительной неизменности конфигурации сверхлокальной сети и задаваться централизованно, но возможно и динамическое заполнение маршрутных таблиц с использованием обычным методов «самообучения». Во-вторых, отказ от подтверждений на уровне звена заметно упрощает аппаратуру контроллера звена. В-третьих, исключение повторов передачи на звене устраняет принципиальную необходимость в выходных буферах порта для хра-

нения уже переданных пакетов на случай возникновения необходимости повторной передачи. В результате при прочих равных условиях интегральные коммутаторы редуцированного интерфейса будут проще и дешевле коммутаторов SRIO, не говоря уже о боле дорогих технологиях вроде ASI.

5 Заключение

Интегрированные коммуникационные микроконтроллеры зарекомендовали себя хорошим «строительным материалом» для недорогого интеллектуального коммуникационного оборудования среднего класса, т. е. оборудования пусть не передовых рубежей телекоммуникации, но зато активно востребованного и массового выпускаемого. На основе ИКМ строятся различные ADSL-модемы, маршрутизаторы для офисов и домашнего применения, небольшие цифровые АТС, шлюзы из сетей ISDN (Integrated Services Digital Network) в пакетные сети и другое оборудование для схожих приложений.

К сожалению, ИКМ пока еще недостаточно используются в отечественных разработках, хотя потенциально весьма привлекательны многими своими достоинствами, в том числе встроенными программируемыми процессорными ядрами и аппаратными адаптерами с поддержкой телекоммуникационных интерфейсов и протоколов. Эти качества ИКМ особенно важны для отечественных разработчиков, традиционно вынужденных решать программными средствами задачи, которые за рубежом решаются аппаратно благодаря доступности новейших достижений технологии сверхбольших интегральных схем.

Архитектура децентрализованной коммутации дает возможность еще больше расширить сферу потенциальной применимости ИКМ, включив в нее высокоскоростные многопортовые пакетные коммутаторы. Децентрализованные пакетные коммутаторы могут строиться на существующих ИКМ, при этом в качестве объединяющей среды архитектуры децентрализованной коммутации можно использовать современные высокоскоростные последовательные интерфейсы. Особенно привлекательно в этом плане выглядит интерфейс SRIO, адаптеры которого интегрируются в некоторые современные ИКМ.

Однако ИКМ, подходящие для реализации узлов децентрализованного коммутатора, т. е. с производительными процессорными ядрами, высокоскоростными коммуникационными портами и поддержкой интерфейса SRIO, недешевы. Соизмеримы с ними по цене и реализующие объединяющую среду интегральные коммутаторы SRIO. Поэтому в целом децентрализованному пакетному коммутатору, реализуемому на основе существующих ИКМ и интерфейсе SRIO, будет трудно конкурировать с массово выпускаемыми аппаратными коммутаторами, несмотря на потенциально более широкие функциональные возможности и гибкость. Более эффективно и, следовательно, конкурентоспособно задача построения децентрализованных высокоскоростных многопортовых пакетных коммутаторов могла бы решаться путем упрощения ИКМ и высокоскоростного последовательного интерфейса специализацией их для задач пакетной коммутации и, в частности, ориентацией на архитектуру децентрализованной пакетной коммутации. В этом плане предложенный в статье редуцированный интерфейс естественно дополняет ранее выдвинутую автором концепцию создания отечественных упрощенных ИКМ для пакетной коммутации [8].

Литература

1. *Егоров В. Б.* Принципы создания коммутационной аппаратуры на основе специализированных микроконтроллеров // Системы и средства автоматики. — М.: Наука, 1999. Вып. 9. С. 44–55.
2. *Егоров В. Б.* Интегрированные коммуникационные микроконтроллеры Freescale Semiconductor: из прошлого в будущее // Электронные компоненты, 2008. № 7. С. 31–40.
3. *Соколов И. А., Егоров В. Б.* Дезинтеграционный подход к архитектуре универсального процессора коммутации пакетов // Информационные технологии и вычислительные системы, 2005. № 2. С. 76–85.
4. *Соколов И. А., Егоров В. Б.* Дезинтегрированная архитектура пакетной коммутации // Информатика и её применения, 2008. Т. 2. Вып. 4. С. 2–11.
5. *Егоров В. Б., Полухин А. Н.* Принципы создания системной шины многопортовых пакетных коммутаторов // Системы и средства информатики. — М.: Наука, Физматлит, 2000. Вып. 10. С. 80–90.
6. *Robinson J.* RACEway interlink adoption and growth // VITA J., September 1997. P. 12–16. http://www.vita.com/sept97vj/rway.pdf.
7. *Богданов А., Станкова Е., Корхов В., Мареев В.* Архитектуры и топологии многопроцессорных вычислительных систем. — М.: ИНТУИТ.ру, Открытые системы, 2004. Гл. 5. http://www.informika.ru/text/teach/topolog/5.htm.
8. *Егоров В. Б.* Концепция создания отечественных интегрированных коммуникационных микроконтроллеров для пакетной коммутации // Информатика и её применения, 2009. Т. 3. Вып. 1. С. 34–46.
9. *Егоров В. Б.* Последовательный интерфейс RapidIO и его применение в пакетной коммутации // Электронные компоненты, 2008. № 12. С. 69–76.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 53–64

ТЕХНОЛОГИЧЕСКАЯ СИСТЕМА ДЛЯ ПОСТРОЕНИЯ ПРОГРАММНЫХ КОМПЛЕКСОВ АВТОМАТИЗАЦИИ ОБРАБОТКИ ТРЕХМЕРНЫХ ДАННЫХ ЛАЗЕРНОГО СКАНИРОВАНИЯ

В. А. Сухомлин[1], И. Н. Горькавый[2]

Аннотация: Рассмотрена технологическая система для создания комплексов автоматической обработки данных лазерного сканирования. Приведены разработанные и применяемые авторами алгоритмы для классификации трехмерных данных и получения высококачественных моделей земного рельефа. Описан инструментарий для визуализации, оценки и коррекции полученных моделей. Обсуждаются достигнутые результаты и направление дальнейших исследований.

Ключевые слова: автоматизация обработки данных; алгоритмы классификации; трехмерные модели; лидар; лазерное сканирование

1 Введение

Традиционные аэрокосмические данные и существующие программы их обработки не дают возможности создать трехмерные модели земной поверхности высокого разрешения. Стереоскопическая обработка космических снимков со спутников SPOT Image приводит к модели с 20-метровым горизонтальным разрешением и вертикальной точностью в 10–20 м при уровне достоверности 90% [1]. Спутниковая программа радарного зондирования Shuttle Radar Topographic Mission, реализованная в 2000 г., получила модель земной поверхности с разрешением в 30 м и вертикальной точностью ~ 4 м [2].

В 1993 г. появился первый коммерческий лидар (LIght Detection And Ranging, http://www.csc.noaa.gov/crs/rs_apps/sensors/lidar.htm) авиационного базирования. Фактически лидарные авиасистемы совершили переворот в генерации трехмерных моделей земной поверхности, позволив получать большие объемы лазерных измерений с горизонтальным разрешением 1–2 м и вертикальной точностью 10–15 см при уровне достоверности 95%. На данный момент лазерное зондирование — это наиболее эффективный и точный способ создания трехмерной модели земного рельефа.

Современные лидарные системы, установленные на самолетах или вертолетах, позволяют сканировать земную поверхность лазерными инфракрасными лучами (длина волны ~ 1 мкм) с частотой выстрелов 100–150 кГц. Для каждого лазерного импульса фиксируется время одного или нескольких отражений от поверхности. Таким образом определяются трехмерные координаты точек отражения луча (наклон лазерного луча к поверхности известен, координаты самолета определяются с помощью GPS (Global Positioning System) и инерционного измерительного устройства). Сканер быстро перемещает лазерный луч поперек траектории самолета, летящего на высоте около километра. Плотность распределения зондируемых точек по земной поверхности (post spacing) обычно близка к одному выстрелу на квадратный метр. Абсолютная вертикальная точность измерений равна 10–15 см, относительная (по отношению к соседней точке) — около 5 см. Горизонтальная точность зависит от высоты полета и от расхождения луча и обычно близка к 50 см. Кроме этого, определяется интенсивность отражения луча от поверхности.

В настоящее время по всему миру работает несколько сотен таких авиационных систем. Полученные данные применяются для точного топографирования местности, для мониторинга состояния старых и планирования новых нефте- и газопроводов, для анализа распространения радиоволн и звука, для моделирования стихийных бедствий и других прикладных задач.

Новый уровень трехмерных данных потребовал новых методов обработки и соответствующих программных средств, которые могли бы наиболее полно использовать информативность лидарных данных. Комплексы для автоматической обработки таких данных стали развиваться только

[1]Факультет вычислительной математики и кибернетики МГУ; Институт проблем информатики Российской академии наук, sukhomlin@mail.ru

[2]Факультет вычислительной математики и кибернетики МГУ, ilya_gor@rambler.ru

в последние годы. Они заметно сложнее своих предшественников, так как умеют распознавать и классифицировать объекты на поверхности. Кроме того, предъявляются повышенные требования к точности получаемых моделей земного рельефа.

Целью данной работы является решение задачи построения комплексов автоматической обработки данных лазерного сканирования с использованием расширяемой компонентной архитектуры.

Статья состоит из восьми разделов. В разд. 2 обсуждаются задачи, возникающие при обработке лидарных данных высокого разрешения, и приведены ключевые особенности трехмерных данных в сравнении с двумерными аналогами; рассмотрено несколько возможных подходов к построению комплексов автоматизации обработки. Раздел 3 посвящен требованиям, которые предъявляются к комплексам обработки трехмерных данных и к получаемым с их помощью моделям рельефа. В разд. 4 рассмотрена модель системы на основе базовой архитектуры и перечислены ее основные компоненты. В разд. 5 приводится формализованная схема представления данных. Раздел 6 посвящен использованным в работе методам классификации данных. В разд. 7 описан инструментарий интерфейса: способы визуализации и интерактивной коррекции данных. В разд. 8 сформулированы основные результаты работы, отмечены особенности предлагаемого решения, указано направление дальнейших исследований.

2 Задачи

Развитие технологий получения высококачественных трехмерных изображений обширных районов земной поверхности привело к быстрому росту объема трехмерных данных и одновременно расширило спектр задач, возникающих при их обработке. В настоящее время лидарные данные используются для получения высокоточных цифровых моделей рельефа (ЦМР, Digital Terrain Model) с разрешением 2 м и выше; для извлечения моделей трехмерных городов, которые используются, например, в Google Earth; для создания контурных, затененных и других типов топографических карт; для генерации инфракрасных изображений местности высокого разрешения; для исследования характеристик лесных массивов и других практических применений [3–7].

При автоматизации решения этих задач используются схожие алгоритмы распознавания и классификации объектов, фильтрации артефактов, интерполяции и триангуляции поверхностей, преобразования координатных систем. Кроме того, при решении любой из них понадобится ввод/вывод данных, визуализация результата, конвертор форматов и другие системные и интерфейсные средства. Разумно было бы создать некоторую базовую архитектуру комплексов по обработке трехмерных данных, и в ее рамках реализовать необходимые программные модули для решения подзадач обработки. Такой метасистемный переход позволил бы создавать целевые решения, обеспечивающие весь жизненный цикл автоматической обработки, не перегруженные лишней функциональностью и при этом повторно использующие код большинства модулей.

Существующие продукты не предусматривают комплексного подхода к этой проблеме, обеспечивая решение некоторых распространенных задач, но не всего цикла обработки данных. Например, программные средства, поставляемые с ALTM (Airborne Laser Terrain Mapper) системами Optech (такие как ZinView, DASHMap, IVS3D Fledermaus), обеспечивают визуализацию больших массивов данных, растеризацию, сбор статистической информации, но не включают интерполирование данных, их классификацию (на рельеф, здания, растительность) и валидацию. Похожая ситуация с ESRI ArcGIS, который хотя и является индустриальным стандартом, но не имеет встроенных алгоритмов классификации и предназначен для работы с уже готовыми решениями.

Принципиальными требованиями к комплексам по автоматизации обработки трехмерных данных лазерного сканирования являются скорость обработки и точность алгоритмов распознавания и классификации объектов. Лидарная аэросъемка генерирует 100–150 ГБ данных на каждую тысячу квадратных километров. Поэтому уже для проектов среднего размера общий объем данных составляет терабайты. Эффективная обработка таких объемов возможна только при наличии интеллектуальных алгоритмов распознавания и классификации объектов.

Имеющиеся решения для потоковой обработки и распознавания двумерных изображений малоприменимы к трехмерным из-за следующих особенностей 3D-данных [7]:

- принципиального отличия значимости высоты объекта от значимости его цвета и разной сферы их применения в практических задачах;
- наличия множества значений высот для каждой точки поверхности, обусловленных последовательными отражениями луча от частично проницаемых поверхностных объектов, тогда как на двумерных изображениях цвет имеет один слой;

– разреженности (неполноты) трехмерных изображений, вызванной их векторным представлением и наличием пробелов в данных из-за сложностей в функционировании лидаров, тогда как растровые изображения, как правило, полны;
– неоднородности трехмерных изображений, связанной со сбоями в работе оборудования или лидарной аэросъемкой, проведенной в разные времена года.

Трехмерные данные более информативны, чем их двумерные аналоги, однако их обработка связана с дополнительными сложностями и необходимы специальные алгоритмы, которые бы смогли максимально использовать имеющуюся информацию об объектах для их автоматической классификации. Возникла потребность в создании высокоавтоматизированных комплексов для обработки больших массивов трехмерных данных. Существует несколько возможных путей построения подобных программных решений, а именно:

(1) **создание полнофункциональной системы**, обеспечивающей весь жизненный цикл обработки данных и включающей в себя решения всех известных задач — классификации объектов, выделения контурных карт, трехмерного моделирования и т. д. Реализация такого комплекса с нуля представляет собой дорогостоящий и сложный проект, а его сопровождение и развитие потребует значительных человеческих ресурсов;

(2) **расширение имеющейся геоинформационной системы (ГИС).** При этом за основу берется коммерческая система и недостающие функции реализуются и интегрируются в нее. К сожалению, встроенные программные средства, такие как скрипт-язык Avenue в ESRI ArcGIS, предназначены больше для развития уже имеющихся в ГИС решений, чем для реализации новых. Получение же исходного кода таких систем, как правило, невозможно. Более перспективным выглядит использование ГИС с открытым кодом, такой как GRASS (Geographic Resources Analysis Support System), но требования GPL (General Public License) не всегда приемлемы;

(3) **создание базовой технологической системы**, включающей набор готовых системных и функциональных компонентов, с последующим ее расширением и конкретизацией к требованиям заданного класса задач. Это позволяет при разработке значительно сократить этапы анализа, проектирования и тестирования системы и получить программное решение с возможностью адаптации к изменяющимся условиям его функционирования. С развитием системы число функциональных компонентов и решаемых ими задач может расти.

В данной работе избран третий вариант решения проблемы — конкретизация базовой архитектуры. Ее использование обусловлено технологичностью, сбалансированностью стоимости и времени на разработку и сопровождение. При этом сохраняется возможность для развития системы и применения ее к другим типам задач.

3 Требования

Проектируемые комплексы по обработке трехмерных данных лазерного сканирования должны отвечать ряду основных требований:

– обеспечивать высокий уровень автоматизации с минимальным участием человека;
– создавать модели рельефа и наземных объектов максимально возможной вертикальной и горизонтальной точности;
– обладать достаточной функциональностью для решения заданного класса задач;
– давать пользователю возможность контроля обработки данных;
– адаптироваться к различному качеству и формату входных данных;
– предоставлять возможность оперативно изменять вид и формат конечных моделей.

Требование вертикальной точности поддается формализации: пользуясь общепринятыми стандартами индустрии, а именно среднеквадратическим отклонением в сантиметрах, взятым по 95% контрольных измерений, имеем:

$$\sigma(z) < 18{,}5\,\text{см}\,, \quad \sigma(z) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - z_i')^2}\,, \qquad (1)$$

где z_i' — высота контрольного измерения, а z_i — интерполированная высота ЦМР в точке измерения. По рекомендациям FEMA (Federal Emergency Management Agency) и ASPRS (American Society for Photogrammetry and Remote Sensing) [8] отклонение не более чем в 18,5 см считается достаточным для построения контурных 60-сантиметровых карт, популярных в индустрии. С развитием лидарной технологии требования к точности будут, вероятно, повышаться. Количество контрольных измерений, как правило, слишком мало, чтобы выявить

все ошибки классификации. Поэтому кроме количественной оценки точности необходима визуальная верификация трехмерной модели оператором. Остальные требования формализуются исходя из характера и объема планируемой работы, сроков ее выполнения, пожеланий пользователей данных.

Анализ перечисленных выше требований и типовых задач обработки позволил сформулировать основные критерии, которым должна удовлетворять базовая технологическая система:

- иметь минимальный набор функций, при этом обеспечивающий решение задач обработки;
- обладать модульной расширяемой архитектурой;
- обеспечивать простоту реализации целевых комплексов обработки;
- минимизировать затраты на сопровождение комплексов.

Архитектура базовой системы, созданной в рамках данной работы, рассмотрена ниже.

4 Модель системы

4.1 Архитектура базовой технологической системы

На основе описанной выше концепции была разработана архитектура базовой технологической системы, состоящей из трех основных виртуальных машин или ядер:

(1) системного ядра, отвечающего за организацию вычислений, поддержку управления процессом обработки, управление ресурсами, доставку, подготовку и хранение данных;

(2) функционального ядра, реализующего математические методы и алгоритмические решения подзадач обработки;

(3) интерактивного ядра, позволяющего пользователю оценивать и оперативно корректировать процесс автоматической обработки данных.

