ISSN 1992-2264

Российская академия наук

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ

ТОМ 1
Выпуск 1
2007

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ

Журнал Отделения информационных технологий и вычислительных систем РАН

Издается с 2007 года
Журнал выходит ежеквартально

Учредители:
Российская академия наук
Институт проблем информатики Российской академии наук

Адрес редакции:

Москва 119333, ул. Вавилова 44, корп. 2, ИПИ РАН,
редакция журнала «Информатика и её применения»
Тел. (495)135-86-92, e-mail asorokin@ipiran.ru

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

Информатика и её применения

Том 1 Выпуск 1 Год 2007

СОДЕРЖАНИЕ

Технический редактор *Л. Кокушкина*
Художественный редактор *М. Седакова*
Сдано в набор 15.04.07. Подписано в печать 01.06.07. Формат 60 x 84 / 8
Бумага офсетная. Печать офсетная. Усл.-печ. л. 10. Уч.-изд. л. 8,5. Тираж 200 экз.

Заказ №

Издательство «ЭЛЕКС-КМ» Москва 115201, Каширское ш., 22-3
elex@elex-km.ru; http://www.elex-km.ru

Отпечатано в ППП «Типография «Наука» с готовых диапозитивов.
Москва 121099, Шубинский пер., д. 6.

Уважаемый читатель!

Предлагаем Вашему вниманию первый номер нового журнала «Информатика и её применения» — научного журнала Отделения информационных технологий и вычислительных систем Российской академии наук (ОИТВС РАН). Базовым институтом РАН, осуществляющим издание данного журнала, является Институт проблем информатики РАН.

Необходимость издания такого журнала вызвана активным развитием информатики и информационных технологий, большой важностью этого научного направления для развития страны, проникновением информационных технологий во все сферы жизни современного общества.

Тематику журнала определяет тот факт, что информатика — это комплексная фундаментальная научная дисциплина, опирающаяся на достижения ряда других наук, в том числе математики, физики, лингвистики и др.; одновременно журнал будет уделять большое внимание современным информационным технологиям, являющимся приложениями результатов информатики как фундаментальной науки.

Мы приглашаем авторов представлять для публикации в новом журнале статьи как с достижениями в области теоретических проблем информатики, так и с изложением результатов ее практического приложения.

Основные направления научных публикаций в журнале:

- теоретические основы информатики;
- математические методы исследования сложных систем и процессов;
- информационные системы и сети;
- информационные технологии;
- архитектура и программное обеспечение вычислительных комплексов и сетей.

В дальнейшем предполагается публиковать в журнале рецензии на наиболее интересные книжные новинки в области информатики и информационных технологий, объявления о крупнейших международных и всероссийских конференциях, различных научных мероприятиях по этой тематике и другие информационные материалы.

В 2007 году выйдут в свет 2 выпуска журнала; начиная с 2008 года журнал будет выходить ежеквартально.

Надеемся, что содержание статей, помещаемых в журнале, вызовет интерес научной общественности. Редколлегия, со своей стороны, сделает все для того, чтобы журнал информировал читателей о новейших достижениях информатики и ее актуальных практических приложениях.

Главный редактор журнала
«Информатика и её применения», академик *С. В. Емельянов*

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007. Т. 1. Вып. 1. С. 3–13

РАЗВИТИЕ ТЕОРИИ ФИЛЬТРОВ ПУГАЧЕВА ДЛЯ ОПЕРАТИВНОЙ ОБРАБОТКИ ИНФОРМАЦИИ В СТОХАСТИЧЕСКИХ СИСТЕМАХ*

И. Н. Синицын[1]

Аннотация: В современной статистической информатике важное место занимают статистические методы оперативной обработки (фильтрации, экстраполяции, интерполяции и т.д.) информации в стохастических системах. Дается аналитический обзор работ и рассматриваются основные тенденции развития нелинейных условно оптимальных фильтров Пугачева. Рассматриваются фильтры Пугачева для регулярных и нерегулярных, непрерывных и дискретных, гауссовских и негауссовских стохастических систем, в том числе с автокоррелированными помехами в наблюдениях. Устанавливается связь между фильтрами Калмана и Пугачева. Обсуждаются фильтры Пугачева по различным вероятностным критериям, а также вопросы совместной условно оптимальной фильтрации и распознавания. Сформулированы направления дальнейшего развития фильтров Пугачева.

Ключевые слова: стохастическая система; условно оптимальная фильтрация; экстраполяция и интерполяция; фильтр Пугачева; оперативная обработки информации; автокоррелированная помеха

1 Введение

Как отмечается [1, 2], к концу 1970-х возникла настоятельна необходимость распространения нашедшей широкое применение калмановской теории линейной фильтрации на нелинейные стохастические системы (СтС). Многочисленные попытки получить решение этой задачи путем линеаризации исходных уравнений СтС с различными уточнениями на протяжении 20 лет не привели к удовлетворительным результатам, а методы субоптимальной фильтрации часто приводили на практике к противоречивым результатам. Владимир Семенович Пугачев предложил принципиально новый подход, основанный на идее условно оптимального оценивания и применения хорошо разработанной к этому времени теории нелинейных дифференциальных СтС. Была разработана теория условно оптимальной оперативной (в реальном масштабе времени) фильтрации процессов в дифференциальных СтС, и дан метод нахождения оптимального по Парето фильтра в классе допустимых фильтров, описываемых дифференциальными уравнениями заданного вида. Разработанный метод позволяет оценивать не только состояние СтС, но и неизвестные параметры в ее уравнениях. Для решения уравнений, определяющих неизвестные функции в дифференциальном уравнении условно оптимального фильтра, было использовано уравнение для одномерной характеристической функции. В этом случае все неизвестные функции определяются заранее в процессе проектирования фильтра, а не во время его оперативной работы. Для проектирования фильтра используется только априорная информация. При практическом применении теории для получения требуемых оценок необходимо лишь интегрирование дифференциального уравнения фильтра в темпе получения текущей (оперативной) информации.

В связи с необходимостью решать задачи статистического анализа дискретных нелинейных СтС при проектировании условно оптимальных дискретных фильтров и экстраполяторов потребовалось распространить хорошо разработанные методы анализа стохастических дифференциальных уравнений на системы, описываемые стохастическими разностными уравнениями. Это было сделано В. С. Пугачевым в [3].

Итогом многолетних исследований В. С. Пугачева, его учеников и последователей в области статистического анализа дифференциальных СтС, а также фильтрации, экстраполяции и оценивания параметров в таких системах является монография [4]. В этом труде дано систематическое изложение теории дифференциальных СтС на базе единого подхода с применением уравнения Пугачева для конечномерной характеристической функции. Развита точная теория линейных дифференциальных СтС, в частности получены явные формулы для конечномерных характеристических функций,

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 07-07-00031) и программы ОИТВС РАН «Фундаментальные основы информационных технологий и систем» (проект 1.5).

[1]ИПИ РАН, sinitsin@dol.ru

а также основные приближенные методы статистического анализа нелинейных систем. Изложены методы нормальной аппроксимации, моментов, семиинвариантов и методы, основанные на ортогональных разложениях плотностей вероятностей, в частности на разложениях по полиномам Эрмита. Выведены основные формулы и уравнения теории оптимальной фильтрации. Рассматриваются теория оптимальной линейной фильтрации и методы субоптимальной нелинейной фильтрации. Впервые дано полное изложение теории условно оптимальной фильтрации, экстраполяции и оценивания параметров в дифференциальных СтС. В эту монографию вошли многие новые научные результаты. Материал иллюстрирован большим количеством примеров и задач, многие из которых имеют самостоятельное значение.

Библиография и краткий обзор работ по теории Фильтра Пугачева (ФП), выполненных до 1998 г., даны в [1, 2]. Рассмотрим современное состояние и развитие теории ФП.

2 Задачи условно оптимального оценивания в непрерывных стохастических системах

Практическое применение субоптимальных методов фильтрации ограничивается высоким порядком фильтров, особенно в задачах большой размерности. Поэтому единственным способом получения практически реализуемых фильтров в задачах большой размерности является понижение порядка фильтров. Чтобы понять, как это может быть достигнуто для непрерывных СтС, достаточно проанализировать структуру уравнения для субоптимальной оценки. Легко видеть, что все эти методы дают для оценки вида $\hat{X}_t = AU_t$ уравнение

$$dU_t = \alpha_t \xi(U_t, Y_t, t)dt + \beta_t \eta(U_t, Y_t, t)dY_t + \gamma_t dt\,, \quad (1)$$

где $\xi(U_t, Y_t, t)$, $\eta(U_t, Y_t, t)$ — некоторые функции текущих значений наблюдаемого процесса Y_t, оценки U_t и времени t; α_t, β_t и γ_t — некоторые функции времени. То, что они становятся известными вместе с ковариационной матрицей ошибки R_t только после интегрирования полной системы уравнений, определяющей все неизвестные параметры функции $p^*(x;\theta)$, аппроксимирующей апостериорную плотность $p_t(x)$, для дальнейших рассуждений не имеет значения. Так, например, в методе нормальной аппроксимации [4] $\xi = \left[\, f^T - f^{(1)T} h^T \,\right]^T$, $\eta = h$. Соответственно, коэффициент α имеет блочную структуру $\alpha_t = [\, I_{n_x}\, I_{n_x} \,]$, где I_{n_x} — единичная $(n_x \times n_x)$-матрица, $\beta_t = I_{n_x}$, $\gamma_t = 0$. Чтобы определить функцию ξ в методе ортогональных разложений [4], подставим в уравнение для оценки выражения значения функций f, $f^{(1)}$, h:

$$f = f_0 + \sum_{k=3}^{N} \sum_{|\nu|=k} f_\nu c_\nu\,,$$

$$f^{(1)} = f_0^{(1)} + \sum_{k=3}^{N} \sum_{|\nu|=k} f_\nu^{(1)} c_\nu\,,$$

$$h = h_0 + \sum_{k=3}^{N} \sum_{|\nu|=k} h_\nu c_\nu$$

и перепишем это уравнение в виде

$$dU_t = \Bigg[f_0 - h_0 f_0^{(1)} + \sum_{k=3}^{N} \sum_{|\nu|=k} \Big(f_\nu - h_0 f_\nu^{(1)} - h_\nu f_0^{(1)} \Big) c_\nu - \sum_{k,l=3}^{N} \sum_{\substack{|\nu|=k \\ |\mu|=l}} h_\nu f_\mu^{(1)} c_\nu c_\mu \Bigg] dt + \left(h_0 + \sum_{k=3}^{N} \sum_{|\nu|=k} h_\nu c_\nu \right) dY_t\,.$$

Отсюда видно, что компонентами векторной функции ξ в этом случае служат все компоненты n_x-мерных векторных функций f_0, $-h_0 f_0^{(1)}$, f_ν, $-h_0 f_\nu^{(1)}$, $-h_\nu f_0^{(1)}$, $-h_\nu f_\mu^{(1)}$ $(|\nu|, |\mu| = 3, \ldots, N)$ и соответственно матрица α_t состоит из горизонтально расположенных диагональных блоков, первые два из которых представляют собой единичную матрицу I_{n_x}, а остальные — произведения I_{n_x} на соответствующие коэффициенты c_ν и на произведения $c_\nu c_\mu$. Если некоторые компоненты векторых функций f_0, $-h_0 f_0^{(1)}$, f_ν, $-h_0 f_\nu^{(1)}$, $-h_\nu f_0^{(1)}$, $-h_\nu f_\mu^{(1)}$ не зависят от U_t, то линейную комбинацию этих компонент с соответствующими столбцами матрицы α_t можно выделить и принять за вектор γ_t в уравнении (1). Матричная функция η представляет собой блочную матрицу, состоящую из всех расположенных по вертикали $(n_x \times n_y)$-матриц h_0, h_ν $(|\nu| = 3, \ldots, N)$, а матрица β_t состоит из горизонтально расположенных блоков, первый из которых представляет собой единичную матрицу I_{n_x}, а остальные — произведения I_{n_x} на соответствующие коэффициенты c_ν $(|\nu| = 3, \ldots, N)$. Аналогично и применение метода моментов или метода семиинвариантов [4] приводит к виду (1).

Если бы коэффициенты α_t, β_t, γ_t в уравнении (1) были известными функциями времени, то уравнение (1) определило бы фильтр того же порядка n_x,

что и уравнение, описывающее поведение системы. Поэтому естественно возникает мысль попытаться непосредственно определить коэффициенты α_t, β_t, γ_t в уравнении (1) как функции времени из условия минимума среднего квадрата ошибки, $\mathrm{M}\left|\hat{X}_t - X_t\right|^2 = \min$, при всех $t > t_0$. Это приводит к теории условно оптимальной фильтрации Пугачева, в которой уравнение фильтра задается заранее и оптимизируются только коэффициенты этого уравнения.

Итак, мы приходим к идее решения задачи оценивания путем нахождения оптимального фильтра в некотором классе допустимых фильтров, определяемом условием, чтобы поведение фильтра описывалось дифференциальным уравнением заданного порядка и заданной формы. Таким образом, мы отказываемся от абсолютной оптимизации и ограничиваемся условной оптимизацией в заданном ограниченном классе фильтров.

Определив класс допустимых фильтров, следует решить вопрос о том, какой фильтр в этом классе считается оптимальным. Следуя В. С. Пугачеву, будем считать оптимальным такой фильтр, который дает в известном смысле наилучшую оценку при всех значениях $t > t_0$. Однако в общем случае нелинейной СтС в классе допустимых фильтров может не быть такого фильтра, который давал бы наилучшую оценку при всех значениях $t > t_0$. В самом деле, такой фильтр был бы оптимальным одновременно по множеству критериев. В каждый данный момент $t > t_0$ условие $\mathrm{M}\left|\hat{X}_t - X_t\right|^2 = \min$ или $\mathrm{M}\left|\hat{X}_t - X_{t+\Delta}\right|^2 = \min$ представляет собой один определенный критерий оптимальности. Требование, чтобы это условие выполнялось для некоторого множества значений t, равносильно требованию оптимальности фильтра одновременно по соответствующему множеству критериев. Иными словами, задача оптимизации фильтра при всех значениях $t > t_0$ представляет собой задачу многокритериальной оптимизации. Такие задачи, как правило, не имеют решения. Фильтр Калмана–Бьюси (ФКБ), дающий оптимальную линейную оценку состояния линейной системы в каждый момент времени $t > t_0$, представляет собой исключение.

Исходя из приведенных соображений, будем считать оптимальным такой фильтр из класса допустимых фильтров, который при любом совместном распределении величин Y_t, X_t, U_t в момент $t \geq t_0$ дает наилучшую оценку $\hat{X}_s$ вектора X_s или вектора $X_{s+\Delta}$, $\Delta > 0$, в бесконечно близкий момент $s > t$, $s \to t$, реализующую минимум среднего квадрата ошибки $\mathrm{M}\left|\hat{X}_s - X_s\right|^2$ или соответственно $\mathrm{M}\left|\hat{X}_s - X_{s+\Delta}\right|^2$. Иными словами, будем считать оптимальным такой допустимый фильтр, который на каждом бесконечно малом интервале времени совершает оптимальный переход из того состояния, в котором он оказался в начале этого интервала, в новое состояние. Такой допустимый фильтр будем называть **условно оптимальным**. Тогда задачи фильтрации и экстраполяции сведутся к нахождению оптимальных значений α_t, β_t и γ_t в уравнении (1) в любой момент времени $t \geq t_0$, обеспечивающих минимум среднего квадрата ошибки фильтрации $\mathrm{M}\left|\hat{X}_s - X_s\right|^2$ или экстраполяции $\mathrm{M}\left|\hat{X}_s - X_{s+\Delta}\right|^2$, $\Delta > 0$, в бесконечно близкий будущий момент $s > t$, $s \to t$.

Условно оптимальный фильтр обладает тем свойством, что в данном классе допустимых фильтров не существует фильтра, который при данном начальном распределении Y_t, X_t и U_t в момент t_0 был бы лучше условно оптимального при всех $t > t_0$. Это значит, по терминологии теории многокритериальной оптимизации, что условно оптимальный фильтр представляет собой один из множества допустимых фильтров, **оптимальный по Парето**.

В задаче условно оптимальной фильтрации возьмем уравнения системы и уравнение наблюдения в общей форме:

$$\begin{aligned} dX_t &= \varphi(X_t, Y_t, t)\,dt + \psi(X_t, Y_t, t)\,dW\,, \\ dY_t &= \varphi_1(X_t, Y_t, t)\,dt + \psi_1(X_t, Y_t, t)\,dW \end{aligned} \tag{2}$$

и будем предполагать, что $W(t)$ представляет собой процесс с независимыми приращениями с нулевым математическим ожиданием и конечной ковариационной функцией

$$\begin{aligned} k_w(t_1, t_2) &= k(\min(t_1, t_2))\,, \\ k(t) &= k(t_0) + \int_{t_0}^{t} \nu(\tau)\,d\tau\,. \end{aligned} \tag{3}$$

Для задачи экстраполяции необходимо ограничиться случаем, когда функции φ и ψ не зависят от наблюдаемого вектора Y_t, процесс $W(t)$ состоит из двух независимых блоков $W_1(t)$, $W_2(t)$ и соответственно матрицы ψ, ψ_1 имеют блочную структуру $\psi = [\psi'\,0]$, $\psi_1 = [\,0\,\psi_1']$, так что $\psi dW = \psi' dW_1$, $\psi_1 dW = \psi_1' dW_2$. В этом случае, отбрасывая штрихи у функций ψ и ψ_1, напишем уравнения (2) в виде:

$$\begin{aligned} dX_t &= \varphi(X_t, t)\,dt + \psi(X_t, t)\,dW_1, \\ dY_t &= \varphi_1(X_t, Y_t, t)\,dt + \psi_1(X_t, Y_t, t)\,dW_2, \end{aligned}$$

где $W_1(t)$, $W_2(t)$ — независимые процессы с независимыми приращениями с нулевыми математиче-

скими ожиданиями и конечными ковариационными функциями вида (3).

Что же касается точности фильтрации, то следует отметить, что методы параметризации распределений дают возможность оценивать по априорным данным точность любого фильтра, описываемого конечным числом стохастических дифференциальных уравнений, независимо от того, каким методом получен этот фильтр. Это позволяет сравнивать по точности методы теории условно оптимальной фильтрации с методами субоптимальной фильтрации. Кроме того, принятый в теории условно оптимальной фильтрации метод построения классов допустимых фильтров позволяет построить класс допустимых фильтров, содержащий любой наперед заданный фильтр, описываемый конечным числом дифференциальных уравнений. Оптимальный фильтр этого класса будет, как правило, лучше и уж во всяком случае не хуже, чем данный фильтр.

Таким образом, теория условно оптимальной фильтрации Пугачева обладает двумя несомненными преимуществами по сравнению с методами субоптимальной фильтрации [4]. Во-первых, она позволяет получать фильтры более низкого порядка и, следовательно, более простые в реализации. Во-вторых, она дает возможность получать фильтры не меньшей, а при желании и большей точности, чем фильтры, даваемые методами субоптимальной нелинейной фильтрации [4].

3 Основные теоремы условно оптимального оценивания для непрерывных стохастических систем

В [4] применительно к СтС (2), где $W(t)$ — процесс с независимыми приращениями, получены уравнения условно оптимальной фильтрации и экстраполяции для различных классов непрерывных СтС.

В частности, для винеровского процесса $W(t)$, если уравнения (2) допускают существования одномерных моментов, тогда входящие в уравнение ФП (1) коэффициенты α_t, β_t, γ_t определяются следующими уравнениями:

$$A\alpha_t m_1 + A\beta_t m_2 + A\gamma_t = m_0\,, \tag{4}$$

где

$$\begin{aligned} m_0 &= \mathrm{M}\varphi(X_t, Y_t, t)\,,\\ m_1 &= \mathrm{M}\xi(Y_t, U_t, t)\,,\\ m_2 &= \mathrm{M}\eta(Y_t, U_t, t)\varphi_1(X_t, Y_t, t)\,; \end{aligned}$$

$$A\beta_t = \kappa_{02}\kappa_{22}^{-1}\,, \tag{5}$$

где

$$\begin{aligned} \kappa_{02} &= \mathrm{M}(X_t - AU_t)\varphi_1(X_t, Y_t, t)^T\eta(Y_t, U_t, t)^T + \\ &\quad + \mathrm{M}\psi(X_t, Y_t, t)\nu(t)\psi_1(X_t, Y_t, t)^T\eta(Y_t, U_t, t)^T\,,\\ \kappa_{22} &= \mathrm{M}\eta(Y_t, U_t, t)\psi_1(X_t, Y_t, t)\nu(t)\times\\ &\quad \times\psi_1(X_t, Y_t, t)^T\eta(Y_t, U_t, t)^T\,; \end{aligned}$$

$$\begin{aligned} &A\alpha_t\kappa_{11} + \mathrm{M}\left(AU_t - X_t\right)\left(\xi^T\alpha_t^T + \gamma_t^T\right)\frac{\partial\xi^T}{\partial u} =\\ &\qquad = \kappa'_{01} - A\beta_t\kappa'_{21}\,, \end{aligned} \tag{6}$$

где

$$\begin{aligned} \kappa_{01} &= \mathrm{M}\left[\,\varphi(X_t, Y_t, t) - m_0\,\right]\xi(Y_t, U_t, t)^T\,,\\ \kappa_{11} &= \mathrm{M}\left[\,\xi(Y_t, U_t, t) - m_1\,\right]\xi(Y_t, U_t, t)^T\,,\\ \kappa'_{21} &= \mathrm{M}\left[\,\eta(Y_t, U_t, t))\varphi_1 - m_2\,\right]\xi(Y_t, U_t, t)^T\,, \end{aligned}$$

$$\begin{aligned} \kappa'_{01} &= \kappa_{01} + \mathrm{M}(X_t - AU_t)\frac{\partial\xi^T}{\partial t} + \mathrm{M}\big\{(X_t - AU_t)\varphi_1^T +\\ &+ \psi\nu\psi_1^T - A\beta_t\eta\psi_1\nu\psi_1^T\big\}\left(\frac{\partial}{\partial y} + \eta^T\beta_t^T\frac{\partial}{\partial u}\right)\xi^T +\\ &+ \frac{1}{2}\mathrm{M}(X_t - AU_t)\Big\{\mathrm{tr}\Big[\psi_1\nu\psi_1^T\left(\frac{\partial}{\partial y} +\\ &+ 2\eta^T\beta_t^T\frac{\partial}{\partial u}\right)\frac{\partial^T}{\partial y}\Big] +\\ &+ \mathrm{tr}\left[\beta_t\eta\psi_1\nu\psi_1^T\eta^T\beta_t^T\frac{\partial}{\partial u}\frac{\partial^T}{\partial u}\right]\Big\}\xi^T dt\,. \end{aligned}$$

Для вычисления математических ожиданий в m_0, m_1, m_2, κ_{02}, κ_{22}, κ_{11}, κ_{21}, κ_{01}, κ'_{01}, κ'_{21} в общем случае необходимо знать совместное распределение векторов Y_t, X_t, U_t при любом $t \geq t_0$, т.е. одномерное распределение составного случайного процесса $Z_t = \left[\,Y_t^T X_t^T U_t^T\,\right]^T$. Это распределение определяется уравнением Пугачева для характеристической функции, соответствующем системе стохастических дифференциальных уравнений (1) и (2):

$$\begin{aligned} \frac{\partial g_1(\lambda_1, \lambda_2, \lambda_3; t)}{\partial t} &= \mathrm{M}\big\{i\lambda_1^T\varphi_1 + i\lambda_2^T\varphi +\\ &+ i\lambda_3^T\left[\alpha\xi(Y_t, U_t, t) + \beta_t\eta(Y_t, U_t, t)\varphi_1 + \gamma_t\right] +\\ &+ \chi(\psi_1^T\lambda_1 + \psi^T\lambda_2 +\\ &+ \psi_1^T\eta(Y_t, U_t, t)^T\beta_t^T\lambda_3; t)\big\}\exp\big\{i\lambda_1^T Y_t +\\ &+ i\lambda_2^T X_t + i\lambda_3^T U_t\big\}\,. \end{aligned} \tag{7}$$

К этому уравнению следует добавить начальное условие

$$g_1(\lambda_1, \lambda_2, \lambda_3; t_0) = g_0(\lambda_1, \lambda_2, \lambda_3)\,,$$

где $g_0(\lambda_1, \lambda_2, \lambda_3)$ — совместная характеристическая функция начальных значений $Y_0 = Y(t_0)$, $X_0 = X(t_0)$, $U_0 = U(t_0)$ процессов Y_t, X_t, U_t. Разумеется, начальное распределение определяемого уравнением (2) процесса U_t не может быть известным. Поэтому его неизбежно надо задавать более или менее произвольно. Единственное требование, которому следует подчинить это распределение, состоит в том, чтобы в начальный момент $t = t_0$ удовлетворялись условие несмещенности оценки и условие $\mathrm{M}(\hat{X}_t - X_t)\xi_t^T = 0$. Только в этом случае фильтр, определяемый формулой $\hat{X}_t = AU_t$ и уравнением (1) будет условно оптимальным.

Уравнение (7) можно решить совместно с уравнениями (4)–(6) любым приближенным методом [4]. При этом, взяв достаточно большое N, можно найти решение с любой степенью точности. Вычисления, необходимые для определения коэффициентов α_t, β_t, γ_t в уравнении (1) условно оптимального фильтра и совместного распределения Y_t, X_t, U_t при любом $t \geq t_0$, конечно, очень сложны, особенно в многомерных задачах. Однако эти вычисления используют только априорные данные и не опираются на результаты наблюдений, поэтому их надо выполнять для каждой конкретной задачи (или класса задач) только один раз при синтезе фильтра (алгоритма фильтрации). Практическое применение фильтра при каждом конкретном эксперименте требует только решения уравнения (1) при известных функциях времени α_t, β_t, γ_t.

Для рассматриваемого фильтра формула для производной ковариационной матрицы ошибки принимает вид:

$$\begin{aligned}\dot{R}_t = \mathrm{M}[&(X_t - \hat{X}_t)\varphi(X_t, Y_t, t)^T + \\ &+ \varphi(X_t, Y_t, t)(X_t^T - \hat{X}_t^T) - \\ &- A\beta_t\eta(Y_t, U_t, t)\psi_1(X_t, Y_t, t)\nu(t) \times \\ &\times \psi_1(X_t, Y_t, t)^T\eta(Y_t, U_t, t)^T\beta_t^T A^T + \\ &+ \psi(X_t, Y_t, t)\nu(t)\psi(X_t, Y_t, t)^T].\end{aligned}$$

В [5–7] основные теоремы условно оптимального оценивания обобщены на случай непрерывных СтС с винеровскими и пуассоновскими шумами:

$$\begin{aligned}dX_t &= \varphi(X_t, Y_t, t)dt + \psi'(X_t, Y_t, t)dW_0 + \\ &\quad + \int\limits_{R_0^q} \psi''(X_t, Y_t, t, v)P^0(dt, dv)\,, \\ dY_t &= \varphi_1(X_t, Y_t, t)dt + \psi_1'(X_t, Y_t, t)dW_0 + \\ &\quad + \int\limits_{R_0^q} \psi_1''(X_t, Y_t, t, v)P^0(dt, dv)\,.\end{aligned}$$

Здесь $W_0 = W_0(t)$ — винеровский процесс, а $P^0(dt, dv)$ — центрированная пуассоновская мера.

Среди новых результатов общей теории условно оптимального оценивания следует выделить следующие:

- оптимальную в среднеквадратическом теорию оценивания при автокоррелированных помехах в наблюдениях [4, 5, 7–9];
- теорию оценивания по различным вероятностным критериям [7, 10–13].
- теорию совместного оценивания и распознавания [4, 7, 14–16].

4 Фильтры Калмана–Бьюси и линейные фильтры Пугачева

Как известно [4, 7], для линейной негауссовской дифференциальной СтС

$$\begin{aligned}dX_t &= (aY_t + a_1X_t + a_0)dt + \psi dW\,, \\ dY_t &= (bY_t + b_1X_t + b_0)dt + \psi_1 dW\,,\end{aligned} \tag{8}$$

где $a, a_1, b, b_1, \psi, \psi_1$ — функции времени t, не зависящие от $X_t = [\,X_1 \ldots X_{n_x}\,]^T$, $Y_t = \left[\,Y_1 \ldots Y_{n_y}\,\right]^T$.

Линейный ФП определяется уравнением:

$$\begin{aligned}d\hat{X}_t = (aY_t + a_1\hat{X}_t + a_0)dt + \\ + \beta_t\left[\,dY_t - (bY_t + b_1\hat{X}_t + b_0)dt\,\right]\,.\end{aligned}$$

Для определения β_t необходимо найти математическое ожидание m_t и ковариационную матрицу K_t случайного вектора $Q_t = [Y_1 \ldots Y_{n_y}X_1 \ldots \hat{X}_{n_x}]^T$, ковариационную матрицу R_t ошибки $\tilde{X}_t = \hat{X}_t - X_t$. Эти уравнения в нашем случае имеют вид:

$$\begin{aligned}\dot{m}_t &= \bar{a}m_t + \bar{a}_0\,, \\ \dot{K}_t &= \bar{a}K_t + K_t\bar{a}^T + \bar{c}_0\nu\bar{c}_0^T\,,\end{aligned}$$

где

$$\bar{a} = \begin{bmatrix} b & b_1 \\ a & a_1 \end{bmatrix}, \quad \bar{a}_0 = \begin{bmatrix} b_0 \\ a_0 \end{bmatrix}, \quad \bar{c}_0 = \begin{bmatrix} \psi_1 \\ \psi \end{bmatrix}.$$

Уравнение для матрицы R_t имеет вид:

$$\begin{aligned}\dot{R}_t = a_1R_t + R_ta_1^T + \psi\nu\psi^T - \\ - \left[\,R_tb_1^T + \psi\nu\psi_1^T\,\right]\kappa_{22}^{-1}\left[\,b_1R_t + \psi_1\nu\psi^T\,\right]\,,\end{aligned}$$

При этом коэффициент β_t равен:

$$\beta_t = \left[\,R_tb_1^T + \psi\nu\psi_1^T\,\right]\kappa_{22}^{-1}\,,$$

где $\kappa_{22} = \psi_1\nu\psi_1^T$.

Для уравнения (8) при $a = b = 0$ при негауссовском белом шуме интенсивности ν линейный ФП (ЛФП) при $N = n_x$ имеет вид ФКБ [4, 7].