Структура такой системы и состав ее базовых компонентов показаны на рис. 1. Системное ядро состоит из блока ввода/вывода данных, конвертора форматов, менеджера данных, менеджера ресурсов и профилировщика. Функциональное ядро содержит блоки преобразования координатных систем, фильтрации артефактов, калибрации, растеризации, интерполяции, классификации объектов, векторизации, моделирования поверхностей и валидации. Интерактивное ядро состоит из компонента визуализации 3D, визуализатора 2D, блока статистического анализа и графического интерфейса.

Каждый компонент ядер решает определенную задачу на некотором наборе входных данных и является необходимым элементом для построения комплексов автоматизации обработки. В компоненты выделены только те задачи, которые являются общими для большинства типов приложений и, соответственно, могут быть повторно использованы.

Применение такой трехъядерной архитектуры даст возможность создавать автоматизированные комплексы для широкого спектра приложений. При этом каждый целевой продукт будет включать только те компоненты ядер, которые необходимы для решения поставленных задач. Реализация выбранных компонентов может быть изменена с учетом требований среды, таких как специфический формат входных данных или изменение интерфейса управления. Компоненты представлены в виде библиотек (динамических и статических) или в виде исполнимых файлов.

Анализ показывает, что подобные работы ведутся в настоящее время в ряде стран. Например, комплекс TerraScan финской компании Terrasolid, среда SCOP++ [3] немецкой фирмы Inpho или трехстадийная архитектура, предложенная итальянской группой в [5].

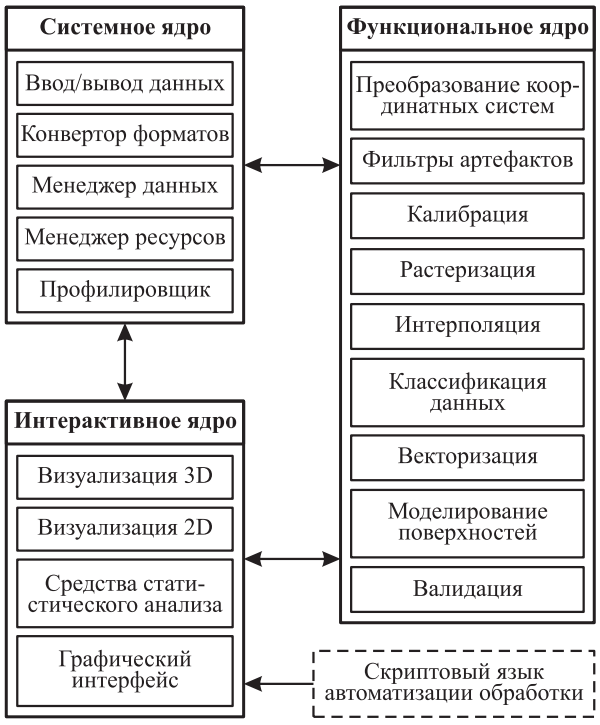


Рис. 1 Базовая архитектура комплексов обработки трехмерных данных

4.2 Компоненты системного ядра

Системное ядро представляет собой программный модуль, постоянно загруженный в память и работающий в скрытом режиме без прямого общения с пользователем. Реализация ядра предоставляет другим компонентам программный интерфейс для унифицированного доступа к данным, запускает на выполнение функциональные компоненты, следит за использованием аппаратных ресурсов и ресурсов операционной системы, ведет статистику времени выполнения компонентов. Реализация выполнена на языке C++ в среде Visual Studio. Рассмотрим подробней функциональность компонентов или модулей ядра.

Ввод/вывод обеспечивает считывание данных с носителей (сотни гигабайт, как правило) и запись готовых результатов. Для упрощения предварительной подготовки данных модуль рекурсивно обходит дерево директорий файловой системы и автоматически распознает пригодные для чтения файлы с данными. Число успешно прочитанных записей лидарных импульсов и записей с поврежденным форматом передается в компонент статистического анализа интерактивного модуля для генерации отчета о входных данных проекта. Скорость операций чтения/записи составляет не менее 10 МБ/с, что позволяет за сутки прочитать и подготовить к обработке терабайт данных. Подробное описание входного формата лидарных данных есть в разд. 5.

С вводом/выводом тесно связан **конвертор форматов**, который используется для перевода одних форм представления данных в другие, в том числе во внутреннее представление. Этот компонент обеспечивает единый метод работы с данными, имеющими различную организацию и структуру. Список поддерживаемых форматов в реализации включает такие популярные форматы, как бинарные BIL (“band interleaved by line”), LAS (“LIDAR data exchange”), GeoTIFF (“geographic tagged image file format”), текстовые ARC/INFO ASCII GRID, ASCII XYZ, и может быть расширен. Практика показала, что узким местом в производительности компонента часто становятся такие базовые преобразования, как переводы STRING $\leftrightarrow$ INT, STRING $\leftrightarrow$ FLOAT. Анализ функций atol( ) и strtol( ) из библиотеки GNU glibc выявил, что в этих функциях бóльшую часть кода занимают операции, связанные с распознаванием недесятичных чисел, использованием языковых локалей и проверкой переполнений. Дополнительные действия, условные переходы и вызовы других функций значительно замедляют код стандартной библиотеки. Поскольку при вводе данных их формат заранее известен, то эти проверки можно опустить и использовать более эффективный код.

Организацию данных во внутреннем представлении и сохранение промежуточных результатов вычислений осуществляет **менеджер данных**. Он сортирует прочитанные входные данные на отдельные участки. Чтобы определить, к какому участку принадлежит лидарный импульс, используется сетка разбиения, которая может быть как равномерной прямоугольной, так и произвольно заданной. Если система координат (картографическая проекция), в которой определена сетка, не совпадает с системой координат входных данных, то производится **преобразование координатных систем** применительно к сетке, так как в ней меньше точек. Каждый готовый участок выводится в отдельный файл. На этом этапе обработки хранение промежуточных результатов неизбежно, иначе пришлось бы считывать всю неструктурированную базу данных проекта при каждом обращении к участку. Здесь под участком понимается совокупность лидарных точек, расположенных в пространственном параллелепипеде, размеры которого задаются с учетом цели обработки и возможностей применяемых компьютеров.

Менеджер данных полагается в своей работе на **менеджер ресурсов**, который контролирует использование ресурсов системы, таких как оперативная память, место на жестких дисках, количество одновременно открытых дескрипторов файлов, и, по возможности, освобождает ресурсы, занятые редко используемыми данными, для более новых.

Профилировщик служит для сбора статистики о производительности вычислений и отдельных функциональных компонентов. Он имеет точность работы в 1 мс и использует механизм timeGetTime( ) как имеющий низкие накладные расходы и наиболее надежный и устойчивый на разных типах систем, включая многопроцессорные. На основе собранной статистики периодически обновляется прогноз о завершении обработки (это может быть несколько часов или несколько дней), а также выдается итоговый отчет. Профилировщик полезен при тестировании различных компонентов комплекса и при поиске узких мест в последовательной работе компонентов (конвейера обработки).

4.3 Компоненты функционального ядра

Функциональное ядро представляет собой набор программных модулей, которые реализуют алгоритмы конвейера автоматической обработки данных. Число модулей в конвейере и порядок их следования могут варьироваться, но в общем случае они соответствуют порядку, указанному на рис. 1 в

функциональном ядре. Модули загружаются в память и исполняются по мере надобности. Отделение их разработки от остальных частей комплекса позволило сделать алгоритмы более адаптивными к различным задачам и дало возможность быстрой перенастройки комплексов. Модули используют системное ядро для доступа к данным и интерактивное ядро для визуализации и обратной связи с пользователем. Модули были реализованы на языках C++ и Fortran 77, а некоторые были портированы на UNIX и использовались как самостоятельные утилиты.

Преобразование координатных систем выполняет перевод между картографическими проекциями. Типичная задача — перевод из универсальной равноугольной поперечно-цилиндрической проекции Меркатора в локальную систему. На первых этапах реализации была сделана попытка использовать для этой цели сторонний продукт Corpscon 5, но его производительность и устаревший код ввода-вывода не позволяли выполнить задачу за приемлемое время. Поэтому был написан собственный модуль, основанный на коде Corpscon, который был переработан для достижения более высокой производительности: около 1 млн преобразований координат в секунду, что в несколько раз быстрее, чем при использовании оригинального кода.

Фильтры артефактов на разных этапах работы проверяют соответствие данных набору определенных условий — от простейших требований к корректности записи до выявления аномальных значений высоты и интенсивности, которые могут быть признаками сбоя в оборудовании или наличия посторонних объектов между лазером и поверхностью земли. Лидарные отражения, не удовлетворяющие условиям, помечаются как ошибочные и в дальнейшем не участвуют в построении цифровых моделей и не входят в конечный результат. Более сложные ошибки, такие как смещение всего массива отражений по одной из координатных осей, выявляются на этапе **валидации** при сравнении с контрольными измерениями. Иногда ошибки приводят к пропускам в данных, в таком случае решением является **интерполяция**.

Модуль **калибрации**, используя нормализацию по выбранному эталону, выравнивает значения интенсивности (яркости) для данных, собранных разной аппаратурой или в разное время года. Описание алгоритма и примеры его работы можно найти в [6].

Лидарные данные после сортировки на участки представляют собой кластер трехмерных точек с некоторым набором атрибутов у каждой. **Растеризация** переводит такие данные в набор двумерных матриц, что позволяет значительно повысить скорость работы и на порядок понизить требования к объему оперативной памяти. При правильном выборе размера элемента растра потеря вертикальной точности практически отсутствует. Горизонтальная точность лидарных данных изначально не очень высока из-за расхождения луча лазера.

Модуль **интерполяции** заполняет пустоты в двух- или трехмерных данных, как правило, связанные с ошибками при сборе данных. При этом используются многопроходные алгоритмы, позволяющие пробелы в данных отличать от пустот, соответствующих водным массивам, которые не дают стабильного отраженного луча. Пробелы заполняются простой интерполяцией информации из соседних непустых элементов растра. Число пробелов может достигать 10% от всей площади участка. Описание реализованных методов интерполяции есть в [6].

Модуль **классификации объектов** сортирует данные на несколько типов (земля, растительность, здания), используя совокупность оригинальных математических методов [4, 7], позволяющих достичь высокой точности классификации (1 ошибка на 10^3–10^4 точек) при сохранении приемлемой производительности ($\sim$ 30 тыс. классификаций в секунду). Методам классификации, положенным в основу данного модуля, посвящен разд. 6.

Растр используется обычно в качестве внутреннего представления для распознавания и классификации, при этом итоговые модели должны быть векторными. Поэтому модуль **векторизации** делает обратное преобразование отклассифицированных на растре моделей в векторную форму. Далее вступает в работу компонент **моделирования поверхностей**, строящий триангуляционную сетку на основе векторной модели. Проверка точности полученной трехмерной модели осуществляется в модуле **валидации**, который формирует набор контрольных измерений и вычисляет среднеквадратическое отклонение (1).

4.4 Компоненты интерактивного ядра

Интерактивное ядро — это интерфейсная часть комплекса, которая предоставляет пользователю средства визуализации, контроля и активного вмешательства в процесс обработки. Основные компоненты ядра реализованы на языке C++ и используют кросс-платформенный стандарт OpenGL 2.1.

Компонент **визуализации 3D** позволяет отобразить трехмерную модель рельефа с возможностью наложения текстурных карт; модель можно произвольно вращать и приближать. В компоненте реализовано большое число оптимизационных техник для обеспечения максимальной производительности. Модуль **визуализации 2D** отвечает за

отображение двумерных карт высот и интенсивностей и имеет стандартные функции сдвига и увеличения изображения.

Средства статистического анализа позволяют узнать характерные свойства распознаваемых объектов — занимаемую площадь, высоту над уровнем земли. Эта информация полезна для классификации с участием оператора. Оболочкой для интерактивного ядра служит **графический интерфейс** (GUI, Graphical User Interface). С его помощью пользователь загружает данные участка с диска, управляет их отображением в 2D- или 3D-виде, следит за качеством автоматической классификации объектов и визуально оценивает получившуюся модель. При обнаружении недостатков оператор может провести интерактивную коррекцию: выделить мышкой проблемную область и пересчитать ее с новыми параметрами классификации. Более подробно этот инструментарий описан в разд. 7.

Контроль обработки человеком улучшает качество моделей, так как создать идеальные методы распознавания и классификации пока не представляется возможным. Однако уже существующий комплекс может провести обработку в автономном режиме и получить модели, удовлетворяющие стандартам точности. Для описания пакетной обработки используется скриптовое представление на основе языка Perl и библиотеки, обеспечивающей доступ к компонентам системы. Выбор языка обусловлен доступностью интерпретаторов Perl для всех основных платформ, его гибкостью при работе со строками и легкостью написания на нем сценариев. Элементы управления интерфейса во многом зависят от применяемых функциональных компонентов и их конфигурационных параметров, поэтому графический интерфейс подвержен частым изменениям при построении целевых решений.

5 Схема данных

Лидарные данные имеют высокую информативность, позволяя передать не просто трехмерные координаты конкретной точки земной поверхности, но и ее цвет в инфракрасном диапазоне, сколько отражающих поверхностей встретил лазерный луч на пути к земле, каковы их координаты, под каким углом и в какое время был выпущен луч и многое другое. Сохранение в процессе обработки этих дополнительных параметров позволяет использовать их на этапе классификации объектов и получать более точные цифровые модели. В общем виде лидарные данные можно представить множеством точек в трехмерном пространстве с некоторым набором обязательных и необязательных атрибутов у каждой точки. Формальную схему можно определить так:

```
<xs:schema>
   <xs:element name="PointData">
      <xs:complexType>
      <xs:sequence>
         <xs:element name="CoordX" type="xs:double" />
         <xs:element name="CoordY" type="xs:double" />
         <xs:element name="HeightZ" type="xs:float" />
         <xs:element name="Intensity" type="xs:float" />
         <xs:element name="ReturnNumber" type="xs:byte" />
         <xs:element name="ScanDirection" minOccurs="0">
           <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:enumeration value="Left to Right" />
             <xs:enumeration value="Right to Left" />
           </xs:restriction>
           </xs:simpleType>
         </xs:element>
         <xs:element name="ScanAngle" type="xs:byte"
                     minOccurs="0" />
         <xs:element name="FlightlineEdgeFlag" type="xs:boolean"
                     minOccurs="0" />
         <xs:element name="ClassType" minOccurs="0">
           <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:enumeration value="Noise" />
```

```
            <xs:enumeration value="Unclassified" />
            <xs:enumeration value="Ground" />
            <xs:enumeration value="Low Vegetation" />
            <xs:enumeration value="Medium Vegetation" />
            <xs:enumeration value="High Vegetation" />
            <xs:enumeration value="Building" />
            <xs:enumeration value="Water" />
          </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="SourceID" type="xs:string"
                    minOccurs="0" />
        <xs:element name="GPS Time" type="xs:time" minOccurs="0" />
        <xs:element name="RedChannel" type="xs:integer"
                    minOccurs="0" />
        <xs:element name="GreenChannel" type="xs:integer"
                    minOccurs="0" />
        <xs:element name="BlueChannel" type="xs:integer"
                    minOccurs="0" />
      </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Здесь `CoordX`, `CoordY` и `HeightZ` задают координаты точки; `Intensity` —интенсивность отражения лазерного луча; `ReturnNumber` определяет, к какому по счету отражению импульса относится данная точка; поля `ScanDirection`, `ScanAngle` и `FlightlineEdgeFlag` характеризуют направление движения лазерного луча во время импульса. Поле `ClassType` содержит класс объекта, к которому относится точка (Unclassified — для еще не классифицированных). Поле `SourceID` определяет источник данных — это может быть имя файла, название проекта или любой другой текстовой идентификатор. Три цветовых компоненты `RedChannel`, `GreenChannel` и `BlueChannel` позволяют окрасить точку, если лидарный сканер использовался совместно с фотокамерой.

Пример записи, соответствующей изложенной схеме данных:

```
<PointData>
  <CoordX> 598867.72 </CoordX>
  <CoordY> 4500731.25 </CoordY>
  <HeightZ> 0.06 </HeightZ>
  <Intensity> 248.92 </Intensity>
  <ReturnNumber> 1 </ReturnNumber>
  <ClassType> Water </ClassType>
  <SourceID> 062603_f1_297 </SourceID>
</PointData>
```

Учитывая то, что проект может содержать более 10^{10} точек, использование XML (eXtensible Markup Language) формата в реализации сопряжено с большими затратами аппаратных ресурсов, поэтому данные хранятся в двоичной форме. При этом текстовые поля заменяются числовыми и максимум полей упаковывается побитно. Формат и длина упакованной записи, включая наличие необязательных полей, определяются в заголовке. Туда же выносятся глобальные свойства данных, такие как информация о картографической проекции, текстовый комментарий к проекту и т. п. Форма XML может быть использована при работе с небольшим подмножеством данных, например при просмотре пользователем точек, принадлежащих к конкретному объекту на поверхности.

6 Методы классификации

Важным компонентом функционального ядра базовой системы является модуль «классификации данных» (см. рис. 1). Для создания качественной модели земной поверхности лидарные данные должны быть классифицированы на отражения от земли и на отражения от растительности или домов, которые для ЦМР являются помехами. Проблема классификации огромного массива данных (около миллиона точек на квадратный километр) представляет собой нетривиальную алгоритмическую проблему. Рассмотрим некоторые методы, которые могут быть положены в основу компонента классификации данных.

В статье [9] был предложен алгоритм фильтрации, основанный на анализе наклона поверхности (slope-based filter или SB-фильтр). Он состоит в том, что для каждой точки лидарного отражения строится фильтрующий элемент — выпуклая поверхность в виде перевернутой чаши, например гиперболоида с вершиной в анализируемой точке. Если в полости гиперболоида нет других точек, то рассматриваемая точка отражения считается точкой земной поверхности. По набору лидарных отражений от земной поверхности методом триангуляции строится модель рельефа.

Такой метод имеет ряд недостатков [10], например он неприменим к крупным зданиям. Кроме того, для земной поверхности с большим наклоном значительная часть точек может быть неправильно отфильтрована как помехи (гиперболоид, построенный в точке крутого склона, легко захватывает соседние точки отражений). В статье [10] предложен адаптивный SB-фильтр, где в качестве фильтрующего элемента взят конус с углом, зависящим от наклона поверхности, что частично устранило недостатки метода.

Одним из интересных подходов является «метод линейного предсказания» [11], совмещающий классификацию и интерполяцию. Это статистический метод, основанный на корреляции между соседними точками.

В статье [12] предложен IPF (Interrelated Processes Forecasting) метод, который также совмещает фильтрацию и интерполирование. По точкам лидарных отражений строится усредненная поверхность первого приближения, и точки, расположенные значительно выше этой поверхности, отбрасываются. Оставшиеся точки используются для генерации модели поверхности второго приближения, и алгоритм повторяется. После ряда итераций получается земная поверхность приемлемого качества.

Авторами настоящей статьи был разработан новый математический метод классификации лазерных отражений, соответствующих земной поверхности, — метод виртуальных поверхностей (МВП). В качестве фильтрующих элементов он использует поверхности с динамически меняющейся степенью выпуклости. Этот метод обладает положительными качествами известных методов: SB-фильтра [9] и его адаптивной версии [10], а также IPF-метода [12]. При этом метод виртуальной поверхности избегает наиболее серьезных недостатков этих алгоритмов. Метод виртуальных поверхностей совмещает фильтрацию и интерполирование, как и IPF-метод, получая в результате виртуальную поверхность, близкую к реальной земной поверхности. Метод виртуальных поверхностей, в отличие от IPF-метода, не является итеративным и делает классификацию поверхности за один проход, что значительно ускоряет обработку данных. Детальное описание МВП можно найти в [4]. Для эффективной работы метода необходима растеризация лидарных данных: перенос на прямоугольную сетку с пикселом определенного размера. С этой целью первичное множество точек лазерных отражений рассортировывается в три двумерные матрицы:

(1) $Z_{\min\, ik}$ — матрицу минимального значения высот отражений;

(2) $Z_{\max\, ik}$ — матрицу максимального значения высот отражений;

(3) $I_{\mathrm{av}\, ik}$ — матрицу среднего значения интенсивности отражения.

Для формализованного представления процесса в [13] изложен метод матричного описания обработки данных (МООД), который сопоставляет каждой процедуре обработки оператор, действующий над матрицей значений.

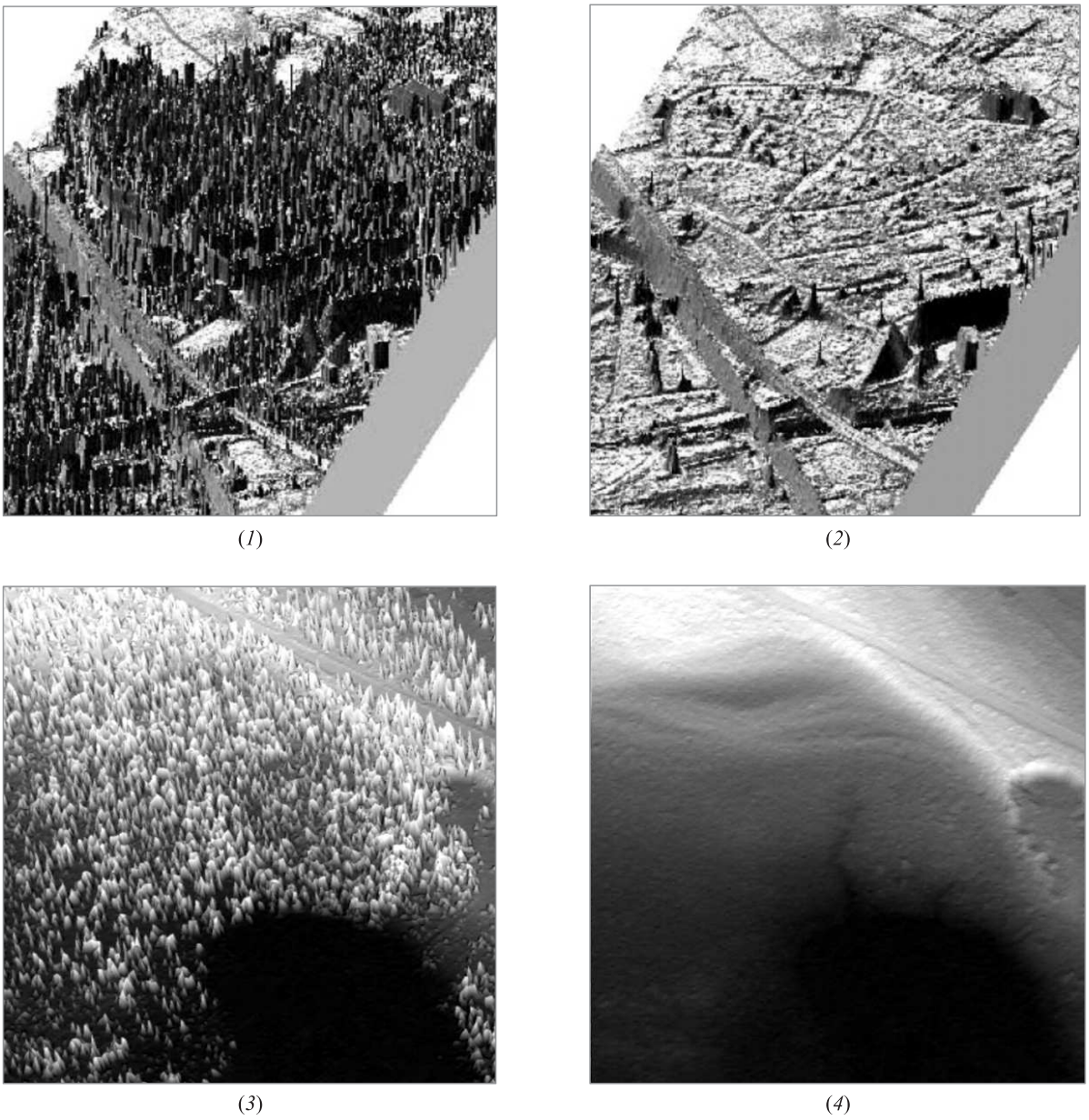
На рис. 2 приведен результат работы классифицирующих алгоритмов комплекса АКС-ЛИДАР-3D, реализованного авторами в рамках предложенной архитектуры, в сравнении с модулем RASCOR [14], разработанным в Ганноверском университете (Германия).

Видно, что RASCOR не смог полностью извлечь растительность и здания, оставив многочисленные пики, в то время как МВП построил достаточно гладкую модель рельефа, убрав всю растительность.

7 Инструменты

В рамках предложенной архитектуры разработан удобный инструментарий активного вмешательства в процесс обработки, который позволяет в течение 1–2 мин пересчитать модель земного рельефа на участке ~ 1 км^2 с уточненными параметрами классификации. Для обнаружения проблемной области в рельефе, связанной с ошибкой в исходных данных или с неточностью автоматической обработки, используется оригинальный 3D-визуализатор [7]. Регулярная структура ЦМР, полученная на этапе классификации, позволяет воспользоваться при отображении иерархической структурой данных.

Для эффективной визуализации необходимо обращаться к ресурсам современных графических адаптеров (GPU, Graphics Processing Unit), которые превосходят CPU (Central Processing Unit) по вычислительной мощности. Для ускорения отображения в модуле используются: группировка вершин



(*1*) (*2*)

(*3*) (*4*)

Рис. 2 Работа алгоритмов классификации: (*1*) и (*2*) — исходные данные и результат работы RASCOR; (*3*) и (*4*) — исходные данные и результат работы комплекса АКС-ЛИДАР-3D

в участки, отсечение по усеченной пирамиде видимости, динамическая детализация, оптимизация по предварительному кэшу и кэшу трансформированных вершин.

Качество и скорость визуализации сравнимы с такими популярными инструментами ГИС, как ESRI ArcGIS Desktop и APL Quick Terrain Modeler. Пример визуализации трехмерных рельефов с помощью разработанного инструмента показан на рис. 3, где изображен карьер глубиной более 300 м, имеющий ступенчатый рельеф стен.

В зависимости от сложности моделируемой области до 90% обработанных участков оказываются хорошего качества. Остальные участки обычно содержат один–два локальных артефакта. После визуализации полученной ЦМР и выявления ошибки в рельефе, оператор может воспользоваться инструментом активной коррекции трехмерных данных, например:

– задать более агрессивную фильтрацию в рамках метода виртуальной поверхности и выгладить любую плохо отфильтрованную зону;

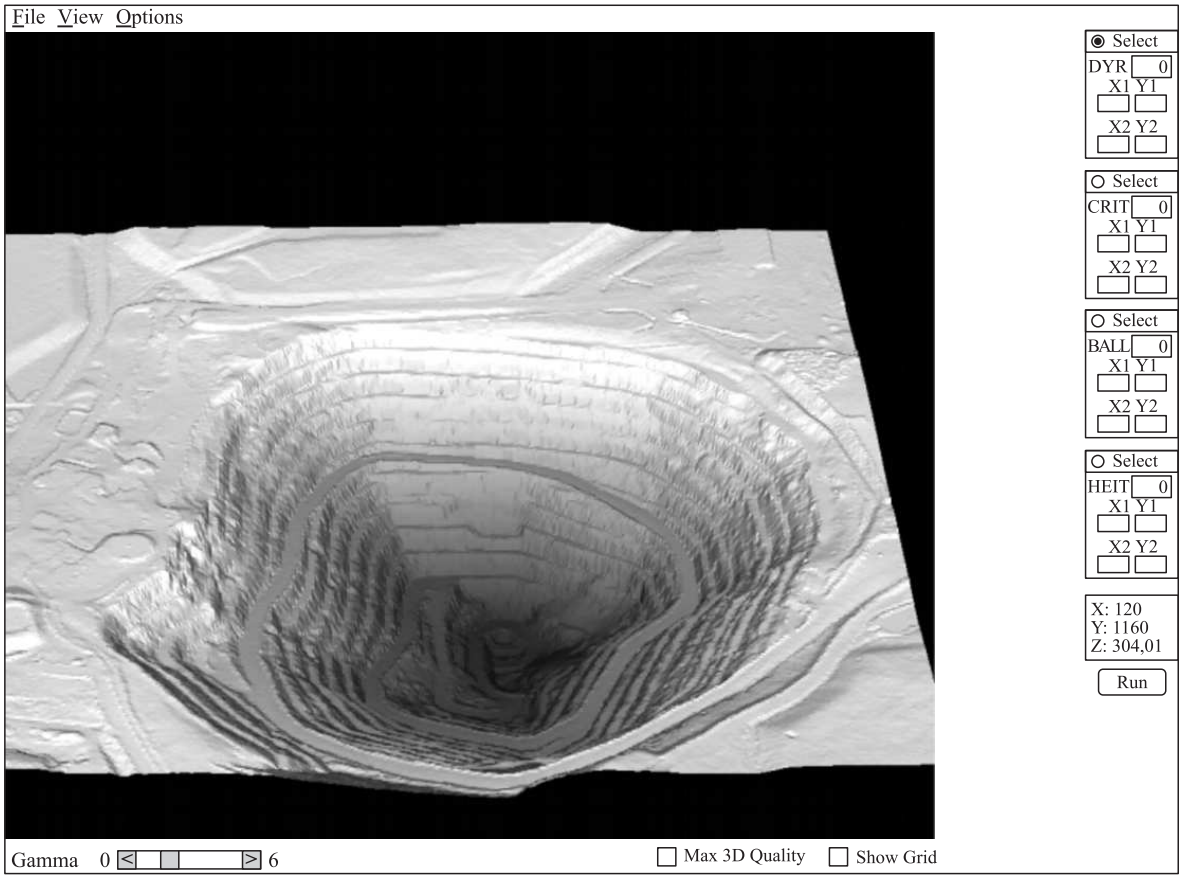


Рис. 3 Окно интерфейса: трехмерная модель земной поверхности

– отменить фильтрацию в определенной зоне и сохранить все имеющиеся там объекты;

– отрезать лидарные отражения выше или ниже заданного уровня;

– усилить функцию заклеивания точечных артефактов в исходных данных и т. д.

8 Заключение

Методы и идеи, на которые опирается данная работа, разрабатывались авторами на протяжении последних шести лет. За это время на основе предложенной технологической системы было создано два типа комплексов автоматизации обработки трехмерных данных лазерного сканирования:

– АКС-ЛИДАР-3D для получения трехмерной цифровой модели рельефа и распознавания поверхностных объектов;

– АКС-ЛИДАР для получения двумерных инфракрасных изображений поверхности.

Для каждого типа было сделано несколько реализаций под конкретные проекты обработки (примеры работы комплексов были приведены на рис. 2 и 3).

В частности, комплексы использовались для обработки лидарных данных территории штата Мэриленд (США). Результаты в виде инфракрасных изображений, цифровых моделей рельефа и сопутствующие метаданные о точности полученных моделей доступны по адресам: http://dnrweb.dnr.state.md.us/gis/data/lidar/ и http://maps.csc.noaa.gov/TCM/.

Общий объем работы, выполненной с помощью комплексов, составил свыше 20 000 км^2 земной поверхности с разрешением 2 м. Скорость полного цикла обработки составляла 500 км^2 в неделю, при этом полученная модель рельефа имела типичное среднеквадратическое отклонение по высоте в 14,3 см, а для качественных данных точность достигала 9,3 см (по данным независимой экспертизы, проведенной Dewberry LLC). В проекте с высококачественными лидарными авиаданными и разрешением в 1,2 м была достигнута точность по высоте в 6–8 см. Точность автоматической класси-

фикации превосходила аналогичные программные продукты [11].

Основными особенностями предлагаемого подхода и реализации программных комплексов, описанных в данной статье, являются:

- использование трехъядерной архитектуры, включающей системное, функциональное и интерактивное ядра, что позволяет повторно использовать максимум кода и сократить трудозатраты на разработку;
- создание базовой технологической системы, включающей необходимый набор основных компонентов и упрощающей реализацию и сопровождение комплексов обработки данных;
- применение оригинальных алгоритмических решений, таких как метод виртуальной поверхности для классификации объектов и матричное представление данных, что обеспечивает высокую точность и скорость обработки;
- реализация собственного визуализатора трехмерных поверхностей, который использует современные техники оптимизации, что позволило работать с моделями, состоящими из миллионов точек;
- наличие инструментария для управления циклом обработки — как в виде интерактивных средств, реализованных в графическом интерфейсе, так и в виде скрипт-языка;
- применение открытого бинарного формата данных, разработанного в соответствии с требованиями индустриального стандарта (LAS 1.2/2008). Этот формат показал высокую эффективность при хранении и преобразовании данных.

Дальнейшее усовершенствование базовой технологической системы ведется в направлении расширения возможностей функционального ядра и повышения качества работы его алгоритмов. В перспективе возможно создание полностью автоматизированного полнофункционального комплекса по обработке трехмерных данных лазерного сканирования.

Литература

1. SPOT DEM product description, 2005. http://www.spotimage.fr/automne_modules_files/standard/public/p807_fileLINKEDFILE_SPOT_DEM_Product_Description_v1-2.pdf.
2. *Rodriguez E., Morris C., Belz J., Chapin E., Martin J., Daffer W., Hensley S.* An assessment of the SRTM topographic products. Technical Report JPL D-31639. — Pasadena, California, 2005. 143 p. http://www2.jpl.nasa.gov/srtm/SRTM_D31639.pdf.
3. *Pfeifer N., Stadler P., Briese C.* Derivation of digital terrain models in the SCOP++ environment // OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models, Stockholm, 2001.
4. *Горькавый И. Н.* Метод виртуальной поверхности для классификации данных LIDAR и генерации трехмерной модели земного рельефа // Труды I Международной научно-практической конференции «Современные информационные технологии и ИТ-образование» / Под ред. В. А. Сухомлина. — М.: ВМК МГУ, 2005. С. 583–597.
5. *Forlani G., Nardinocchi C., Scaioni M., Zingaretti P.* Complete classification of raw LIDAR data and 3D reconstruction of buildings // Pattern Analysis & Applications, 2006. Vol. 8. No. 4. P. 357–374.
6. *Горькавый И. Н.* Комплексный подход к автоматизации процесса обработки данных LIDAR для получения инфракрасных изображений высокого разрешения // Известия высших учебных заведений. Геодезия и аэрофотосъемка, 2007. Вып. 5. С. 148–162.
7. *Горькавый И. Н.* Автоматизированные программные средства обработки трехмерных данных лазерного сканирования // Известия высших учебных заведений. Геодезия и аэрофотосъемка, 2008. Вып. 4. С. 22–34.
8. ASPRS guidelines: Vertical accuracy reporting for lidar data, 2004. http://www.asprs.org/society/committees/lidar/Downloads/Vertical_Accuracy_Reporting_for_Lidar_Data.pdf.
9. *Vosselman G.* Slope based filtering of laser altimetry data // Int. Arch. Photogrammetry Remote Sensing, 2000. Vol. 33.
10. *Sithole G.* Filtering of laser altimetry data using a slope adaptive filter // Int. Arch. Photogrammetry Remote Sensing, 2001. Vol. 34.
11. *Lohmann P., Koch A., Schaeffer M.* Approaches to the filtering of laser scanner data // Int. Arch. Photogrammetry Remote Sensing, 2000. Vol. 33.
12. *Kraus K., Pfeifer N.* Advanced DTM generation from LIDAR data // Int. Arch. Photogrammetry Remote Sensing, 2001. Vol. 34.
13. *Горькавый И. Н.* Программные средства и математические методы обработки и классификации трехмерных данных // Труды III Международной научно-практической конференции «Современные информационные технологии и ИТ-образование» / Под ред. В. А. Сухомлина. — М.: ВМК МГУ, 2008. С. 297–313.
14. *Jacobsen K., Lohmann P.* Segmented filtering of laser scanner DSMs // ISPRS WG III/3 Workshop. Dresden, 2003.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2009. Т. 3. Вып. 2. С. 65–81

СЕМИОТИЧЕСКАЯ МОДЕЛЬ ВЗАИМОСВЯЗЕЙ КОНЦЕПТОВ, ИНФОРМАЦИОННЫХ ОБЪЕКТОВ И КОМПЬЮТЕРНЫХ КОДОВ*

И. М. Зацман[1]

Аннотация: Рассматривается семиотическая модель, которая была разработана в процессе исследования проблем генерации и эволюции целевых систем знаний (ЦСЗ), а также отображения в электронных библиотеках и других видах информационных систем процессов их эволюции во времени и пространстве знаковых систем. Эти проблемы относятся к новому направлению исследований, получившему название «когнитивная информатика». Предлагаемая в статье семиотическая модель предназначена для описания взаимосвязей денотатов, концептов, информационных объектов и компьютерных кодов. Эта модель позиционируется как теоретическая основа для разработки методов трехкомпонентной кодировки денотатов, концептов и информационных объектов, в том числе построения взаимно однозначных отношений между концептами систем знаний человека и компьютерными кодами.