В случае независимых негауссовских шумов с интенсивностями ν_1 и ν_2 ЛФП совпадает с ФКБ для независимых шумов с интенсивностями ν_1 и ν_2.

Таким образом, ЛФП обладает следующими особенностями по сравнению в ФКБ:

- линейный ФП в отличие от ФКБ справедлив для дифференциальных СтС (8), содержащих одновременно и переменные состояния X_t, и наблюдения Y_t, при этом белые шумы могут быть негауссовскими;
- при $N = n_x$ для гауссовских шумов в (8) не зависящих от Y_t $(a = b = 0)$, ЛФП совпадает с ФКБ. При этом порядок уравнений для синтеза фильтра составляет $Q_{\text{ФКБ}} = n_x(n_x + 3)/2$;
- при одинаковой точности (более простых в реализации) порядок ЛФП может быть меньше $Q_{\text{ФКБ}}$.

В [4, 7] содержатся новые результаты по теории линейного условно оптимального оценивания в случае автокоррелированной помехи в наблюдениях, а также совместного условно оптимального оценивания и распознавания. Получены необходимые и достаточные условия глобальной устойчивости ЛФП.

Наконец, упомянутые результаты удалось распространить на линейные дифференциальные СтС (8) с параметрическими шумами, когда коэффициенты при dW имеют следующий вид:

$$\psi = c_{10} + \sum_{r=1}^{n_y} c_{1,r} Y_r + \sum_{r=1}^{n_x} c_{1,n_y+r} X_r \,,$$

$$\psi_1 = c_{20} + \sum_{r=1}^{n_y} c_{12,r} Y_r + \sum_{r=1}^{n_x} c_{12,n_y+r} X_r \,,$$

а также на случай автокоррелированной помехи в наблюдениях.

5 Особенности условно оптимального оценивания для дискретных стохастических систем

Те же рассуждения, которые привели нас в разд. 2 к классам допустимых фильтров, описываемых дифференциальными уравнениями, подсказывают мысль использовать дискретные фильтры, описываемые разностными уравнениями, и дискретные наблюдения в задачах оценивания. Для решения этих задач в реальном масштабе времени рассмотрим принцип дискретного условно оптимального оценивания Пугачева. Этот принцип состоит в отказе от абсолютной оптимальности и ограничении оптимальными оценками для некоторых узких классов допустимых оценок, удовлетворяющих некоторым простым в реализации разностным уравнениям, которые могут быть вычислены на основе результатов наблюдений в масштабе реального времени. Главная трудность при синтезе условно оптимальных фильтров состоит в выборе класса допустимых фильтров. Обычно в практических задачах за класс допустимых дискретных условно оптимальных фильтров принимают множество фильтров, описываемых конечномерными разностными уравнениями с некоторыми неизвестными коэффициентами. В этом случае проблема оптимизации сводится к определению оптимальных значений всех неизвестных коэффициентов, которые в общем случае зависят от времени.

Первой особенностью нелинейного условно оптимального оценивания служит то обстоятельство, что такое оценивание является многокритериальным, поскольку требуется минимизация $\mathrm{M}\left|\hat{X}_k - X_k\right|^2$ для любого момента времени k из некоторого интервала.

Второй особенностью нелинейного условно оптимального оценивания является то, что оптимальные коэффициенты фильтров должны определяться в ходе проектирования фильтра только априорными данными без использования текущих наблюдений, как это имеет место в фильтрах Калмана. Данные текущих наблюдений используются только в процессе фильтрации при рекуррентном решении уравнений фильтра.

Поставим задачу найти условно оптимальную оценку $\hat{X}_k$ для любого момента времени k случайных величин $\hat{X}_k$, используя наблюдения случайных величин $Y_1^k = \{Y_1, \ldots, Y_k\}$ в классе допустимых фильтров.

Рассмотрим сначала нелинейную регрессионную дискретную СтС:

$$\begin{aligned} X_{k+1} &= \omega_k(X_k, Y_k, V_k)\,, \\ Y_k &= \omega_{1k}(X_k, Y_k, V_k) \quad (k = 1, 2, \ldots)\,. \end{aligned} \tag{9}$$

Определим класс допустимых фильтров формулой

$$\hat{X}_k = AU_k \tag{10}$$

и разностным уравнением

$$U_{k+1} = \delta_k \zeta_k(Y_k, U_k) + \gamma_k \,. \tag{11}$$

Здесь A — некоторая постоянная $(n_x \times N)$-матрица, $N \geq n_x$, ранга n_x; ζ_k — некоторые известные,

так называемые структурные функции (в общем случае векторные функции размерности q); δ_k — произвольные $(N \times q)$-матрицы коэффициентов фильтров; а γ_k — произвольные $(N \times 1)$-матрицы столбцы смещений нуля. Каждому выбору значений δ_k, γ_k соответствует определенный допустимый фильтр, а все возможные значения δ_k, γ_k определяют класс допустимых фильтров для данных функций ζ_k. Различные последовательности функций $\{\zeta_k\}$ определяют различные классы допустимых фильтров. Каждому выбору $\{\zeta_k\}$ соответствует определенный класс допустимых фильтров.

Последовательность функций $\{\zeta_k\}$ в (11) может быть в принципе произвольной. Но точность фильтрации зависит от выбора $\{\zeta_k\}$. Таким образом, встает вопрос о рациональном выборе $\{\zeta_k\}$. Априори можно только сказать, что чем больше размерность структурных векторных функций ζ_k, тем выше может быть точность фильтрации.

Следуя В. С. Пугачеву, примем за оптимальный такой допустимый фильтр, который минимизирует средний квадрат ошибки $\mathrm{M}\left|\hat{X}_{k+1} - X_{k+1}\right|^2$ на каждом шаге (при каждом k) путем выбора δ_k, γ_k в (11) при данных значениях δ_h, γ_h, $h \le k$, найденных в результате предыдущих шагов. Такой фильтр называется **дискретным условно оптимальным фильтром** или **дискретным ФП**. Значения δ_k и γ_k в (11), соответствующие условно оптимальному фильтру, принимаются за оптимальные значения δ_k, γ_k.

Уравнение (11) показывает, каким образом допустимый фильтр использует в каждый момент времени $(k+1)$ информацию, содержащуюся в предыдущих результатах наблюдений $Y_1, \ldots, Y_k$. А именно, эта информация используется только через U_k. И только текущий результат наблюдения Y_k используется непосредственно при формировании оценки $\hat{X}_{k+1}$ в момент времени k. Таково условие, при котором $\mathrm{M}\left|\hat{X}_{k+1} - X_{k+1}\right|^2$ минимизируется в каждый момент времени $(k+1)$ условно оптимальным фильтром.

Таким образом, задача синтеза ФП сводится к нахождению оптимальных последовательностей $\{\delta_k\}$ и $\{\gamma_k\}$ в (11).

Как показано в [3, 7], при условии существования одномерных моментов в основе теории условно оптимального оценивания, согласно (10), (11) для СтС (9) лежат следующие рекуррентные уравнения:

$$
\begin{aligned}
A\gamma_k &= m_{k+1} - A\delta_k\rho_k\,,\\
A\delta_k &= D_k B_k^{-1}\,,\\
m_{k+1} &= \mathrm{M}\omega_k(X_k, V_k)\,,\\
\rho_k &= \mathrm{M}\zeta_k(\omega_{1k}(X_k, V_k), U_k)\,,\\
B_k &= \mathrm{M}[\zeta_k(\omega_{1k}(X_k, V_k), U_k) - \\
&\qquad - \rho_k]\zeta_k(\omega_{1k}(X_k, V_k), U_k)^T\,,\\
D_k &= \mathrm{M}\left[\omega_k(X_k, V_k) - m_{k+1}\right] \times \\
&\qquad \times \zeta_k(\omega_{1k}(X_k, V_k), U_k)^T\,,\\
g_{1,k+1}(\lambda,\mu) &= \mathrm{M}\exp\left\{i\lambda^T\omega_k(X_k, V_k) + \right.\\
&\quad \left. + i\mu^T\left[\,\delta_k\zeta_k(\omega_{1k}(X_k, V_k), U_k) + \gamma_k\,\right]\right\}\,. \qquad (12)
\end{aligned}
$$

Начальным условием для рекуррентного уравнения (12) служит начальное значение характеристической функции

$$
g_{1,1}(\lambda,\mu) = \mathrm{M}\exp\left\{i\lambda^T X_1 + i\mu^T U_1\right\},
$$

вычисляемое для начальных значений X_1 и U_1.

Для нелинейной авторегрессионной дискретной СтС вида:

$$
\begin{aligned}
X_{k+1} &= \varphi_k(X_k, Y_k) + \psi_k(X_k, Y_k)V_k\,,\\
Y_k &= \varphi_{1k}(X_k, Y_k) + \psi_{1k}(Y_k)V_k \quad (k = 1, 2, \ldots)
\end{aligned} \qquad (13)
$$

уравнения условно оптимального фильтра имеют вид:

$$
\begin{aligned}
\hat{X}_{k+1} &= \alpha_k\xi_k(\hat{X}_k) + \beta_k\eta_k(\hat{X}_k)Y_k + \gamma_k\,,\\
\alpha_k\kappa_{11}^{(k)} + \beta_k\kappa_{21}^{(k)} &= \kappa_{01}^{(k)}\,,\\
\alpha_k\kappa_{12}^{(k)} + \beta_k\kappa_{22}^{(k)} &= \kappa_{02}^{(k)}\,,\\
\gamma_k &= \rho_0^{(k)} - \alpha_k\rho_1^{(k)} - \beta_k\rho_2^{(k)}\,,
\end{aligned}
$$

где

$$
\begin{aligned}
\rho_k &= \left[\,\rho_1^{(k)T}\rho_2^{(k)T}\,\right]^T\,,\\
\rho_1^{(k)} &= \mathrm{M}\xi_k(\hat{X}_k)\,, \quad \rho_2^{(k)} = \mathrm{M}\eta_k(\hat{X}_k)\varphi_{1k}(X_k)\,,\\
\kappa_{01}^{(k)} &= \mathrm{M}\left[\,\varphi_k(X_k) - m_{k+1}\,\right]\xi_k(\hat{X}_k)^T\,,\\
\kappa_{02}^{(k)} &= \mathrm{M}\left[\,\varphi_k(X_k) - m_{k+1}\,\right]\varphi_{1k}(X_k)^T\eta_k(\hat{X}_k)^T + \\
&\qquad + \mathrm{M}\psi_k(X_k)\nu_k\psi_{1k}(X_k)^T\eta_k(\hat{X}_k)^T\,;\\
m_{k+1} &= \rho_0^{(k)}, \quad \rho_0^{(k)} = \mathrm{M}\varphi_k(X_k)\,,\\
B_k &= \begin{bmatrix}\kappa_{11}^{(k)} & \kappa_{12}^{(k)}\\ \kappa_{21}^{(k)} & \kappa_{22}^{(k)}\end{bmatrix}\,, \quad \det|B_k| \neq 0\,,\\
\kappa_{11}^{(k)} &= \mathrm{M}\left[\xi_k(\hat{X}_k) - \rho_1^{(k)}\right]\xi_k(\hat{X}_k)^T\,,\\
\kappa_{12}^{(k)} &= \kappa_{21}^{(k)T} = \mathrm{M}\left[\xi_k(\hat{X}_k) - \right.\\
&\qquad \left. - \rho_1^{(k)}\right]\varphi_{1k}(X_k)^T\eta_k(\hat{X}_k)^T\,,\\
\kappa_{22}^{(k)} &= \mathrm{M}\left[\eta_k(\hat{X}_k)\varphi_{1k}(X_k) - \right.\\
&\qquad \left. - \rho_2^{(k)}\right]\varphi_{1k}(X_k)^T\eta_k(\hat{X}_k)^T + \\
&\quad + \mathrm{M}\eta_k(\hat{X}_k)\psi_{1k}(X_k)\nu_k\psi_{1k}(X_k)^T\eta_k(\hat{X}_k)^T\,,
\end{aligned}
$$

$$g_{1,k+1}(\lambda,\mu) = \mathrm{M} h_k\big(\psi_k(X_k)^T\lambda + \\ + \psi_{1k}(X_k)^T\eta_k(\hat{X}_k)\beta_k^T\mu\big)\exp\big\{i\lambda^T\varphi_k(X_k) + \\ + i\mu^T[\alpha_k\xi_k(\hat{X}_k) + \beta_k\eta_k(\hat{X}_k)\varphi_{1k}(X_k) + \gamma_k]\big\},$$

где h_k — характеристическая функция V_k.

Следует обратить внимание, что в своих работах В. С. Пугачев трактовал дискретный фильтр Калмана не так, как это принято большинством авторов. Он принимал за дискретный фильтр Калмана обычный **одношаговый линейный предсказатель** и так же построил изложенную выше нелинейную **дискретную** условно оптимальную фильтрацию. В западной литературе делают различие между алгоритмами фильтрации и одношаговыми предсказателями [7]. А В. С. Пугачев отождествлял их. Это просто разная трактовка термина **фильтрация**. Владимир Семенович Пугачев понимал под **дискретной фильтрацией** то, что невозможно в один и тот же момент времени k одновременно и получать наблюдение, и тут же (мгновенно в этот же момент времени) оценивать состояние. Поэтому, например, структуру линейного дискретного фильтра он записывал в следующей форме:

$$X_{k+1} = \alpha_k X_k + \beta_k Y_k + \gamma_k \qquad (14)$$

и таким образом получался либо ФП, либо «одношаговый предсказатель»:

$$(X_k, Y_k) \xrightarrow[\text{прогноз}]{} X_{k+1}\,,$$

который В. С. Пугачев называл **дискретным фильтром Калмана** [3]. В западной литературе (и Калман сам) записывают дискретную версию фильтра Калмана «по определению», когда в один и тот же момент времени k мы и получаем наблюдение, и тут же (мгновенно) производим оценку состояния. Тогда уравнение для оценки запишется в следующем виде:

$$X_{k+1} = \bar{\alpha}_k X_k + \bar{\beta}_k Y_{k+1} + \bar{\gamma}_k\,. \qquad (15)$$

Вследствие разницы форм (14) и (15) получаются разные уравнения для коэффициентов усиления фильтров $\alpha_k, \beta_k, \gamma_k$.

Наиболее полное изложение современной теории дискретных ЛФП для линейных дискретных СтС, в том числе с параметрическими, гауссовскими и негауссовскими шумами, а также в случае дискретной автокоррелированной помехи в наблюдениях содержится в [3, 4, 7]. Теория условно оптимальной интерполяции для СтС (9) и (13) разработана в [7, 17, 18]. Вопросы дискретной теории условно оптимального оценивания по различным вероятностным критериям, теории совместного дискретного условно оптимального оценивания, идентификации и распознавания обсуждаются в [4, 7, 19–27].

6 Заключение

В настоящее время теория условно оптимальных линейных и нелинейных ФП для конечномерных непрерывных и дискретных СтС получила широкое применение для оперативной обработки информации в информационных и информационно-управляющих системах морской, авиационно-космической, медицинской и другой техники. Создание программных средств (ПС), реализующих ФП, представляет собой нетривиальную задачу. Сложность задачи для нелинейных СтС заключается в том, что ПС должны автоматически по исходным нелинейным уравнениям СтС составлять и решать систему уравнений высокого порядка для определения неизвестных параметров распределения, а также вычислять коэффициенты ФП. Известные ПС в основном представляют собой отдельные программы, предназначенные для решения конкретных задач. Проблема разработки информационной технологии и базового программного обеспечения впервые была поставлена и решена в ИПИ РАН. Вопросам создания и применения ПС для ФП посвящена обширная литература (см., например, [4, 7, 28–41]).

Среди направлений дальнейшего развития ФП можно выделить следующие:

- совершенствование методического обеспечения синтеза ФП на базе использования эффективных методов параметризации распределений (эллипсоидальных фильтров [4, 7, 40, 41], модифицированных методов параметризации ненормированных распределений [7], квазилинейных методов [4, 7, 42] и др.), а также непараметрических методов [43, 44];
- развитие алгоритмического обеспечения с учетом архитектурных особенностей используемых средств вычислительной техники [45–47];
- создание новых комплексных информационных технологий для анализа и синтеза ФП для СтС, в том числе в бесконечных пространствах для контроля и мониторинга, управления функционированием и обеспечения безопасности [7, 48–62].

Литература

1. *Доступов Б. Г., Емельянов С. В., Казаков И. Е., Кибзун А. И., Кузнецов Н. А., Мизин И. А., Синицына И. В., Синицын И. Н., Синицын В. И.* Обзор научных трудов академика В. С. Пугачева // Автоматика и телемеханика, 1998. № 11. С. 8–20.

2. *Синицын И. Н., Синицын В. И., Соколов И. А.* О работах академика В. С. Пугачева в области математических методов информатики // В кн. «Системы средства информатики». Спец. вып. «Математические методы информатики». М.: Наука, Физматлит, 2001. С. 5–15.
3. *Пугачев В. С.* Теория вероятностей и математическая статистика. Учеб. пособие. М.: Наука, 1-е изд. 1979; 2-е изд. 2002.
4. *Пугачев В. С., Синицын И. Н.* Стохастические дифференциальные системы. Анализ и фильтрация. М.: Наука, 1-е изд. 1985; 2-е изд. 1990. [Англ. пер.: Stochastic differential systems. Analysis and filtering. Chichester, New York: John Wiley, 1987.]
5. *Пугачев В. С., Синицын И. Н.* Стохастические системы. Теория и программное обеспечение. Юбилейный сборник трудов институтов Отделения информатики, вычислительной техники и автоматизации РАН. М.: Изд. ОИТВС РАН, 1993. Т. 1. С. 75–93.
6. *Пугачев В. С., Синицын И. Н.* Прикладные методы анализа стохастических систем. М.: Изд-во ОИТВС РАН. Вестник МАИ, 1994. № 1. С. 39–47.
7. *Синицын И. Н.* Фильтры Калмана и Пугачева. М.: Изд-во Логос, 2006.
8. *Sinitsyn I. N.* Ill problems of on-line conditionally optimal filtering // Ill-Posed Problems in Natural Sciences: Proceedings of the International Conference. Moscow, August, 19–25, 1991. Utrecht: VSP. Moscow: TVP Sci. Publ. P. 174–183.
9. *Синицын И. Н., Синицын В. И., Корепанов Э. Р., Белоусов В. В.* Современное методическое и программное обеспечение анализа качества и моделирования стохастических систем управления // В кн. Труды III международной конференции «Идентификация систем и задачи управления» (SICPRO'04), 2004. CD-ROM. С. 17–43.
10. *Синицын И. Н., Шин В. И.* Условно оптимальная фильтрация процессов в стохастических дифференциальных системах по сложным статистическим критериям // Докл. АН СССР, 1991. Т. 320. № 4. С. 813–817.
11. *Синицын И. Н., Мощук Н. К., Шин В. И.* Условно оптимальная фильтрация в стохастических дифференциальных системах по бейесовым критериям // Докл. РАН, 1993. Т. 330. № 4. С. 436–439.
12. *Синицын И. Н., Шин В. И., Корепанов Э. Р.* Теория условно оптимальной фильтрации стохастических процессов по сложным статистическим критериям // В кн. «Системы и средства информатики». Вып. 5. М.: Наука, 1993. С. 106–120.
13. *Sinitsyn I. N., Korepanov E. R., Shin V. I.* Methods, algorithms and software tools for CAE of stochastic control systems // EUROSIM'98 Congress Proceedings. Helsinki University of Technology, Espoo, Finland. April 14–15, 1998. P. 200–205.
14. *Синицын И. Н., Корепанов Э. Р.* Теория и программное обеспечение условно оптимальной фильтрации и распознавания сигналов в стохастических системах // Тез. докл. 2-й Всероссийской конференции «Распознавание образов и анализ изображений: новые информационные технологии» (РОАИ-2-95). Ульяновск, 1995. Часть 2. С. 8–9.
15. *Синицын И. Н.* Условно оптимальная фильтрация и распознавание сигналов в стохастических дифференциальных системах // Автоматика и телемеханика, 1997. № 3. С. 124–130.
16. *Синицын И. Н., Шин В. И.* Распознавание процессов, определяемых стохастическими дифференциальными уравнениями // Докл. РАН, 1998. Т. 359. № 2. С. 1–5.
17. *Синицын И. Н., Шин В. И.* Условно оптимальная интерполяция случайных последовательностей, определяемых разностными уравнениями // Докл. РАН, 1994. Т. 336. № 4. С. 453–456.
18. *Синицын И. Н., Шин В. И., О М.* Условно оптимальная интерполяция случайных процессов с фиксированной точкой в стохастических дифференциальных системах // Автоматика и телемеханика, 1997. № 2. С. 224–233.
19. *Shin V. I.* Statistical analysis and suboptimal filtering of random processes in nonlinear stochastic systems // Workshop (International) on Advanced Electronics Technology-95. Moscow, Russia, 1995. P. 11–15.
20. *Shin V. I.* Statistical analysis and filtering of processes in nonlinear stochastic systems. Seoul, Korea: The Korean Mathematical Society, 1995. Vol. 32. No. 1. P. 4–8.
21. *Lee Y., Cho Y., Oh M., Shin V. I.* Iterated conditionally optimal filters // Automation and Remote Control, 1997. Vol. 58. No. 6. P. 961–968.
22. *Менхо О., Чангхи Хан, Синицын И. Н., Шин В. И.* Рекуррентная фильтрация в дискретных нелинейных системах с неизвестными параметрами // Автоматика и телемеханика, 1998. № 1. С. 44–63.
23. *Oh M., Shin V. I., Lee Y., Choi U. J.* Suboptimal discrete filters for stochastic systems with different types of observations // Computer and Mathematics with Applications, 1998. Vol. 35. No. 3. P. 17–27.
24. *Oh M., Han C., Sinitsyn I. N., Shin V. I.* Recursive filtering in discrete non-linear systems with unknown parameters // Automation and Remote Control, 1998. Vol. 59. No. 1. P. 36–43.
25. *Азаров С. В., Менхо О., Синицын И. Н., Шин В. И.* Метод нормальной аппроксимации в задачах адаптивной дискретной фильтрации и распознавания // Автоматика и телемеханика, 1998. № 11. С. 21–31.
26. *Lee Y., Oh M., Shin V. I.* Adaptive nonlinear continuous-discrete filtering // Applied Numerical Mathematics, 2001. No. 47. P. 45–56.
27. *Shin V. I., Shevlyakov G.* An optimal mean square combination of estimates with application to filtering problems // 7th Conference (International) on Pattern Recognition and Image Analysis (PRIA-7-2004) Proceedings. St. Petersburg, Russia. Vol. 2. P. 394–397.
28. *Пугачев В. С., Синицын И. Н.* Направления развития математического обеспечения для исследования стохастических систем // В кн. «Информатика: проблемы, перспективы». М.: Наука, 1986. С. 30–38.

29. *Pugachev V. S., Sinitsyn I. N., Shin B. I.* Problems of analysis and on-line conditionally optimal filtering of processes in nonlinear stochastic systems // 2nd IFAC Symposium on Stochastic Control. Vil'nius, USSR, 1986. Moscow. Preprints. Part 1. P. 4–18.
30. *Синицын И. Н., Петрова Е. В., Шин В. И.* Алгоритмическое и программное обеспечение для фильтрации случайных процессов с использованием ПЭВМ // Тез. докл. 3-го межведомственного семинара по актуальным вопросам вычислительной техники и информатики «Электронный офис для научных исследований». М.: ИАЭ им. Курчатова, 1988. С. 20–23.
31. *Пугачев В. С., Синицын И. Н.* Современное состояние и перспективы развития математического обеспечения для исследования стохастических систем // Тез. докл. Всесоюзного совещания «Проблемы управления-89». Ташкент, 1989. Т. 1. С. 504–505.
32. *Синицын И. Н., Карпенко А. П., Чередничено, Корепанов Э. Р.* Диалоговый комплекс для исследования и моделирования стохастических систем на базе ПЭВМ // В сб. трудов 3-й Всесоюзной школы «Прикладные проблемы управления макросистемами». Апатиты. М.: Изд-во ВНИИСИ, 1989. С. 237–239.
33. *Sinitsyn I. N.* Problems of signal analysis and conditionally optimal processing in stochastic differential systems // Latvian Signal Processing Conference (International) Proceedings. Riga, 1990. Vol. 2. P. 60–64.
34. *Пугачев В. С., Синицын И. Н.* (ред.). Принципы разработки интеллектуализированных ППП для построения условно оптимальных фильтров. Пакет прикладных программ «СтС-Фильтр». Г. К. Алдушин, Р. Н. Бабкина, В. Ф. Бурлака, В. Ю. Вигдерович, Б. И. Гершиков, Э. Р. Корепанов, О. А. Куленко, В. С. Пугачев, В. И. Синицын, И. Н. Синицын, А. П. Хатунцев. В. И. Шин. Препринт. М.: ИПИ АН СССР, 1991.
35. *Синицын И. Н., Маишева Е. Ю., Корепанов Э. Р., Мощук Н. К., Огнева О. С., Синицын В. И., Шин В. И., Хатунцев А. П.* Программные средства для анализа и моделирования случайных процессов, проектирования фильтров и идентификаторов на ПЭВМ // Тез. докл. IV Всесоюзной научно-технической конференции «Перспективные методы планирования и анализа экспериментов при исследовании случайных полей и процессов». Петрозаводск, 1991. М.: Изд-во МЭИ, 1991. С. 82–83.
36. *Пугачев В. С., Синицын И. Н., Хатунцев А. П., Шин В. И., Корепанов Э. Р., Синицын В. И.* Проблемы разработки математического обеспечения для проектирования дискретных условно оптимальных фильтров // В кн. «Системы и средства информатики». Вып. 3. М.: Наука, 1992. С. 3–19.
37. *Пугачев В. С., Синицын И. Н., Хатунцев А. П., Шин В. И., Корепанов Э. Р., Синицын В. И.* Математическое обеспечение для проектирования условно оптимальных фильтров и анализа процессов в дискретных стохастических системах // Автоматика и телемеханика, 1992. № 6. С. 78–85.
38. *Sinitsyn I. N., Karpenko A. P.* Combined parallel statistical and analysis modeling methods, algorithms and software for dynamical stochastic systems research // EUROSIM 1996 Conference (International) Proceedings. L. Dekker, W. Smit, J. C. Zuidervaat (eds.). Elsevier Science B.V., 1996. P. 187–194.
39. *Корепанов Э. Р.* Развитие алгоритмического и программного обеспечения для синтеза фильтров Пугачева // В кн. «Системы и средства информатики». Спец. вып. М.: Наука, Физматлит, 2001. С. 37–42.
40. *Синицын И. Н., Синицын В. И., Корепанов Э. Р., Белоусов В. В.* Информационные технологии синтеза параметризованных фильтров Пугачева. М.: Наукоемкие технологии, 2004. № 7. С. 50–79.
41. *Синицын И. Н., Синицын В. И., Корепанов Э. Р., Белоусов В. В., Чумин Ю. В.* Методические и программное обеспечение анализа качества и моделирования сингулярных стохастических систем // Труды IV международной конференции «Идентификация систем и проблемы управления» (SICPRO'05), 2005. CD-ROM. С. 1713–1743.
42. *O M., Shin V. I.* Modified quasilinear filtering method for estimation of processes in multidimentional nonlinear stochastic systems // Kybernetika, 1997. Vol. 33. No. 4. P. 399–408.
43. *Добровидов А. В., Кошкин Т. М.* Непараметрическое оценивание сигналов. М.: Наука, Физматлит, 1997.
44. *Васильев В. А., Добровидов А. В., Кошкин Г. М.* Непараметрическое оценивание функционалов от распределений стационарных последовательностей. М.: Наука, 2004.
45. *Sinitsyn I. N.* Parallel simulation technologies for stochastic systems // In Lectures Notes in Computer Science, 1277. Parallel Computing Technologies, 4th Conference (International) PACT-97 Proceedings. Springer, 1997. P. 383–388.
46. *Синицын И. Н., Синицын В. И., Степанов А. М., Ушмаев О. С.* Проблемы реализации вычислительных методов обработки и анализа сигналов и изображений на архитектурах с ассоциативной памятью // Труды 1-й Всероссийской конференции «Методы и средства обработки информации». М.: Изд-во МГУ им. М. В. Ломоносова, 2003. С. 137–141.
47. *Синицын И. Н., Степанов А. М., Ушмаев О. С.* Проблемы синтеза фильтров и идентификаторов с ассоциативной памятью // Труды III международной конференции «Идентификация систем и проблемы управления» (SICPRO'04), 2004. CD-ROM. С. 1896–1911.
48. *Казаков И. Е., Мальчиков С. В.* Приближенное построение фильтров Пугачева заданной сложности // Автоматика и телемеханика, 1981. № 12. С. 48–55.
49. *Пугачев В. С.* Управление летными и испытаниями летательных аппаратов как средство повышения их надежности // В кн. «Проблемы надежности летательных аппаратов». М.: Машиностроение, 1985. С. 25–37.

50. *Казаков И. Е., Гладков Д. И.* Методы оптимизации стохастических систем. М.: Наука, Гл. ред. физматлит, 1987.

51. *Руденко Е. А.* Оптимальная структура дискретных нелинейных фильтров произвольного порядка // В кн. «Статистические методы в теории управления ЛА». Тем. сб. науч. тр. МАИ. М.: Изд-во МАИ, 1990. С. 53–60.

52. *Руденко Е. А.* Адаптивный дискретный нелинейный фильтр для реализации на борту ЛА // В кн. «Управление и навигация ЛА в условиях параметрических неопределенности». Тем. сб. науч. тр. МАИ. М.: Изд-во МАИ, 1991. С. 23–30.

53. *Корепанов Э. Р.* Дискретные условно оптимальные фильтры с памятью // Докл. РАН, 1992. Т. 324. № 1. С. 50–55.

54. *Панков А. Р.* Рекуррентная условно-минимаксная фильтрация процессов в разностных нелинейных стохастических системах // Изв. АН СССР. Техническая кибернетика, 1992. № 3. С. 63–70.

55. *Руденко Е. А.* Оптимальная структура дискретных алгоритмов конечномерной непрерывно-дискретной нелинейной фильтрации при марковских помехах // В кн. «Оптимизация алгоритмов обработки информации и управления». Тем. сб. науч. тр. МАИ. М.: Изд-во МАИ, 1992. С. 62–70.

56. *Pankov A. R.* Conditionally-minimax nonlinear filter for differential system with discrete observations // Advances in Modelling and Analysis. Ser. B. AMSE Press., 1993. Vol. 28. No. 1. P. 31–39.

57. *Pankov A. R., Bosov A. V.* Conditionally-miminax algorithm of nonlinear system state estimation // IEEE Trans. Autom. Control, 1994. Vol. 39. No. 8. P. 1617–1620.

58. *Синицын И. Н.* Из опыта преподавания статистических основ информатики в технических университетах // В кн. «Системы и средства информатики». Спец. вып. 8, посвященный II международному конгрессу ЮНЕСКО «Образование и информатика». М.: Наука, Физматлит, 1996. С. 68–73.

59. *Шин В. И.* Фильтры Пугачева для комплексной обработки информации // Автоматика и телемеханика, 1998. № 11. С. 195–206.

60. *Босов А. В.* Условно-минимаксные алгоритмы оценивания и управления для нелинейных стохастических систем // Автоматика и телемеханика, 1998. № 11. С. 46–59.

61. *Синицын И. Н., Синицын В. И., Корепанов Э. Р., Белоусов В. В., Ильясов Д. Ф., Ушмаев О. С.* Субоптимальные обучающиеся информационные технологии и системы // Материалы межрегиональной научно-технической конференции «Интеллектуальные информационные системы» (Интеллект-2003). Тула: Изд-во Тульского государственного университета, 2003. С. 25–27.

62. *Синицын И. Н., Синицын В. И., Корепанов Э. Р., Белоусов В. В., Лавренюк Ю. А.* Условно оптимальные стохастические информационные технологии контроля сложных динамических систем // В кн. «Системы и средства информатики». ИПИ РАН. Вып. 16. М.: Наука, 2006. С. 72–108.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007, т. 1, №1: 14–26

СРЕДСТВА ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ ПРИЛОЖЕНИЙ*

В. Н. Захаров[1], В. А. Козмидиади[2]

Аннотация: Рассмотрены проблемы построения отказоустойчивых серверов, возникающие в связи с недетерминированностью поведения приложений. Предложена формальная модель, описывающая поведение приложения, основными объектами которой являются ресурсы и события. Предложены алгоритмы протоколирования работы приложения на резервном узле кластера, а также восстановления и продолжения его работы при отказе основного узла. При этом для клиентов сбой остается незаметным, за исключением некоторого увеличения времени обслуживания.

Ключевые слова: сервер приложений; прозрачная отказоустойчивость; процесс; ресурс; событие; контрольная точка; детерминированность

1 Введение

Средства вычислительной техники стали использоваться в областях, требующих безотказной работы систем в течение многих лет (24 часа в сутки, 365 дней в году).

К таким областям относятся, например, центры хранения и обработки данных в сетях (системы резервирования билетов, биллинговые, банковские и т.д.), массированные распределенные вычисления (GRID-вычисления) и др.

Обычно в подобных системах применяются частные решения, ориентированные, в основном, на обеспечение надежного хранения данных (например, файловые серверы, использующие для хранения RAID-контроллеры) и корректного их состояния при отказах (серверы баз данных с транзакционным выполнением запросов). Однако большинство приложений не гарантируют, что не произойдет потери части данных при отказе системы. Обычно предполагается, что клиентские средства должны повторять запросы после восстановления серверов, для того чтобы данные не были потеряны, или что можно сделать возврат по времени на некоторое время назад и повторить работу с этого места. Однако далеко не все клиентские средства и условия применения приложений допускают это.

Отказоустойчивые системы для критически важных приложений, корректно решающие проблемы восстановления после сбоев, предлагаемые ведущими производителями, как правило, дороги. Кроме того, они включают специфические серверные и клиентские приложения, не совместимые со стандартными приложениями, не обеспечивающими отказоустойчивость. Примером такого подхода к решению проблемы отказоустойчивости хранения данных являются системы NetApp FAS компании Network Appliance, работающие на базе специализированной операционной системы Data ONTAP [1].

Построение отказоустойчивых систем, использующих серверы со стандартными приложениями, в свете вышесказанного, является актуальной проблемой, вызывающей значительный интерес. Рассмотрение методов достижения прозрачной отказоустойчивости таких систем и является предметом статьи.

2 Основные понятия и подходы

Под ***сервером*** в данной работе понимается вычислительный центр (отдельный компьютер или кластер) в сети, предоставляющий клиентам (пользователям, клиентским компьютерам) определенные услуги, разделяя между ними свои ресурсы. Подобные серверы названы ***серверами приложений***. Широко распространенным примером сервера такого типа является файловый сервер, обеспечивающий удаленный коллективный доступ к файловой системе. Часто используются вычислительные серверы, предоставляющие клиентам возможность выполнять на них свои программы (например, в центрах коллективного пользования).

*Работа выполнена при поддержке РФФИ, гранты № 06-07-89188 и № 06-07-08072.

[1]Институт проблем информатики РАН, vzakharov@ipiran.ru

[2]Институт проблем информатики РАН, kozmidiady_v@tochka.ru

Обычно приложение представляет собой программу или группу программ, работающих в операционной среде, создаваемой операционной системой (в другой терминологии — один или несколько взаимодействующих ***процессов*** или потоков (*threads*)), которые реализуют функциональность сервера. Для построения отказоустойчивых серверов приложений широко используется кластерная технология. Следуя [2], ***кластером***, названа разновидность параллельной или распределенной системы, которая:

– состоит из нескольких компьютеров (узлов кластера), связанных как минимум необходимыми коммуникационными каналами;

– используется как единый, унифицированный компьютерный ресурс.

Прозрачная отказоустойчивость (*Transparent Fault Tolerance*, *TFT*) сервера приложений — это такое его поведение при возникновении аппаратных или программных отказов либо отказов в сети, при котором:

– отказ не вызывает потери или искажения данных, находящихся в базе данных сервера;

– сервер продолжает нормально функционировать, несмотря на имевшие место отказы.

Клиенты сервера «не замечают» произошедших отказов. Единственным допустимым отклонением сервера от нормального поведения с точки зрения клиента является некоторое увеличение времени обслуживания (на несколько секунд или десятков секунд).

Обычно приложения, работающие на серверах приложений, не ориентированы на прозрачную отказоустойчивость. Они «заботятся» лишь о собственной целостности (например, состоянии файловой системы или базы данных). Восстановление работоспособности сервера приводит к разрыву соединений с клиентами и потере их запросов. Это замечают клиенты — запросы следует повторять, на что клиентские приложения далеко не всегда рассчитаны. В данной работе предполагается, что приложения (прикладные программные средства), выполняемые на сервере, являются стандартными, то есть не имеют специальных средств, обеспечивающих отказоустойчивость.

Серьезные исследования в области обеспечения отказоустойчивости серверов были развернуты после создания вычислительных серверов, предназначенных для решения задач, требующих больших вычислительных ресурсов. Решение этих задач выполняется на суперкомпьютерах, обеспечивающих массово-параллельные вычисления и представляющих собой кластеры из сотен и тысяч узлов (процессоров). Однако даже на этих «монстрах» решение может требовать десятков или сотен часов, и одиночный сбой, если не предприняты специальные меры, может привести к необходимости начинать работу сначала. Обычно решение вычислительной задачи в таких случаях осуществляется в модели относительно редко взаимодействующих между собой процессов, выполняемых на разных узлах кластера. Эти взаимодействия нужны для координации работы процессов, в частности, для обмена данными и промежуточными результатами. Взаимодействия опираются на специальный протокол, называемый *MPI* (*Message-Passing Interface*) и представляющий собой стандарт “de facto” [3].

Для преодоления последствий сбоя достаточно давно была разработана и широко применяется технология, опирающаяся на механизм ***контрольных точек*** (*checkpoints*) [4–6]. По этой технологии система должна иметь ***стабильную память***, которая не меняется при отказах. Соответствующие программные средства периодически сохраняют информацию о состоянии процессов приложения в стабильной памяти. Все процессы также имеют доступ к ***устройству стабильной памяти***. В случае отказа или сбоя записанная в стабильную память информация используется для повторения вычисления с момента, когда была записана эта информация, то есть выполняется ***откат*** назад по времени. Данные, сохранение которых позволяет выполнить откат, называются ***контрольной точкой***. В качестве устройства стабильной памяти может использоваться дисковый том, энергонезависимая оперативная память, память другого узла или узлов кластера. В последнем случае узел, которому требуется сохранить информацию, пересылает ее через быстрый канал связи на другой узел. Стабильная память после отказа одного из узлов должна быть доступной узлу, на котором делается повтор.

Однако решение, опирающееся только на контрольные точки, не является прозрачным, поскольку не скрывает от клиентов факт отказа системы и требует от них выполнения определенных действий. Так как при работе процессы обмениваются сообщениями, возможны два варианта решения проблемы. Первый — все процессы выполняют записи контрольных точек одновременно, что затруднительно. Второй вариант, при несоблюдении синхронности, — возврат в каждом процессе к такому скоординированному набору контрольных точек, при котором невозможна противоречивая ситуация. Такая ситуация возникает, когда один процесс вернулся к контрольной точке, после которой он должен получить сообщение от другого процесса,

а этот другой процесс вернулся к точке, которая следует за выдачей этого сообщения. Однако при повторе ожидаемое первым процессом сообщение не поступит. В этом случае возможен эффект домино, в результате процессы оказываются отброшены как угодно далеко назад.

В этом состоит **первая проблема**, которую необходимо преодолеть.

Если нужно, чтобы последствия отказа узла не были видны клиенту, это означает, что клиент не должен:

- терять и потом восстанавливать соединения с сервером;
- повторять свои запросы;
- повторно получать сообщения, которые он уже получил.

Вторая проблема, которую надо решать, связана с недетерминированностью поведения сервера приложений. Приведем пример. Пусть имеется система продажи билетов на самолеты. Два клиента одновременно обратились к системе с запросом билета на один и тот же рейс. Клиентам безразлично, какие места им зарезервирует система. Система выполняет запросы клиентов параллельно, поэтому в какой-то момент между процессами, обрабатывающими эти запросы, может возникнуть конкуренция за ресурс — в данном случае, скажем, рейс. Один из процессов захватывает ресурс первым, резервирует место и освобождает ресурс. Потом второй процесс проделывает то же самое.

Порядок, в котором в этом примере процессы захватили ресурс, зависит от многих факторов и, в конечном счете, случаен. Однако это не мешает правильному функционированию системы, поскольку клиентам важно одно — получить билеты, причем на разные места. Однако отсутствие детерминизма в поведении приложения приводит к тому, что при повторном выполнении могут быть получены другие результаты: например, клиенту уже сообщено, что ему зарезервировано место № 5, а при повторе может получиться, что зарезервировано место № 6. Система должна устранить это несоответствие и сделать его невидимым для клиента.

Недетерминированность поведения системы — это следствие, по крайней мере, двух обстоятельств. Во-первых, это присущая системам с разделением времени неопределенность в порядке выполнения процессов. Во-вторых, это конкуренция процессов за общие ресурсы. Перечислим некоторые причины недетерминированного поведения приложений:

- синхронизация процессов с помощью семафоров или атомарных операций над операндами в общей памяти процессов;
- зависимость от порядка получения клиентских запросов;
- время, затраченное процессом на обработку полученного запроса;
- генераторы случайных чисел;
- системное управление процессами и потоками;
- локальные таймеры;
- доступ к реальному времени.

По различным причинам время, которое тратится на выполнение вычислительной задачи с одними и теми же исходными данными, не является константой, то есть повторное выполнение может дать другое время. Процессы используют общие ресурсы, обращение к которым требует организации очередности выполнения (сериализации) — первым пришел, первым захватил. И, наконец, результат работы процесса может зависеть от состояния ресурса, а это состояние может изменить другой процесс, ранее захвативший ресурс. Все это создает значительные трудности при попытках воспроизведения поведения процессов с сохраненной контрольной точки.

Прозрачная отказоустойчивость серверов приложений обычно осуществляется переносом приложения на другой узел кластера, идентичный первому по конфигурации аппаратных средств и операционной среды. Это делается методом, называемым *snapshot/restore*. На основном узле (оригинале) периодически фиксируется состояние приложения на этом узле кластера (так называемый ***снимок*** или *snapshot*). После отказа оригинала на резервном узле (копии) делается ***восстановление*** (*restore*), то есть восстанавливается последнее зафиксированное состояние приложения. Операционная среда при этом приводится в состояние, которое соответствует моменту изготовления снимка. После этого узел-копия продолжает работу с зафиксированного места. Сравнение метода *snapshot/restore* с другими подходами приведено в [7].

Ниже рассматриваются информационные технологии, позволяющие решить ряд принципиальных вопросов, связанных с реализацией прозрачной отказоустойчивости серверов приложений. Ими являются:

- виртуализация операционной среды, в которой работает серверное приложение;
- отказоустойчивая реализация протокола управления передачей TCP (*transmission control protocol*);

– создание контрольных точек состояния приложения и файловой системы, которые делаются внешним по отношению к приложению образом;
– восстановление серверного приложения на основании контрольной точки.

3 Модель описания поведения приложения

Предлагаемый подход опирается на построение модели вычислений, связанной с использованием понятия времени в многопроцессных приложениях. Впервые подобные проблемы были изучены в классической работе Л. Лампорта [8].

Многопроцессными приложения называются потому, что в них параллельно работают несколько процессов. Процесс ведет себя детерминированно, пока в предписанном кодом порядке выполняет процессорные инструкции. Конечно, его работа может быть прервана практически в любой момент и процессор передан другому процессу или ядру. Поэтому абсолютное время, которое затрачивает процесс на выполнение определенной работы, не константа, а случайная величина. То же относится к относительному времени, то есть времени, которое процесс занимал процессор, поскольку одни и те же обращения к операционной среде могут вызвать работы разной длительности, а значит потребовать разное время на свое выполнение.

Кэшированность инструкций и данных, а также длина хэш-списков влияют на действительное время пребывания в операционной среде. Утрачивает смысл понятие ***одновременность действий***, поскольку нельзя установить, выполнили ли два разных процесса какие-либо действия одновременно или одно из них предшествовало другому. Таким образом, с процессом можно связать только его локальное время, которое линейно упорядочивает события, происходившие в этом процессе. Глобальное время, линейно упорядочивающее действия во всех процессах, отсутствует. Расстояние (в этом качестве используется время) между действиями оказывается случайной величиной.

Эти соображения важны, поскольку процессы в интересующих нас приложениях взаимодействуют и используют общие ресурсы. Для взаимодействия они используют средства синхронизации, предоставляемые операционной средой — например, наборы семафоров SVR4 (System V Release 4), POSIX-семафоры, бинарные семафоры и другие примитивы взаимного исключения (POSIX mutual exclusion locks) и т.д. Подобные средства операционной среды, которые позволяют процессам синхронизировать свою деятельность друг с другом или сериализовать обращения к совместно используемым объектам, будут ниже называться ***ресурсами***.

С каждым ресурсом связано свое локальное время, линейно упорядочивающее ***события*** в жизни ресурса. Например, в случае двоичных семафоров это создание семафора, а также его захват и освобождение процессом. Заметим, что событие — это не намерение процесса (например, захватить бинарный семафор), а сам факт захвата семафора процессом (то есть успешное выполнение намерения). От изъявления намерения до его осуществления может многое произойти. Например, семафор, который хочет захватить рассматриваемый процесс, принадлежал другому процессу, потом тот процесс его освободил, но семафор был сначала передан операционной средой третьему процессу, который также на него претендовал, и т.д. Поведение рассматриваемого процесса в это время нас не интересует — он ресурсом еще не овладел, а только его захват определяет его дальнейшее поведение. По причинам, изложенным выше, расстояние между двумя событиями — случайная величина. Однако события замечательны тем, что они одновременно присутствуют и в локальном времени процесса, и в локальном времени ресурса. Поэтому все, что произошло в истории процесса или/и ресурса до этого события, предшествует ему. Далее будет считаться, что истории и ресурсов, и процессов состоят только из событий, причем между двумя последовательными событиями в жизни процесса последний ведет себя детерминированно.

Это означает, что на поведении процесса сказывается только его предыдущая история, то есть состояние ресурсов, с которыми он взаимодействовал. Это свойство процессов ниже будет называться ***локальной детерминированностью***. Этим свойством не обладают ресурсы, поскольку следующее событие в истории ресурса не определяется однозначно по его предыдущей истории. Утверждение, касающееся детерминированного поведения процессов, неявно опирается на предположение, что учтены все ресурсы, которые могут привести к недетерминированности процессов.

Таким образом, описанное нами очень неформально время в многопроцессном комплексе представляет собой отношение частичного порядка, введенное на множестве событий. Зная полное состояние комплекса в некоторый момент времени, нельзя однозначно определить, какое событие в истории ресурса наступит следующим. Можно говорить только о вероятности наступления того или иного события. Недетерминированность поведения есть следствие двух обстоятельств. Во-

первых, это неопределенность времени, которое тратит процесс на переход от одного события к другому. Во-вторых, конкуренция процессов за общие ресурсы.

Выполнение приложения, на множестве событий которого введена частичная упорядоченность, можно описать направленным ациклическим графом выполнения. Вершинами этого графа являются события, с каждым из которых связаны две входящие в него дуги. Одна дуга начинается в событии, которое непосредственно предшествует данному событию в истории процесса, другая — в истории ресурса.

Построение средств обеспечения прозрачной отказоустойчивости приложений опирается на следующее утверждение:

Для восстановления работы приложения после отказа достаточно располагать:

- *контрольной точкой, которая отражает на некоторый момент времени состояния процессов и других ресурсов, образующих приложение;*
- *графом выполнения приложения, который описывает работу приложения, начинающуюся с контрольной точки и заканчивающуюся отказом.*

Данные, которые нужны для построения графа выполнения, далее называются протоколом. Вся эта информация должна находиться в стабильной памяти, не разрушающейся при отказе.

Ниже неформально описан алгоритм восстановления работы приложения после отказа, который опирается на наличие контрольной точки и графа выполнения. Будем считать, что в распоряжении имеются средства, позволяющие остановить процесс в тот момент, когда он **намерен** совершить некоторую операцию над ресурсом. Заметим, что событие в графе выполнения соответствует не изъявлению намерения, а его удовлетворению, то есть завершению выполнения операции.

Предварительно сделаем следующее:

- используя контрольную точку, приведем приложение в состояние, соответствующее этой контрольной точке;
- в графе выполнения пометим все вершины (события) как «не наступившие». У некоторых вершин графа отсутствуют им непосредственно предшествующие; соответствующие события наступили сразу же после создания контрольной точки. Для каждой такой вершины включим в граф дополнительную вершину, ей предшествующую в истории процесса, и отметим эту дополнительную вершину как «наступившую»;
- разрешим процессам приложения выполняться.

Пусть некоторый процесс проявляет намерение выполнить операцию над каким-либо ресурсом. Отыщем для этого процесса в его истории последнее наступившее событие. Следующее в его истории событие — это то, которое соответствует требуемой операции. Посмотрим, наступило ли событие в истории ресурса, которое ему предшествует. Если нет, переведем процесс в состояние ожидания, отметив в предшествующем событии, что данный процесс ожидает его наступления. Если да, разрешим процессу выполняться, то есть выполнить операцию над ресурсом.

Пусть некоторый процесс объявляет, что он выполнил операцию над каким-либо ресурсом (это соответствует моменту протоколирования при оригинальном выполнении). Отыщем для этого процесса в его истории последнее наступившее событие и перейдем к следующему событию в его истории. Это опять то событие, которое мы рассматриваем. Отметим его как «наступившее». Если наступления этого события ожидал какой-нибудь процесс, выведем этот процесс из состояния ожидания. Наконец, разрешим процессу, выполнившему операцию, продолжаться дальше.

Когда выясняется, что наступили все события графа выполнения, повторное выполнение считается законченным.

Естественным следствием из сказанного является следующее утверждение:

Для того, чтобы размер протокола не рос неограниченно, нужно периодически создавать контрольные точки, очищая при этом протокол.

4 Формальное описание модели поведения многопроцессного приложения

Опишем формально поведение приложения, неформальное описание которого было приведено выше. Рассматриваются два типа объектов:

- ***ресурсы*** (r), например наборы семафоров (POSIX- или SVR4-семафоры), бинарные семафоры (POSIX-mutex's), таймер реального времени, сокеты (*sockets*), то есть двусторонние виртуальные соединения с внешним миром;
- ***процессы*** (p), например процессы или потоки (*threads*) пользователя.

Процесс может выполнять над ресурсом ***операции***. Примерами операций являются: захват

свободного бинарного семафора или его освобождение; изменение значений семафоров в наборе; получение запроса от клиента или посылка ему сообщения. Предполагается, что два процесса не могут одновременно производить операции над одним и тем же ресурсом, а могут это делать только последовательно. Операция процесса p над ресурсом r называется ***событием*** и обозначается как $\{p, r, m, n\}$ (назначение переменных m и n будет объяснено ниже).

С каждым процессом p связывается его история H_p — последовательность событий, в которых он участвовал, то есть его операций над ресурсами. В обозначении события m — это номер события в истории H_p, $m > 0$.

С каждым ресурсом r связывается его история H_r — последовательность событий, в которых он участвовал, то есть операций, которые над ним производили процессы. В обозначении события n — это номер события в истории H_r, $n > 0$.

В начало истории каждого процесса p включим ***фиктивное событие*** $\{p, \varnothing, 0, 0\}$, а в начало истории каждого ресурса r — фиктивное событие $\{\varnothing, r, 0, 0\}$ ($\varnothing$ — отсутствующий объект).

Если один и тот же процесс p выполнил две операции над одним и тем же ресурсом r (события $\{p, r, m_1, n_1\}$ и $\{p, r, m_2, n_2\}$), причем если $m_1 < m_2$, то $n_1 < n_2$. Верно и обратное, то есть если $n_1 < n_2$, то $m_1 < m_2$.

Событие $\{p_1, r_1, m_1, n_1\}$ непосредственно предшествует ($\ll_p$ ($\ll_r$)) в истории H_p (H_r) событию $\{p_2, r_2, m_2, n_2\}$, если $p_1 = p_2$ и $m_1 < m_2$ ($r_1 = r_2$ и $n_1 < n_2$), причем в истории H_p (H_r) нет события $\{p_3, r_3, m_3, n_3\}$ такого, что $p_1 = p_3$ и $m_1 < m_3 < m_2$ ($r_1 = r_3$ и $n_1 < n_3 < n_2$).

Транзитивное замыкание отношения $\ll_p$ ($\ll_r$) дает отношение предшествования в истории H_p (H_r), которое обозначим как $<_p$ ($<_r$). Для этих отношений должны выполняться следующие аксиомы, которые их связывают:

$$\{p, r, m_1, n_1\} <_p \{p, r, m_2, n_2\} \equiv \equiv \{p, r, m_1, n_1\} <_r \{p, r, m_2, n_2\};$$

$$\{p_1, r_1, m_1, n_1\} <_p \{p_1, r_2, m_2, n_2\} <_r <_r \{p_2, r_2, m_3, n_3\} <_p \{p_2, r_1, m_4, n_4\} \supset \supset \{p_1, r_1, m_1, n_1\} <_r \{p_2, r_1, m_4, n_4\};$$

$$\{p_1, r_1, m_1, n_1\} <_r \{p_2, r_1, m_2, n_2\} <_p <_p \{p_2, r_2, m_3, n_3\} <_r \{p_1, r_2, m_4, n_4\} \supset \supset \{p_1, r_1, m_1, n_1\} <_p \{p_1, r_2, m_4, n_4\}.$$

Введем отношение ***непосредственного предшествования событий***, $\Rightarrow$, следующим образом (в обозначениях событий переменные m и n для краткости опущены):

$$\{p_1, r_1\} \Rightarrow \{p_2, r_2\} \stackrel{\text{def}}{=} \stackrel{\text{def}}{=} ((p_1 = p_2)\&(\{p_1, r_1\} \ll_p \{p_2, r_2\})) \vee \vee ((r_1 = r_2)\&(\{p_1, r_1\} \ll_r \{p_2, r_2\})).$$

Будем рассматривать события как вершины направленного графа G, причем от вершины V_1 ведет дуга к вершине V_2, если и только если $V_1 \Rightarrow V_2$. Транзитивное замыкание отношения непосредственного предшествования дает ***отношение предшествования событий***, которое мы обозначим через $\rightarrow$. Вершина V_1 предшествует вершине V_2, если в графе существует путь, ведущий от V_1 к V_2. Граф G называется ***графом выполнения приложения***. Добавим теперь две аксиомы, которые позволяют доказать, что граф G не может иметь циклов, то есть является направленным ациклическим графом:

$$(\{p, r_i, m_1, n_1\} \rightarrow \{p, r_j, m_2, n_2\}) \supset (m_1 < m_2),$$
$$(\{p_i, r, m_1, n_1\} \rightarrow \{p_j, r, m_2, n_2\}) \supset (n_1 < n_2).$$

Подграф G' графа G называется ***правильным***, если вместе с любой вершиной V подграфа G' он содержит также все вершины V' графа G такие, что $V' \rightarrow V$ в графе G. Отношение $\rightarrow$ описывает частично упорядоченное время в многопроцессорном узле, процессы которого ведут себя недетерминировано. Укажем на имеющий место ***дуализм*** процессов и ресурсов: все аксиомы остаются верными, если всюду под процессами иметь в виду ресурсы, и наоборот.

Вершины графа, соответствующие событиям вида $\{\varnothing, r\}$, называются входными вершинами или описателями ресурсов. Входные вершины не имеют предшествующих им вершин.

Два выполнения одного и того же приложения называются ***одинаковыми***, если их графы совпадают.

В отношении приложений предполагается, что они используют только штатные средства операционной среды, в которой работают. В частности, считается, что они не требуют внесения изменений в код операционной системы и не применяют собственные примитивы синхронизации и блокировки, которые могут быть построены, например, с помощью ассемблерных программ, применяющих атомарные машинные инструкции.

Для того чтобы имелась возможность восстановить работу приложения после отказа, необходимо заносить в стабильную память:

- контрольные точки для процессов, которые образуют приложение, работающее на основном узле. Создание контрольных точек нужно делать периодически;

– информацию о получаемых от клиентов запросах и других событиях, влияющих на детерминированность поведения приложения.

Это занесение, называемое далее протоколированием, нужно делать динамически, по мере необходимости. Весь набор подобных записей называется ***протоколом***.

Протокол, находящийся в стабильной памяти, должен быть достаточным для построения графа выполнения приложения после последней контрольной точки. Если это условие выполняется, можно утверждать следующее:

1. На основании последней контрольной точки перед отказом и графа выполнения приложения от этой точки можно так организовать повторение выполнения приложения, что граф этого повторного выполнения будет совпадать с исходным графом.
2. Повторное выполнение приложения не может привести к тупиковым ситуациям, когда выполняемые процессы блокируют друг друга.

Алгоритмы ведения протоколов и повторного выполнения приложения рассмотрены также в [9].

5 Аппаратная и операционная платформы сервера приложений

Практическая реализация технологии прозрачной отказоустойчивости основана на том, что аппаратная платформа сервера приложений должна представлять собой многоузловой кластер. В нем все узлы, кроме одного, являются основными, а один — резервным. Оперативная память резервного узла выступает в роли стабильной памяти. На основных узлах происходит оригинальное выполнение приложения.

В случае отказа одного из основных узлов на резервном узле происходит восстановление состояния приложения, выполнявшегося на отказавшем узле, после чего он берет на себя функции основного узла. Оперативная память резервного узла может выступать в роли стабильной памяти, поскольку она:

– сохраняет свое содержимое при отказе основного узла, на котором происходило оригинальное исполнение;
– доступна при восстановлении, которое делается на резервном узле.

Реализация предлагаемых методов подразумевает внесение некоторых изменений в код ядра системы, поэтому предполагается, что узлы кластера работают под управлением UNIX-подобной системы с открытым кодом.

6 Состав приложения

В начале работы приложения для него создается исходный процесс, например, с помощью функции *execve()*. Далее приложение во время работы может создавать (открывать) новые ресурсы (процессы и потоки, файлы, трубы (*pipes*), наборы семафоров, бинарные семафоры (*mutex's*) и т.д.) и закрывать (уничтожать) их. В каждый момент работы приложения оно состоит из множества открытых ресурсов, которое будет называться ***составом приложения*** в данный момент. В начале жизни приложения в его состав входит один единственный ресурс — исходный процесс. Свойство ресурса «принадлежать составу приложения» является наследуемым, то есть каждый новый ресурс, созданный процессами или потоками, уже входящими в состав приложения, включается в этот состав, а уничтоженный — исключается из него. Заметим, что создаваемым ресурсом может быть новый процесс или поток. Предполагаем, что приложение не может использовать ресурсы, не входящие в его состав.

Для упрощения изложения будем считать, что на сервере приложений функционирует только одно приложение, для которого должна быть обеспечена отказоустойчивость. Оно называется ***контролируемым***. Помимо процессов и потоков контролируемого приложения на кластере могут выполняться другие процессы, но они не должны влиять на работу контролируемого приложения.

7 Функциональные компоненты реализации прозрачной отказоустойчивости

В разработанной реализации восстановление состояния приложения после отказа и продолжение его работы реализуется компонентами, выполняющими следующие функции:

– перехват обращений к ресурсам;
– определение, входит ли ресурс в состав контролируемого приложения;
– виртуализация операционной среды (VIRT-компонент);

- создание контрольных точек состояния состава приложения и восстановление на основании сохраненной контрольной точки аналогичного состояния на другом экземпляре операционной среды, включая состояния всех ресурсов, файловой системы и т.д. (SNAPSHOT/RESTORE-компонент);
- отказоустойчивая реализация протокола TCP (FT-TCP-компонент);
- протоколирование работы приложения (LOGGING-компонент);
- повторение работы приложения, начинающееся с последней контрольной точки (REPLAY-компонент);
- драйвер стабильной памяти.

Перечисленные компоненты можно разделить на две группы:

1. Компоненты, реализация которых в той или иной степени известна из литературы. Это VIRT-компонент [10, 11], SNAPSHOT/ RESTORE-компонент [4–6], драйвер стабильной памяти [12]. В ряде случаев к ним предъявляются дополнительные требования. Во-первых, необходимо минимизировать эти требования, и, во-вторых, требуется их точно сформулировать.
2. Новые компоненты. Кроме обеспечения их функциональности основное требование к таким компонентам — уменьшение накладных расходов при оригинальном выполнении приложения. Поэтому рекомендуется минимальная функциональность, реализуемая на основном узле, даже если это приводит к непропорциональному увеличению загрузки резервного узла или увеличению времени восстановления после отказа.