Ключевые слова: семиотическая модель; целевые системы знаний; денотаты; концепты; информационные объекты; компьютерные коды; трехкомпонентная кодировка; отношения между концептами и компьютерными кодами

1 Введение

Анализ работ [1–3] позволяет утверждать, что в настоящее время в информатике как в информационно-компьютерной науке [4] формируется новая область исследований, целью которых является разработка теоретических основ создания информационно-коммуникационных технологий (ИКТ), обеспечивающих процессы целенаправленного формирования (или другими словами, генерации) новых систем знаний. Целенаправленно формируемые новые системы знаний предлагается называть *целевыми*. Эта новая область исследований относится к когнитивной информатике[2], которая находится в начальной стадии описания ее предметной области и составляющих ее проблем [5–7].

Целью создания ИКТ, которые должны обеспечивать процессы формирования ЦСЗ, является решение как минимум трех актуальных проблем. Это:

(1) идентификация стадий формирования и эволюции ЦСЗ — *проблема идентификации ЦСЗ*;

(2) обеспечение анализа и оценивания степени релевантности разных вариантов ЦСЗ технологическим и другим общественно значимым потребностям, в интересах которых они были целенаправленно сформированы — *проблема релевантности ЦСЗ*;

(3) обеспечение средствами ИКТ целенаправленного влияния на процессы эволюции целевых систем знаний, формируемых в интересах технологических или других общественно значимых потребностей — *проблема направляемого развития ЦСЗ*.

В процессе постановки проблем идентификации, релевантности и направляемого развития ЦСЗ, а также исследования процессов генерации и эволюции ЦСЗ, ключевыми являются следующие вопросы:

(1) категоризация концептов и формулировка признаков, позволяющих зафиксировать границы между категориями личностных, коллективных и конвенциональных концептов, а затем использовать их при описании процессов генерации и эволюции ЦСЗ;

(2) разработка моделей и методов установления отношений между денотатами, концептами, информационными объектами и компьютерными кодами.

*Работа выполнена при частичной поддержке РФФИ, грант № 09-07-00156.

[1]Институт проблем информатики Российской академии наук, iz_ipi@a170.ipi.ac.ru

[2]Когнитивная информатика — направление в информатике как в информационно-компьютерной науке, которое при исследовании вычислительных процессов и разработке информационных систем использует методы когнитивной науки, изучающей ментальные процессы (познавательные и креативные) и ментальные объекты (концепты), а при исследовании форм представления концептов, их эволюции, познавательных и креативных процессов использует методы информатики.

Статья посвящена второму вопросу. Первый вопрос — категоризация концептов — был рассмотрен в работах [8, 9]. В этих работах в процессе описания трех категорий концептов использовалась система базовых терминов информатики, построенная на основе результатов работ [10–14]. В предложенном подходе к категоризации концептов лексически акцентированы различия между этими тремя категориями и сформулированы их отличительные признаки. Предложенная категоризация концептов на личностные, коллективные и конвенциональные не является только терминологическим результатом, так как определение трех категорий концептов и их признаков необходимо для описания новых направлений исследований и разработок, относящихся к проблематике генерации и эволюции ЦСЗ.

Целью статьи является описание взаимосвязей денотатов, концептов, информационных объектов и компьютерных кодов в виде семиотической модели с использованием системы базовых терминов, определенной в работах [8, 9]. В процессе создания этой модели была использована идея распределения объектов предметной области информатики по разным сферам (средам) в зависимости от их природы (например, ментальной и материальной сфер, социальной и цифровой сред). Разные формы изложения этой идеи, использованные в широком спектре проблем информатики, можно найти в работах [15–18].

Основная идея статьи заключается в том, что для каждой из трех вершин семиотического треугольника Г. Фреге[1] [19–21] предлагается использовать свою систему кодировки, в том числе и для вершины концептов. Предлагаемое введение специальной системы кодировки для концептов, которые в семиотическом треугольнике являются значениями знаков, является необходимым условием построения взаимно однозначных отношений между концептами и компьютерными кодами. Построение таких отношений является основой для решения проблем идентификации, релевантности и направляемого развития ЦСЗ, а также для решения задач концептуального индексирования и семантического поиска [17, 22].

Семиотическая модель, предлагаемая в статье, позиционируется как теоретическая основа для разработки методов трехкомпонентной кодировки[2] всех трех вершин семиотического треугольника, в том числе построения взаимно однозначных отношений между концептами систем знаний человека и компьютерными кодами.

Взаимно однозначные отношения между концептами и компьютерными кодами существенно отличаются от взаимно однозначных отношений между литерами и цифровыми двоичными кодами, которые устанавливаются с помощью различных таблиц кодировки множества литер (ASCII — American Standard Code for Information Interchange, EBCDIC — Extended Binary Coded Decimal Interchange Code, UNICODE и т. д.). Это отличие проиллюстрируем на примере слов естественного языка. Если нужно закодировать слово с помощью таблицы кодировки, то оно разделяется на литеры, которые заменяются кодами в соответствии с выбранной таблицей. Отметим, что каждая литера слова в отдельности не является носителем смыслового содержания. Однако вся последовательность литер, образующих одно слово, например «поле», может иметь несколько смысловых значений (концептов) в зависимости от контекста (ржаное поле, футбольное поле, магнитное поле и т. д.).

Следовательно, с помощью таблиц кодировки множества литер могут кодироваться только формы знаков, но не их значения, т. е. таблицы кодировки применимы только к одной вершине семиотического треугольника Г. Фреге. В информационных системах для обозначения именно концептов, но не форм их представления используется, как правило, следующий подход. Сначала (первый этап) концепт как значение некоторого знака обозначают с помощью формы этого знака, а затем (второй этап) форме знака ставится в соответствие некоторый компьютерный код.

Так как естественные языки как знаковые системы обладают свойством асимметрии [23] и построенное на первом этапе соответствие концептов и форм их представления в общем случае является многозначным, то задача построения взаимно однозначных отношений между концептами и компьютерными кодами *через кодировку форм представления концептов* (но не самих концептов напрямую) в общем виде становится трудноразрешимой.

Однако при решении прикладных задач явление асимметрии знаковых систем может быть частично учтено, если на первом этапе концепту как значению знака с помощью программы семантического анализа текста ставится в соответствие не форма некоторого знака, а тот дескриптор тезауруса, который соответствует именно этому концепту [22].

[1]Три вершины семиотического треугольника Г. Фреге — это значение знака (его концепта), формы знака (как частный случай информационного объекта) и денотата знака (материальной, цифровой или иной природы).

[2]Под трехкомпонентной кодировкой, которая далее будет рассмотрена подробнее, понимается присвоение компьютерных кодов трем вершинам семиотического треугольника (денотату, концепту как значению знака и информационному объекту как форме знака).

Это возможно реализовать в том случае, если в информационных системах для кодировки концептов используются тезаурусы, поддерживающие семантические отношения между дескрипторами и фиксирующие разные виды знаковой асимметрии. Например, в тезаурусе с помощью дескрипторов могут быть обозначены (представлены) несколько концептов, имеющих одну и ту же форму представления (например, слово «поле»). При этом каждый концепт будет представлен только одним дескриптором, соответствующим *только одному значению слова «поле»*. Таким образом, использование тезаурусов при решении прикладных задач в явно определенной предметной области позволяет устанавливать взаимно однозначные отношения между концептами и компьютерными кодами.

Предлагаемая семиотическая модель при ее прикладном применении ориентирована в основном на использование тезаурусного подхода к кодировке концептов с помощью компьютерных кодов, но с теоретической точки зрения она допускает использование иных подходов, а не только тезаурусного (см. описание способа кодировки в микрокомпьютере Altair 8800 в разд. 3).

2 Система базовых терминов

В работах [8, 9] дано описание тех 12 базовых терминов информатики, которые необходимы для этой статьи и перечислены на рис. 1. На этом рисунке они пронумерованы против часовой стрелки. Для целей настоящей статьи понадобятся следующие четыре ключевые положения из этого описания.

Во-первых, в этих работах термин *коды* был определен как компьютерные эквиваленты литер двоичных цифр (или их последовательностей), которые могут представлять собой намагниченность или ее отсутствие в цифровой среде, наличие электрического тока или его отсутствие, способность к отражению света или ее отсутствие. Литеры «0» и «1», о которых говорится в дефиниции термина *коды*, по определению являются сущностями *среды социальных коммуникаций* между пользователями информационной системы (ИС), а их компьютерные эквиваленты — сущностями *цифровой среды* ИС.

Во-вторых, среди всех возможных компьютерных кодов цифровой среды в работах [8, 9] были выделены три следующие категории, необходимые для этой статьи:

(1) коды, соотнесенные в явном виде с концептами знаний пользователей ИС, — *коды первой категории, или семантические коды*;

(2) коды, соотнесенные с эксплицитными и отчужденными от человека сенсорно воспринимаемыми знаковыми формами представления элементарных концептов (формы знаков) и

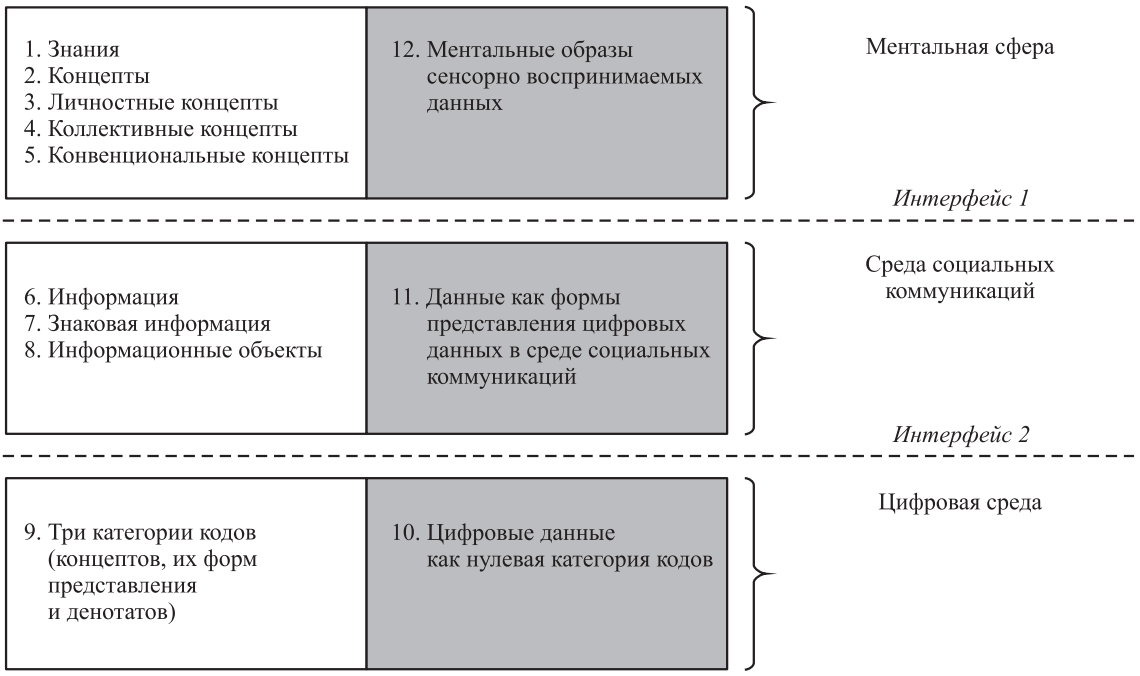


Рис. 1 Система базовых терминов (определения всех перечисленных 12 терминов приведены в работах [8, 9])

сложных концептов (например, фразы на естественном языке) в среде социальных коммуникаций ИС, — *коды второй категории, или информационные коды*;

(3) коды, соотнесенные с материальными объектами и явлениями, а также с алгоритмами, программами и другими видами денотатов; эти коды являются компьютерными идентификаторами денотатов в цифровой среде ИС — *коды третьей категории, или объектные коды*.

В-третьих, была определена еще одна *нулевая категория кодов*, названная *цифровыми данными*, к которой были отнесены все компьютерные коды цифровой среды ИС, не относящиеся в явном виде к трем вышеопределенным категориям. К цифровым данным относятся результаты любых измерений, полученных с помощью цифровых технических систем (устройств) и хранящихся в ИС, а также *результаты любых вычислений*, не являющиеся итогом генерации и представления знаний пользователями ИС.

И последнее, четвертое, положение заключается в том, что все 12 терминов разделены на три группы в зависимости от природы обозначаемых ими сущностей, ментальной, социальной или цифровой [24]:

(1) знания, ментальные образы сенсорно воспринимаемых данных, концепты как родовой термин, личностные, коллективные и конвенциональные концепты как виды этого родового термина (сущности ментальной сферы, т. е. сферы знаний пользователей ИС);

(2) информация, данные, знаковая информация, информационные объекты (сущности среды социальных коммуникаций пользователей ИС);

(3) компьютерные коды всех категорий (сущности цифровой среды ИС).

Четыре рассмотренных положения дополним двумя небольшими пояснениями. Во-первых, *денотатами материальной природы* являются любые физические объекты, названия которых присутствуют в естественных языках, в то время как *денотатами цифровой природы* являются такие сочетания компьютерных кодов, которые имеют уникальные названия в рамках ИС (например, цифровые программные объекты), а эти названия могут использоваться разработчиками ИС при профессиональных коммуникациях. Во-вторых, некоторый физический объект может одновременно являться еще и носителем формы знака, например круг из жести или картона с дорожным знаком «проезд запрещен».

Таким образом, с одной стороны, перечисленные 12 терминов разделены на три группы в зависимости от природы обозначаемых ими сущностей, с другой стороны, они разделены на два класса относительно первоначального источника их возникновения: человека, создающего знания, и технической системы как генератора цифровых данных. Первый класс терминов обозначен на рис. 1 белым фоном, второй класс — серым.

Разделение 12 терминов на два класса основано на следующем теоретическом положении Брукса: информация не является идентичной сенсорным данным. Процесс создания информации может зависеть от результатов наблюдений или измерений, но данные, полученные таким образом, должны быть сначала интерпретированы когнитивными структурами знаний человека, чтобы затем стать концептами[1] [25, 26].

Таким образом, данные как результат наблюдений или измерений по определению не являются знаковой информацией, так как не являются формами вербальных и/или невербальных знаков или знаковых образований, отчужденными от участников социальных коммуникаций, т. е. не являются формами представления знаний человека в сфере социальных коммуникаций.