8 Организация перехвата обращений к ресурсам и определение принадлежности к приложению

Перехват обращений к ресурсам необходим, прежде всего, для того чтобы осуществить протоколирование. Перехват нужен также в случае отказа основного узла для повторения работы приложения, начиная с последней контрольной точки. Способ перехвата зависит от типа ресурса и может осуществляться как в пользовательском режиме, так и в режиме ядра. При выборе режима следует учитывать два обстоятельства:

(1) системную реализацию обращений к ресурсу. Если при таком обращении всегда делается вызов модуля ядра, перехват обычно можно делать в любом режиме. Так, например, обстоит дело с процессами. Если же обращения при некоторых условиях выполняются без вызова ядра, перехват необходимо делать в пользовательском режиме;

(2) необходимость блокировки занесения записей в протокол. Например, если один процесс выполняет некоторую операцию над набором семафоров, может оказаться, что после этого другой процесс, который находился в ожидании, теперь может выполнить свою операцию над набором, которую до этого он выполнить не мог. Таким образом, в протоколе должны быть отмечены два события, причем первое предшествует второму. Возникают трудности, если этот порядок в протоколе не соблюдается. Чтобы этого избежать, требуются специальные средства блокировки, однако в ядре они могут присутствовать, хотя и для других целей. Именно так обстоит дело с наборами семафоров, поэтому перехват для них делается в режиме ядра.

Для реализации перехватов в ядре необходимо внести изменения в код ядра. Например, чтобы организовать перехват обращений к наборам семафоров, нужно внести изменения в модуль ядра, который ответствен за обслуживание семафоров. Перехват обращений в пользовательском режиме не требует внесения изменений в библиотечные функции, а делается с помощью определенной «надстройки» над ними. О структуре программ перехвата, которые содержат функции обращения к LOGGING-компоненту и REPLAY-компоненту, будет сказано далее. Здесь отметим, что перехват может выполняться как при обращении к ресурсу для выполнения операции, так и при обращении к нему с подтверждением, что намерение удовлетворено, то есть операция выполнена.

Перехватив обращение к ресурсу, следует определить, входит ли он в состав приложения, поскольку если нет, протоколировать такое обращение не нужно. Аналогично и при повторном выполнении. Компонент определения основан на том, что свойство принадлежности является ***наследуемым***. Относительно исходного процесса приложения его вхождение в состав должно быть известно. Все ресурсы, созданные процессами, входящими в состав приложения, считаются также входящими в этот состав. Учитывая, что перехват обращений к ресурсам всегда выполняется в контексте обращающегося процесса, при создании новых процессов

(потоков) в их контексте можно отметить, что новый процесс или поток также входит в состав приложения. Если происходит обращение к ресурсу, отличному от создания или уничтожения процесса (потока), то достаточно проверить, входит ли обращающийся процесс (поток) в состав приложения. Если это так, он может обращаться только к ресурсам, входящим в этот состав.

9 Создание контрольных точек

Контрольная точка (или снимок) представляет данные, необходимые для восстановления состояния всех процессов и ресурсов, входящих в состав приложения, а также состояния операционной среды. Существенно, что контрольная точка создается в тот момент, когда в истории каждого из процессов уже наступило некоторое событие, но еще не наступило следующее событие. Фактически это означает, что контрольная точка — это событие в истории всех процессов, то есть такая вершина в графе выполнения, которой непосредственно предшествуют события в историях всех процессов.

Контрольная точка включает для каждого процесса следующие элементы (это далеко не полный список):

- его псевдоним (описание см. ниже);
- его состояние с точки зрения операционной среды;
- содержимое его адресного пространства;
- характеристики всех его локальных ресурсов, которые открыты на момент создания контрольной точки. В частности, это могут быть потоки процесса, для каждого из которых требуется псевдоним, содержимое его стека, аппаратных регистров, указателя текущей инструкции, состояние с точки зрения операционной среды и т.д.;
- состояние открытых сокетов.

Контрольная точка для всего приложения в целом должна включать следующие элементы (список также далек от полноты):

- псевдонимы и содержимое сегментов общей (межпроцессной) памяти;
- состояние файловых систем целиком и отдельных файлов;
- состояние наборов семафоров и их псевдонимы;
- состояние межпроцессных труб.

При каждом создании контрольной точки она заносится в стабильную память, замещая предыдущую контрольную точку. При восстановлении выполняется обратная работа, то есть на основании последней созданной контрольной точки воссоздаются процессы и их потоки, а также все их ресурсы в том состоянии, в котором они находились в момент создания этой точки. В частности, это касается состояния файловых систем и отдельных файлов, для чего применяется специальная технология клонов.

10 Протоколирование работы приложения между контрольными точками

При восстановлении приложения в другом экземпляре среды неизбежно повторение работы приложения, начинающееся с последней контрольной точки, достигнутой при работе приложения в оригинальной среде. Это приводит к необходимости протоколировать при оригинальном выполнении приложения факты обращений к некоторым ресурсам, то есть хранить историю ресурсов, которую после размещения в стабильной памяти можно называть ***журналом***. Журнал — это именованный список последовательных записей событий. Порядок записей в списке соответствует порядку наступления событий в истории ресурса. Имена списков находятся во взаимно однозначном соответствии с именами ресурсов.

Перехват обращения к ресурсу делается непосредственно после выполнения операции над ресурсом, и в протокол заносятся сведения о факте выполнения операции. Протоколирование начинается перед началом работы приложения, то есть перед тем, как уже существующий процесс превращается в исходный процесс приложения. Такое протоколирование и есть основное назначение LOGGING-компонента. Поскольку он работает в контексте процессов приложения при оригинальном выполнении, эффективность его программной реализации очень важна. Время выполнения отдельного запроса клиента линейно зависит от времени работы LOGGING-компонента, связанного с обслуживанием запроса.

Далее предполагается атомарность занесения записей в протокол, которая означает выполнение следующих четырех предположений:

Предположение 1: *Занесение происходит* ***синхронно*** *с выполнением процесса. Например, это может означать, что драйвер стабильной памяти всегда работает в режиме сквозной записи (write-through);*

Предположение 2: *Занесение записи, соответствующей некоторому событию, происходит* ***своевременно****, то есть до того, как для этого же ресурса может наступить следующее событие;*

Предположение 3: *Занесение происходит* ***неделимо****, то есть если в момент занесения происходит отказ, в протоколе запись либо будет присутствовать, либо нет, промежуточного состояния протокола с искаженной записью быть не может;*

Предположение 4: *Занесение записи по некоторому событию происходит* ***одновременно*** *с событием, то есть если происходит отказ, то возможно лишь одно из двух: событие наступило, и о нем есть запись в протоколе; либо событие не наступило, и о нем нет записи в протоколе.*

Предположение 1 (о синхронности) может привести к значительному увеличению времени ответа на запросы. Если в качестве стабильной используется память резервного узла, синхронность означает, что процесс, который затребовал занесения записи в протокол, приостанавливается до тех пор, пока от резервного узла не будет получено подтверждение приема. Поэтому время реакции системы на запрос клиента увеличивается на некоторую величину. Эта величина равна промежутку времени от момента запроса на занесение записи в протокол до момента получения подтверждения. Известен способ, позволяющий ослабить это предположение.

Предположение 2 (о своевременности). Наступление события, например, увеличение значения семафора некоторым процессом, может привести к тому, что в другом процессе становится возможным наступление другого события (уменьшение значения семафора). Своевременность гарантирует, что в протоколе первое событие (увеличение значения) будет предшествовать второму событию (уменьшение значения).

Предположение 3 (о неделимости) — это требование к драйверу стабильной памяти, реализация его не представляет особых трудностей.

Предположение 4 (об одновременности) является очень жестким, однако отказаться от него нелегко, а решение может зависеть от типа ресурса.

Возможно, что определенный выше протокол избыточен, и в нем зафиксированы лишние события. Уменьшение количества фиксируемых событий может опираться на знание причинно-следственных отношений между событиями, а такое знание отсутствует.

Следствием принятых предположений является следующее утверждение.

Теорема 1. *По журналам, находящимся в стабильной памяти, однозначно восстанавливается граф, являющийся правильным подграфом графа выполнения приложения, начиная с последнего сделанного снимка.*

11 Повторение работы приложения

Если на основном узле происходит отказ, на резервном узле выполняется восстановление состояния приложения на момент создания контрольной точки. Однако на основном узле после создания контрольной точки приложение еще некоторое время работало, и эта работа зафиксирована в протоколе. Основное назначение REPLAY-компонента — повторить выполнение приложения, начиная с момента создания последней контрольной точки и до момента возникновения отказа. Такое повторение преследует две цели:

(1) избежать недетерминированности при повторении работы, так как если поведение приложения при повторении отличается от поведения его же при работе на оригинале, это может привести к различным результатам, доступным внешнему миру;

(2) компенсировать то обстоятельство, что при повторении приложение не может получить от внешнего мира сообщения, которые были получены при оригинальном выполнении до возникновения отказа. Аналогично, во внешний мир не могут быть посланы сообщения, которые уже были посланы при оригинальном выполнении до момента возникновения отказа.

REPLAY-компонент, располагая протоколом выполнения приложения после создания последней контрольной точки, сначала строит граф выполнения приложения. Это выполняется в режиме ядра. Далее начинается повторение. Оно опирается на перехват обращений к ресурсам, причем в отличие от LOGGING-компонента нужен перехват как непосредственно **до**, так и непосредственно **после** выполнения операции над ресурсом. Получив управление до выполнения операции, REPLAY-компонент проверяет, действительно ли в соответствии с графом выполнения в истории ресурса должно наступить соответствующее событие. Если это не так, процесс переводится в состояние ожидания и пребывает в нем, пока это условие не будет выполнено. Если же это так, процессу разрешается выполнить операцию над ресурсом. Получив управление после операции, REPLAY-компонент отмечает в графе выполнения, что событие наступило, и проверяет, не ожидает ли какой-нибудь процесс этого события. Компонент отчасти рабо-

тает в режиме ядра, отчасти — в пользовательском режиме.

Помимо описанных выше основных функций, которые выполняются после отказа, REPLAY-компонент выполняет и некоторые вспомогательные функции.

Более формально алгоритм восстановления работы приложения после отказа можно описать следующим образом.

Свяжем с журналом каждого ресурса r натуральное число $n(r)$, равное номеру первой, еще не обработанной записи в журнале. Пусть $n(r) = 0$, если журнал пуст или если обработаны все его записи (записи в журнале нумеруются, начиная с 1). Пусть n таково, что $0 < n \leq N_r$ (N_r — число записей в журнале ресурса r); определим функцию $p_r(n)$ равной идентификатору процесса из n-й записи журнала.

Используя снимок (контрольную точку), приведем приложение в состояние, соответствующее этому снимку, после чего разрешим процессам приложения выполняться. Когда процесс p проявляет намерение выполнить операцию над ресурсом r, проверяется условие $(n(r) > 0)\&(p_r(n(r)) = p)$. Если это условие не выполняется, процесс переводится в состояние ожидания. Если выполняется, процессу дается возможность выполнить операцию над ресурсом. Для выполнения операции могут потребоваться специфические данные (например, реальное время, полученное на узле-оригинале). Эти данные берутся из записи журнала. После того как процесс p выполнил операцию над ресурсом r, следующим образом изменяется $n(r)$:

$$n(r) = (n(r) + 1)(\mathrm{mod}\ N_r + 1)\,.$$

Затем просматриваются все процессы, ранее переведенные в состояние ожидания. Если среди них оказывается процесс, для которого выполняется условие $(n(r) > 0)\&(p_r(n(r)) = p)$, но уже с новым значением $n(r)$, процесс выводится из ожидания и ему предоставляется возможность выполнить желаемую операцию. Если оказывается, что для всех r $n(r) = 0$, все процессы, переведенные в ожидание, выводятся из этого состояния. Восстановление работоспособного состояния приложения после этого считается завершенным.

Сформулируем два утверждения, касающиеся правильности описанного алгоритма.

Теорема 2. *Выполнение приложения на основном узле после изготовления последнего снимка, описанное журналами, находящимися в стабильной памяти, одинаково с выполнением приложения в соответствии с приведенным выше алгоритмом.*

Доказательство этого утверждения опирается на Теорему 1 и делается индукцией по суммарной длине всех журналов, находящихся в стабильной памяти.

Теорема 3. *Выполнение приложения в соответствии с приведенным выше алгоритмом не может привести к тупиковым ситуациям.*

Справедливость этого утверждения есть прямое следствие подразумеваемой корректности приложения. Это означает, что независимо от порядка поступления клиентских запросов и времени, затрачиваемого процессами для перехода от одного события к другому, приложение не должно зацикливаться или попадать в тупиковую ситуацию.

12 Некоторые вопросы программной реализации

Протоколируемые события

Набор протоколируемых событий и содержимое записей о каждом событии должны быть достаточными:

- для построения графа выполнения приложения;
- для идентичного оригинальному повторного выполнения, начинающегося с последней контрольной точки.

Классы учитываемых ресурсов

Учитываемые ресурсы — это те ресурсы, работа с которыми может повлиять на детерминированность поведения приложения, а потому обращения к ним следует протоколировать. Для многих приложений **классы ресурсов**, которые нужно учитывать, исчерпываются следующим списком:

- процессы (PROCESS);
- потоки (POSIX_THREAD);
- бинарные семафоры (POSIX_MUTEX);
- блокировки чтение–запись (POSIX_RWLOCK);
- межпроцессные и внутрипроцессные семафоры (POSIX_SEM);
- наборы семафоров (SVR4_SEM);
- сокеты (SOCKET).

Кроме перечисленных классов в список включается также класс ***административных*** ресурсов, который используется для служебных целей.

Типы протоколируемых событий

Операции над любыми ресурсами можно разделить на четыре группы — создание (открытие), чтение, изменение, уничтожение (закрытие). В составе каждой из групп, в зависимости от класса ресурса, может быть нуль или несколько операций. Однако для построения графа выполнения важно деление операций на три **типа**:

(1) создание (открытие) ресурса — CREATE;

(2) уничтожение (закрытие) ресурса — DESTROY;

(3) прочие операции — OTHER.

Для повторного выполнения не требуется дальнейшая детализация операций типа OTHER, однако включение конкретных операций в этот тип (то есть необходимость их протоколирования) зависит от класса ресурса. Для ресурсов некоторых классов могут отсутствовать операции типа CREATE и DESTROY. Примером могут служить бинарные семафоры.

Запись протокола и ее состав

Запись протокола — описание одного события в историях процесса и ресурса. Это событие заключается в том, что процесс выполнил некоторую операцию над ресурсом. Удобно, чтобы длина записи была кратна 8. Запись заносится в протокол и содержит:

(1) класс ресурса;

(2) псевдоним или системный идентификатор конкретного ресурса, над которым выполнена операция. Выбор зависит от того, в каком режиме происходит протоколирование;

(3) тип операции;

(4) псевдоним или системный идентификатор процесса, выполнившего операцию;

(5) системный идентификатор процесса;

(6) признак, указывающий на необходимость выполнить синхронизацию протокола;

(7) необязательные дополнительные данные и их длина, например это могут быть данные, полученные из сокета.

Протоколирование создания процессов и потоков осуществляется в пользовательском режиме. Поэтому при создании процесса в поле 4 записи вносится псевдоним процесса-родителя, в поле 2 — псевдоним созданного процесса, а в поле 5 — системный идентификатор этого процесса. Во всех остальных случаях (для других ресурсов или других событий) поле 5 не используется. При обращениях к любым ресурсам, отличающимся от процессов, если протоколирование происходит в пользовательском режиме, в поле 4 вносится псевдоним процесса, выполнившего операцию, а в поле 2 — псевдоним ресурса, над которым выполнена операция. При протоколировании в режиме ядра в поле 4 вносится системный идентификатор процесса, а в поле 2 — системный идентификатор ресурса.

Поле 5 необходимо, поскольку протоколирование обращений к ресурсам разных классов может происходить в разных режимах. При этом в записях могут появляться как псевдонимы, так и системные идентификаторы одного и того же процесса. Поэтому, чтобы построить историю процесса в графе выполнения, необходимо знать их соответствие.

Протоколирование

При реализации протоколирования необходимо учитывать следующие обстоятельства:

- занесение записей в протокол может выполняться и в пользовательском режиме, и в режиме ядра;
- должна быть предусмотрена возможность занесения в протокол записей и фиксированной, и переменной длины. Для перечисленных классов ресурсов только обращение к сокетам требует использования записей переменной длины, поскольку, например, запись о чтении из сокета должна сопровождаться прочитанными данными. Однако частота протоколирования подобных записей обычно на несколько порядков меньше частоты записей фиксированной длины;
- размер записей фиксированной длины достаточно мал ($\sim$ 32 байта), поэтому для лучшего использования канала, связывающего основной и резервный узлы, необходимо объединение записей во ***фреймы***, то есть блоки, длина которых близка к стандартной, которую использует этот канал;
- должна быть возможность синхронизации протокола. Это средство, позволяющее приостановить процесс приложения до тех пор, пока от резервного узла не будет получено уведомление, что указанная запись ему доставлена;
- необходимо обеспечить минимальные накладные расходы при протоколировании, поскольку оно выполняется в контексте процесса приложения и от этих расходов линейно зависит время выполнения запроса клиента.

13 Заключение

Рассмотрены некоторые вопросы реализации прозрачной отказоустойчивости серверов приложений. В отличие от наиболее часто применяемых методов использования специальных программных систем сервером и приложениями предлагаемый подход позволяет работать со стандартными серверными и клиентскими приложениями. Предлагаемая технология связана с внесением некоторых изменений в состав ядра операционной системы. Поэтому эта технология может быть реализована либо в кластерах, работающих под UNIX-подобными системами с открытым кодом, либо самими организациями-разработчиками используемых операционных систем.

Основная идея предлагаемой реализации прозрачной отказоустойчивости приложений состоит в совместном использовании технологий виртуализации операционной среды, протоколирования событий и создания контрольных точек, восстановления работы серверного приложения после отказа на основании последней контрольной точки и протокола. Каждая из этих технологий не является новой, они применялись в различных системах, однако пока неизвестны прозрачные отказоустойчивые системы, построенные на сумме этих технологий и обеспечивающие безотказную работу стандартных серверных приложений.

При реализации предлагаемых методов важным аспектом является минимизация накладных расходов, связанных с дополнительными вычислительными работами, необходимыми для организации протоколирования работы в стабильной памяти. Проведенные исследования показали причины основных накладных расходов и позволили конкретизировать направление исследований с целью их снижения. По-видимому, накладные расходы удастся довести до 30% или немного меньшей величины. Такую потерю производительности можно рассматривать как приемлемую плату за реализацию важной функциональности по обеспечению прозрачной отказоустойчивости сервера приложений. Целью продолжающейся работы является разработка эффективных алгоритмов и их программная реализация, позволяющая достичь именно таких параметров.

Литература

1. http://www.netapp.com/products/filer/index.html (сайт фирмы Network Appliance Inc.).
2. *Pfister G.* In search of clusters: The coming battle in lowly parallel computing. N.Y.: Prentice Hall, 1995.
3. MPI-2: Extensions to the Message-Passing Interface. http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html.
4. *Zhong H., Nieh J.* CRAK: Linux checkpoint/restart as a kernel module. Technical Report CUCS-014-01. Department of Computer Science, Columbia University, 2001.
5. *Roman E.* A survey of checkpoint/restart implementations. Berkeley: Lawrence Berkeley National Laboratory Technical Report (publication LBNL-54942), 2003.
6. *Sankaran S., Squyres J. M., Barrett B., Lumsdaine A., Duell J., Hargrove P., Roman E.* The LAM/MPI checkpoint/restart framework: System-initiated checkpointing. LACSI Symposium, Sante Fe, New Mexico, USA, 2003.
7. *Захаров В. Н., Козмидиади В. А.* Реализация отказоустойчивости серверов приложений // Системы высокой доступности, 2006. Т. 2. № 2. С. 56–62.
8. *Lamport L.* Time, clocks, and the ordering of events in a distributed system // Communications of the ACM, 1978. V. 21. № 7. P. 558–565.
9. *Захаров В. Н., Козмидиади В. А., Кузьмин А. В.* Описание недетерминированного поведения и проблема отказоустойчивости приложений // В кн.: Системы и средства информатики / Под. ред. И.А. Соколова. М.: Наука, 2005. Вып. 15. С. 338–358.
10. *King S. T., Dunlap G. W., Chen P. M.* Operating system support for virtual machines. 2003 Annual USENIX Technical Conference Proceedings, 2003.
11. *Захаров В. Н.* Виртуализация как информационная технология // В кн.: Системы и средства информатики: Спец. вып. Научно-методологические проблемы информатики / Под ред. К. К. Колина. М.: ИПИ РАН, 2006. С. 279–298.
12. *Maloy J.* Transparent inter process communication (TIPC). http://tipc.sourceforge.net/.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007. Т. 1. Вып. 1. С. 27–39

МНОГОЛИНЕЙНАЯ СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ С КОНЕЧНЫМ НАКОПИТЕЛЕМ И НЕНАДЕЖНЫМИ ПРИБОРАМИ*

А. В. Печинкин[1], И. А. Соколов[2], В. В. Чаплыгин[3]

Аннотация: Рассматривается многолинейная система массового обслуживания с полумарковским входящим потоком, обслуживанием фазового типа, накопителем конечной емкости и ненадежными приборами, отказывающими независимо друг от друга и от всего процесса функционирования системы. Получены математические соотношения для расчета основных стационарных показателей функционирования системы при некоторых вариантах процесса отказов–восстановлений приборов.

Ключевые слова: система массового обслуживания; ненадежные приборы

1 Описание системы

В последние несколько лет стремительное развитие и активное внедрение особого класса инфотелекоммуникационных систем, главной отличительной чертой которых является обработка потоков заданий путем их разделения и передачи на выполнение нескольким исполнителям или группам исполнителей, привлекли внимание к математическим моделям, позволяющим получать характеристики подобных систем и их отдельных узлов, учитывая особенности их функционирования. Для систем этого класса, включающего, например, системы распределенных вычислений, разнообразные Грид-системы и др., необходимо учитывать различного рода перерывы в работе, сбои и неисправности, возникающие как на этапе передачи, так и на этапе выполнения заданий отдельным устройством или кластером устройств. Поэтому важным направлением в современной теории массового обслуживания является изучение систем массового обслуживания (СМО) с ненадежными приборами.

Несмотря на то, что и ранее публиковались работы, посвященные СМО с различными видами сбоев и задержек при обслуживании заявок (см., например, [1–8]), появление новых эффективных методов расчета СМО с немарковскими входящими потоками позволяет рассчитывать характеристики этих систем, дополнительно рассматривая факторы, препятствующие их безотказному функционированию. Такая возможность обусловлена спецификой появившихся методов, поскольку учет в модели негативных воздействий на систему (например с помощью введения отрицательных заявок или дополнительных состояний прибора) хотя и приводит к усложнению математических построений, но не меняет конструкции самого метода, позволяя оставаться в русле идей, в нем заложенных. В частности, в [9] на основе результатов, полученных в работе [10], найдены соотношения, позволяющие вычислять основные стационарные показатели функционирования многолинейной СМО $SM/MSP/n/\infty$ с полумарковским входящим потоком, марковским процессом обслуживания заявок, ненадежными приборами и накопителем бесконечной емкости при различных вариантах процессов отказов и восстановлений приборов.

В настоящей работе рассматривается СМО $SM/MSP/n/r$, аналогичная исследованной в [9], но с накопителем конечной емкости. Применяемые здесь методы также основываются на результатах работы [10].

Рассмотрим многолинейную СМО с полумарковским входящим потоком заявок, распределением времени обслуживания фазового типа, накопителем конечной емкости r и работающими независимо друг от друга ненадежными приборами и опишем общие принципы ее функционирования. Поскольку для систем с емкостью накопителя, равной нулю или одной заявке, расчетные формулы несколько отличаются от общего случая, далее будем предполагать, что $r \geq 2$.

*Работа выполнена при поддержке РФФИ, гранты 05-07-90103 и 06-07-89056.

[1]Институт проблем информатики РАН, apechinkin@ipiran.ru
[2]Институт проблем информатики РАН, isokolov@ipiran.ru
[3]Институт проблем информатики РАН, vchaplygin@ipiran.ru

В системе имеется n идентичных приборов, которые обслуживают поступающие на них однотипные заявки. Будем называть прибор занятым, если на нем находится заявка, и свободным в противном случае. Каждый из n приборов может находиться либо в исправном, либо в неисправном состоянии.

Состояние прибора будем считать исправным, если на приборе находится заявка и прибор занят ее обслуживанием или если прибор свободен, готов принять заявку и немедленно начать ее обслуживание.

Состояние прибора будем считать неисправным, если на приборе находится заявка, но прибор ее не обслуживает, или если прибор свободен, но не может немедленно начать обслуживание заявки, если таковая на него поступит. Если при отказе прибора (переходе прибора из исправного состояния в неисправное) на нем находится заявка, то она остается на приборе до момента восстановления (перехода прибора из неисправного состояния в исправное) и затем, в зависимости от варианта функционирования системы, либо дообслуживается, либо обслуживается заново.

Если в некоторый момент времени на обслуживание поступает очередная заявка, но все приборы заняты, то эта заявка попадает в накопитель, становясь в очередь на обслуживание. Заявки из очереди на обслуживание выбираются в порядке их поступления в накопитель. Если поступающая заявка при поступлении застает r заявок в накопителе, то она сразу же, не обслуживаясь, покидает систему (теряется). Далее для сокращения записи положим $R = n + r$.

Далее всюду, за исключением раздела 7, будет рассматриваться экспоненциальная модель процессов отказов–восстановлений приборов, при которой отказ прибора, где обслуживается заявка, происходит с интенсивностью α, а окончание ремонта этого прибора — с интенсивностью β. В некоторых вариантах функционирования системы с интенсивностями α^* и β^* может также отказывать и восстанавливаться свободный прибор.

Полумарковский входящий поток заявок определяется полумарковским процессом с конечным множеством состояний $\{1, 2, \ldots, I\}$, $1 \le I < \infty$. В каждый момент изменения состояния полумарковского процесса в систему поступает заявка. Вероятность того, что полумарковский процесс за время меньше x перейдет из состояни i сразу в состояние j, $i, j = \overline{1, I}$, равна $A_{i,j}(x)$. Обозначим через $A(x)$ матрицу из элементов $A_{i,j}(x)$, через $A = A(\infty)$ — матрицу переходных вероятностей вложенной цепи Маркова полумарковского процесса и через $\vec{\pi}_a$ — вектор-строку стационарных вероятностей вложенной цепи Маркова. Вектор $\vec{\pi}_a$ можно определить из системы уравнений равновесия (СУР)

$$\vec{\pi}_a A = \vec{\pi}_a$$

с условием нормировки

$$\vec{\pi}_a \vec{1} = 1$$

(здесь и далее через $\vec{1}$ будем обозначать вектор-столбец из единиц, а через E — единичную матрицу, размерность и порядок которых определяются из контекста). Среднее время между поступлениями заявок в стационарном режиме функционирования системы можно записать в виде

$$\overline{a} = \vec{\pi}_a \int_0^\infty x \, dA(x) \, \vec{1} \, .$$

Будем предполагать, что вложенная цепь Маркова полумарковского процесса является неприводимой и непериодической, а $\overline{a} < \infty$. Кроме того, там, где речь пойдет о стационарном распределении по времени, будем считать, что времена генерации заявок не могут принимать только значения jt, где t — положительное число, а $j = 0, 1, \ldots$.

Распределение фазового типа (PH-распределение) времени обслуживания заявки можно трактовать следующим образом. Исправный прибор, обслуживающий заявку, может находиться на одной из J фаз обслуживания, $1 \le J < \infty$. Поступающая на исправный свободный прибор заявка с вероятностью h_i, $i = \overline{1, J}$, начинает обслуживаться с фазы i. Если в некоторый момент времени прибор обслуживает заявку на фазе i, $i = \overline{1, J}$, то за «малое» время Δ с вероятностью $h_{i,j}\Delta + o(\Delta)$, $j = \overline{1, J}$, $j \ne i$, фаза обслуживания меняется на j-ю и с вероятностью $h_i^*\Delta + o(\Delta)$, где

$$h_i^* = -\sum_{j=1}^{J} h_{i,j} \, ,$$

обслуживание заявки заканчивается и она покидает систему. Вектор-строку с координатами h_i будем обозначать через $\vec{h}$, а матрицу с элементами $h_{i,j}$ — через H. Тогда функцию распределения фазового типа времени обслуживания заявки можно записать в виде

$$H(x) = 1 - \vec{h} \, e^{Hx} \vec{1} \, .$$

Цель настоящей работы заключается в нахождении программно-реализуемых математических соотношений для расчета основных стационарных показателей функционирования описанной выше СМО с ненадежными приборами. Не оговаривая это особо, всюду в дальнейшем будем считать, что

отказы приборов происходят *независимо друг от друга*.

Будут рассмотрены следующие варианты функционирования системы при экспоненциальном процессе отказов–восстановлений приборов:

- обслуживание заявки заново, свободные приборы находятся только в исправном состоянии;
- дообслуживание заявки, свободные приборы находятся только в исправном состоянии;
- обслуживание заявки заново, свободные приборы могут отказывать, заявки поступают на все приборы, вне зависимости от того, в исправном или неисправном состоянии они находятся;
- обслуживание заявки заново, свободные приборы могут отказывать, заявки поступают только на исправные приборы.

Кроме того, в разделе 7 рассмотрен вариант системы с отказами только работающих приборов и марковским процессом отказов–восстановлений приборов.

2 Общая модель

В этом разделе мы рассмотрим общую базовую модель СМО с накопителем конечной емкости, на которой будут основываться наши дальнейшие выкладки.

Общая модель представляет собой многолинейную СМО с накопителем конечной емкости, надежными приборами, полумарковским входящим потоком (процессом генерации) заявок, описанным в предыдущем разделе, и марковским процессом обслуживания заявок, определяемым следующим образом.

Если в системе находится k заявок (далее будем говорить также, что процесс обслуживания находится на слое k), $k = \overline{0, R}$, то процесс обслуживания может находиться в одном из l_k состояний (фаз обслуживания), $l_k < \infty$. Далее, если в некоторый момент в системе находится k заявок, $k = \overline{1, R}$, и фаза обслуживания равна i, $i = \overline{1, l_k}$, то за «малое» время Δ с вероятностью $\lambda_{i,j}^{(k)}\Delta + o(\Delta)$ фаза изменится на j, $j = \overline{1, l_k}$, $j \neq i$, и все заявки будут продолжать обслуживаться, а с вероятностью $n_{i,j}^{(k)}\Delta + o(\Delta)$ фаза изменится на j, $j = \overline{1, l_{k-1}}$, но обслуживание одной из заявок закончится и она покинет систему. Матрицы из элементов $\lambda_{i,j}^{(k)}$ и $n_{i,j}^{(k)}$ будем обозначать через Λ_k и N_k, $k = \overline{1, R}$. Если же в системе отсутствуют заявки, то за «малое» время Δ с вероятностью $\lambda_{i,j}^{(0)}\Delta + o(\Delta)$ фаза изменится с i на j, $i, j = \overline{1, l_0}$, $j \neq i$, естественно, без окончания обслуживания заявки. Матрицу из элементов $\lambda_{i,j}^{(0)}$ будем обозначать через Λ_0.

Кроме того, будем предполагать, что $l_k = l$ при $k = \overline{n, R}$, матрицы $\Lambda_k = \Lambda$ совпадают при $k = \overline{n, R}$, а матрицы $N_k = N$ совпадают при $k = \overline{n+1, R}$.

Предположим, что матрица $\Lambda^* = \Lambda + N$ является неразложимой, а матрица N — ненулевой. Более того, будем предполагать, что при исходных параметрах рассматриваемой СМО введенная далее вложенная цепь Маркова будет неприводимой.

Наконец, при $k = \overline{0, n-1}$ будем предполагать, что если в момент поступления очередной заявки в системе имеется k других заявок и фаза обслуживания равна i, $i = \overline{1, l_k}$, то после поступления новой заявки фаза обслуживания с вероятностью $\omega_{i,j}^{(k)}$ изменится на j, $j = \overline{1, l_{k+1}}$. Соответственно, матрицу из элементов $\omega_{i,j}^{(k)}$ будем обозначать через Ω_k.

Поскольку описанная выше модель отличается от модели, рассмотренной в [9], только емкостью накопителя, то здесь мы остановимся лишь на тех изменениях, которые нужно внести в результаты из [9].

Инфинитезимальная матрица L и матрица $B(t)$ из вероятностей переходов за время t марковского процесса обслуживания заявок отличаются от аналогичных матриц из [9] только тем, что они получены «урезанием» последних на R-м уровне:

$$L = \begin{pmatrix}
\Lambda_0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
N_1 & \Lambda_1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
0 & N_2 & \Lambda_2 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & \Lambda_{n-1} & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & N_n & \Lambda & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & N & \Lambda & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & N & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & \Lambda & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & N & \Lambda
\end{pmatrix},$$

$$B(t) = \begin{pmatrix} B_{0,0}(t) & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ B_{1,0}(t) & B_{1,1}(t) & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ B_{2,0}(t) & B_{2,1}(t) & B_{2,2}(t) & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \dots & 0 \\ B_{n-1,0}(t) & B_{n-1,1}(t) & B_{n-1,2}(t) & \dots & B_{n-1,n-1}(t) & 0 & 0 & \dots & 0 \\ B_{n,0}(t) & B_{n,1}(t) & B_{n,2}(t) & \dots & B_{n,n-1}(t) & B_0(t) & 0 & \dots & 0 \\ B_{n+1,0}(t) & B_{n+1,1}(t) & B_{n+1,2}(t) & \dots & B_{n+1,n-1}(t) & B_1(t) & B_0(t) & \dots & 0 \\ B_{n+2,0}(t) & B_{n+2,1}(t) & B_{n+2,2}(t) & \dots & B_{n+2,n-1}(t) & B_2(t) & B_1(t) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ B_{R,0}(t) & B_{R,1}(t) & B_{R,2}(t) & \dots & B_{R,n-1}(t) & B_r(t) & B_{r-1}(t) & \dots & B_0(t) \end{pmatrix}.$$

Матрица переходных вероятностей вложенной цепи Маркова, порожденной количеством заявок и фазами процессов генерации и обслуживания заявок непосредственно после моментов поступления заявок в систему, будет иметь вид:

$$B = \begin{pmatrix} B_{1,1} & B_{1,2} & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ B_{2,1} & B_{2,2} & B_{2,3} & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \dots & 0 & 0 \\ B_{n-1,1} & B_{n-1,2} & B_{n-1,3} & \dots & B_{n-1,n} & 0 & 0 & \dots & 0 & 0 \\ B_{n,1} & B_{n,2} & B_{n,3} & \dots & B_{n,n} & B_0 & 0 & \dots & 0 & 0 \\ B_{n+1,1} & B_{n+1,2} & B_{n+1,3} & \dots & B_{n+1,n} & B_1 & B_0 & \dots & 0 & 0 \\ B_{n+2,1} & B_{n+2,2} & B_{n+2,3} & \dots & B_{n+2,n} & B_2 & B_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{R-1,1} & B_{R-1,2} & B_{R-1,3} & \dots & B_{R-1,n} & B_{r-1} & B_{r-2} & \dots & B_1 & B_0 \\ B_{R,1} & B_{R,2} & B_{R,3} & \dots & B_{R,n} & B_r & B_{r-1} & \dots & B_2 & B_1 + B_0 \end{pmatrix}.$$

В приведенных выше формулах матрицы $B_{k,s}(t)$, $B_k(t)$, B_k и $B_{k,s}$ определены теми же самыми формулами, что и в [9].

Обозначим через $\vec{p}_k^*$, $k = \overline{1,R}$, вектор-строку, координатами которой $p^*_{k,m}$, где $m = \overline{1, Il_k}$ при $k = \overline{1,n-1}$ и $m = \overline{1,Il}$ при $k = \overline{n,R}$, являются стационарные вероятности по вложенной цепи Маркова того, что в системе находится k заявок, а фазы процессов генерации и обслуживания заявок равны i и j соответственно. Здесь положено $m = (\vec{e}_i' \otimes \vec{e}_j'')\vec{u}_k$, где $\otimes$ — символ кронекерова произведения матриц, $\vec{u}_k = (1, 2, \dots, Il_k)^T$ — вектор-столбец размерности Il_k или Il, а $\vec{e}_i'$ и $\vec{e}_j''$ — вектор-строки размерностей I и l_k или l, в которых i-я и j-я координаты соответственно равны единице, а остальные нулю (это позволит нам в дальнейшем использовать кронекерово произведение матриц).

Векторы $\vec{p}_k^*$, $k = \overline{1,n}$, находятся из СУР:

$$\vec{p}_1^* = \sum_{i=1}^{R} \vec{p}_i^* B_{i,1}\,, \tag{1}$$

$$\vec{p}_k^* = \sum_{i=k-1}^{R} \vec{p}_i^* B_{i,k}\,, \quad k = \overline{2,n}\,, \tag{2}$$

$$\vec{p}_k^* = \sum_{i=k-1}^{R} \vec{p}_i^* B_{i-k+1}\,, \quad k = \overline{n+1, R-1}\,, \tag{3}$$

$$\vec{p}_R^* = \vec{p}_{R-1}^* B_0 + \vec{p}_R^*(B_0 + B_1) \tag{4}$$

с условием нормировки

$$\sum_{k=1}^{R} \vec{p}_k^* \vec{1} = 1\,. \tag{5}$$

Для решения СУР (1)–(4) с условием нормировки (5) можно воспользоваться методом, изложенным в [11], с. 22. Суть этого метода заключается в последовательном исключении состояний цепи Маркова. Как показывают многочисленные расчеты, метод позволяет проводить расчеты стационарных вероятностей состояний с очень высокой точностью.

Приведем формулы для вычисления вектор-строки $\vec{p}_k$, $k = \overline{0,R}$, координатами которой $p_{k,m}$, где $m = \overline{1, Il_k}$ при $k = \overline{0, n-1}$ и $m = \overline{1,Il}$ при $k \geq n$, являются стационарные вероятности по времени того, что в системе находится k заявок, а фазы процессов генерации и обслуживания заявок равны i

и j (здесь, как и прежде, $m = (\vec{e}_i' \otimes \vec{e}_j'')\vec{u}_k$). Эти формулы имеют следующий вид:

$$\vec{p}_0 = \frac{1}{a}\sum_{t=1}^{R}\vec{p}_t^{\,*}\int_0^\infty \left(E - A^{(d)}(x)\right)\otimes B_{t,0}(x)\,dx\,, \quad (6)$$

$$\vec{p}_k = \frac{1}{a}\sum_{t=k}^{R}\vec{p}_t^{\,*}\int_0^\infty \left(E - A^{(d)}(x)\right)\otimes B_{t,k}(x)\,dx\,, \quad k = \overline{1,n-1}\,, \quad (7)$$

$$\vec{p}_k = \frac{1}{a}\sum_{t=k}^{R}\vec{p}_t^{\,*}\int_0^\infty \left(E - A^{(d)}(x)\right)\otimes B_{t-k}(x)\,dx\,, \quad k = \overline{n,R}\,, \quad (8)$$

где $A^{(d)}(x)$ — диагональная матрица с элементами

$$A_{i,i}^{(d)}(x) = \sum_{j=1}^{I} A_{i,j}(x)$$

на главной диагонали.

Обозначим через $\vec{p}_k^{\,-}$, $k = \overline{0,R}$, вектор-строку, координатами которой $p_{k,m}^-$, где $m = \overline{1,Il_k}$ при $k = \overline{0,n-1}$ и $m = \overline{1,Il}$ при $k = \overline{n,R}$, являются стационарные вероятности того, что поступающая заявка застанет в системе k других заявок, а фазы процессов генерации и обслуживания заявок равны i и j. Здесь, как и ранее, $m = (\vec{e}_i' \otimes \vec{e}_j'')\vec{u}_k$. Векторы $\vec{p}_k^{\,-}$, $k \geq 0$, находятся из соотношений

$$\vec{p}_0^{\,-} = \sum_{t=1}^{R}\vec{p}_t^{\,*}\hat{B}_{t,0}\,, \quad (9)$$

$$\vec{p}_k^{\,-} = \sum_{t=k}^{R}\vec{p}_t^{\,*}\hat{B}_{t,k}, \qquad k = \overline{1,n-1}\,, \quad (10)$$

$$\vec{p}_k^{\,-} = \vec{p}_{k+1}^{\,*}, \qquad k = \overline{n,R-2}\,, \quad (11)$$

$$\vec{p}_k^{\,-} = \sum_{j=k}^{R}\vec{p}_j^{\,*}B_{j-k}, \qquad k = R-1, R\,, \quad (12)$$

где матрица $\hat{B}_{t,k}$ определена в [9].

Стационарное распределение $W(x)$ времени ожидания начала обслуживания заявки может быть записано в виде

$$W(x) = 1 - \frac{1}{1-\pi}\sum_{j=0}^{r-1}\left(\sum_{i=j+1}^{r}\vec{p}_{i+n-1}^{\,-}\left(\vec{1}_m\otimes I_l\right)\right)B_j(x)\vec{1}\,, \quad (13)$$

где $\pi = \vec{p}_R^{\,-}\vec{1}$ — стационарная вероятность потери заявки.

В дальнейшем, наряду с линейной нумерацией состояний процесса обслуживания, будем использовать также мультииндексную нумерацию, при которой номер состояния определяется некоторым мультииндексом или объединением мультииндексов. В частности, рациональный способ перехода от мультииндексной нумерации состояний процесса обслуживания к линейной нумерации, необходимой для программирования, для системы $SM/PH/n/r$ предложен в [10]. Там же показано, как в этом случае формируются элементы матрицы L. Теперь для вычисления стационарных показателей функционирования СМО $SM/PH/n/r$ осталось воспользоваться приведенными выше формулами (1)–(13).

Для СМО $SM/PH/n/r$ нетрудно найти также стационарное распределение $V(x)$ времени пребывания заявки в системе, определяемое формулой свертки:

$$V(x) = \int_0^x W(x-y)\,dH(y)\,, \quad (14)$$

где $H(x)$ — распределение времени пребывания заявки на приборе.

Более подробный анализ алгоритмов расчета основных и вспомогательных стационарных характеристик СМО $SM/PH/n/r$ можно найти в [10].

3 Отказы работающих приборов с обслуживанием заявки заново

Наиболее просто к общей модели из разд. 2 приводится n-линейная СМО $SM/PH/n/r$ с отказами только тех приборов, на которых обслуживаются заявки. При отказе прибора заявка ждет окончания ремонта этого прибора и затем обслуживается заново. Прибор, на котором заявка отсутствует, всегда находится в исправном состоянии.

Пусть прибор, на котором находится заявка, с интенсивностью α может отказать и перейти в неисправное состояние, прекратив обслуживание этой заявки, и с интенсивностью β может вновь перейти в исправное состояние и продолжить обслуживание заявки.

Время обслуживания заявки прибором, находящимся в исправном состоянии, имеет распределение фазового типа с параметрами матрицей H порядка J и вектор-строкой $\vec{h}$ размерности J.

Для каждого из n приборов добавим к J фазам обслуживания еще одну фазу, соответствующую нахождению прибора в неисправном состоянии. Тогда рассматриваемая система с точки зрения таких

характеристик, как распределения длины очереди и времени пребывания в системе, будет эквивалентна СМО $SM/PH/n/r$ без отказов приборов с распределением времени обслуживания фазового типа с матрицей G порядка $S = J + 1$ и вектор-строкой $\vec{g}$ размерности $S = J + 1$, определяемыми соотношениями

$$G = \begin{pmatrix} H - \alpha E & \alpha\vec{1} \\ \beta\vec{h} & -\beta \end{pmatrix}, \qquad \vec{g} = (h_1, \ldots, h_J, 0)\,.$$

Теперь для нахождения основных стационарных показателей функционирования СМО $SM/PH/n/r$ с описанным типом отказов приборов осталось воспользоваться результатами, изложенными в предыдущем разделе для системы $SM/PH/n/r$ без отказов.

4 Отказы работающих приборов с дообслуживанием заявки

Так же просто к общей модели из разд. 2 приводится n-линейная СМО, отличающаяся от описанной в предыдущем разделе только тем, что заявка не начинает обслуживаться заново на восстановленном приборе, а дообслуживается.

Определяя квадратную матрицу G порядка $S = 2J$ и вектор-строку $\vec{g}$ размерности $S = 2J$ следующим образом:

$$G = \begin{pmatrix} H - \alpha E & \alpha E \\ \beta E & -\beta E \end{pmatrix}, \quad \vec{g} = (h_1, \ldots, h_J, 0, \ldots, 0)$$

и рассматривая СМО $SM/PH/n/r$ без отказов приборов с такими матрицей G и вектор-строкой $\vec{g}$, мы снова приходим к формулам расчета основных стационарных характеристик (1)–(14), полученным для общей модели.

5 Отказы всех приборов с обслуживанием заявки заново; заявки поступают на все приборы

Следующая СМО отличается от СМО из разд. 3 лишь тем, что отказывать могут все приборы, а не только те, на которых обслуживаются заявки. Однако ее исследование уже несколько сложнее. Положим, что заявки могут поступать как на исправный, так и на неисправный прибор.

Будем предполагать, что вероятность отказа не занятого обслуживанием заявки исправного прибора за «малое» время Δ равна $\alpha^*\Delta + o(\Delta)$, а вероятность восстановления не занятого обслуживанием заявки неисправного прибора равна $\beta^*\Delta + o(\Delta)$, где α^* и β^* — соответственно интенсивности отказов и восстановлений приборов, не занятых обслуживанием заявок.

Если в системе имеется свободный исправный прибор, то поступающая в систему заявка идет на этот прибор; если же свободных исправных приборов нет, но есть свободный восстанавливаемый прибор, то заявка идет на него. Интенсивность восстановления такого прибора становится равной β (вместо β^*).

Как и прежде, время обслуживания заявки прибором, находящимся в исправном состоянии, имеет распределение фазового типа с параметрами матрицей H порядка J и вектор-строкой $\vec{h}$ размерности J.

Назовем слоем k, $k = \overline{0, R}$, множество всех состояний процесса обслуживания, в которых общее число заявок в системе равно k.

Слой k при $k = \overline{n, R}$ будет иметь вид

$$\{(i_1, \ldots, i_n)\}\,,$$

где $i_1, \ldots, i_n = \overline{1, J+1}$. Состояние $(i_1, \ldots, i_n)$ означает, что первый прибор обслуживает заявку на фазе i_1, …, n-й прибор — на фазе i_n и еще $k - n$ заявок находятся в очереди.

Слой k при $k = \overline{0, n-1}$ представляет собой множество

$$\{(i_1, \ldots, i_k; m)\}\,, \quad m = \overline{0, n-k}\,,$$

где состояние $(i_1, \ldots, i_k; m)$ означает, что первый прибор обслуживает заявку на фазе i_1, …, k-й прибор — на фазе i_k и m приборов свободны от заявок и восстанавливаются.

Матрицы Λ_k, $k = \overline{0, n-1}$, N_k, $k = \overline{0, n}$, Λ и N, порождающие инфинитезимальную матрицу L общей модели разд. 2, и матрицы Ω_s, $s = \overline{0, n-1}$, формируются точно так же, как и для аналогичной системы с бесконечным накопителем [9]. Полученные матрицы L и Ω_s, $s = \overline{0, n-1}$, позволяют по формулам (1)–(8) из разд. 2 определить стационарные вероятности состояний по вложенной цепи Маркова и по времени, а по формуле (13) — стационарную функцию распределения времени ожидания начала обслуживания заявки.

Обратимся теперь к вычислению стационарного распределения времени пребывания заявки в системе.

Обозначим через $G_1(x)$ и $G_2(x)$ распределения фазового типа с одинаковой матрицей

$$G = \begin{pmatrix} H - \alpha E & \alpha\vec{1} \\ \beta\vec{h} & -\beta \end{pmatrix}$$

порядка $J+1$, но с разными векторами $\vec{g}_1$ и $\vec{g}_2$ размерности $J+1$:

$$\vec{g}_1 = (0,\dots,0,1)\,, \qquad \vec{g}_2 = (h_1,\dots,h_J,0)\,.$$

Если заявка поступает в систему, в которой находится k других заявок, $k=\overline{0,n-1}$, и все свободные $n-k$ приборов восстанавливаются, то время ее обслуживания будет иметь функцию распределения $G_1(x)$; в противном случае время обслуживания заявки будет иметь функцию распределения $G_2(x)$.

Обозначим через $p_k^{(1)}$, $k=\overline{0,n-1}$, вероятность того, что поступающая в систему заявка застанет в ней k других заявок и среди свободных приборов будут отсутствовать исправные, а через $p_k^{(2)}$, $k=\overline{0,n-1}$, — вероятность того, что поступающая заявка застанет в системе k других заявок и среди свободных приборов будут исправные. Ясно, что $p_k^{(1)}$ представляет собой сумму координат вектора $\vec{p}_k^{\,-}$ по всем возможным фазам генерации заявок и всем возможным индексам $i_1,\dots,i_k$, отвечающим состояниям процесса обслуживания $(i_1,\dots,i_k;n-k)$, а $p_k^{(2)}$ — сумма остальных координат вектора $\vec{p}_k^{\,-}$. Напомним, что вектора $\vec{p}_k^{\,-}$ стационарных вероятностей того, что поступающая заявка застанет в системе k других заявок, $k=\overline{0,R}$, определяются из соотношений (9)–(12).

Положим

$$p^{(1)} = \sum_{k=0}^{n-1} p_k^{(1)}\,, \qquad p^{(2)} = \sum_{k=0}^{n-1} p_k^{(2)}\,.$$

С учетом введенных обозначений стационарное распределение $V(x)$ времени пребывания заявки в системе можно записать в виде

$$V(x) = \frac{1}{1-\pi}\left(p^{(1)}G_1(x) + p^{(2)}G_2(x) + \right.$$
$$+ \sum_{i=n}^{R-1} \vec{p}_i^{\,-} G_2(x)\vec{1} -$$
$$\left. - \sum_{j=0}^{r-1} \left(\sum_{i=n+j}^{R-1} \vec{p}_i^{\,-} \right) \int_0^x B_j(x-y)\, dG_2(y)\vec{1} \right).$$

6 Отказы всех приборов с обслуживанием заявки заново; заявки поступают только на исправные приборы

Как и в системе из предыдущего раздела, в рассматриваемой здесь СМО отказывать могут все приборы, а не только те, на которых обслуживаются заявки. По-прежнему, если в системе имеется свободный исправный прибор, то поступающая в систему заявка идет на этот прибор; однако если свободных исправных приборов нет, то заявка становится в очередь. Остальные предположения и обозначения предыдущего раздела остаются в силе.

В отличие от случая бесконечного накопителя [9], эта система уже не приводится к общей системе, разобранной в разд. 2.

Как и прежде, назовем слоем k, $k=\overline{0,R}$, множество всех состояний процесса обслуживания, в которых общее число заявок в системе равно k.

Рассмотрим сначала случай $k<r$. Тогда слой k при $k\ge n$ имеет вид

$$\{(i_1,\dots,i_n)\cup(i_1,\dots,i_{n-1})\cup\dots\cup(0)\}\,,$$

где $i_1,\dots,i_n=\overline{1,J+1}$. Состояние $(i_1,\dots,i_n)$ означает, что первый прибор обслуживает заявку на фазе i_1, $\dots$, n-й прибор — на фазе i_n и еще $k-n$ заявок находятся в очереди; состояние $(i_1,\dots,i_{n-1})$ — первый прибор обслуживает заявку на фазе i_1, $\dots$, $(n-1)$-й прибор — на фазе i_{n-1}, один прибор восстанавливается (без заявки) и $k-n+1$ заявок находятся в очереди; $\dots$; состояние (0) — все приборы восстанавливаются (без заявок) и k заявок находятся в очереди. При этом число фаз обслуживания заявки равно $J+1$. Последняя, $(J+1)$-я, фаза соответствует восстановлению неисправного прибора (на котором находится заявка). Слой k при $k=\overline{0,n-1}$ представляет собой множество

$$\{(i_1,\dots,i_k;m)\cup(i_1,\dots,i_k)\cup$$
$$\cup(i_1,\dots,i_{k-1})\cup\dots\cup(0)\}\,, \quad m=\overline{0,n-k-1}\,,$$

где состояние $(i_1,\dots,i_k;m)$ означает, что первый прибор обслуживает заявку на фазе i_1, $\dots$, k-й прибор — на фазе i_k, а m приборов свободны от заявок и восстанавливаются; состояние $(i_1,\dots,i_k)$ означает, что первый прибор обслуживает заявку на фазе i_1, $\dots$, k-й прибор — на фазе i_k, а остальные $n-k$ приборов свободны от заявок и восстанавливаются; состояние $(i_1,\dots,i_{k-1})$ — первый прибор обслуживает заявку на фазе i_1, $\dots$, $(k-1)$-й прибор — на фазе i_{k-1}, $n-k+1$ приборов свободны от заявок и восстанавливаются и имеется одна заявка в очереди; $\dots$; состояние (0) — n приборов восстанавливаются (без заявок) и k заявок в очереди.

Перейдем к случаю $k\ge r$. Теперь слой k при $k\ge n$ имеет вид

$$\{(i_1,\dots,i_n)\cup(i_1,\dots,i_{n-1})\cup\dots\cup(i_1,\dots,i_{k-r})\}\,.$$

Здесь состояния имеют тот же смысл, что и прежде, но последнее состояние $(i_1,\dots,i_{k-r})$ означает, что

первый прибор обслуживает заявку на фазе $i_1, \ldots,$ $(k-r)$-й прибор — на фазе i_{k-r}, $R-k$ приборов восстанавливаются (без заявок) и r заявок находятся в очереди (накопитель полон — все места ожидания заняты). Слой k при $k = \overline{0, n-1}$ представляет собой множество

$$\{(i_1, \ldots, i_k; m) \cup (i_1, \ldots, i_k) \cup (i_1, \ldots, i_{k-1}) \cup \ldots \cup \cup (i_1, \ldots, i_{k-r})\}, \quad m = \overline{0, n-k-1},$$

причем и теперь отличие от случая $k < r$ состоит в уменьшении числа состояний за счет ограничения емкости накопителя.

Обозначим через D матрицу переходных вероятностей вложенной цепи Маркова, порожденной фазами полумарковского процесса поступления заявок и фазами обслуживания сразу же после моментов поступления заявок в систему. Очевидно, эта матрица имеет вид:

$$D = \begin{pmatrix} D_{1,1} & D_{1,2} & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 \\ D_{2,1} & D_{2,2} & D_{2,3} & \ldots & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ldots & 0 & 0 \\ D_{n-1,1} & D_{n-1,2} & D_{n-1,3} & \ldots & D_{n-1,n} & 0 & \ldots & 0 & 0 \\ D_{n,1} & D_{n,2} & D_{n,3} & \ldots & D_{n,n} & D_{n,n+1} & \ldots & 0 & 0 \\ D_{n+1,1} & D_{n+1,2} & D_{n+1,3} & \ldots & D_{n+1,n} & D_{n+1,n+1} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ D_{R-1,1} & D_{R-1,2} & D_{R-1,3} & \ldots & D_{R-1,n} & D_{R-1,n+1} & \ldots & D_{R-1,R-1} & D_{R-1,R} \\ D_{R,1} & D_{R,2} & D_{R,3} & \ldots & D_{R,n} & D_{R,n+1} & \ldots & D_{R,R-1} & D_{R,R} \end{pmatrix}.$$

Для нахождения элементов $D_{k,s}$ матрицы D обратимся к аналогичной СМО с бесконечным накопителем. Матрица B переходных вероятностей вложенной цепи Маркова системы с бесконечным накопителем, где алгоритм построения матриц $B_{k,s}$ получен в [9], имеет вид:

$$B = \begin{pmatrix} B_{1,1} & B_{1,2} & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots \\ B_{2,1} & B_{2,2} & B_{2,3} & \ldots & 0 & 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ldots \\ B_{n-1,1} & B_{n-1,2} & B_{n-1,3} & \ldots & B_{n-1,n} & 0 & 0 & 0 & \ldots \\ B_{n,1} & B_{n,2} & B_{n,3} & \ldots & B_{n,n} & B_0 & 0 & 0 & \ldots \\ B_{n+1,1} & B_{n+1,2} & B_{n+1,3} & \ldots & B_{n+1,n} & B_1 & B_0 & 0 & \ldots \\ B_{n+2,1} & B_{n+2,2} & B_{n+2,3} & \ldots & B_{n+2,n} & B_2 & B_1 & B_0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

Для матрицы B используем также обозначение:

$$B = \begin{pmatrix} B_{1,1} & B_{1,2} & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots \\ B_{2,1} & B_{2,2} & B_{2,3} & \ldots & 0 & 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ldots \\ B_{n-1,1} & B_{n-1,2} & B_{n-1,3} & \ldots & B_{n-1,n} & 0 & 0 & 0 & \ldots \\ B_{n,1} & B_{n,2} & B_{n,3} & \ldots & B_{n,n} & B_{n,n+1} & 0 & 0 & \ldots \\ B_{n+1,1} & B_{n+1,2} & B_{n+1,3} & \ldots & B_{n+1,n} & B_{n+1,n+1} & B_{n+1,n+2} & 0 & \ldots \\ B_{n+2,1} & B_{n+2,2} & B_{n+2,3} & \ldots & B_{n+2,n} & B_{n+2,n+1} & B_{n+2,n+2} & B_{n+2,n+3} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Заметим теперь, что $D_{k,s} = B_{k,s}$ при $s < k$, поскольку в этом случае как в системе с конечным, так и в системе с бесконечным накопителями потери заявки при поступлении не происходит. Кроме того, $D_{k,s} = B_{k,s}$ и при $k < r$, так как тогда при любом числе отказавших приборов имеются места ожидания и заявка в обеих системах не теряется.

Обратимся к случаю $k \geq r$. Тогда если на предыдущем шаге вложенная цепь Маркова находилась в каком-либо состоянии, соответствующему

состоянию $(i_1,\dots,i_{k-r})$ слоя k процесса обслуживания заявок, то в системе с бесконечным накопителем она переходит в некоторое состояние, соответствующее состоянию $(i_1,\dots,i_{k-r})$, но слоя $k+1$ процесса обслуживания заявок, в то время как в системе с конечным накопителем переходит в состояние, соответствующее состоянию $(i_1,\dots,i_{k-r})$ того же слоя k процесса обслуживания заявок, причем с той же самой фазой полумарковского входящего процесса. Поэтому получаем при $k \geq r$, что

$$D_{k,k} = B_{k,k} + B_{k,k+1}E_k\,,$$

где E_k — матрица того же размера, что и $B_{k+1,k}$, все элементы которой равны 0, кроме равных единице элементов, определяющих переходы вложенной цепи Маркова из состояний, соответствующих всевозможным состояниям $(i_1,\dots,i_{k-r})$ слоя $k+1$ процесса обслуживания заявок (для системы с бесконечным накопителем), в состояния $(i_1,\dots,i_{k-r})$ слоя k процесса обслуживания заявок (для системы с конечным накопителем), причем каждая пара состояний вложенной цепи Маркова имеет одну и ту же фазу полумарковского входящего процесса. Соответственно, матрица $D_{k,k+1}$ получается из матрицы $B_{k,k+1}$ удалением столбцов, соответствующих всем состояниям вложенной цепи Маркова (системы с бесконечным накопителем), при нахождении в которых процесс обслуживания заявок пребывает в каком-либо состоянии $(i_1,\dots,i_{k-r})$ слоя $k+1$.

Обозначая, как и в разд. 2, через $\vec{p}_k^{\,*}$, $k=\overline{1,R}$, вектор-строку, координатами которой p^*_{km}, где $m=\overline{1,Il_k}$ при $k=\overline{1,n-1}$ и $m=\overline{1,Il}$ при $k=\overline{n,R}$, являются стационарные вероятности по вложенной цепи Маркова того, что в системе находится k заявок, а фазы процессов генерации и обслуживания заявок равны i и j соответственно, получаем СУР

$$\vec{p}_1^{\,*} = \sum_{i=1}^{R} \vec{p}_i^{\,*} B_{i,1}\,,$$

$$\vec{p}_k^{\,*} = \sum_{i=k-1}^{R} \vec{p}_i^{\,*} B_{i,k}\,, \quad k=\overline{2,R}$$

с условием нормировки

$$\sum_{k=1}^{R} \vec{p}_k^{\,*}\vec{1} = 1\,,$$

решение которой можно получить методом, упомянутым в разд. 2.