Однако данные могут быть *интерпретированы* или *концептуализированы* с помощью системы знаний человека. Иначе говоря, на основе анализа данных человеком сначала могут быть сгенерированы концепты в рамках некоторой системы его знаний, используемой для их интерпретации, а затем эти концепты могут быть выражены в виде отчужденных форм представления концептов, т. е. в виде знаковой информации.

Приведенные в этом разделе положения позволяют уточнить формулировку цели статьи: предлагается семиотическая модель для описания взаимосвязей денотатов, концептов, информационных объектов и компьютерных кодов всех категорий, *кроме нулевой*. Нулевая категория компьютерных кодов отсутствует в семиотической модели по следующей причине. Основная цель построения семиотической модели заключается в создании теоретической основы для разработки методов трехкомпонентной кодировки денотатов, концептов и информационных объектов, которым по определению соответствуют только коды трех категорий, т. е. нулевая категория кодов для них не

[1] Результаты интерпретации данных наблюдений или измерений Брукс называет не концептами, а ментальной информацией как структурной составляющей систем знаний человека. Однако это различие в названиях является не понятийным, а чисто лексическим, т. е. одно и то же рассматриваемое понятие по-разному называется в этой статье и в работе Брукса.

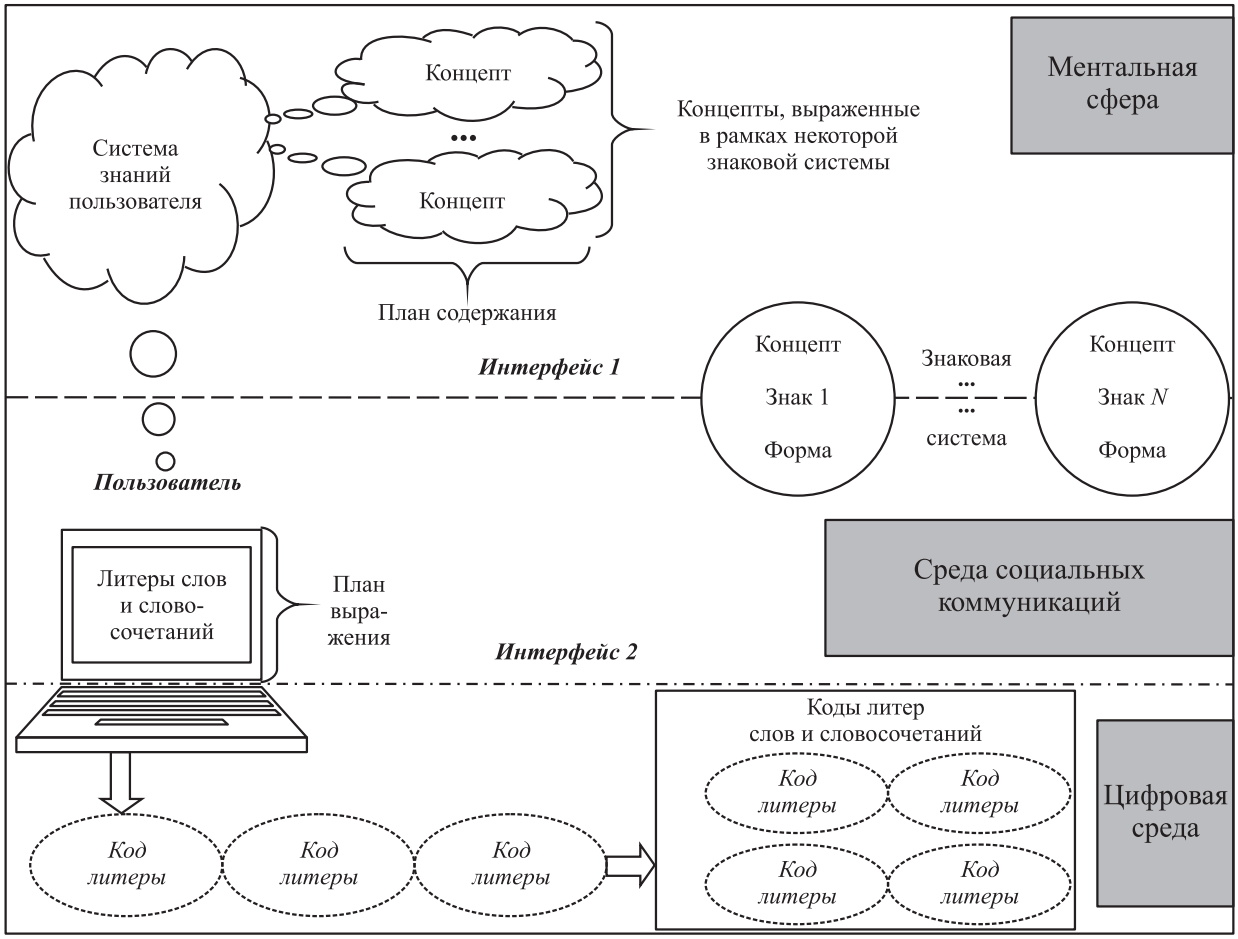


Рис. 2 Два интерфейса между сущностями ментальной сферы, социальной и цифровой сред

применяется. Иначе говоря, в предлагаемой семиотической модели, являющейся развитием семиотического треугольника Г. Фреге, коды нулевой категории исключены из рассмотрения, так как для трех вершин треугольника достаточно иметь только три категории кодов для денотатов, концептов и информационных объектов[1].

3 Концепты, информационные объекты и компьютерные коды

Традиционные таблицы кодировки множества литер непосредственно применимы только к тем информационным объектам, которые являются *линейными конкатенациями литер или символов*. Если концепты могут быть представлены в виде таких информационных объектов как форм знаков и разделены на литеры (I этап), то затем этим литерам могут быть поставлены в соответствие компьютерные коды (II этап).

Рисунок 2 иллюстрирует двухэтапный процесс назначения компьютерных кодов концептам, которые являются значениями вербальных знаков. Сначала концепты выражаются с помощью форм знаков (реализация интерфейса 1), т. е. словами или устойчивыми словосочетаниями. Затем слова делятся на литеры и каждой литере ставится в соответствие компьютерный код этой литеры (реализация интерфейса 2). Знаки как двуединые сущности, каждая из которых включает концепт и форму его представления, обозначены в виде кругов, размещенных на линии интерфейса 1 (см. рис. 2). При таком двухэтапном процессе необходимо учитывать явление асимметрии, о чем говорилось в разд. 1.

Рассмотрим более подробно интерфейсы 1 и 2.

[1]Однако для постановки и решения задач формирования концептов на основе уже имеющихся цифровых данных [8], которые здесь не рассматриваются, понадобится другая семиотическая модель, включающая коды всех категорий, в том числе нулевой.

На рис. 1 и 2 показаны два интерфейса между сущностями ментальной сферы, социальной и цифровой сред:

- интерфейс 1 между значениями знаков (концептами ментальной сферы) и формами знаков (информационными объектами среды социальных коммуникаций) условно обозначен штриховой линией;
- интерфейс 2 между литерами, формами знаков (информационными объектами) и компьютерными кодами цифровой среды обозначен штрихпунктирной линией.

На рис. 2 выше штриховой линии изображена ментальная сфера, ниже штрихпунктирной линии — цифровая среда. Между ними — среда социальных коммуникаций. Слева на этом рисунке изображен компьютер. Пользователь компьютера представляет свои концепты в виде вербальных текстов, а компьютер переводит литеры слов в коды ИС.

Первый интерфейс традиционно исследуют в таких областях знаний, как семиотика, лингвистика, информационная наука, информатика как информационно-компьютерная наука [4, 27–31]. Согласно Шрейдеру, «наиболее принципиальные вопросы информатики всегда возникали *на стыке информации и знания* (курсив — **И. З.**) — там, где речь шла о превращении одного в другое» [29, с. 50].

Описание первого интерфейса в семиотике и лингвистике часто ведется в терминах «план содержания», «план выражения», «знак», «значение знака», «форма знака» и «знаковая система». В качестве примера описания первого интерфейса процитируем основное положение концепции, лежащей в основе «Толково-комбинаторного словаря современного русского языка» (ТКС)[1]. Главная идея создания этого словаря состоит в том, что «естественный язык есть система, устанавливающая соответствие между заданным смыслом и всеми выражающими его текстами; соответственно, лингвистическое описание некоторого языка должно представлять собой множество правил, ставящих в соответствие *всякому смыслу все тексты данного языка, несущие этот смысл*» (курсив — **И. З.**) [32, с. 4].

Естественный язык по определению является знаковой системой, но все естественные языки являются только одной из категорий знаковых систем. В качестве примера еще одной категории знаковых систем можно назвать язык карты и другие геоязыки [33]. Согласно Барту, знаковые системы всех категорий объединяет их возможность членить знания человека на концепты и устанавливать соответствие между множеством концептов и множеством форм их представления в виде вербальных и/или невербальных текстов [34].

В общем случае соответствие между этими двумя множествами, обеспечиваемое знаковой системой, обладает двумя особенностями. *Во-первых*, со временем может изменяться «объем значения» любого концепта, например выражаемого словом или устойчивым словосочетанием естественного языка. *Во-вторых*, разные знаковые системы различаются по способам и правилам членения систем знаний на концепты. В силу этих особенностей «объемы значений» практически любых соотносимых в двуязычных словарях пар слов не совпадают [35].

Таким образом, именно знаковые системы являются основой для реализации первого интерфейса. Именно они традиционно обеспечивают между концептами и информационными объектами как формами знаков взаимосвязи, обладающие двумя перечисленными особенностями. Отметим, что содержание рис. 2 имеет два существенных отличия от содержания приведенного положения концепции ТКС.

Во-первых, в общем случае при реализации первого интерфейса в качестве информационных объектов могут выступать не только слова и словосочетания текстов на естественных языках, но и изображения (образные тексты) — графики, диаграммы, рисунки, карты и т. д.

Во-вторых, на рис. 2 первый интерфейс рассматривается в сочетании со вторым интерфейсом — между литерами и компьютерными кодами цифровой среды.

Второй интерфейс в информатике как компьютерной науке часто изучается независимо от первого. При этом информационные объекты в процессе изучения, как правило, изначально определяются не как формы представления концептов, а как последовательности литер некоторого алфавита или заданного множества символов. Эти последовательности в частном случае могут представлять собой фразы на естественном языке. При постановке и решении широкого спектра проблем информатики как компьютерной науки последовательности литер используются в качестве исходных данных

[1] ТКС — это словарь синтеза, его цель — помочь построить текст, т. е., во-первых, дать читателю словаря возможность найти все мыслимые формы выражения нужной ему идеи (концептов) и отобрать среди них те, которые наилучшим образом подходят к данному контексту, а во-вторых, указать правильный способ сочетания выбранных форм. ТКС можно использовать и в обратном направлении, т. е. от текста к смыслу (концептам), но его организующий принцип — это движение от смысла к тексту (сочетанию информационных объектов) [32, с. 5].

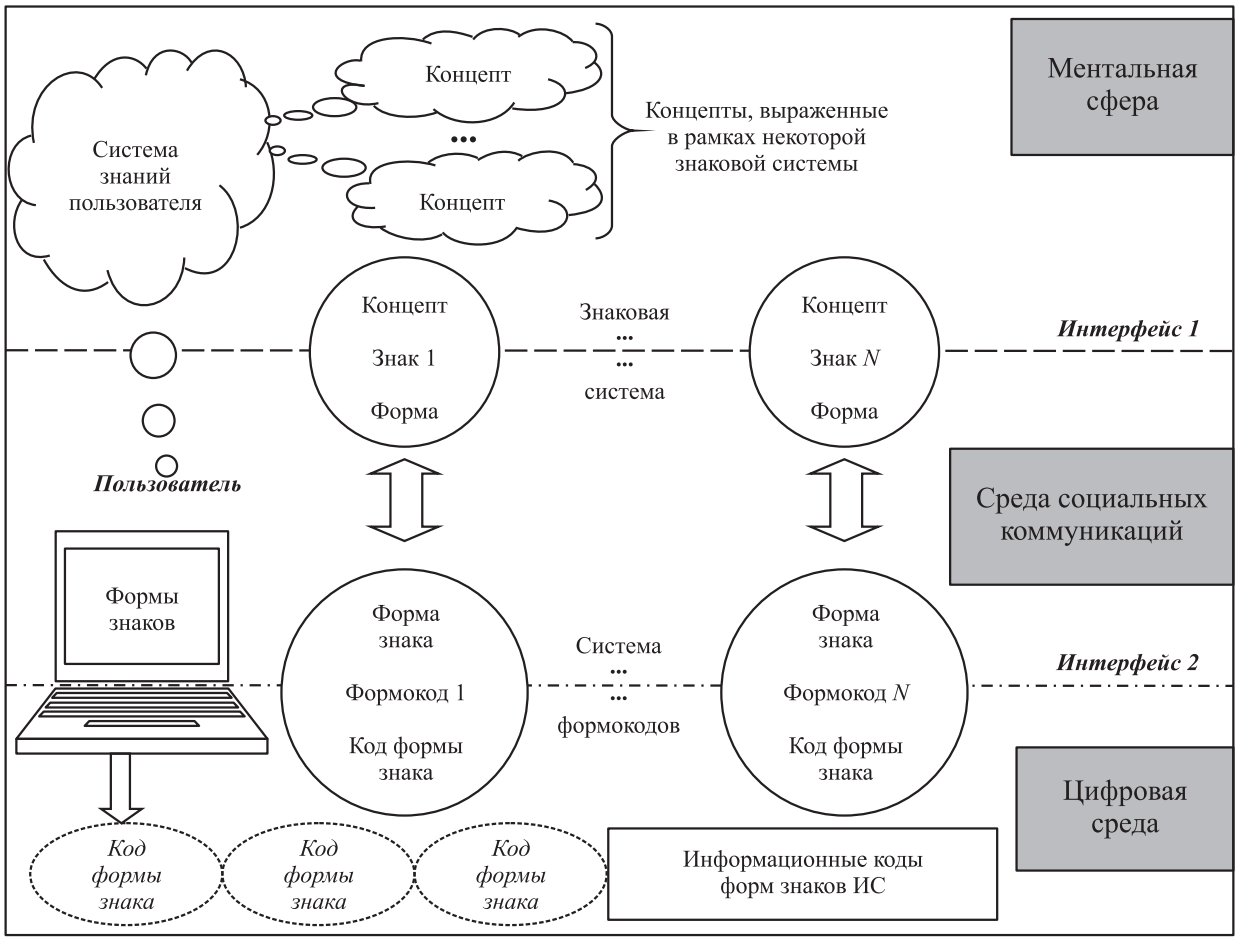


Рис. 3 Два интерфейса, система формокодов и знаковая система

как в классических работах [36], так и в современных [37].

Для любой фразы естественного языка литеры составляющих ее слов хранятся в памяти компьютера в виде кодов. При отображении слов на экране монитора и/или при печати их на бумаге используются таблицы кодировки, которые обеспечивают соответствие между этими компьютерными кодами и литерами слов. Иначе говоря, эти таблицы обеспечивают второй интерфейс методом кодировки литер. Естественно, что для кодирования лексических значений слов и смысла фраз естественного языка эти таблицы кодировки не предназначены.

Теоретически имеется возможность кодировать не отдельные литеры вербального текста, а каждое слово (форму вербального знака) этого текста как единое целое, если в ИС в дополнение к таблицам кодировки литер использовать систему формокодов, подробное описание которой дано в работе [38].

Отметим, что по определению из работы [38] *формокодом* называется общепринятое или нормативно-заданное сочетание кода ИС и информационного объекта, являющегося формой знака. На рис. 3 показано, что каждый знак является двуединой сущностью, включающей концепт как значение знака и форму знака, а каждый формокод является двуединой сущностью, включающей форму знака и код этой формы. Знаки обозначены в виде кругов, размещенных на линии интерфейса 1, а формокоды — в виде кругов, размещенных на линии интерфейса 2. Формокодовая кодировка, т. е. кодировка форм знаков компьютерными кодами, принципиально отличается от кодировки литер, так как формокодовая кодировка применима к более широкому спектру знаковых систем, в том числе к образным знаковым системам, что будет показано в разд. 5. Таким образом, рис. 3 отличается от рис. 2 использованием формокодовой системы для кодировки форм знаков, с помощью которой могут быть сформированы информационные коды форм знаков.

Если бы в знаковых системах отсутствовало явление знаковой асимметрии, то каждому элемен-

тарному концепту соответствовала бы только одна форма его представления, и наоборот. В таком случае уникальный в пределах ИС код формы знака можно было бы использовать и как код соответствующего этой форме концепта. Следовательно, можно было бы поставить во взаимно однозначное соответствие концепт как значение знака и компьютерный код. Однако из-за асимметрии знаковых систем (например, явления синонимии и полисемии [39] в естественных языках) соответствие концептов и форм их представления в общем случае является многозначным.

Как уже было отмечено, отсюда следует важный вывод о том, что задача построения взаимно однозначных отношений между концептами и компьютерными кодами *через кодирование форм представления концептов* (но не самих концептов или дескрипторов тезауруса) в общем случае является *трудноразрешимой*.

Если в ИС используется тезаурус, то система формокодов может быть построена на его основе с использованием дескрипторов тезауруса. Этот способ построения систем формокодов, использующий тезаурус, является более универсальным, так как он применим для форм вербальных и образных знаков, не являющихся линейными последовательностями литер, символов и других знаковых примитивов [38].

4 Третий интерфейс и семиотическая модель

Обсудив вопросы использования формокодовой системы для кодировки форм знаков, с помощью которой могут быть сформированы информационные коды форм знаков, а также ее отличий от таблиц кодировок множества литер, рассмотрим задачу формирования семантических кодов концептов. Для решения этой задачи необходимо рассмотреть еще один интерфейс между концептами как значениями знаков и компьютерными кодами.

До сих пор рассмотрение двух интерфейсов велось с использованием рис. 2 и 3, с помощью которых иллюстрировались отличия формокодовой системы для кодировки форм знаков от таблиц кодировок множества литер. Однако существует еще один интерфейс — третий — между концептами ментальной сферы как значениями знаков и компьютерными кодами, который на этих рисунках не был изображен. На рис. 4 приведены все три интерфейса, среди них и интерфейс 3 между ментальной сферой и цифровой средой, который обозначен сплошной двойной линией.

На рис. 4 показано, что, кроме информационных кодов форм знаков, формируются семантические коды концептов. Две стрелки, расположенные ниже компьютера, условно обозначают процесс кодировки форм знаков и концептов. Для формирования кодов концептов необходимо определить еще одну двуединую сущность, аналогичную формокоду, что будет сделано чуть позже. Как уже отмечалось, по определению коды первой категории ИС соответствуют концептам, коды второй категории ИС — формам представления концептов.

Для формирования кодов первой и второй категории используется тезаурус. Описания нескольких тезаурусов и особенностей их формирования можно найти в работах [40–44]. Отметим, что на рис. 4 денотаты и коды третьей категории не показаны.

Примером реализации третьего интерфейса без использования тезауруса является первый коммерчески доступный микрокомпьютер MITS Altair 8800. В базовом комплекте поставки у него не было никаких периферийных устройств в современном понимании. На микрокомпьютере можно было программировать, задавая определенные пользователем наборы двоичных последовательностей вручную. Для их ввода в этот микрокомпьютер использовались двухпозиционные переключатели, расположенные на передней панели компьютера. С помощью одного переключателя можно было ввести только одну двоичную цифру (ноль или единицу). Результат своей работы микрокомпьютер показывал, включая или выключая лампочки на передней панели[1]. С помощью двухпозиционных переключателей и лампочек в этом микрокомпьютере был реализован простейший вариант третьего интерфейса[2].

Этот пример человеко-машинного интерфейса интересен тем, что человек мог почти напрямую взаимодействовать с кодами компьютера, используя только двоичные переключатели и лампочки на передней панели как формы представления компьютерных кодов вне цифровой среды микрокомпьютера.

Вернемся к рис. 4, на котором третий интерфейс обозначен двойной сплошной линией. Выше этой двойной линии находится ментальная сфера, ниже — цифровая среда. Для описания третьего интерфейса предлагается применить систему семо-