Дальнейшее исследование рассматриваемой СМО, в том числе вычисление стационарных распределений, связанных с временем пребывания заявки в системе, также проводится по алгоритму разд. 2.

7 Отказы работающих приборов; марковский процесс отказов–восстановлений приборов

В заключение рассмотрим вариант СМО с марковским процессом отказов–восстановлений. При этом, так же как и в [9], ограничимся рассмотрением СМО, в которой отказывают только работающие приборы, причем отказы приборов проявляются в изменении скорости обслуживания заявок. Сохраняются все предположения разд. 4 относительно параметров функционирования системы, за исключением экспоненциальности времен, проводимых системой в исправном и неисправном состояниях.

Марковский процесс отказов–восстановлений приборов определим следующим образом. Имеются инфинитезимальная матрица $F=(f_{i,j})$ порядка M, вектор $\vec{f}=(f_1,\dots,f_M)$ размерности M и диагональная матрица D порядка M с неотрицательными элементами $d_{i,i}$ на главной диагонали. Число M назовем числом фаз процесса отказов–восстановлений. Элемент $d_{i,i}$ будем называть скоростью обслуживания заявки при фазе i процесса отказов–восстановлений приборов.

В момент поступления заявки на прибор с вероятностью f_i выбирается фаза i процесса отказов–восстановлений прибора и далее с вероятностью h_k выбирается фаза обслуживания k фазового распределения процесса обслуживания заявки. Затем начинается фазовое обслуживание заявки с матрицей $d_{i,i}H$. Обслуживание при фазе i процесса отказов–восстановлений прибора происходит либо до момента окончания обслуживания, либо до момента, когда в соответствии с интенсивностью $f_{i,j}$ фаза процесса отказов–восстановлений не изменится на j-ю, после чего обслуживание продолжится, но уже с матрицей $d_{j,j}H$, и так далее до момента окончания обслуживания.

Для того чтобы привести систему к общей модели, необходимо положить

$$G = H\otimes D + E\otimes F\,, \qquad \vec{g} = \vec{h}\otimes\vec{f}\,.$$

Теперь осталось воспользоваться результатами разд. 2, позволяющими получить соотношения для вычисления основных стационарных характеристик системы.

В частности, для системы из разд. 4 с экспоненциальными временами пребывания прибора в исправном и неисправном состояниях

$$F = \begin{pmatrix} -\alpha & \alpha \\ \beta & -\beta \end{pmatrix}, \quad \vec{f} = (1, 0), \quad D = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Если же времена пребывания прибора в исправном и неисправном состояниях (для прибора, обслуживающего заявку) имеют фазовые распределения с матрицами $F_1 = (f_{1,i,j})$ порядка M_1 и $F_2 = (f_{2,i,j})$ порядка M_2 и векторами $\vec{f}_1 = (f_{1,1}, \ldots, f_{1,M_1})$ размерности M_1 и $\vec{f}_2 = (f_{2,1}, \ldots, f_{2,M_2})$ размерности M_2, то нужно положить

$$M = M_1 + M_2, \quad \vec{f} = (f_{1,1}, \ldots, f_{1,M_1}, 0, \ldots, 0),$$
$$F = \begin{pmatrix} F_1 & \vec{f}_2 \otimes (F_1\vec{1}) \\ \vec{f}_1 \otimes (F_2\vec{1}) & F_2 \end{pmatrix}, \quad D = \begin{pmatrix} E & 0 \\ 0 & 0 \end{pmatrix}.$$

Введение марковского процесса отказов–восстановлений приборов, как правило, существенно увеличивает размерность процесса, описывающего функционирование СМО.

8 Примеры численного расчета характеристик СМО с ненадежными приборами

На основе математических соотношений для общей модели *SM/MSP/n/r* с надежными приборами был разработан программный комплекс, который, в частности, позволяет рассчитывать стационарные показатели функционирования для некоторых вариантов СМО с ненадежными приборами. Приведем несколько примеров таких расчетов.

Расчеты производились для двух вариантов СМО с ненадежными приборами, в которых могут происходить отказы только исправных приборов, причем в первом случае заявка обслуживается заново (см. разд. 3), а во втором — дообслуживается (см. разд. 4).

Пример 1а. Рассмотрим СМО $SM_2/PH_2/8/5$ с надежными приборами. В системе имеется 8 идентичных приборов и 5 мест ожидания. Полумарковский процесс генерации заявок имеет следующую матрицу переходных вероятностей вложенной цепи:

$$A = \begin{pmatrix} 0{,}8 & 0{,}2 \\ 0{,}6 & 0{,}4 \end{pmatrix}.$$

Условные функции распределения времени пребывания на фазе полумарковского процесса генерации заявок определим следующим образом. Пусть $d_{11} = 0{,}25$ и $d_{22} = 0{,}025$ — времена между соответствующими сменами фаз полумарковского процесса генерации заявок для детерминированных условных функций распределения, а $\lambda_{12} = 4$ и $\lambda_{21} = 2{,}5$ — интенсивности соответствующих смен фаз полумарковского процесса генерации заявок для экспоненциальных условных функций распределения. Время обслуживания заявки каждым прибором имеет распределение фазового типа с матрицей

$$H = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix}$$

и вектором

$$\vec{h} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Пример 1б. Рассматривается СМО из примера *1а*, но теперь прибор, на котором находится заявка, может отказывать с интенсивностью $\alpha = 2{,}5$ и восстанавливаться с интенсивностью $\beta = 12$. Заявки после восстановления обслуживаются заново. Свободные приборы находятся только в исправном состоянии.

Пример 1в. Рассматривается СМО из примера *1б*, но с интенсивностями отказа $\alpha = 2{,}5$ и восстановления $\beta = 32{,}5$.

Пример 1г. Рассматривается СМО из примера *1б*, но с интенсивностями отказа $\alpha = 2$ и восстановления $\beta = 12$.

Пример 1д. Рассматривается СМО из примера *1б*, но с интенсивностями отказа $\alpha = 2$ и восстановления $\beta = 32{,}5$.

Сравнительные результаты расчетов для этих примеров приведены в табл. 1, 2 и на рис. 1.

Пример 2а. Рассматривается СМО $SM_2/PH_2/4/5$ с 4 надежными приборами и 5 местами ожидания. Полумарковский процесс генерации заявок того же типа, что в примере *1а*, но с параметрами $d_{11} = 0{,}5$, $d_{22} = 0{,}05$, $\lambda_{12} = 2$ и $\lambda_{21} = 1{,}25$. Обслуживания фазового типа те же, что и в примере *1а*.

Пример 2б. Рассматривается СМО из примера *2а*, но с ненадежными приборами и дообслуживанием заявок. Свободные приборы находятся только в исправном состоянии. Прибор, на котором находится заявка, может отказывать с интенсивностью $\alpha = 3$ и восстанавливаться с интенсивностью $\beta = 3{,}75$.

Пример 2в. Рассматривается СМО из примера *2б* с интенсивностями отказов $\alpha = 3$ и восстановлений $\beta = 5$.

Пример 2г. Рассматривается СМО из примера *2б* с интенсивностями отказов $\alpha = 2{,}25$ и восстановлений $\beta = 3{,}75$.

Пример 2д. Рассматривается СМО из примера *2б* с интенсивностями отказов $\alpha = 2{,}25$ и восстановлений $\beta = 5$.

Результаты расчетов для примеров *2а–2д* приведены в табл. 3, 4 и на рис. 2.

Таблица 1 Распределение числа заявок в системе по времени

Распределение числа заявок	*1а*	*1б*	*1в*	*1г*	*1д*
p_0	0,01487	0,00031	0,00077	0,00077	0,00148
p_1	0,05038	0,00179	0,00406	0,00406	0,00725
p_2	0,12344	0,00629	0,01322	0,01321	0,02232
p_3	0,21045	0,01674	0,03291	0,03288	0,05257
p_4	0,23628	0,03613	0,06551	0,06545	0,09730
p_5	0,18028	0,06405	0,10466	0,10461	0,14123
p_6	0,10173	0,09354	0,13466	0,13466	0,16190
p_7	0,04735	0,11367	0,14174	0,14180	0,14993
p_8	0,02020	0,11861	0,12685	0,12694	0,11741
p_9	0,00864	0,11667	0,10634	0,10644	0,08596
p_{10}	0,00378	0,11470	0,08900	0,08905	0,06288
p_{11}	0,00164	0,11301	0,07440	0,07441	0,04588
p_{12}	0,00069	0,10917	0,06079	0,06074	0,03266
p_{13}	0,00028	0,09530	0,04509	0,04499	0,02124

Таблица 2 Агрегированные характеристики

Характеристики	*1а*	*1б*	*1в*	*1г*	*1д*
Интенсивность входящего потока	4,00000	4,00000	4,00000	4,00000	4,00000
Интенсивность обслуживания	8,00000	4,07427	4,57143	4,57143	5,02415
Загрузка	0,50000	0,98177	0,87500	0,87500	0,79615
Вероятность потери заявки	0,00031	0,07996	0,03793	0,03785	0,01808
Среднее число занятых приборов	3,99877	7,22613	6,73451	6,73508	6,25405
Средняя длина очереди	0,02529	1,59832	0,97615	0,97571	0,58620
Среднее число заявок в системе	4,02406	8,82445	7,71066	7,71078	6,84025
Среднее время ожидания начала обслуживания	0,00632	0,43431	0,25366	0,25352	0,14925
Среднее время обслуживания заявки	1,00000	1,96354	1,75000	1,75000	1,59231
Среднее время пребывания заявки в системе	1,00632	2,39785	2,00366	2,00352	1,74156

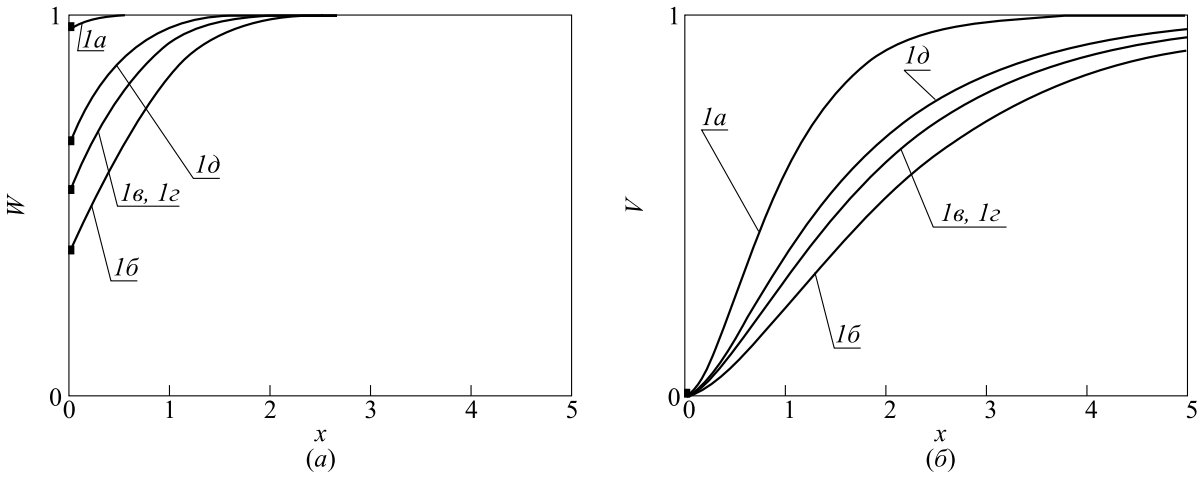

Рис. 1 Стационарные распределения времени ожидания начала обслуживания $W(x)$ (*а*) и времени пребывания заявки в системе $V(x)$ (*б*) для примеров *1а–1д*

Таблица 3 Распределение числа заявок в системе по времени

Распределение числа заявок	*2а*	*2б*	*2в*	*2г*	*2д*
p_0	0,09418	0,01386	0,02385	0,02389	0,03471
p_1	0,25648	0,04807	0,07823	0,07913	0,10989
p_2	0,32943	0,10798	0,16334	0,16385	0,21190
p_3	0,19683	0,15921	0,21000	0,20879	0,23709
p_4	0,07473	0,16184	0,17797	0,17619	0,16977
p_5	0,02835	0,14153	0,12716	0,12626	0,10259
p_6	0,01200	0,12183	0,08993	0,08994	0,06249
p_7	0,00510	0,10363	0,06302	0,06357	0,03798
p_8	0,00210	0,08388	0,04209	0,04299	0,02214
p_9	0,00082	0,05818	0,02440	0,02540	0,01142

Таблица 4 Агрегированные характеристики

Характеристики	*2а*	*2б*	*2в*	*2г*	*2д*
Интенсивность входящего потока	2,00000	2,00000	2,00000	2,00000	2,00000
Интенсивность обслуживания	4,00000	2,22222	2,50000	2,50000	2,75862
Загрузка	0,50000	0,90000	0,80000	0,80000	0,72500
Вероятность потери заявки	0,00092	0,04855	0,02087	0,02169	0,01014
Среднее число занятых приборов	1,99816	3,42522	3,13323	3,13060	2,87059
Средняя длина очереди	0,08010	1,32251	0,78646	0,79578	0,48721
Среднее число заявок в системе	2,07826	4,74773	3,91968	3,92638	3,35780
Среднее время ожидания начала обслуживания	0,04009	0,69500	0,40161	0,40671	0,24610
Среднее время обслуживания заявки	1,00000	1,80000	1,60000	1,60000	1,45000
Среднее время пребывания заявки в системе	1,04009	2,49500	2,00161	2,00671	1,69610

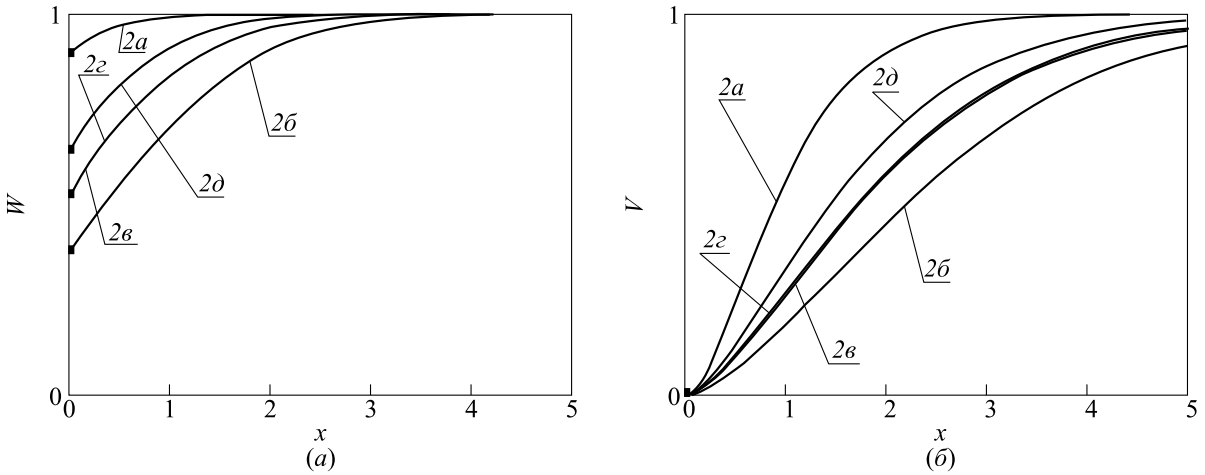

Рис. 2 Стационарные распределения времени ожидания начала обслуживания W(x) (*а*) и времени пребывания заявки в системе $V(x)$ (*б*) для примеров *2а–2д*

9 Заключение

В настоящей работе получены математические соотношения для расчета стационарных характеристик СМО с полумарковским входящим потоком, обслуживанием фазового типа, конечным накопителем и ненадежными приборами. Рассмотрены различные варианты функционирования СМО с независимо отказывающими приборами при экспоненциальном процессе отказов–восстановлений. Кроме этого, рассмотрен вариант системы с отказами только работающих приборов и марковским процессом отказов–восстановлений. Для нескольких СМО с ненадежными приборами на основе полученных в настоящей работе результатов проведены численные расчеты следующих стационарных показателей функционирования: распределения очереди по моментам поступления заявок и по времени, среднего числа заявок в системе, среднего числа занятых приборов, среднего числа заявок в очереди, распределений времени ожидания начала обслуживания и времени пребывания заявки в системе и их средних значений.

Литература

1. *Kabak I. V.* Blocking and delays in $M^{(n)}/M/c$ queuing systems // Operations Res., 1968. Vol. 16. P. 830–840.
2. *Mytrany I. L., Avi-Itzhak B.* A many-server queue with service interruptions // Operations Res., 1968. Vol. 16. P. 628–638.
3. *Neuts M. F., Lucantoni D. M.* A Markovian queue with N servers subject to breakdowns and repairs. Mgmt. Sci., 1979. Vol. 25. P. 849–861.
4. *Бабицкий А. В., Дудин А. Н., Клименок В. И.* К расчету характеристик ненадежной системы массового обслуживания с конечным источником // Автоматика и телемеханика, 1996. № 1. С. 92–103.
5. *Дудин А. Н.* Оптимальное гистерезисное управление ненадежной системой BMAP/SM/1 с двумя режимами работы // Автоматика и телемеханика, 2002. № 10. С. 58–72.
6. *Dudin A. N., Kazimirsky A. V., Klimenok V. I.* $BMAP/G/1$ system unreliable in an idle state // Bulletin of Kerala Mathematics Association, 2004. No. 2. P. 1–19.
7. *Микадзе И. С., Хочолава В. В., Хуродзе Р. А.* Виртуальное время ожидания в однолинейной СМО с ненадежным прибором // Автоматика и телемеханика, 2004. № 12. С. 119–128.
8. *Mikadze I. S., Khocholava V. V.* Studing the queue length in a single server queuing system with unreliable server // Automation and Remote Control, 2005. Vol. 6. No. 1. P. 65–73.
9. *Печинкин А. В., Соколов И. А., Чаплыгин В. В.* Многолинейные системы массового обслуживания с независимыми отказами и восстановлениями приборов // Системы и средства информатики: спец. выпуск «Математическое и алгоритмическое обеспечение информационно-телекоммуникационных систем». М.: Изд-во Института проблем информатики РАН, 2006. С. 99–123.
10. *Печинкин А. В., Чаплыгин В. В.* Стационарные характеристики системы массового обслуживания $SM/MSP/n/r$ // Автоматика и телемеханика, 2004. № 9. С. 85–100.
11. *Bocharov P. P., D'Apice C., Pechinkin A. V., Salerno S.* Queueing Theory. Utrecht, Boston: VSP, 2004.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007, т. 1, №1: 40–53

НОВЫЙ МЕТОД ВЕРОЯТНОСТНО-СТАТИСТИЧЕСКОГО АНАЛИЗА ИНФОРМАЦИОННЫХ ПОТОКОВ В ТЕЛЕКОММУНИКАЦИОННЫХ СЕТЯХ*

Д. А. Батракова[1], В. Ю. Королев[2], С. Я. Шоргин[3]

Аннотация: В данной работе предлагается метод исследования стохастической структуры хаотических информационных потоков в сложных телекоммуникационных сетях. Предлагаемый метод основан на стохастической модели телекоммуникационной сети, в рамках которой она представляется в виде суперпозиции некоторых простых последовательно-параллельных структур. Эта модель естественно порождает смеси гамма-распределений для времени выполнения (обработки) запроса сетью. Параметры получаемой смеси гамма-распределений характеризуют стохастическую структуру информационных потоков в сети. Для решения задачи статистического оценивания параметров смесей экспоненциальных и гамма-распределений (задачи разделения смесей) используется EM-алгоритм. Чтобы проследить изменение стохастической структуры информационных потоков во времени, EM-алгоритм применяется в режиме скользящего окна. Описывается программный инструментарий для применения полученного решения к реальным статистическим данным. Приводится интерпретация результатов.

Ключевые слова: телекоммуникационные сети; информационные потоки; разделение смесей распределений; метод скользящего окна; программа для разделения смесей

1 Введение

Развитие телекоммуникационных сетей, их усложнение поставило перед инженерами важную прикладную задачу исследования характеристик информационных потоков, возникающих в этих сетях. Здесь под информационным потоком мы будем понимать упорядоченное движение любого вида информации по сети.

Если на заре эры телекоммуникаций, в эпоху первых телефонных линий и телеграфа эта проблема не была столь насущной, то со временем, при постепенном охвате мирового пространства сетями возникла необходимость в построении и исследовании адекватных моделей сетей и процессов, происходящих в них.

Современные сети — это *конвергентные* сети, т.е. совокупность крайне разнородных как по топологии, так и по физической архитектуре сетей, которые предлагают конечному пользователю самые разнообразные сервисы. Это — огромное виртуальное и физическое пространство, состоящее из миллионов процессоров, операционных платформ, линий передачи данных и стыковочных узлов. Существует множество классификаций телекоммуникационных сетей по различным признакам:

- масштабу (локальные сети — LAN, масштаба города — MAN, широкого масштаба — WAN);
- топологии, или логической организации («звезда», «кольцо», «шина»);
- физической организации (оптические сети, радио);
- предлагаемым услугам (сотовые сети, для доступа в Интернет);
- назначению (военные, гражданские) и др.

Конвергентная сеть входит во все эти классы, причем, как правило, обладает всеми этими признаками. Поэтому построение модели для ее анализа является и очень важной, и очень сложной задачей.

Существуют достаточно многочисленные математические методы, ориентированные на моделирование и анализ телекоммуникационных сетей. В большинстве своем они основываются на теории массового обслуживания, то есть разделе теории вероятностей, посвященном описанию функционирования сложных систем обслуживания (в том числе телекоммуникационных сетей и систем) с помощью стохастических процессов особого вида

* Работа выполнена при поддержке РФФИ, проекты №№ 04-01-00671, 05-07-90103.

[1] ИПИ РАН, daria.batrakova@gmail.com

[2] Факультет вычислительной математики и кибернетики МГУ им. М. В. Ломоносова, ИПИ РАН, vkorolev@comtv.ru

[3] ИПИ РАН, sshorgin@ipiran.ru

и анализу таких процессов. Указанная теория является весьма развитой и широко применяется на практике. Тем не менее, ее применимость ограничена — во-первых, все возрастающей сложностью структур и дисциплин обслуживания в рассматриваемых реальных сетях. Эта сложность во многих случаях принципиально не может найти адекватного отображения в моделях массового обслуживания, даже несмотря на постоянно растущую сложность самих этих моделей. В результате даже модели, допускающие точный математический анализ, дают возможность расчета всего лишь приближенных значений характеристик реальных сетей, ибо предположения, принимаемые при построении моделей, во многих случаях не соответствуют практике. Во-вторых, для описания телекоммуникационной сети в виде сети массового обслуживания исследователь должен располагать детальным описанием структуры сети, что далеко не всегда имеет место на практике. В-третьих, разработано крайне мало моделей массового обслуживания, в которых используется в качестве входной информация о наблюдаемых (статистических) показателях функционирования сети; в то же время, такая информация очень часто доступна исследователю, и ее использование при анализе сети весьма целесообразно.

В данной работе предлагается в определенной степени альтернативный подход к моделированию сложных телекоммуникационных сетей. Строится и исследуется вероятностная модель сложной телекоммуникационной сети как суперпозиции достаточно простых структур. При этом практически никакая априорная информация о структуре исследуемой сети не используется — наоборот, в результате исследования модели исследователь получает приближенное представление об этой структуре. Характеристики типовых простых структур, составляющих в совокупности модель сети, и сети в целом при этом подходе описываются гамма-распределениями. Ставится задача оценки параметров модели на основе статистических данных о функционировании сети, а также предлагается математическое решение этой задачи. В статье описан также созданный на основе разработанных математических методов программный инструментарий и приведены результаты расчетов для реального трафика.

2 Математическая модель и постановка задачи

2.1 Логическая модель сети

Рассмотрим абстрактную *конвергентную телекоммуникационную сеть*. Это может быть как крупномасштабная транспортная сеть (WAN), сеть отдельного оператора масштаба города (MAN) с различными сервисами, так и локальная сеть (LAN).

Любой из этих случаев можно описать как $(p,\ q)$-*сеть*.

Определение 1. В теории графов и сетей под $(p,\ q)$-сетью понимается набор вида $S = (G,\ V',\ V'')$, где G — граф, а V' и V'' — выборки из множества $V(G)$ (вершин графа) длины p и q соответственно. При этом выборка V' (V'') считается *входной* (*выходной*) выборкой, а ее i-я вершина называется *i-м входным* (*выходным*) *полюсом* или, иначе, *i-м входом* (*выходом*) сети S. Вершины, не участвующие во входной и выходной выборках сети, считаются ее внутренними вершинами [1].

Сеть S (рис. 1) имеет p точек входа — точек соединения с внешней средой (это могут быть точки стыковки разнородных сетей, сетей различных операторов, физические подключения к интерфейсам маршрутизаторов и т.п.). Под *внешней средой* будем понимать другие сети, которые передают данные в сеть S. Отдельные «единицы» данных (кадры, сообщения, датаграммы, пакеты) поступают на входы сети S, обрабатываются и подаются на каждый из q выходов, которые могут быть соединены как с конечными пользователями, так и с другими сетями.

Следует отметить, что структура сложных телекоммуникационных сетей обладает свойством некоторого самоподобия, т.е. на каком бы уровне сетевой архитектуры мы ни рассматривали поведение информационных потоков, мы можем выделить некоторые базовые структуры, субпотоки, суперпозицией которых мы можем получить модель конкретной сети, какой бы уровень «детализации» сегментов сети мы ни взяли. Так, например, физические подключения к интерфейсам оптического кросс-коннекта в этом смысле подобны «виртуальным» подключениям к портам TCP на сервере приложений.

Итак, независимо от уровня сетевой архитектуры мы можем рассматривать некоторую величину, характеризующую количество каких-либо ресурсов сети S, занимаемых в процессе передачи и обработки данных. Это могут быть ресурсы, относящиеся как к «объему» (памяти сетевого устройства, количеству занятых линий, размеру пакета), так и ко «времени» (времени обслуживания заявки, времени простоя). Эта величина случайна, т.к. мы не можем абсолютно точно сказать для сложной телекоммуникационной сети, какое сообщение на какой из входов поступит и какого размера оно будет. Таким образом, случайный характер данной величины определяется случайностью поведения внешней среды.

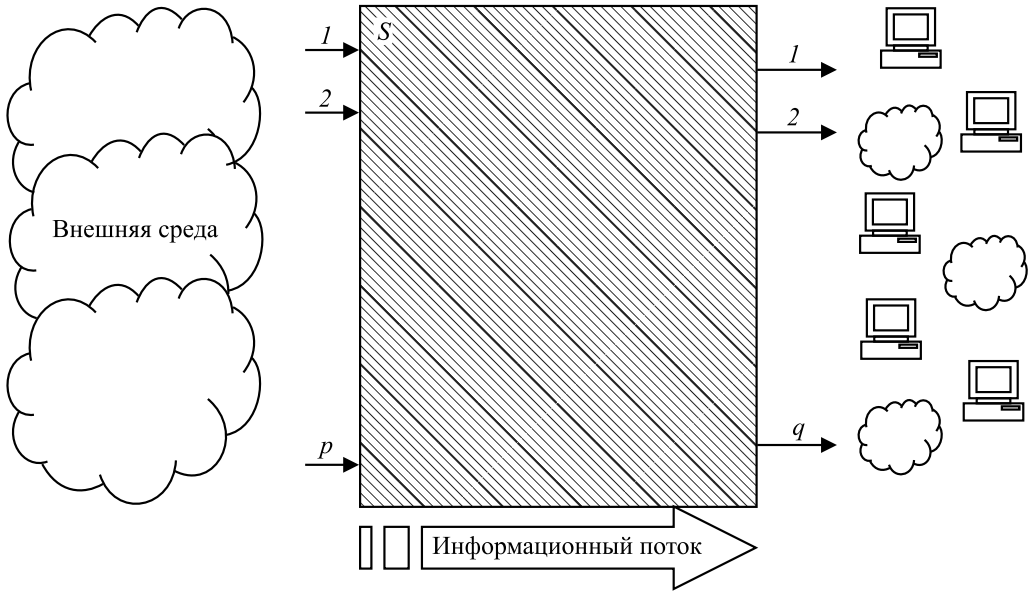

Рис. 1 Абстрактная телекоммуникационная сеть

Пусть R — это описанная выше случайная величина, $R > 0$. Далее, не ограничивая общности, будем подразумевать под ней время, необходимое для какой-либо операции сети (обработки «единицы» данных), предполагая, что время обработки прямо зависит от объема сообщения.

2.2 Вероятностная модель сети

Даже не зная реальной топологии сети, мы можем описать функционирование некоторых ее участков как процесс выполнения операций (задач сети) в последовательном порядке (например, если доступен только один канал данных) или как процесс одновременного выполнения субопераций (когда доступно более одного пути выполнения). Это значит, что мы можем представить функционирование сложной телекоммуникационной сети как *суперпозицию* таких «последовательных» и «параллельных» блоков.

Для построения вероятностной модели распределения R используется комбинация асимптотического подхода, основанного на предельных теоремах теории вероятностей, и принципа максимальной неопределенности (энтропии).

Рассмотрим следующую модель. Предположим, что мы можем разделить сеть S на несколько сегментов S_i. Пусть T — случайная величина, время выполнения операции в отдельно взятом блоке S_i (сегменте сети).

Если операции выполняются *параллельно*, то время, необходимое для их выполнения — это максимальное время, затрачиваемое на какую-либо субоперацию:

$$T = \max_i T_i \,,$$

где T_i — случайные величины для соответствующих субопераций. Модель такого выполнения представлена на рис. 2.

Известно, что предельное распределение экстремальных значений для выборок — это экспоненциальное распределение с плотностью [2]

$$f(x) = \begin{cases} \lambda e^{-\lambda x}\,, & x > 0\,, \\ 0\,, & x \le 0\,, \end{cases}$$

где $\lambda > 0$ — параметр масштаба.

Учитывая также энтропийный подход, естественно будет считать распределение T экспоненциальным, т.к. экспоненциальное распределение обладает наибольшей энтропией среди всех распределений с $x > 0$.

Если же операции сети выполняются *последовательно*, то величина T — это сумма времен T_i, необходимых для выполнения каждой субоперации:

$$T = \sum_i T_i \,,$$

где T_i — случайные величины для соответствующих субопераций. Такая модель представлена на рис. 3.

Это значит, что распределение общей длительности T выполнения блока — это свертка распределений «элементарных» времен T_i (экспоненциально распределенных).

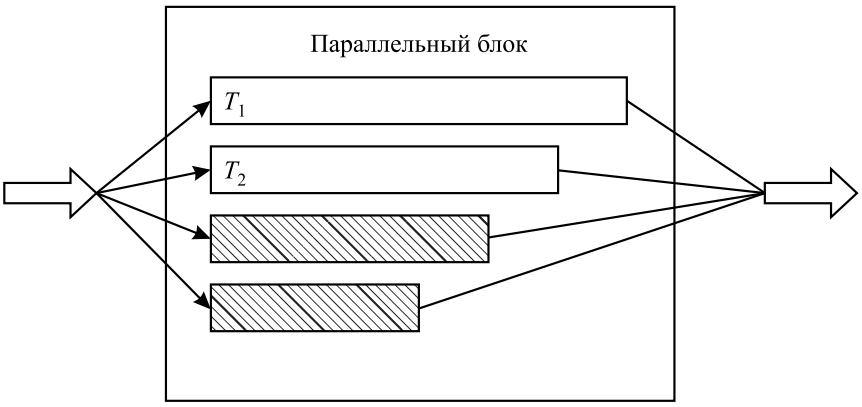

Рис. 2 Параллельное выполнение

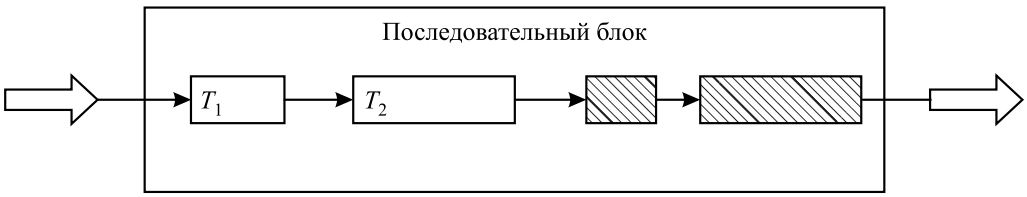

Рис. 3 Последовательное выполнение

Известно, что результатом свертки экспоненциальных распределений является гамма-распределение, определяемое плотностью

$$g_{\lambda,\alpha}(x) = \begin{cases} \dfrac{\lambda_0^{\alpha_0}}{\Gamma(\alpha_0)}\, x^{\alpha_0-1} e^{\lambda_0 x}, & x > 0\,, \\ 0, & x \le 0\,, \end{cases}$$

где $\alpha > 0$ — параметр формы, $\lambda > 0$ параметр масштаба, а $\Gamma(z)$ — гамма-функция Эйлера:

$$\Gamma(z) = \int\limits_0^\infty x^{z-1} e^{-x}\, dx\,.$$

Известно [2], что класс гамма-распределений замкнут над операцией свертки, поэтому результирующее распределение случайной величины R будет также гамма-распределением

$$g_{\lambda,\alpha}(x) = \begin{cases} \dfrac{\lambda^{\alpha}}{\Gamma(\alpha)}\, x^{\alpha-1} e^{-\lambda x}, & x > 0\,, \\ 0, & x \le 0\,. \end{cases}$$

В силу случайного характера ввода данных в сеть S из внешней среды маршрут передачи данных становится случайным, что представлено на рис. 4. Это означает, что параметры результирующего распределения R тоже случайны. Отсюда имеем следующую модель *смеси гамма-распределений*, определяемой плотностью

$$p(x) = \iint g_{\lambda,\alpha}(x)\, dH(\lambda,\, \alpha)\,, \qquad (1)$$

где $H(\lambda, \alpha)$ — смешивающая функция, функция распределения входных параметров.

Поясним понятие *смеси распределений*.

Определение 2. Пусть имеется двухпараметрическое семейство n-мерных плотностей распределения

$$F = \{f_\omega(x;\, \theta(\omega))\}\,, \qquad (2)$$

где одномерный (целочисленный или непрерывный) параметр ω в качестве нижнего индекса функции f определяет специфику общего вида каждого компонента — распределения смеси, а в качестве аргумента при многомерном, вообще говоря, параметре θ определяет зависимость значений хотя бы части компонентов этого параметра от того, в каком именно составляющем распределении f_ω он присутствует. Кроме того, пусть $P = \{P(\omega)\}$ — *семейство смешивающих функций* распределения.

Функция плотности распределения

$$f(x) = \int f_\omega(x;\, \theta(\omega))\, dP(\omega) \qquad (3)$$

называется P-*смесью* (или просто *смесью*) *распределений* семейства F, интеграл в (3) понимается в смысле Лебега–Стильтьеса [3].

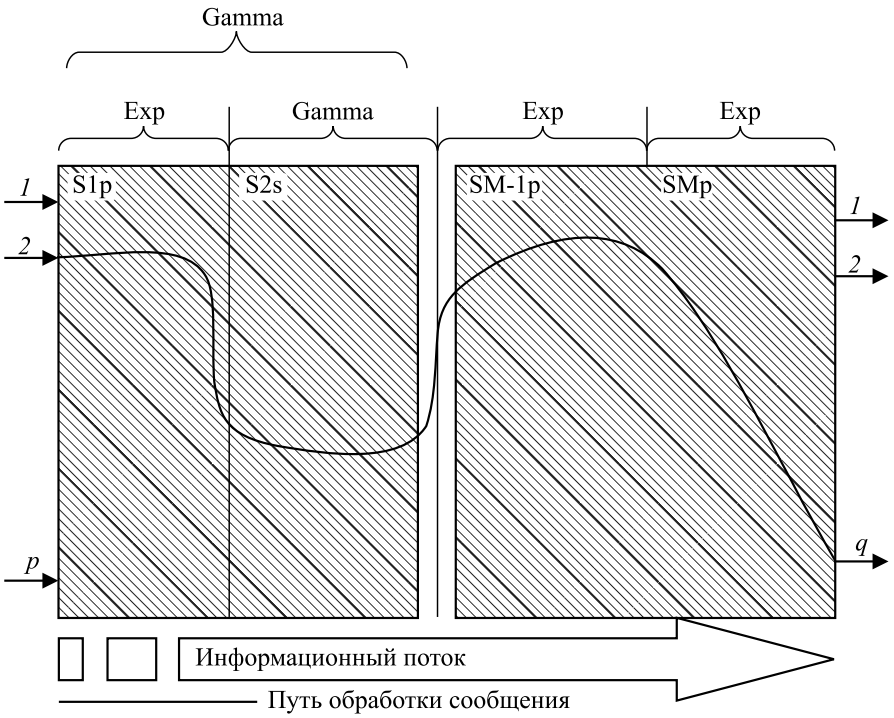

Рис. 4 Модель пути обработки сообщения сетью S

Определение 3. Семейство смесей (3) называется *идентифицируемым* (*различимым*), если из равенства

$$\int f_\omega(x;\,\theta(\omega))\,dP(\omega) = \int f_\omega(x,\,\theta(\omega))dP^*(\omega)$$

следует, что $P(\omega) \equiv P^*(\omega)$ для всех $P \in P(\omega)$ [3].

2.3 Постановка задачи

Перед нами встает задача *разделения* такой смеси. Вообще говоря, задача разделения смесей распределений со смешивающими функциями общего вида является *некорректно поставленной*, т.к. она допускает существование нескольких решений. Поэтому будем искать решение в классе *конечных идентифицируемых смесей распределений*, где смешивающая функция дискретна.

Для этого сузим данное выше определение и будем рассматривать в дальнейшем лишь случай конечного числа k возможных значений параметра ω, что соответствует конечному числу скачков смешивающих функций $P(\omega)$. Величины этих скачков как раз и будут играть роль *удельных весов* (*априорных вероятностей*) p_j компонентов смеси. Более того, в нашем случае мы постулируем также однотипность компонентов распределений f_j, т.е. принадлежность всех f_j к одному общему параметрическому семейству $\{f(X;\,\theta)\}$, где θ — многомерный, вообще говоря, параметр. Так что (3) в этом случае может быть записано в виде

$$p(x) = \sum_{j=1}^{k} p_j f_j(x;\,\theta_j)\,. \tag{4}$$

Переформулируем понятие идентифицируемости (различимости) смесей специально применительно к такому виду смесей.

Определение 4. *Конечная смесь* (3) называется *идентифицируемой* (*различимой*), если из равенства

$$\sum_{j=1}^{k} p_j f_j(x;\,\theta_j) = \sum_{l=1}^{k^*} p_l^* f_l(x;\,\theta_l^*)$$

следует, что $k = k^*$ и для любого j $(1 \le j \le k)$ найдется такое l $(1 \le l \le k^*)$, что $p_j = p_l^*$ и $f_j(x;\,\theta_j) = f_l(x;\,\theta_l^*)$ [3].

Решить эту задачу в выборочном варианте — значит по выборке классифицируемых наблюдений $X_1, \ldots, X_n$, извлеченной из генеральной совокупности, являющейся смесью (3) генеральных совокупностей типа (2) (при заданном общем виде составляющих смесь функций $f_j(x;\,\theta_j)$), построить статистические оценки для числа компонентов смеси k, их удельных весов p_j и, главное, для каждого из компонентов функции f_j однозначно определяются своими параметрами θ_j: $f_j = f(x;\,\theta_j)$.

Однако не следует ставить знак тождества между задачей разделения смеси и задачей статистического оценивания параметров в модели (4) по выборке $X_1, \ldots, X_n$, поскольку задача разделения сохраняет смысл и применительно к генеральным совокупностям, т.е. в теоретическом варианте [3].

Итак, для статистического анализа на основе реальных данных мы аппроксимируем нашу общую модель (1) следующей:

$$p(x) \approx \hat{p}(x) = \sum_{j=1}^{k} p_j g_{\lambda_j,\alpha_j}(x)\,,$$

где p_j — дискретные смешивающие параметры, $g_{\lambda_j,\alpha_j}(x)$ — плотности гамма-распределений.

Такая аппроксимация не только позволяет решить поставленную статистическую задачу, но и получить наглядную визуализацию результатов. Существуют достаточно эффективные методики разделения смесей распределений, среди них — семейство так называемых *EM-алгоритмов* (*Expectation-Maximization Algorithms*).

Полученные результаты могут быть достаточно просто интерпретированы. Число компонентов смеси символизирует число типичных параллельных или последовательных структур. Значения параметров составляющих смесь гамма-распределений показывают «степень параллелизма» соответствующей структуры: чем ближе параметр формы к 1, тем выше эта «степень». И наоборот, чем дальше значение параметра формы от 1, тем больше последовательных операций выполняется в соответствующем блоке.

Веса компонентов характеризуют примерную долю использования ресурсов для сообщений, соответствующих каждому распределению входных данных.

Итак, предложенный подход позволяет получить представление о стохастической структуре телекоммуникационной сети.

3 EM-алгоритм разделения смесей распределений

3.1 Описание алгоритма

Определяемый ниже итерационный алгоритм решения поставленной в предыдущем разделе задачи относится к процедурам, базирующимся на *методе максимального правдоподобия*.

Этот алгоритм позволяет находить максимум логарифмической функции правдоподобия по параметрам $p_1, p_2, \ldots, p_k, \theta_1, \theta_2, \ldots, \theta_k$ при фиксированном k по выборке $X_1, \ldots, X_n$, т.е. решение оптимизационной задачи вида

$$\sum_{i=1}^{n} \ln\left(\sum_{j=1}^{k} p_j f_j(X_i;\,\theta_j)\right) \to \max_{p_j,\,\theta_j}. \qquad (5)$$

Конкретные алгоритмы, построенные по этой схеме, часто называют *алгоритмами типа EM*, поскольку в каждом из них можно выделить два этапа, находящихся по отношению друг к другу в последовательности итерационного взаимодействия: *оценивание* (*Estimation*) и *максимизация* (*Maximization*) [4].

Введем в рассмотрение так называемые апостериорные вероятности g_{ij} принадлежности наблюдения X_i к j-му классу:

$$g_{ij} = \frac{p_j f(X_i;\,\theta_j)}{\sum\limits_{l=1}^{k} p_l f(X_i;\,\theta_l)}\ (i = 1, \ldots, n;\ j = 1, \ldots, k)\,. \qquad (6)$$

Очевидно, что для всех $i = 1, \ldots, n$ и $j = 1, \ldots, k$

$$g_{ij} \geq 0, \quad \sum_{j=1}^{k} g_{ij} = 1\,.$$

Далее обозначим $\Theta = (p_1, \ldots p_k, \theta_1, \ldots, \theta_k)$ и представим анализируемую логарифмическую функцию правдоподобия

$$\ln L(\Theta) = \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{k} p_j f_j(X_i;\,\theta_j)\right)$$

в виде

$$\ln L(\Theta) = \sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij} \ln p_j + \\ + \sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij} f(X_i;\,\theta_j) - \sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij} \ln g_{ij}\,. \qquad (7)$$

Справедливость этого тождества легко проверяется с учетом

$$\sum_{j=1}^{k} g_{ij} = 1\,.$$

Далее идея построения итерационного алгоритма вычисления оценок $\hat{\Theta} = (\hat{p}_1, \ldots, \hat{p}_k,\ \hat{\theta}_1, \ldots, \hat{\theta}_k)$ для параметров $\Theta = (p_1, \ldots, p_k,\ \theta_1, \ldots, \theta_k)$ состоит в следующем:

1. Выбирается некоторое *начальное приближение* $\hat{\Theta}^0$.
2. **E-step:** вычисляются по формулам (6) начальные приближения g_{ij}^0 для апостериорных вероятностей g_{ij} — *этап оценивания*.

3. **M-step:** затем, возвращаясь к (7), при вычисленных значениях g_{ij}^0 следует определить значения $\hat{\Theta}^1$ из условия максимизации отдельно каждого из первых двух слагаемых правой части (7), поскольку первое слагаемое

$$\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij}\ln p_j$$

зависит только от параметров p_j, а второе слагаемое

$$\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij} f(X_i;\,\theta_j)$$

зависит только от параметров θ_j — *этап максимизации*.

4. Проверяется *условие останова*:

$$\| \Theta^{(t)} - \Theta^{t-1} \| < \varepsilon\,,$$

где t — номер итерации, а $\| \bullet \|$ — евклидова норма, для некоторого $\varepsilon > 0$.

Очевидно, решение оптимизационной задачи

$$\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij}^{(t)}\ln p_j \to \max_{p_j}$$

дается выражением (с учетом $\sum_{j=1}^{k} p_j = 1$):

$$p_{ij}^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} g_{ij}^{(t)}\,,$$

где t — номер итерации, $t = 0, 1, 2, \ldots$

Решение оптимизационной задачи

$$\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij}^{(t)} f(X_i;\,\theta_j) \to \max_{\theta_j}$$

получить намного проще решения задачи (5): выражение для θ_j записывается с учетом знания конкретного вида функций $f(X,\,\theta)$ [3].

3.2 О сходимости алгоритма

В работе М. И. Шлезингера [5], где эта схема (позднее названная EM-схемой) впервые предложена, установлены и основные свойства реализующих ее алгоритмов. В частности, было доказано, что при достаточно широких предположениях *предельные точки* всякой последовательности, порожденной итерациями EM-алгоритма, являются стационарными точками оптимизируемой логарифмической функции правдоподобия $\ln L(\Theta)$ и что найдется неподвижная точка алгоритма, к которой будет сходиться каждая из таких последовательностей. Если дополнительно потребовать положительной определенности информационной матрицы Фишера для $\ln L(\Theta)$ при истинных значениях параметра Θ, то можно показать, что асимптотически по $n \to \infty$ (т.е. при больших выборках) существует единственное сходящееся (по вероятности) решение $\hat{\Theta}(n)$ уравнений метода максимального правдоподобия и, кроме того, существует в пространстве параметров Θ норма, в которой последовательность $\Theta^{(t)}(n)$, порожденная EM-алгоритмом, сходится к $\hat{\Theta}(n)$, если только начальная аппроксимация $\hat{\Theta}^0$ не была слишком далека от $\hat{\Theta}(n)$.

Таким образом, результаты исследования свойств EM-алгоритмов метода максимального правдоподобия разделения смеси и их практическое использование показали, что они являются достаточно работоспособными (при известном числе компонентов смеси) даже при большом числе k компонентов и при высоких размерностях анализируемого признака X [3].

3.3 Уравнения для смеси экспоненциальных распределений

Применим описанный выше алгоритм к разделению смеси экспоненциальных распределений:

$$p(x) = \sum_{j=1}^{k} p_j \lambda_j e^{-\lambda_j x}\,.$$

Получаем следующие итерационные уравнения:

$$g_{ij}^{(t+1)} = \frac{p_j^{(t)}\lambda_j^{(t)} e^{-\lambda_j^{(t)} X_i}}{\sum\limits_{l=1}^{k} p_l^{(t)}\lambda_l^{(t)} e^{-\lambda_l^{(t)} X_i}}\,,$$

$$p_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} g_{ij}^{(t)}\,.$$

Чтобы найти оценки λ_j, подсчитаем первую производную функции

$$\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij}^{(t)}\ln(\lambda_j e^{-\lambda_j X_i}):$$

$$\left(\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij}^{(t)}\ln\left(\lambda_j e^{-\lambda_j X_i}\right)\right)' \lambda_j =$$

$$= \left(\sum_{j=1}^{k}\sum_{i=1}^{n} g_{ij}^{(t)}\ln(\lambda_j - \lambda_j X_i)\right)' \lambda_j =$$

$$= \sum_{i=1}^{n} g_{ij}^{(t)}\left(\frac{1}{\lambda_j} - X_i\right).$$

Разрешая уравнение

$$\sum_{i=1}^{n} g_{ij}^{(t)} \left(\frac{1}{\lambda_j} - X_i \right) = 0$$

относительно λ_j, получаем следующее итерационное уравнение:

$$\lambda_j^{(t+1)} = \frac{\sum\limits_{i=1}^{n} g_{ij}^{(t)}}{\sum\limits_{i=1}^{n} g_{ij}^{(t)} X_i} .$$

3.4 Уравнения для смеси гамма-распределений

Применим теперь EM-алгоритм к смеси гамма-распределений вида

$$p(x) = \sum_{j=1}^{k} p_j \frac{\alpha_j^{\alpha_j} x^{\alpha_j - 1}}{\lambda_j^{\alpha_j} \Gamma(\alpha_j)} e^{-(\alpha_j/\lambda_j)x} .$$

Такая параметризация удобна для нахождения оценок α_j [6].

Аналогичным способом выписываются итерационные уравнения:

$$g_{ij}^{(t+1)} = \frac{p_j^{(t)} \dfrac{(\alpha_j^{\alpha_j})^{(t)} x^{\alpha_j - 1}}{(\lambda_j^{\alpha_j})^{(t)} \Gamma(\alpha_j)} e^{-(\alpha_j/\gamma_j)^{(t)} x}}{\sum\limits_{l=1}^{k} p_l^{(t)} \dfrac{(\alpha_l^{\alpha_l})^{(t)} x^{\alpha_l - 1}}{(\lambda_l^{\alpha_l})^{(t)} \Gamma(\alpha_l)} e^{-(\alpha_l/\lambda_l)^{(t)} x}} ,$$

$$p_j^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} g_{ij}^{(t)} .$$

Далее найдем оценки λ_j для данного случая, приравнивая производную

$$\sum_{j=1}^{k} \sum_{i=1}^{n} g_{ij}^{(t)} \ln \left(\frac{\alpha_j^{\alpha_j} x^{\alpha_j - 1}}{\lambda_j^{\alpha_j} \Gamma(\alpha_j)} e^{-(\alpha_j/\lambda_j)x} \right) \qquad (8)$$

по λ_j к нулю и разрешая относительно λ_j уравнение:

$$\sum_{i=1}^{n} g_{ij}^{(t+1)} \left(\frac{\alpha_j^{(t)}}{\lambda_j^{(t)}} - \frac{\alpha_j^{(t)} X_i}{\left(\lambda_j^{(t)}\right)^2} \right) = 0 .$$

Получаем

$$\lambda_j^{(t+1)} = \frac{\sum\limits_{i=1}^{n} g_{ij}^{(t)} X_i}{\sum\limits_{i=1}^{n} g_{ij}^{(t)}} .$$

Для того чтобы получить итерационные уравнения для α_j, найдем первую производную (8):

$$\left(\sum_{j=1}^{k} \sum_{i=1}^{n} g_{ij}^{(t)} \ln \left(\frac{\alpha_j^{\alpha_j} x^{\alpha_j - 1}}{\lambda_j^{\alpha_j} \Gamma(\alpha_j)} e^{-(\alpha_j/\lambda_j)x} \right) \right)' \alpha_j =$$

$$= \left(\sum_{j=1}^{k} \sum_{i=1}^{n} g_{ij}^{(t)} \ln \left(\frac{\alpha_j^{\alpha_j}}{\lambda_j^{\alpha_j}} \right) - \ln \Gamma(\alpha_j) + \right.$$

$$\left. + (\alpha_j - 1) \ln X_i - \frac{\alpha_j}{\lambda_j} X_i \right)' \alpha_j =$$

$$= \sum_{i=1}^{n} g_{ij}^{(t)} \left(\ln \alpha_j + 1 - \ln \lambda_j - \frac{\Gamma'(\alpha_j)}{\Gamma(\alpha_j)} + \right.$$

$$\left. + \ln X_i - \frac{X_i}{\lambda_j} \right) ;$$

$$\sum_{i=1}^{n} g_{ij}^{(t)} \left(\ln \alpha_j + 1 - \ln \lambda_j - \right.$$

$$\left. - \frac{\Gamma'(\alpha_j)}{\Gamma(\alpha_j)} + \ln X_i - \frac{X_i}{\lambda_j} \right) = 0 ;$$

$$\frac{\Gamma'(\alpha_j)}{\Gamma(\alpha_j)} = \frac{\sum\limits_{i=1}^{n} g_{ij}^{(t)} \left(\ln \alpha_j + 1 - \ln \lambda_j + \ln X_i - \dfrac{X_i}{\lambda_j} \right)}{\sum\limits_{i=1}^{n} g_{ij}^{(t)}} . \qquad (9)$$

Здесь $\Gamma'(\alpha_j)/\Gamma(\alpha_j)$ — это *логарифмическая производная гамма-функции*. Для нее существует так называемое *разложение Абрамовитца–Стигана* [4]:

$$\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} = \log \alpha - \frac{1}{2\alpha} - \frac{1}{12\alpha^2} + \ldots$$

Подставим первые три члена разложения в (9) и разрешим это уравнение относительно α_j:

$$\alpha_{ij}^{(t+1)} = \frac{\sum\limits_{i=1}^{n} g_{ij}^{(t+1)}}{2 \sum\limits_{i=1}^{n} g_{ij}^{(t+1)} \left(\dfrac{X_i}{\lambda_j^{(t)}} - \ln \dfrac{X_i}{\lambda_j^{(t)}} - 1 \right)} .$$

В итоге получаем итерационные уравнения для α_j.

4 Описание программного обеспечения (программа EM)

4.1 Назначение программы

Разработанная авторами статьи программа EM предназначена для решения задачи разделения сме-

сей экспоненциальных и гамма-распределений, поставленной в разд. 2, с использованием EM-алгоритма и формул, описанных в разд. 3.

4.2 Инструменты разработки

Для создания программы была использована среда разработки Microsoft Visual Studio .NET 2005 и объектно-ориентированный язык C#. Для визуализации результатов была использована свободно распространяемая графическая библиотека ZedGraph [7].

4.3 Возможности программы

- Загрузка выборочных данных из текстового файла
- Оценивание по выборке параметров смеси экспоненциальных распределений
- Оценивание по выборке параметров смеси гамма-распределений
- Отслеживание изменений параметров смесей распределений во времени в режиме «скользящего окна»
- Построение гистограммы по выборке

4.4 Входные и выходные данные. Функционирование программы

В качестве *входных данных* программа EM получает:

- выборочные данные из текстового файла;
- число компонентов смеси;
- размер «скользящего окна»;
- размер класса гистограммы.

На *выходе* мы получаем:

- точечные оценки параметров смеси экспоненциальных распределений;
- точечные оценки параметров смеси гамма-распределений;
- графическое изображение результирующей смеси распределения;
- графическое изображение компонентов каждой смеси;
- графическое изображение того, как меняются параметры смесей распределений с течением времени в режиме «скользящего окна»;
- гистограмма, построенная по выборке;
- значение статистического теста.

Выборочные данные загружаются из текстового файла в память программы и подаются на вход двум независимо работающим реализациям EM-алгоритма — для идентификации смеси экспоненциальных распределений и для идентификации смеси гамма-распределений. Результатом их работы являются наборы значений оцениваемых параметров модели, предложенной в разд. 2. Кроме того, результирующие распределения визуализируются в виде графиков. В программе можно запустить режим «скользящего окна», который для всех подвыборок заданного размера с помощью EM-алгоритма оценивает параметры смесей распределений этих подвыборок. Все действия программы документируются в окне информации.

5 Описание тестовых расчетов

С использованием разработанной программы были проведены тестовые расчеты на выборочных данных реального сетевого трафика.

На вход программы EM были поданы выборки трафика:

I Между лабораторией Lawrence Berkeley (Berkeley, California) и внешним миром размера примерно 7000 [8] — *выборка 1.*

II Сети радиодоступа ЗАО «Синтерра» размера примерно 1000 [9] — *выборка 2.*

5.1 Выборка 1 “Berkeley”

При числе компонентов смеси 5 и случайном начальном приближении были получены результаты, представленные в табл. 1.

Данные результаты иллюстрирует рис. 5.

Гистограмма на рис. 6 показывает статистическую значимость полученных результатов.

Данная выборка обладает той особенностью, что она собиралась в течение достаточно длительного времени и в ней агрегирован самый разнородный трафик. Поэтому в ней присутствует не только большое количество «коротких» сообщений (что обычно для выборок из телетрафика), но и некоторый массив сообщений средней длины, а также определенный «выброс» больших сообщений. Это свидетельствует о *пиковости* телетрафика на довольно больших промежутках времени.

Как мы видим, EM-алгоритм удачно справился с задачей идентификации смеси.

Таблица 1 Результаты применения EM-алгоритма к выборке 1 “Berkeley”

№	Начальное приближение	Результат
	P	
0	0,2	0,1896
1	0,2	0,1858
2	0,2	0,1830
3	0,2	0,2259
4	0,2	0,2154
	α	
0	2,7028	10,9783
1	3,6273	5,8621
2	5,7598	2,7092
3	0,2315	1,0235
4	0,9110	0,4772
	λ	
0	85,2066	137,1714
1	23,9592	136,7349
2	63,8425	132,6482
3	1,8026	116,7317
4	98,3882	102,5278

Таблица 2 Результаты применения EM-алгоритма к выборке 2 “Synterra”

№	Начальное приближение	Результат
	P	
0	0,2	0,3815
1	0,2	0,3594
2	0,2	0,2589
3	0,2	$0{,}4401 \cdot 10^{-9}$
4	0,2	0,0
	α	
0	6,0804	1,5833
1	3,1838	0,8554
2	1,4886	0,4557
3	4,6407	0,2278
4	3,7843	0,1139
	λ	
0	17,3387	15,8682
1	47,8294	16,9150
2	54,1984	19,2866
3	8,6254	19,2866
4	5,7252	19,2866

5.2 Выборка 2 “Synterra”

Результаты применения EM-алгоритма к выборке “Synterra” представлены в табл. 2.

Данные результаты иллюстрируют рис. 7.

Эти результаты также отражают действительную картину, как показано на рис. 8.

Этот трафик был снят с базовой станции «Лукойл-Юго-Запад» сети широкополосного радиодоступа ЗАО «Синтерра». Сеть радиодоступа является реализацией так называемой «последней мили», переносящей два разных вида трафика: данные (Ethernet пакеты) и голос (IP-телефония, VoIP). Поэтому здесь присутствуют в качестве основной массы короткие, но интенсивные сообщения (пакеты SIP и голосовые фреймы), а также длинные сообщения, содержащие данные.

Как мы видим, программная реализация EM-алгоритма успешно справилась с задачей разделения смесей распределений для этих двух выборок, что делает данную программу удобным инструментом построения стохастической картины конкретной сети. По полученным данным, используя метод интерпретации, предложенный в разд. 2, можно получить представление о количестве последовательных и параллельных структур вероятностной модели сети.

5.3 Режим «скользящего окна»

Результаты для выборки “Berkeley” в режиме «скользящего окна» представлены на рис. 9.

Данные графики показывают изменение параметров распределений подвыборок выборки “Berkeley”. Видно, что параметры распределений подвыборок не остаются неизменными во времени, наоборот, они имеют внешне случайный характер. На рис. 9, *в* видна даже своеобразная пульсация первой компоненты. На основании расчетов можно сделать вывод о том, что пиковость трафика обусловливается как формой, так и интенсивностью сообщений.

6 Заключение

В данной работе исследована вероятностная модель информационных потоков, возникающих в сложных телекоммуникационных конвергентных сетях, построенная с помощью асимптотического и энтропийного подходов. Эта модель предполагает, что функционирование сложной телекоммуникационной сети можно представить в виде суперпозиции довольно простых стохастических структур — последовательных и параллельных, которые порождают смеси гамма-распределений для случайной величины времени обработки и передачи сообщений в сети. Предложена простая интерпретация параметров данной модели.