[1]Двухпозиционные переключатели можно трактовать как промежуточную среду перехода от концептов пользователя микрокомпьютера MITS Altair 8800 к кодам. Однако эту среду не будем учитывать при рассмотрении третьего интерфейса.

[2]Описание микрокомпьютера MITS Altair 8800 и его цветную фотографию можно найти по адресу http://historywired.si.edu/object.cfm?ID=339.

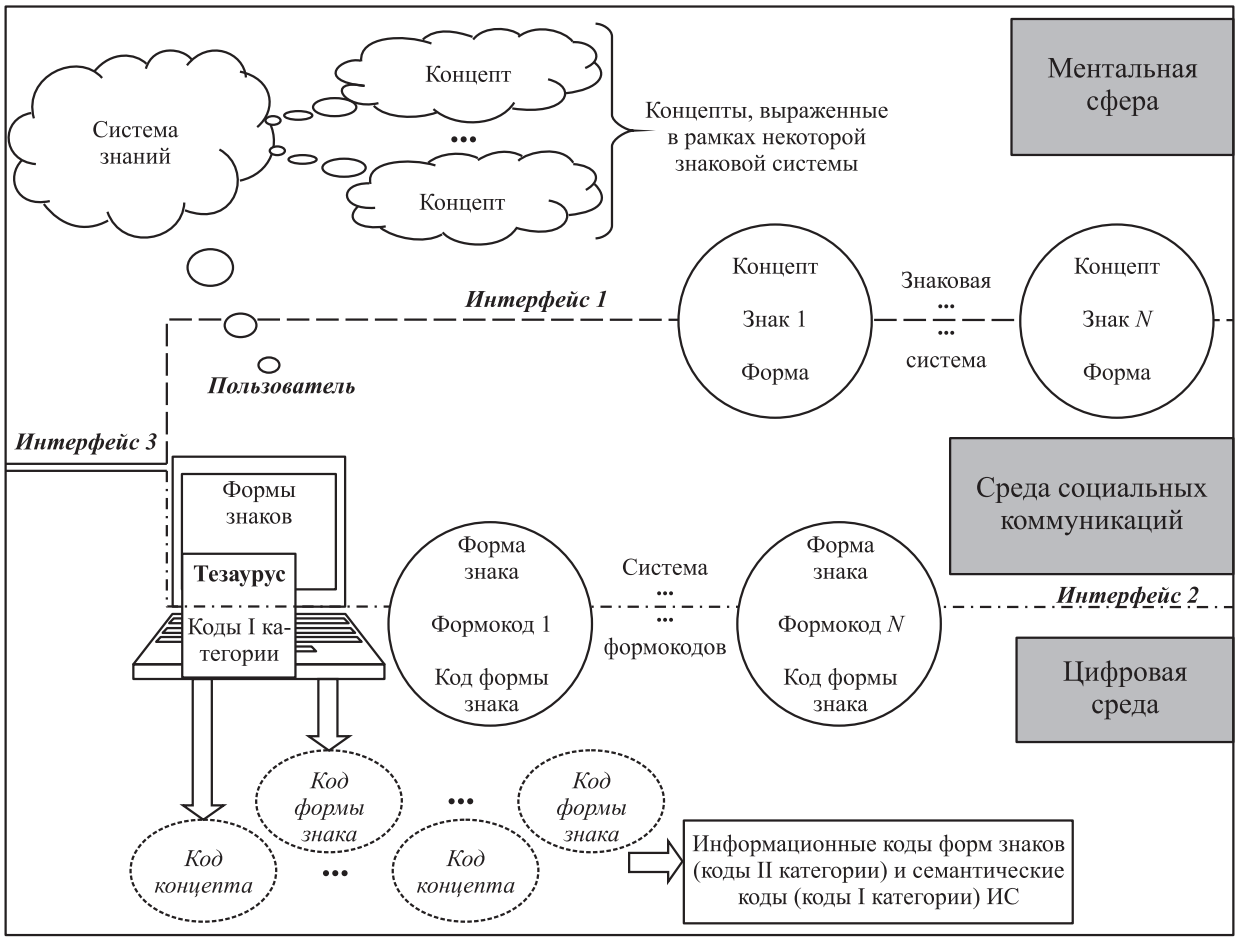


Рис. 4 Три интерфейса

кодов (рис. 5). Отметим, что по определению из работы [38] *семокодом* называется общепринятое или нормативно-заданное сочетание в виде бинарного отношения концепта как значения знака и компьютерного кода первой категории. Семокодовая кодировка используется для формирования семантических кодов концептов ИС (см. рис. 4 и 5).

Подводя итоги рассмотрения трех интерфейсов ИС, процессов формирования информационных и семантических кодов, показанных на рис. 4 и 5, построим табл. 1, которая содержит:

- названия ментальной сферы, социальной и цифровой сред (в первой строке и в первой колонке);
- моносферные[1] термины «концепт (значение знака)», «информационный объект (форма знака)» и «компьютерный код» (расположены на главной диагонали, кроме первой ячейки первой строки);
- полисферные[2] термины «знак», «формокод» и «семокод», которые являются по своей природе двуединими сущностями (в остальных ячейках таблицы).

Таблица 1 построена только в целях компактного описания принадлежности моносферных терминов к одной сфере (среде), а полисферных — к нескольким.

Для построения семиотической модели будем дополнительно рассматривать сферу материальных объектов и денотаты, а также учитывать определенное в разд. 2 разделение компьютерных кодов концептов, информационных объектов и денотатов на три категории. Рассмотрение сферы материальных объектов и принадлежащих ей денотатов обусловлено тем, что в ряде задач возникает необходимость формирования кодов денотатов в дополнение к кодам форм знаков и концептов.

[1]Моносферными будем называть те термины, которые обозначают сущности, относящиеся только к одной сфере или среде.
[2]Полисферными будем называть те термины, которые обозначают сущности, относящиеся более чем к одной сфере или среде.

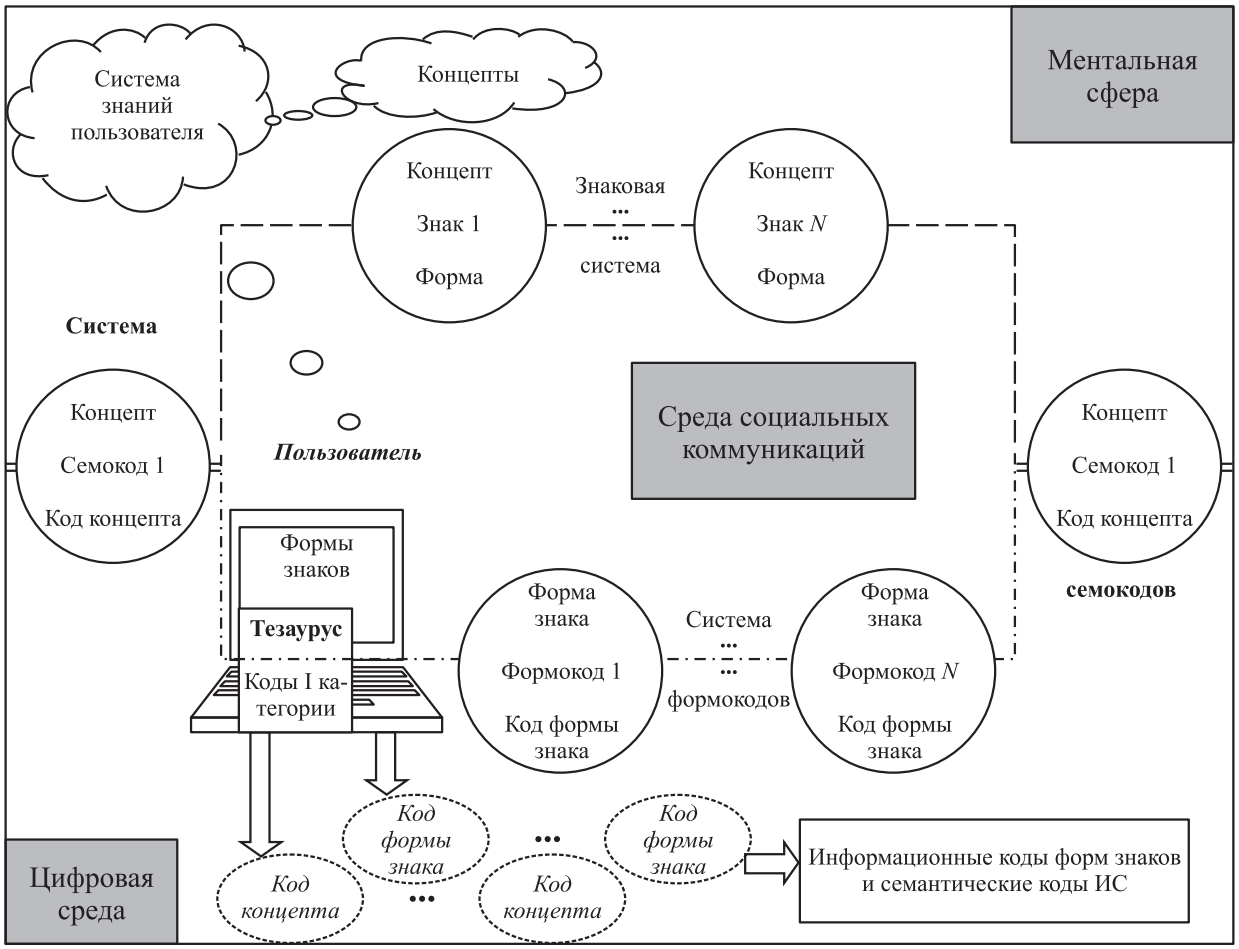


Рис. 5 Три интерфейса и три двуединых сущности: знак, семокод и формокод

Таблица 1 Моносферные и полисферные термины

Названия одной сферы и двух сред → ↓	Ментальная сфера (знаний человека)	Среда социальных коммуникаций	Цифровая среда
Ментальная сфера (знаний человека)	*Концепт (значение знака)*	Знак	Семокод
Среда социальных коммуникаций	Знак	*Информационный объект (форма знака)*	Формокод
Цифровая среда	Семокод	Формокод	*Компьютерный код*

В процессе построения семиотической модели для описания взаимосвязей денотатов, концептов, информационных объектов и компьютерных кодов всех категорий, кроме *нулевой*, будем использовать рис. 6. На нем кроме ментальной сферы, социальной и цифровой сред изображена сфера материальных объектов и явлений, а также один денотат как физический объект или явление.

Отметим, что в общем случае денотат может принадлежать и к другим средам, например цифровой. Так в описании проблем направляемой генерации и эволюции ЦСЗ рассматриваются денотаты, представляющие собой совокупности цифровых данных [8, 9], которые по определению из разд. 2 относятся к цифровой среде.

Таким образом, предлагаемая в статье семиотическая модель включает пять составляющих. Это:

(1) ментальная сфера, материальная сфера физических объектов и явлений, социальная и цифровая среды;

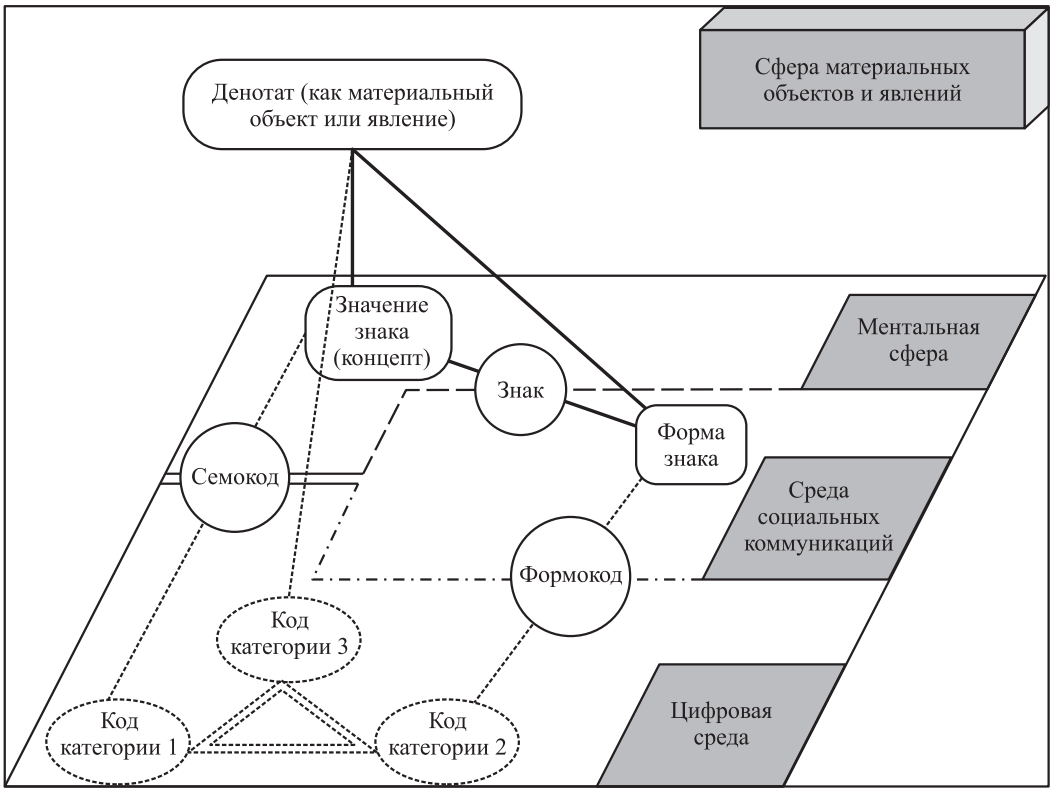


Рис. 6 Семиотическая модель для описания взаимосвязей денотатов, концептов, информационных объектов как форм знаков и компьютерных кодов

(2) денотат, соответствующие ему концепт (значение знака) ментальной сферы и информационный объект (форма знака) социальной среды, а также компьютерные коды трех категорий цифровой среды для денотата, концепта и информационного объекта;

(3) знак, объединяющий концепт как значение знака и информационный объект как форму знака, являющийся элементом знаковой системы ИС;

(4) формокод, объединяющий информационный объект как форму знака и компьютерный код второй категории, являющийся элементом формокодовой кодировки для формирования информационных кодов форм знаков ИС;

(5) семокод, объединяющий концепт как значение знака и компьютерный код первой категории, являющийся элементом семокодовой кодировки для формирования семантических кодов концептов ИС.

Естественно, что рассматриваемая семиотическая модель основана на треугольнике Г. Фреге, тремя вершинами которого являются значение знака (концепт), форма знака (как частный случай информационного объекта) и денотат знака (материальной, цифровой или иной природы). Новыми компонентами в этой модели являются формокод и семокод как ключевые элементы формокодовой и семокодовой кодировок, а также три категории кодов. Они образуют в цифровой среде треугольник, который предлагается называть «компьютерным треугольником». Включение в модель формокода, семокода и кодов трех категорий позволяет разрабатывать принципиально новые методы трехкомпонентной кодировки, обеспечивающие в дополнение к литерной (символьной) кодировке формирование:

– информационных кодов форм знаков с помощью формокодовой кодировки;

– семантических кодов концептов с помощью семокодовой кодировки;

– объектных кодов денотатов (в статье определены коды денотатов, но система их кодировки не рассматривается).

В следующем разделе приводится пример построения фрагментов формокодовой и семокодовой кодировок для электронной библиотеки геоизображений из работы [38]. Эти фрагменты

применяются для формирования кодов форм вербальных и образных знаков с помощью формокодовой кодировки и кодов концептов геообъектов, являющихся устьевыми областями рек, с помощью семокодовой кодировки. С помощью семокодовой кодировки в пределах электронной библиотеки геоизображений обеспечивается взаимно однозначное отношение между концептами геообъектов, являющихся устьевыми областями рек, и компьютерными кодами.

5 Пример трехкомпонентной кодировки

Рассмотрим пример построения трехкомпонентной кодировки, основанной на рассмотренной семиотической модели. Эта кодировка была разработана для электронной библиотеки геоизображений в целях описания взаимосвязей шести геообъектов (денотатов), соответствующих им концептов и информационных объектов с компьютерными кодами трех категорий [38].

Для шести геообъектов, являющихся устьевыми областями рек, был разработан фрагмент вербально-образного тезауруса. В нем каждый из шести геообъектов (денотатов) и соответствующий ему концепт были представлены *тремя дескрипторами*: для названия геообъекта на русском языке, названия на английском языке и для изображения геообъекта, которое является стилизованной пиктограммой соответствующей устьевой области реки.

В соответствии с определением семиотическая модель для этого примера включает такие компоненты, как:

- ментальная сфера знаний пользователей электронной библиотеки геоизображений, материальная сфера физических объектов и явлений (шесть геообъектов, являющихся устьевыми областями рек), социальная и цифровая среды электронной библиотеки;
- геообъект (денотат), соответствующие ему *один концепт* ментальной сферы и *три информационных объекта* социальной среды (название на русском языке, название на английском языке и изображение геообъекта), а также компьютерные коды трех категорий цифровой среды для денотата, концепта и информационных объектов;
- знак, объединяющий концепт как значение знака и информационный объект как форму знака, являющийся элементом знаковой системы электронной библиотеки;
- формокод, объединяющий информационный объект как форму знака и компьютерный код второй категории, а также используемый для формирования *трех видов информационных кодов* форм знаков электронной библиотеки;
- семокод, объединяющий концепт как значение знака и компьютерный код первой категории, а также используемый для формирования *семантических кодов* концептов электронной библиотеки.

Коды денотатов в этом примере не используются. Для построения трех видов информационных кодов форм знаков (для названий на русском языке, названий на английском языке и изображений геообъектов) и семантических кодов концептов электронной библиотеки в качестве исходных данных будем использовать классификацию устьевых областей рек и их частей по морфологическим признакам из работы [45]. Отметим, что на основе именно этой классификации в вербально-образном тезаурусе электронной библиотеки ранее были построены вербальные и образные дескрипторы для шести рассматриваемых геообъектов, в том числе их связи с другими дескрипторами тезауруса (рис. 7).

Основные 6 видов устьевых областей рек приведены в табл. 2, в третьем столбце которой их названия выделены курсивом. Кроме этой таблицы в классификации используются 6 стилизованных пиктограмм устьевых областей рек, которые приведены в нижней части рис. 7. Построение трехкомпонентной кодировки осуществляется в три этапа.

На первом этапе в целях формирования трех видов информационных кодов форм знаков образуем группу из 12 вербальных и шести образных знаков для описания шести геообъектов, являющихся устьевыми областями рек. В примере будет использоваться только эта группа из 18 знаков, а не вся знаковая система электронной библиотеки. Рисунок 7 и табл. 2 позволяют получить первое представление о рассматриваемой группе знаков, используемых для описания геообъектов. В общем случае *первый принцип* построения трехкомпонентной кодировки заключается в конструктивном описании знаковой системы электронной библиотеки, включающем ответы на следующие вопросы:

1. Какие естественные языки используются для названий денотатов (в примере используются русский и английский языки)?
2. Что должен включать словарь электронной библиотеки для каждого естественного языка (в примере словарь каждого языка включает шесть названий устьевых областей рек)?

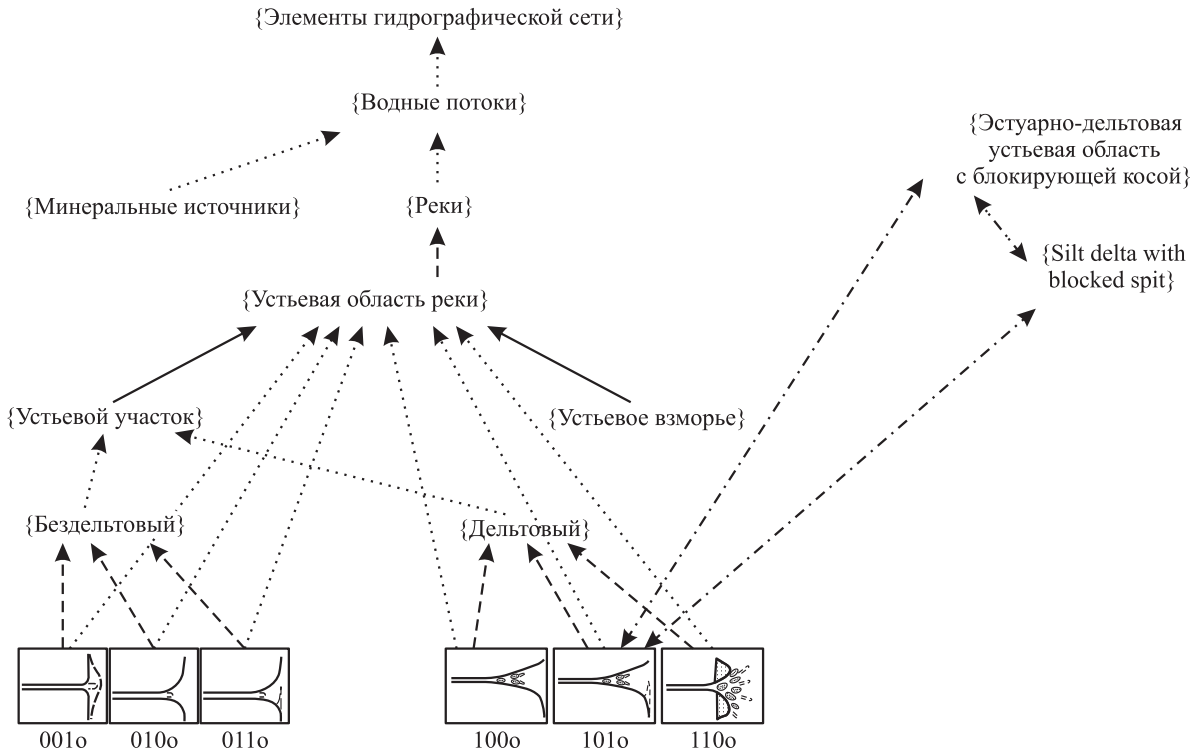


Рис. 7 Фрагмент вербально-образного тезауруса

Таблица 2 Классификация устьевых областей рек [45]

Устьевой участок реки	Устьевое взморье	Устьевая область реки
Однорукавный (бездельтовый)	Открытое без блокирующей косы	*Простая без блокирующей косы*
	Полузакрытое – без блокирующей косы – с блокирующей косой	*Эстуарная* – *без блокирующей косы* – *с блокирующей косой*
Мало- и многорукавный (дельтовый)	Полузакрытое – без блокирующей косы – с блокирующей косой	*Эстуарно-дельтовая* – *без блокирующей косы* – *с блокирующей косой*
	Открытое	*Дельтовая с дельтой выдвижения*

3. Будут ли использоваться изображения денотатов (в примере используются стилизованные пиктограммы)?

4. Что должен включать словарь изображений денотатов (в примере словарь изображений включает шесть стилизованных пиктограмм, приведенных на рис. 7)?

На втором этапе будем использовать вербально-образный тезаурус электронной библиотеки, в рамках которого сформируем группу из 12 вербальных и 6 образных дескрипторов, используемых для описания 6 геообъектов, являющихся устьевыми областями рек (см. рис. 7). Интересующие нас 6 форм представления образных дескрипторов этой группы расположены в нижней части рис. 7, на котором приведены названия только для двух из двенадцати вербальных дескрипторов, соответствующих одному геообъекту (эстуарно-дельтовая устьевая область с блокирующей косой). В общем случае *второй принцип* построения трехкомпонентной кодировки заключается в конструктивном описании вербальных и/или образных дескрипторов вербально-образного тезауруса электронной библиотеки, определяемых на основе группы вербальных и/или образных знаков, сформированной на первом этапе.

Таблица 3 Коды 6 концептов, 12 названий вербальных дескрипторов и 6 пиктограмм образных дескрипторов вербально-образного тезауруса электронной библиотеки

Коды концептов геообъектов, являющихся устьевыми областями рек	Названия устьевых областей на русском и английском языках и коды названий (даны в скобках)	Коды форм образных дескрипторов (см. рис. 7)
001к	Простая без блокирующей косы (001р) — Simple river mouth without blocked spit (001a)	001о
010к	Эстуарная без блокирующей косы (010р) — Estuary without blocked spit (010a)	010о
011к	Эстуарная с блокирующей косой (011р) — Estuary with blocked spit (011a)	011о
100к	Эстуарно-дельтовая без блокирующей косы (100р) — Silt delta without blocked spit (100a)	100о
101к	Эстуарно-дельтовая с блокирующей косой (101р) — Silt delta with blocked spit (101a)	101о
110к	Дельтовая с дельтой выдвижения (110р) — Protruding delta (110a)	110о

На третьем этапе всем 12 вербальным названиям и шести пиктограммам для 6 геообъектов и соответствующим 6 концептам в электронной библиотеке назначаются компьютерные коды (табл. 3)[1]: 6 кодов для концептов (001к–110к, где литера «к» обозначает концепт), 6 кодов для русскоязычных названий геообъектов (001р–110р, где литера «р» обозначает русский язык), 6 кодов для англоязычных названий (001а–110а, где литера «а» обозначает английский язык) и 6 кодов для пиктограммам геообъектов (001о–110о, где литера «о» обозначает визуальные образы денотатов).

В общем случае *третий принцип* построения трехкомпонентной кодировки заключается в назначении компьютерных кодов денотатам (в примере не рассматриваются), соответствующим им концептам и информационным объектам. Значения кодов определяются на основе идентификаторов соответствующих дескрипторов вербально-образного тезауруса (см. коды в табл. 3 и на рис. 7).

На основе шести концептов геообъектов, объединенных с образными формами их представления (пиктограммами), ранее были построены образные дескрипторы вербально-образного тезауруса [38]. В соответствии с определением формокода из разд. 3 формы образных знаков, которые объединены с соответствующими им кодами, указанными в третьей колонке табл. 3, образуют группу *образных формокодов*. На основе этой группы формируется *первый вид информационных кодов* форм образных знаков электронной библиотеки, а именно коды 001о–110о.

На основе шести концептов, объединенных с названиями геообъектов на русском языке, ранее были построены вербальные дескрипторы русскоязычной части вербально-образного тезауруса [38]. В соответствии с определеним формокода из разд. 3 названия шести устьевых областей рек на русском языке, которые объединены с соответствующими им кодами, указанными во второй колонке табл. 3, образуют группу *вербальных формокодов* русскоязычной формокодовой системы электронной библиотеки. На основе этой группы формируется *второй вид информационных кодов* форм русскоязычных знаков электронной библиотеки, а именно коды 001р–110р.

На основе шести концептов, объединенных с названиями геообъектов на английском языке, ранее были построены вербальные дескрипторы англоязычной части вербально-образного тезауруса [38]. В соответствии с определеним формокода из разд. 3 названия шести устьевых областей рек на английском языке, которые объединены с соответствующими им кодами, указанными во второй колонке табл. 3, образуют группу *вербальных формокодов* англоязычной формокодовой системы электронной библиотеки. На основе этой группы формируется *третий вид информационных кодов* форм англоязычных знаков электронной библиотеки, а именно коды 001а–110а.

В соответствии с определением семокода из разд. 3 концепты, которые объединены с соответствующими им кодами, указанными в первой колонке табл. 3, образуют группу семокодов электрон-

[1]Отметим, что в таблице приведены не сами коды цифровой среды, а их алфавитно-цифровые обозначения в среде социальных коммуникаций между пользователями электронной библиотеки.

ной библиотеки. На основе этой группы формируются *семантические коды* концептов (значений) знаков электронной библиотеки, а именно коды 001к–110к.

Семантические отношения между дескрипторами вербально-образного тезауруса на рис. 7 были сформированы на основе системы отношений между геообъектами, используемой в системе классификации из работы [45]. Точечными стрелками условно обозначены родовидовые отношения в тезаурусе, сплошными — отношения «часть–целое», штриховыми — ассоциативные отношения, штрихпунктирными — отношения между дескрипторами разных модальностей (вербальной и образной), но с одинаковыми концептами (значениями). Стрелка «штрих-двойной-пунктир» обозначает отношения между дескрипторами разных естественных языков (русский и английский), но с одинаковыми концептами (значениями).

В результате выполнения трех этапов были построены следующие коды:

- информационные коды форм образных знаков электронной библиотеки 001о–110о;
- информационные коды форм русскоязычных знаков электронной библиотеки 001р–110р;
- информационные коды форм англоязычных знаков электронной библиотеки 001а–110а;
- семантические коды концептов (значений) знаков электронной библиотеки 001к–110к.

Как отмечалось ранее, объектные коды в этом примере не рассматривались. Семантические коды концептов обеспечивают *взаимно однозначное соответствие* между концептами и компьютерными кодами, т. е. каждому концепту соответствует один код, а каждому семантическому коду — один концепт.

Рассмотренный пример построения трехкомпонентной кодировки, основанной на рассмотренной семиотической модели, наглядно иллюстрирует следующие ее отличия от кодировки символов или литер:

- при кодировании текстов и изображений в электронных библиотеках и других ИС имеется возможность однозначной идентификации и кодирования концептов с помощью семантических кодов;
- коды концептов, полученные в результате трехкомпонентной кодировки, не зависят от модальности форм представления концептов (вербальной или образной), что позволяет организовать концептуальное индексирование и семантический поиск в электронных геобиблиотеках [38];
- при кодировании слов и словосочетаний текстов на разных естественных языках в электронных библиотеках и других ИС совокупность информационных кодов форм вербальных знаков разных языков, по сути, представляет собой индекс межязыковых переходов [46, 47].

6 Заключение

Важно отметить, что частным случаем построенной семиотической модели, в которой отсутствуют формокод, семокод и компьютерные коды всех трех категорий, является семиотический треугольник Г. Фреге. Спектр приложений, в которых рассмотренная семиотическая модель может быть использована, во многом будет определяться уровнем и/или средствами ее реализации в ИС: прикладная задача, среда разработки прикладных задач ИС, система управления базами данных, форматы данных ИС, язык программирования, операционная система и т. д.

Однако существует два основных ограничения на использование рассмотренной семиотической модели, которые не зависят от уровня и средств ее реализации в ИС. Первое ограничение является следствием стационарности семиотической модели, т. е. эту модель можно применять только в случаях стационарных взаимосвязей денотатов, концептов, информационных объектов и компьютерных кодов.

Второе ограничение, которое уже упоминалось, является следствием исключения из рассмотрения кодов нулевой категории. Эту модель можно применять только в тех случаях, когда денотаты, концепты, информационные объекты, их бинарные отношения с компьютерными кодами, а также знаки, формокоды и семокоды ИС не зависят от кодов нулевой категории.

Разработка методов трехкомпонентной кодировки, основанных на рассмотренной семиотической модели, в том числе построение формокодовых и семокодовых кодировок, как правило, существенно зависит от назначения тех ИС, для которых строится трехкомпонентная кодировка. От назначения ИС в первую очередь зависит число видов информационных, семантических и объектных кодов, которые должны формироваться в процессе трехкомпонентной кодировки.

Главный результат описания взаимосвязей денотатов, концептов, информационных объектов и компьютерных кодов в виде семиотической модели заключается в том, что предложена теоретическая основа для разработки методов трехкомпонентной

кодировки денотатов, концептов и информационных объектов, в том числе для построения взаимно однозначных отношений между концептами систем знаний человека и компьютерными кодами. Иначе говоря, использование предлагаемой семиотической модели позволяет исключить *влияние асимметрии знаковых систем* на отношения между концептами систем знаний человека и компьютерными кодами.

Полученные результаты были сформулированы в процессе развития идеи семиотического треугольника Г. Фреге. Предлагаемое развитие этой идеи позволяет позиционировать компьютерные коды трех категорий как «представителей» денотатов, концептов и информационных объектов в цифровой среде электронных библиотек и других видов ИС.

В заключение отметим, что необходимость в дальнейшем развитии семиотической модели связана с разделением всех концептов на три категории (см. рис. 1) в контексте постановки актуальных проблем генерации ЦСЗ с учетом их эволюции во времени. Поэтому категоризация концептов и их эволюция во времени должны в будущем найти свое отражение в *нестационарной семиотической модели* описания взаимосвязей денотатов, авторских, коллективных, конвенциональных концептов, информационных объектов и компьютерных кодов всех категорий.

Литература

1. FP7 Exploratory Workshop 4 “Knowledge Anywhere Anytime.” http://cordis.europa.eu/ist/directorate_f/f_ws4.htm.
2. CORDIS ICT Programme Home. http://cordis.europa.eu/fp7/ict/programme/home_en.html.
3. ICT FP7 Work Programme. ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/ict-wp-2007-08_en.pdf.
4. *Gorn S.* Informatics (computer and information science): Its ideology, methodology, and sociology // The studies of information: Interdisciplinary messages / Eds. F. Machlup, U. Mansfield. — N.Y.: John Wiley and Sons, Inc., 1983. P. 121–140.
5. *Wang Y.* Cognitive informatics: A new transdisciplinary research field // Brain and Mind, 2003. Vol. 4. No. 2. P. 115–127.
6. *Wang Y.* On cognitive informatics // Brain and Mind, 2003. Vol. 4. No. 2. P. 151–167.
7. *Bryant A.* Cognitive informatics, distributed representation and embodiment // Brain and Mind, 2003. Vol. 4. No. 2. P. 215–228.
8. *Зацман И. М.* Концептуализация данных наукометрических исследований в научных электронных библиотеках // Труды X Всероссийской конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». — Дубна: ОИЯИ, 2008. С. 45–54.
9. *Зацман И. М., Косарик В. В., Курчавова О. А.* Задачи представления личностных и коллективных концептов в цифровой среде // Информатика и её применения, 2008. Т. 2. Вып. 3. С. 54–69.
10. *Nonaka I.* The knowledge-creating company // Harvard Business Review, 1991. Vol. 69. No. 6. P. 96–104.
11. *Nonaka I., Takeuchi H.* The knowledge-creating company. — N. Y.: Oxford University Press, 1995. [Пер.: Нонака И., Такеучи Х. Компания — создатель знания. — М.: ЗАО «Олимп-бизнес», 2003.]
12. *Шемакин Ю. И., Романов А. А.* Компьютерная семантика. — М.: НОЦ «Школа Китайгородской», 1995.
13. *McArthur D.* Information, its forms and functions: The elements of semiology. — Lewinton: The Edwin Mellen Press, Ltd., 1997.
14. *Колин К. К.* Становление информатики как фундаментальной науки и комплексной научной проблемы // Системы и средства информатики. Спец. вып. «Научно-методологические проблемы информатики» / Под ред. К. К. Колина. — М.: ИПИ РАН, 2006. С. 7–58.
15. *Колин К. К.* О структуре научных исследований по комплексной проблеме «Информатика» // Социальная информатика. — М.: ВКШ при ЦК ВЛКСМ, 1990. С. 19–33.
16. *Колин К. К.* Эволюция информатики и проблемы формирования нового комплекса наук об информации // Научно-техническая информация, 1995. Сер. 1. № 5. С. 1–7.
17. *Зацман И. М.* Концептуальный поиск информационных объектов в электронных библиотеках научных документов // Компьютерная лингвистика и интеллектуальные технологии. Труды международной конференции Диалог-2003. — М.: Наука, 2003. С. 710–716.
18. *Gladney H. M., Bennet J. L.* What do we mean by authentic? What’s the real McCoy? // D-Lib Magazine, 2003. Vol. 9. No. 7/8.
19. *Успенский В. А.* К публикации статьи Г. Фреге «Смысл и денотат» // В кн.: Семиотика и информатика. Вып. 35. — М.: Языки русской культуры, 1997. С. 351–352.
20. *Фреге Г.* Смысл и денотат // В кн.: Семиотика и информатика. Вып. 35. — М.: Языки русской культуры, 1997. С. 352–379.
21. *Фреге Г.* Понятие и вещь // В кн.: Семиотика и информатика. Вып. 35. — М.: Языки русской культуры, 1997. С. 380–396.
22. *Добров Б. В., Лукашевич Н. В.* Тезаурус и автоматическое концептуальное индексирование в университетской информационной системе «Россия» // Труды III Всероссийской конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». — Петрозаводск: КарНЦ РАН, 2001. С. 78–82.

23. *Гак В. Г.* Асимметрия // Большой энциклопедический словарь «Языкознание». — М.: Большая российская энциклопедия, 1998. С. 47.
24. *Зацман И. М.* Семиотические основания и элементарные технологии информатики // Информационные технологии, 2005. № 7. С. 18–31.
25. *Brookes B. C.* The foundations of information science. Part I. Philosophical aspects // J. Inf. Sci., 1980. No. 2. P. 125–133.
26. *Зацман И. М., Кожунова О. С.* Предпосылки и факторы конвергенции информационной и компьютерной наук // Информатика и её применения, 2008. Т. 2. Вып. 1. С. 77–97.
27. *Eco U.* A theory of semiotics. — Bloomington: Indiana University Press, 1976.
28. *Шрейдер Ю. А.* ЭВМ как средство представления знаний // Природа, 1986. № 10. С. 14–22.
29. *Шрейдер Ю. А.* Информация и знание // В кн.: Системная концепция информационных процессов. — М.: ВНИИСИ, 1988. С. 47–52.
30. *Гиляревский Р. С.* Основы информатики. — М.: Экзамен, 2003.
31. Информатика как наука об информации: Информационный, документальный, технологический, экономический, социальный и организационный аспекты / Под ред. Р. С. Гиляревского. — М.: ФАИР-ПРЕСС, 2006.
32. *Мельчук И. А.* Русский язык в модели «Смысл⇔Текст». — Москва–Вена: Школа «Языки русской культуры», Венский славистический альманах, 1995.
33. *Лютый А. А.* Язык карты: сущность, система, функция. 2-е изд., испр. — М.: ИГ РАН, 2002.
34. *Барт Р.* Основы семиологии // В кн.: Французская семиотика: От структурализма к постструктурализму. — М.: Прогресс, 2000. С. 247–310.
35. *Кибрик А. Е.* Язык // Большой энциклопедический словарь «Языкознание». — М.: Большая российская энциклопедия, 1998. С. 604–606.
36. *Turing A. M.* On computable numbers, with an application to the Entscheidungsproblem. 1936 (http://www.abelard.org/turpap2/tp2-ie.asp).
37. *Fox E. A.* Digital libraries of the future: Integration through the 5S Framework // Труды VI Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». — Пущино: Ин-т математических проблем биологии РАН, 2004. С. 1–2.
38. *Зацман И. М.* Концептуальный поиск и качество информации. — М.: Наука, 2003.
39. *Шмелев Д. Н.* Полисемия // Большой энциклопедический словарь «Языкознание». — М.: Большая российская энциклопедия, 1998. С. 382.
40. Roget's international thesaurus. — New York: Thomas Y. Crowell Co., 1954.
41. *Морковкин В. В.* Идеографические словари. — М.: Изд-во МГУ, 1970.
42. Тезаурус научно-технических терминов / Под ред. Ю. И. Шемакина. — М.: Воениздат, 1972.
43. *Баранов О. С.* Идеографический словарь русского языка. — М.: ЭТС, 1995.
44. *Лукашевич Н. В., Добров Б. В.* Тезаурус для автоматического концептуального индексирования как особый вид лингвистического ресурса // Компьютерная лингвистика и ее приложения. Тр. Междунар. семинара Диалог'2001. В 2-х т. Т. 2 / Под ред. А. С. Нариньяни. — М.: РосНИИ ИИ, 2001. С. 273–279.
45. *Михайлов В. Н.* Гидрология устьев рек: Методическое пособие. — М.: Изд-во МГУ, 1996.
46. Vossen P., ed. EuroWordNet General Document (Version 3) (http://www.illc.uva.nl/EuroWordNet/docs/GeneralDoc).
47. *Кожунова О. С.* Eurowordnet: задачи, структура и отношения // Информатика и её применения, 2008. Т. 2. Вып. 4. С. 85–92.

ABSTRACTS

A PROBABILISTIC ANALYSIS OF FAULT DETECTION LATENCY IN A NETWORK OF FINITE STATE MACHINES

A. V. Pechinkin[1] and S. L. Frenkel[2]

[1]IPI RAN, apechinkin@ipiran.ru
[2]IPI RAN, fsergei@mail.ru

This paper suggests an approach to the computation of time probability distribution function (PDF) of Fault Detection Latency (FDL) in case, when a system is modeled as a combination of interacting finite state machines (FSMs) under random inputs, where the interactions deal with switching each of the FSMs from a working mode to a testing one. Fault detection latency is a period of fault detection after it occurs in certain inner states. Traditionally, the FDL of an FSM is modeled as the time (numbers of its state transition steps) to absorption for a Markov chain with the state space generated by a product of fault-free and faulty (i.e., corrupted by a fault) FSMs. The principal problem of using this model for the networks of sub-FSMs is that random transitions of the product of the fault-free and faulty networked automata even under independent inputs random vectors are not the Markovian ones. Thanks to an extension of the transition space of the networked FSMs by some additional states corresponding to the number of steps between transitions to the modes mentioned above for each of sub-FSMs, this model is extended to the case of an FSM decomposed (in a designing process) into a number of components of sub-FSMs. A way to compute the FDL PDF in terms of FDL PDF of initial FSM (that is not decomposed) and the FSMs of corresponding sub-FSMs is shown.

Keywords: testing; Finite State Machine; Markov chains

BACKUP USING SNAPSHOTS

V. A. Kozmidiady

IPI RAN, v.kozmidiady@gmail.com

Complication of the reserve copying (backup) using differences between two successive snapshots is examined. A formal model in which it is possible exactly to put a problem is offered. An algorithm which solves this problem for almost linear time of the volume of differences is offered.

Keywords: backup; snapshots; file system recovery; journaling file system

REGISTRATION OF DISTORTIONS IN AUTOMATIC FINGERPRINT IDENTIFICATION

O. S. Ushmaev

IPI RAN, oushmaev@ipiran.ru

The problem of biometric system adjusting to the variable operational environment is considered. The impact of operational distortion is analyzed with automatic fingerprint identification as an example. The general approach for operation distortion registration is suggested. Based on this approach, the method for adjusting of a biometric system was developed. The author's experiments showed the effectiveness of the suggested approach.

Keywords: biometrics; automatic fingerprint identification; recognition distortions

APPROXIMATE METHOD OF CALCULATION OF NODE CHARACTERISTICS IN TELECOMMUNICATION NETWORK WITH REPETITIVE TRANSMISSIONS

Ya. M. Agalarov

IPI RAN, agglar@yandex.ru

A model of packet commutation node with repetitive transmissions for two schemes of buffer memory distributions — fully accessible and fully shared — is considered. The approximate method of calculation of stream intensity and probabilities of node blocking is proposed. The necessity and sufficiency conditions for existence and unicity of solution of equation were derived for streams in node when operating regime became established, and the convergence of iteration method for solving proposed equation was proved.

Keywords: packet commutation node; buffer memory; repetitive transmissions; probabilities of blocking; iteration method

SOME IMPLEMENTATION ASPECTS OF THE CONNECTING MEDIUM IN THE DECENTRALIZED PACKET SWITCHING ARCHITECTURE

V. B. Egorov

IPI RAN, vegorov@ipiran.ru

Specific requirements to the connecting medium in the decentralized packet switching architecture are defined. Some medium implementation aspects are discussed. Special attention is paid to high-speed serial interfaces used as a connecting medium, with a reduced version of such an interface assigned to packet switching being proposed.

Keywords: packet switch; decentralized switching; integrated communications controller; superlocal network; high-speed serial interface

TECHNOLOGICAL SYSTEM FOR AUTOMATIC PROCESSING OF THREE-DIMENSIONAL LIDAR DATA

V. A. Sukhomlin[1] and I. N. Gorkavyi[2]

[1] IPI RAN, sukhomlin@mail.ru
[2] Faculty of Computational Mathematics and Cybernetics,
M. V. Lomonosov Moscow State University, ilya_gor@rambler.ru

New approach to development of software suites for automatic processing of LIDAR data is considered. The elaborated algorithms for classification of three-dimensional (3D) data and generation of high quality models of bare-Earth relief are described, tools for visualization of 3D models and interactive correction of models by operator are presented. The paper discusses goals, achieved results, and perspectives of new approach.

Keywords: automatic data processing; classification methods; three-dimensional models; LIDAR; laser scanning

SEMIOTIC MODEL FOR COMPUTER CODING CONCEPTS AND INFORMATION OBJECTS

I. M. Zatsman

IPI RAN, iz_ipi@a170.ipi.ac.ru

The semiotic model, which has been developed during research of problems of generation and evolution of goal-oriented knowledge systems in digital libraries and other kinds of information systems, is considered. These problems concern to the new direction of the researches which has been named "Cognitive Informatics." The semiotic model suggested is intended for the description of computer coding concepts and information objects. This model is positioned as theoretical foundations for development of methods of the three-component coding denotata, concepts, and information objects, including construction of one-to-one correspondence between concepts of goal-oriented knowledge systems and computer codes.

Keywords: semiotic model; goal-oriented knowledge systems; denotata; concepts; information objects; computer codes; three-component coding; one-to-one correspondence between concepts and computer codes

Об авторах

Агаларов Явер Мирзабекович (р. 1952) — кандидат технических наук, доцент, ведущий научный сотрудник ИПИ РАН

Горькавый Илья Николаевич (р. 1982) — аспирант кафедры автоматизации систем вычислительных комплексов факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова

Егоров Владимир Борисович (р. 1948) — кандидат технических наук, и.о. ведущего научного сотрудника ИПИ РАН

Зацман Игорь Моисеевич (р. 1952) — кандидат технических наук, заведующий отделом ИПИ РАН

Козмидиади Владимир Александрович (р. 1936) — доктор технических наук, главный научный сотрудник ИПИ РАН

Печинкин Александр Владимирович (р. 1946) — доктор физико-математических наук, профессор, главный научный сотрудник ИПИ РАН; профессор Российского университета дружбы народов (РУДН)

Сухомлин Владимир Александрович (р. 1945) — доктор технических наук, профессор кафедры автоматизации систем вычислительных комплексов факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова; ведущий научный сотрудник ИПИ РАН

Ушмаев Олег Станиславович (р. 1981) — кандидат технических наук, ведущий научный сотрудник ИПИ РАН

Френкель Сергей Лазоревич (р. 1951) — кандидат технических наук, доцент, старший научный сотрудник ИПИ РАН, доцент факультета вычислительных машин и систем Московского института радиотехники, электроники и автоматики (МИРЭА)

About Authors

Agalarov Yaver M. (b. 1952) — Candidate of Technical Sciences (PhD); leading scientist, Institute of Informatics Problems, Russian Academy of Sciences

Egorov Vladimir B. (b. 1948) — Candidate of Technical Sciences (PhD), leading scientist, Institute of Informatics Problems, Russian Academy of Sciences

Frenkel Sergey L. (b. 1951) — Candidate of Technical Sciences (PhD); senior scientist, Institute of Informatics Problems, Russian Academy of Sciences; associate professor, Moscow Institute of Radio, Electronics, and Automation, Faculty of Computer Systems

Gorkavyi Ilia N. (b. 1982) — postgraduate student, Department of Automation of Computational Systems, Faculty of Computational Mathematics and Cybernetics, M. V. Lomonosov Moscow State University

Kozmidiady Vladimir A. (b. 1936) — Doctor of Science in Technology; proncipal scientist, Institute of Informatics Problems, Russian Academy of Sciences

Pechinkin Alexander V. (b. 1946) — Doctor of Science in physics and mathematics; principal scientist, Institute of Informatics Problems, Russian Academy of Sciences; professor, Peoples' Friendship University of Russia

Sukhomlin Vladimir A. (b. 1945) — Doctor of Science in Technology; Department of Automation of Computational Systems, Faculty of Computational Mathematics and Cybernetics, M. V. Lomonosov Moscow State University; leading scientist, Institute of Informatics Problems, Russian Academy of Sciences

Ushmaev Oleg S. (b. 1981) — Candidate of Technical Sciences (PhD), leading scientist, Institute of Informatics Problems, Russian Academy of Sciences

Zatsman Igor M. (b. 1952) — Candidate of Technical Sciences (PhD); department head, Institute of Informatics Problems, Russian Academy of Sciences

Правила подготовки рукописей статей для публикации в журнале «Информатика и её применения»

Журнал «Информатика и её применения» публикует теоретические, обзорные и дискуссионные статьи, посвященные научным исследованиям и разработкам в области информатики и ее приложений. Журнал издается на русском языке. Тематика журнала охватывает следующие направления:

- теоретические основы информатики;
- математические методы исследования сложных систем и процессов;
- информационные системы и сети;
- информационные технологии;
- архитектура и программное обеспечение вычислительных комплексов и сетей.

1. В журнале печатаются результаты, ранее не опубликованные и не предназначенные к одновременной публикации в других изданиях. Публикация не должна нарушать закон об авторских правах. Направляя свою рукопись в редакцию, авторы автоматически передают учредителям и редколлегии неисключительные права на издание данной статьи на русском языке и на ее распространение в России и за рубежом. При этом за авторами сохраняются все права как собственников данной рукописи. В связи с этим авторами должно быть представлено в редакцию письмо в следующей форме: Соглашение о передаче права на публикацию:

   «Мы, нижеподписавшиеся, авторы рукописи « », передаем учредителям и редколлегии журнала «Информатика и её применения» неисключительное право опубликовать данную рукопись статьи на русском языке как в печатной, так и в электронной версиях журнала. Мы подтверждаем, что данная публикация не нарушает авторского права других лиц или организаций. Подписи авторов: (ф. и. о., дата, адрес)».

   Редколлегия вправе запросить у авторов экспертное заключение о возможности опубликования представленной статьи в открытой печати.

2. Статья подписывается всеми авторами. На отдельном листе представляются данные автора (или всех авторов): фамилия, полные имя и отчество, телефон, факс, e-mail, почтовый адрес. Если работа выполнена несколькими авторами, указывается фамилия одного из них, ответственного за переписку с редакцией.

3. Редакция журнала осуществляет самостоятельную экспертизу присланных статей. Возвращение рукописи на доработку не означает, что статья уже принята к печати. Доработанный вариант с ответом на замечания рецензента необходимо прислать в редакцию.

4. Решение редакционной коллегии о принятии статьи к печати или ее отклонении сообщается авторам. Редколлегия не обязуется направлять рецензию авторам отклоненной статьи.

5. Корректура статей высылается авторам для просмотра. Редакция просит авторов присылать свои замечания в кратчайшие сроки.

6. При подготовке рукописи в MS Word рекомендуется использовать следующие настройки. Параметры страницы: формат — A4; ориентация — книжная; поля (см): внутри — 2,5, снаружи — 1,5, сверху — 2, снизу — 2, от края до нижнего колонтитула — 1,3. Основной текст: стиль — «Обычный»: шрифт Times New Roman, размер 14 пунктов, абзацный отступ — 0,5 см, 1,5 интервала, выравнивание — по ширине. Рекомендуемый объем рукописи — не свыше 25 страниц указанного формата. Ознакомиться с шаблонами, содержащими примеры оформления, можно по адресу в Интернете: http://www.ipiran.ru/journal/template.doc.

7. К рукописи, предоставляемой в 2-х экземплярах, обязательно прилагается электронная версия статьи (как правило, в форматах MS WORD (.doc) или LaTex (.tex), а также — дополнительно — в формате .pdf) на дискете, лазерном диске или по электронной почте. Сокращения слов, кроме стандартных, не применяются. Все страницы рукописи должны быть пронумерованы.

8. Статья должна содержать следующую информацию на русском и английском языках: название, Ф.И.О. авторов, места работы авторов и их электронные адреса, аннотация (не более 100 слов), ключевые слова. Ссылки на литературу в тексте статьи нумеруются (в квадратных скобках) и располагаются в порядке их первого упоминания. Все фамилии авторов, заглавия статей, названия книг, конференций и т. п. даются на языке оригинала, если этот язык использует кириллический или латинский алфавит.

9. Присланные в редакцию материалы авторам не возвращаются.
10. При отправке файлов по электронной почте просим придерживаться следующих правил:
    - указывать в поле subject (тема) название журнала и фамилию автора;
    - использовать attach (присоединение);
    - в случае больших объемов информации возможно использование общеизвестных архиваторов (ZIP, RAR);
    - в состав электронной версии статьи должны входить: файл, содержащий текст статьи, и файл(ы), содержащий(е) иллюстрации.
11. Журнал «Информатика и её применения» является некоммерческим изданием, и гонорар авторам не выплачивается.

Адрес редакции: Москва 119333, ул. Вавилова, д. 44, корп. 2, ИПИ РАН
Тел.: +7 (499) 135-86-92 Факс: +7 (495) 930-45-05 E-mail: rust@ipiran.ru

Выпускающий редактор Л. Кокушкина
Художественный редактор М. Седакова
Сдано в набор 19.03.09. Подписано в печать 05.06.09. Формат $60 \times 84/8$
Бумага офсетная. Печать цифровая. Усл.-печ. л. 11. Уч.-изд. л. 10,5. Тираж 200 экз.

Заказ №

Издательство «ТОРУС ПРЕСС», Москва 119991, ул. Косыгина, д. 4
torus@torus-press.ru; http://www.torus-press.ru

Отпечатано в ППП «Типография «Наука» с готовых диапозитивов
Москва 121099, Шубинский пер., д. 6