Для решения вытекающей из модели задачи предложен итерационный алгоритм, базирующийся на методе максимального правдоподобия — EM-алгоритм, для которого получены формулы для конкретного вида смесей — экспоненциальных и

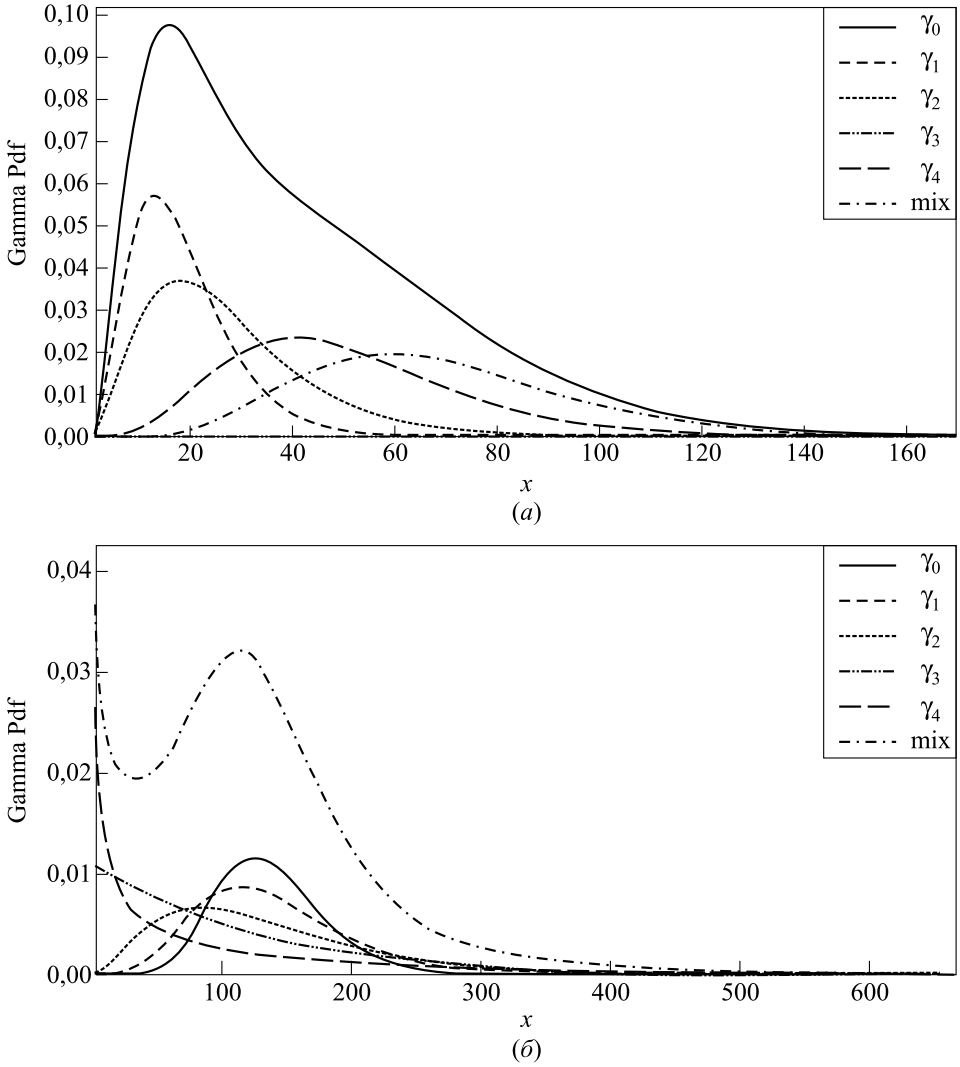

Рис. 5 Компоненты смеси начального приближения (*а*) и результата (*б*) для выборки 1 “Berkeley”

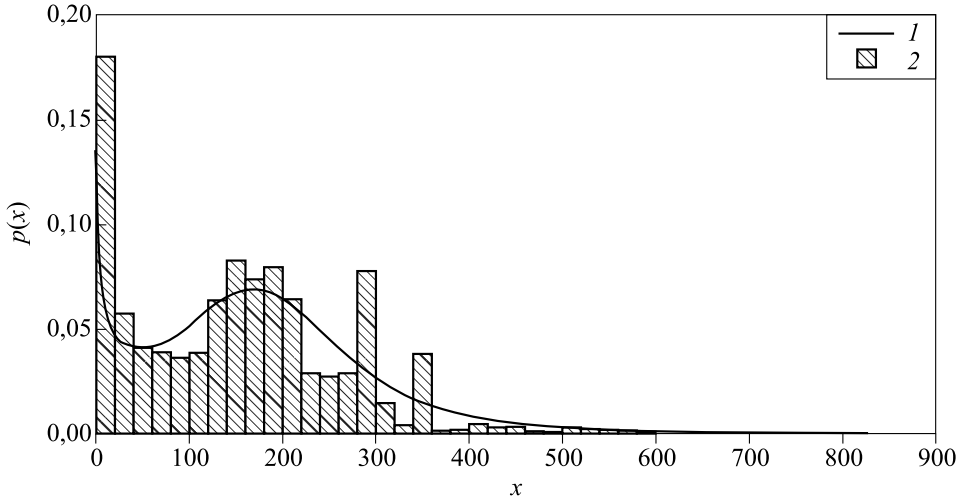

Рис. 6 График смеси распределений (*1*) и гистограмма для выборки 1 “Berkeley” (*2*)

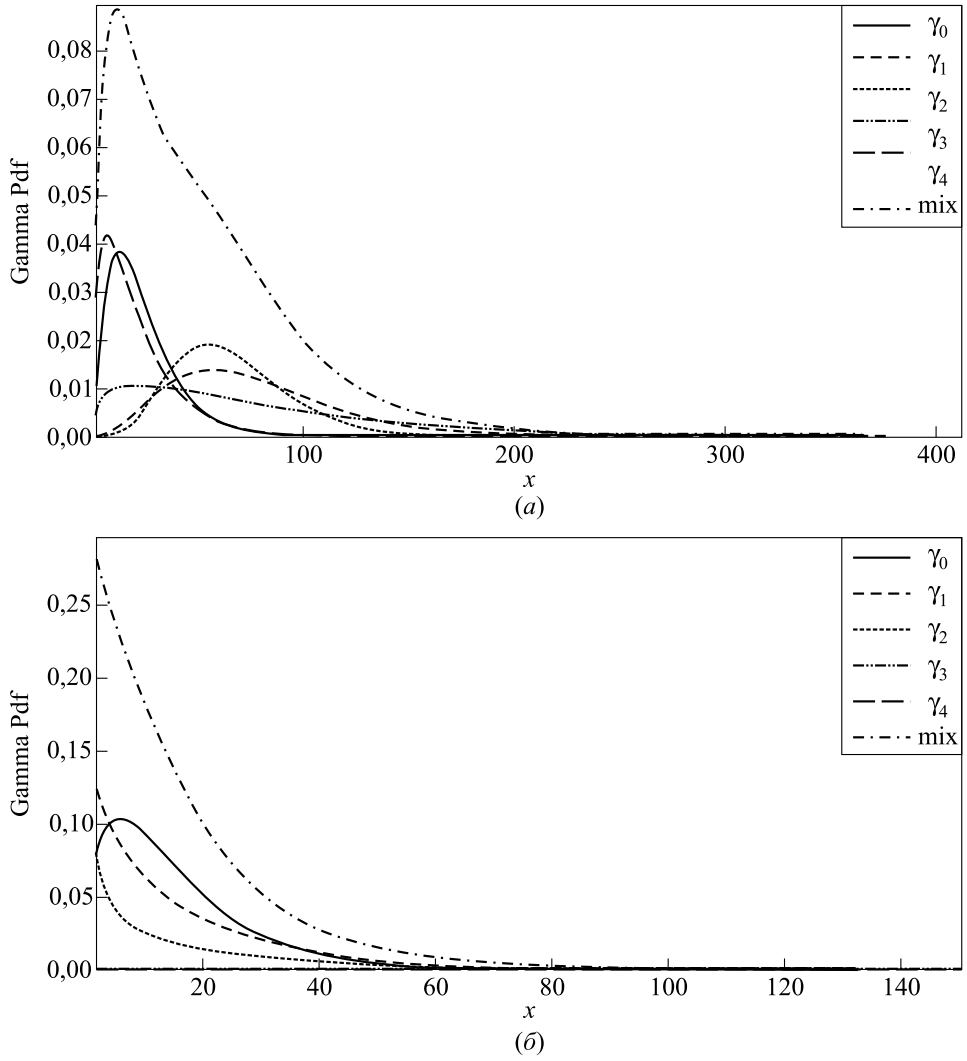

Рис. 7 Компоненты смеси начального приближения (*а*) и результата (*б*) для выборки 2 “Synterra”

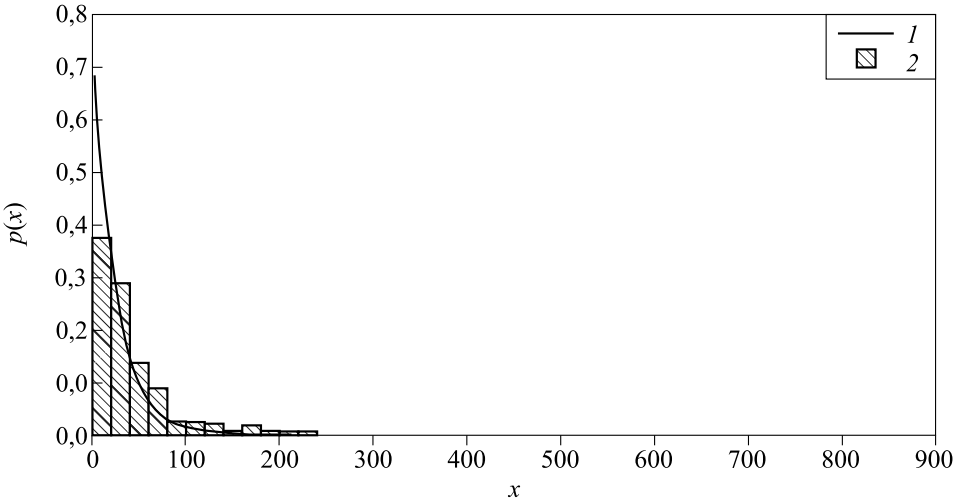

Рис. 8 График смеси распределений (*1*) и гистограмма для выборки 2 “Synterra” (*2*)

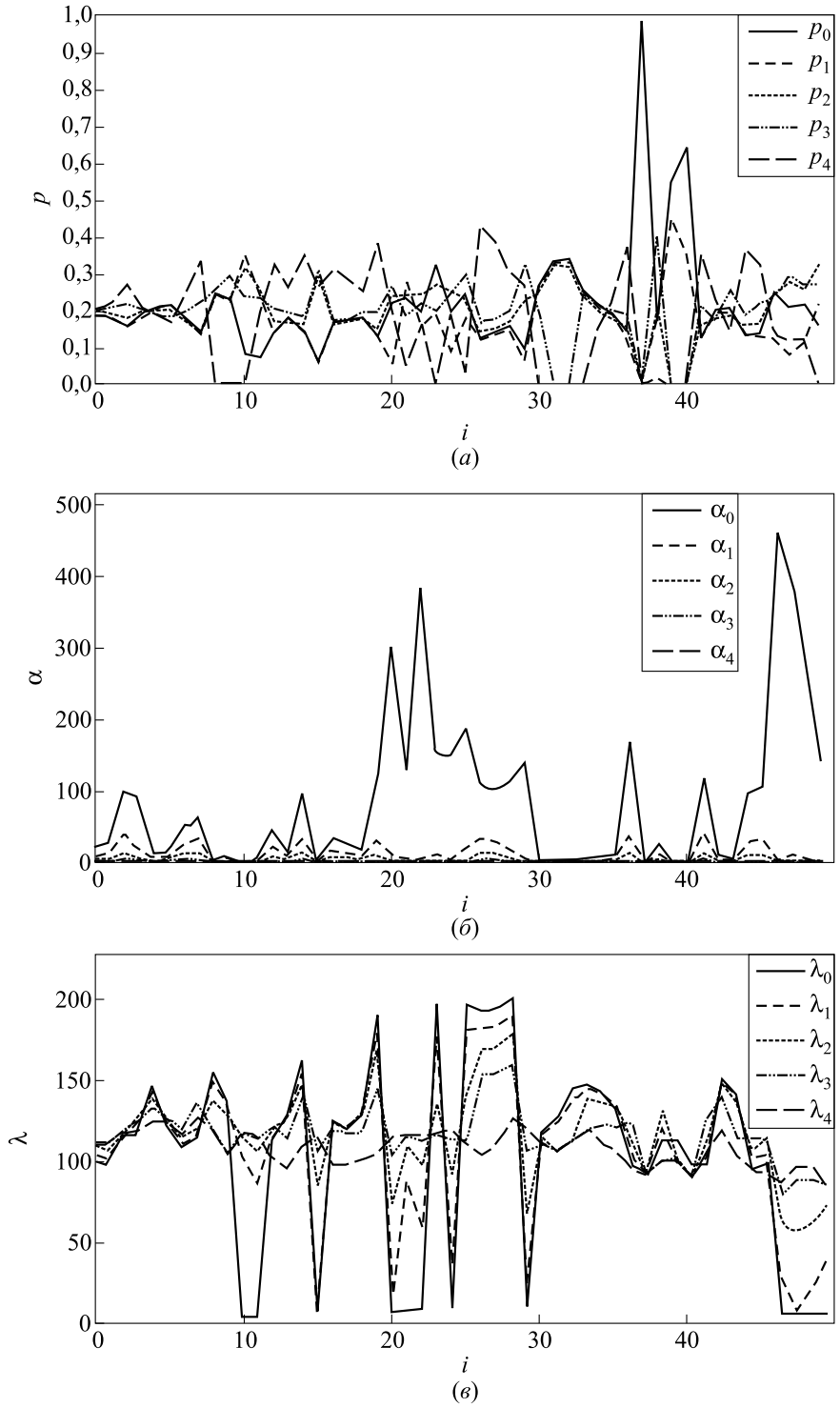

Рис. 9 Изменение смешивающих параметров (*а*), параметров формы (*б*) и параметров масштаба (*в*) во времени для выборки 1 “Berkeley”

гамма-распределений. Кроме того, разработан программный инструментарий для оценки параметров предложенной модели на выборках из реальных трафиковых данных. Проведены исследования, которые подтвердили предположения вероятностной модели.

Получение информации о стохастической структуре телекоммуникационных сетей и наличие программных инструментов для выявления более или менее стабильных структур позволит понять причины возникновения неожиданных больших нагрузок, предотвратить такие нагрузки, а также поможет в будущем в проектировании надежных, оптимальных по стоимости и уровню сервиса телекоммуникационных сетей нового поколения.

Литература

1. Teletraffic Engeneering Handbook. International Telecommunication Union, Geneva, 2005 http://www.itu.int.
2. *Севастьянов Б. А.* Курс теории вероятностей и математической статистики. М., 2004.
3. *Айвазян С. А., Бухштабер В. М., Енюков И. С, Мешалкин Л. Д.* Прикладная статистика. Классификация и снижение размерности // Финансы и статистика. М., 1989.
4. *Bilmes J. A.* A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Berkeley, CA, USA: International Computer Science Institute, 1998.
5. *Шлезингер М. И.* О самопроизвольном различении образов // Шлезингер М. И. Читающие. автоматы. Киев: Наукова думка, 1965. С. 38–45.
6. *Hsiao I.-T., Rangarajan A., Gindi G.*. Joint-MAP reconstruction/segmentation for transmission tomography using mixture-models as priors. Yale University, 1998.
7. http://zedgraph.org.
8. http://ita.ee.lbl.gov/html/contrib/LBL-PKT.html.
9. http://www.synterra.ru.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007, т. 1, №1: 54–65

ЛИНГВИСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ДЛЯ СИСТЕМ МАШИННОГО ПЕРЕВОДА И ОБРАБОТКИ ЗНАНИЙ

Е. Б. Козеренко[1]

Аннотация: Данная работа посвящена актуальным проблемам создания семантико-синтаксических представлений для систем машинного перевода и извлечения знаний из естественно-языковых текстов. Целью наших исследований является построение целостной лингвистической модели на основе синергетического подхода, использующего лингвистические знания, статистические методы и механизмы машинного обучения для извлечения новых грамматических правил из текстовых корпусов и разрешения неоднозначности. Для формализации лингвистических знаний используется когнитивная трансферная грамматика (КТГ), являющаяся семантически мотивированным вариантом унификационно-порождающей грамматики. Для подготовки обучающих компонентов систем и получения статистических данных о языковых структурах создается многоязычный лингвистический ресурс, представляющий собой банк синтаксических деревьев (Treebank) и корпус семантически выровненных параллельных текстов на русском, английском и ряде других европейских языков.

Ключевые слова: машинный перевод; формальные грамматики; лингвистическая модель; выравнивание параллельных текстов; семантика; синтаксис; языковые структуры

1 Введение

Современный период развития исследований и разработок в области машинного перевода и систем извлечения знаний из текстов характеризуется интенсивным процессом «гибридизации» подходов и моделей. Потребность в этом носит объективный характер. Значительные вычислительные ресурсы современных систем позволяют накапливать и использовать ранее переведенные текстовые фрагменты, обеспечивать машинный перевод, основанный на прецедентах (“Example-Based Machine Translation”) [1–3], эффективно поддерживать компоненту «переводческой памяти» (“Translation Memory”) [4–6].

Создатели систем, основанных на правилах, вводят в правила различные стохастические модели, которые позволяют отобразить динамику и разнообразие языковых форм и значений, порождаемых в процессе речевой деятельности, а сторонники статистических методов построения лингвистических моделей все чаще обращаются к подходам, основанным на лингвистических знаниях, рассматривая это как средства «интеллектуализации» систем [4–8].

Машинный перевод и извлечение знаний из текстов — это разные задачи, каждая из которых задает свойственный именно ей фокус внимания к лингвистическим объектам. Извлечение знаний (фактографической информации) из текстов и построение информационных систем поддержки аналитических решений (ИСПАР) требует проработки лексико-семантических представлений, создания развитых тезаурусов и онтологий, предметно-ориентированных семантических словарей.

Для машинного перевода наиболее сложной проблемой является реализация языковых трансформаций, которые необходимо производить при переводе с одного языка на другой. Текущий этап развития систем машинного перевода характеризуется исследованиями в области когнитивной семантики, вероятностных языковых моделей и разработкой семантико-синтаксических представлений, учитывающих многозначность и неоднозначность синтаксических структур. Новое содержание проблеме языковых трансформаций придают современные реалии: необходимость проектировать и развивать обучающие компоненты систем машинного перевода и обработки текстовых знаний на основе уже существующих и вновь создающихся корпусов параллельных текстов.

При создании компьютерных моделей русской грамматики доминировали подходы, основанные на грамматиках зависимостей [9] или локально-синтаксических представлениях, при этом не разрабатывались грамматики составляющих. Основ-

[1]ИПИ РАН, kozerenko@mail.ru

ные причины этого — как объективные: русский язык характеризуется относительно свободным порядком слов в предложении, так и субъективные: точка зрения на Хомскианские грамматики как на грамматики, применимые только для английского языка и других языков с фиксированным порядком слов.

Предлагаемый нами подход дает возможность компактного представления структуры составляющих предложения (грамматика фразовых структур), с одной стороны, а с другой стороны — учитывает механизмы зависимости между узлами дерева предложения. Эта система формальной грамматики, получившая название КТГ, дает возможность строить такие алгоритмизируемые представления, которые не ведут к экспоненциальному росту правил и вычислительных затрат.

2 Когнитивная трансферная грамматика

Основу КТГ составляют прототипические структуры исследуемых языков (в исходной модели — русского и английского), их наиболее вероятные позиции в предложении, статистические данные о дистрибутивных характеристиках структур (т.е. информация о контекстных условиях употребления исследуемых объектов — о структурных контекстах), схемы полного разбора предложений.

Создание и развитие КТГ предполагает:

- семантический подход к анализу языкового значения и языковой формы (форм);
- построение формально-грамматических представлений с учетом структур составляющих и механизмов линеаризации, а также отношений зависимости между узлами синтаксического дерева (подход, имеющий черты сходства с HPSG [10]: наследование признаков через головные вершины фразовых структур);
- включение вероятностных характеристик языковых объектов;
- создание пространств когнитивного трансфера (ПКТ), сформированных в виде экспертных лингвистических правил и расширяемых посредством выявления изосемичных языковых структур в параллельных текстах на различных языках.

В отличие от подходов на основе «переводческой памяти» (“Translation Memory”), реализующих возможность наращивания языковой компетенции системы за счет ранее переведенных текстовых фрагментов и в значительной степени основанных на аппарате регулярных выражений, КТГ предназначена для реализации механизма ***структурной памяти***, который моделирует языковую компетенцию взрослого обучаемого (“Adult Learning Memory”). Таким образом, *структурная память* предполагает следующие компоненты:

(1) исходный базовый набор грамматических правил, представленных в формализованном виде (КТГ);

(2) механизмы расширения и уточнения системы правил, которые реализуются посредством методов машинного обучения на параллельных текстах.

Наши исследования в значительной степени базируются на концепциях функционального подхода, который был применен нами для многоязычной ситуации. При разработке лингвистического процессора, обеспечивающего англо-русский и русско-английский трансфер, нами было предложено понятие полей функционального переноса (ПФП) [11, 12], явившихся базисом сегментации языковых структур для решения задач машинного перевода. Основная идея такого поля состоит в принятии гипотезы о том, что в основе грамматических структур лежат структуры когнитивные (ментальные фреймы); функционально-семантическое поле отражает взаимодействие элементов разных языковых уровней [12]. В результате формализации описания синтаксических и семантических свойств полей функционального переноса была разработана грамматика когнитивного переноса, или КТГ [11]. В настоящее время понятие полей функционального переноса расширено для многоязычной ситуации, и мы используем понятие ПКТ. Основной конструктивной единицей ПКТ является *трансфема*.

Трансфема — это единица когнитивного переноса, т.е. переводимое смысловое целое в единстве своих категориальных и функциональных характеристик, устанавливающая семантическое соответствие между языковыми структурами, принадлежащими различным языковым уровням и языковым системам.

Типы трансфем определяются ***рангом трансфемы***:

- ранг 1 — лексема как структурный знак, т.е. слово, рассматриваемое как категориально-функциональная единица без учета конкретного лексического значения данного слова;
- ранг 2 — словосочетание, т.е. синтаксическая структура, состоящая из двух и более синтаксически связанных слов, но не являющаяся законченным предложением или клаузой;

– ранг 3 — клауза, т.е. зависимое (придаточное) предложение;

– ранг 4 — независимое предложение (простое предложение или главное предложение сложноподчиненного предложения);

– ранг 5 — рассеянная структура, т.е. группа слов, обладающая синтактико-семантическим единством, но не расположенная контактно, т.е. между членами группы появляются другие языковые объекты, не являющиеся членами этой группы;

– ранг 0 — морфологические единицы, не являющиеся самостоятельными словами, а входящие в состав лексем исходного языка, которые на языке перевода могут быть выражены единицами других рангов, например: суффикс → клауза (*implementable → который может быть реализован*).

Построена целостная система межъязыковых соответствий русского и английского языков как модуль синтактико-семантических структурных моделей для лингвистического процессора автоматического перевода методом трансфера [11, 12].

3 Функциональный подход к лингвистическому моделированию

Разработка понятия функции, являющегося центральным в функциональной грамматике, связана с широкой проблематикой функций языка [13–21]. Функции связаны со значениями языковых единиц, но они не тождественны им. Исследование функции некоторой языковой формы включает анализ ее значения (или ряда значений в случае многозначности).

На современном этапе лингвистических исследований и разработок необходимо синергетическое сочетание функционального и уровневого подходов. Функциональный подход интегрирует разноуровневые языковые средства (синтаксические, лексические, словообразовательные и словоизменительные) на основе их функционально-семантических характеристик.

Отношение между понятиями *категория* и *функция* можно иначе выразить как *исходное структурное значение* и *реализуемое структурное значение*. Подобная концепция была высказана Леонардом Блумфилдом в его работе «Язык». Так, с точки зрения Л. Блумфилда, каждая лексическая форма связана с грамматическими формами в двух направлениях. С одной стороны, лексическая форма, даже когда она взята сама по себе, абстрагированно, обнаруживает значимую грамматическую *структуру*. С другой стороны, лексическая форма в любом конкретном высказывании, являясь особой языковой формой, всегда сопровождается той или иной грамматической формой. Она выступает в определенной функции, и случаи, в которых преимущественно данная лексическая форма встречается, составляют в совокупности ее грамматическую функцию. Лексические формы, выполняющие какие-либо общие функции, принадлежат к одному формальному классу. На основе различных функций могут возникать частично совпадающие формальные классы. Так, выполнение функции действующего лица характерно для субстантивных выражений и для типично инфинитивных словосочетаний (*to scold the boys would be foolish* «бранить мальчиков было бы глупо») [22].

Здесь Блумфилд называет функцией действующего лица субъектную функцию, тем самым подчеркивая ее «именной» характер. То есть функциональная потенция «быть субъектом предложения» исходно заложена в категории имени существительного. Глагольная же единица (в данном случае инфинитив), исполняя функцию, свойственную имени, как бы «надевает» на себя новую — именную форму — поверх своей исходной, глагольной.

Таким образом, получается *композиционная категория* Глагол–Имя, или VerbNoun в нотации КТГ [12].

Грамматика данного типа рассматривает в единой системе средства, относящиеся к разным языковым уровням, но объединенные на основе их семантических функций; при описании языкового материала используется подход «от семантики к ее формальному выражению» («от функций к средствам») как основной, определяющий построение грамматики, в сочетании с подходом «от формы к семантике» («от средств к функции»). Под единицами строя языка подразумеваются, прежде всего, грамматические формы слова и синтаксические конструкции, а также единицы «строевой лексики» (по Л. В. Щербе) [23]: модальные и фразовые глаголы, слова типа *вчера, обычно, часто, прежде, долго* и т.п.

Степанов Ю. С. [21] вводит понятие функтора как языкового средства транспозиции одного множества языковых единиц в другое множество языковых единиц того же языка. Функция есть свойство или значение функтора. Приводится пример функции: если принять за исходное множество единиц русские глаголы типа *сообщать*, *выражать*, *исполнять* и т.п., а за производное множество единиц *со-*

общение, *выражение*, *исполнение* и нечто *сообщается*, *исполняется*, то отношение первого множества ко второму будет функцией, а языковые формы *-ение, нечто — -(ает)ся* будут языковыми средствами этой функции, функторами.

Понятие функции является одним из центральных в коммуникативной грамматике Г. А. Золотовой. Функция — это предназначенность элемента к определенному способу существования в системе, к определенному служению этой системе [24]. Если за целое принимаем предложение в его коммуникативном назначении, то функции его элементов, его составных частей определяются как их строительные, комбинаторные потенции, реализуемые в построении предложения.

Функции реализуются при взаимодействии языковых объектов и их контекстов.

Для рассмотрения семантики способов конфигурирования языковых структур мы пользуемся понятием структурного знака [25], предложенным в семиотической лингвистике С. К. Шаумяном. При этом слово также рассматривается нами не с точки зрения его лексического значения, а как функционально-категориальная единица, минимальный структурный знак. Такой подход принимается нами как определенный этап исследований структурных знаков, продиктованный необходимостью максимально полного извлечения семантической информации из возможных способов конфигурирования языковых объектов и изучения когнитивных механизмов линеаризации языковых структур.

Семиотическая лингвистика вводит понятие *суперпозиции функций*, полагая, что каждый языковой объект обладает исходной *первичной* функцией, а происходящие в действующем языке сдвиги значений — это наложение вторичной и других функций на исходную. Таким образом, использование инструмента суперпозиции категорий [26, 27] дает возможность выразить функциональные свойства языковых объектов.

4 Изофункциональные трансформации при переводе

Функциональный подход, исследующий отношения «функциональной синонимии» разнородных и разноуровневых единиц языка, чрезвычайно актуален в настоящий момент, когда проводятся эксперименты по выявлению изофункциональных и изосемичных языковых структур из параллельных текстовых корпусов. Именно этот подход позволяет найти соответствия в текстах на разных языках. В самом деле, заранее нельзя с полной достоверностью определить, каким именно образом была переведена та или иная языковая структура в текстовом корпусе. Поэтому необходимо строить и исследовать различные гипотезы при проектировании лингвистического процессора.

Отсутствие полного совпадения между английскими и русскими языковыми конструкциями в научно-технических текстах можно обнаружить при изучении сравнительной частоты употребления в них отдельных частей речи, что важно для построения систем перевода, использующих машинное обучение.

Для научного изложения в целом характерен признак номинативности, т.е. более широкое использование существительных, чем в других функциональных стилях. При этом сопоставительный анализ переводов показывает, что в русском языке эта тенденция выражена более четко, и при переводе английские глаголы нередко заменяются существительными. Проведенные нами статистические исследования параллельных текстов позволяют сделать вывод о том, что русский текст приблизительно на 35% более номинативен, чем английский. Рассмотрим следующие примеры глагольно-именных трансформаций при англо-русском переводе.

(1) *The fuel system is designed to store liquid gasoline and to deliver it to the engine cylinders in the form of vapor mixed with air.*

Система питания предназначается для заправки жидким топливом и подачи его в цилиндры в виде смеси паров бензина с воздухом.

to store and to deliver → для заправки и подачи

(2) *A similar approach has marked the EU's efforts to expand the current club of 15 countries to embrace former communist countries further east.*

Точно таким же подходом характеризуются усилия ЕС по расширению нынешнего клуба 15 стран дальше на восток путем присоединения к нему бывших коммунистических стран.

to embrace → по расширению

Нами были проведены исследования на материале имеющихся в нашем распоряжении параллельных переводов научных статей и отдельно взятых примеров высказываний с исследуемыми конструкциями, а также мы обращались к опросу экспертов-переводчиков. Наиболее продуктивные типы глагольно-именных трансформаций при англо-русском переводе коррелируют со следующими функциональными значениями.

Обстоятельства цели и следствия, выраженные инфинитивом (58% в письменных переводах и 71%

при опросе респондентов – профессиональных переводчиков):

(3) *In order to understand the phenomenon, one should consider the laws of motion.*

Для понимания этого явления надо рассмотреть законы движения.

In order to understand → Для понимания

Составное сказуемое с инфинитивом (be + инфинитив) (51% в письменных переводах и 59% — при опросе респондентов – профессиональных переводчиков):

(4) *The difficulty will be to obtain the substance in question.*

Трудность будет состоять в получении рассматриваемого вещества.

to obtain → в получении

Инфинитив после относительных местоимений which и whom с предшествующим предлогом часто переводится отглагольным существительным с предлогом *для*; в этом случае относительное местоимение с предлогом не переводится (48% в письменных переводах и 52% при опросе респондентов – профессиональных переводчиков):

(5) *In vacuum, molecules have large space in which to move.*

В вакууме молекулы имеют большое пространство для движения.

in which to move → для движения

Адъективные трансформации инфинитива (существительное + инфинитив в определительной функции) (практически 100% в обоих случаях):

(6) *The amount of polonium to be obtained from a uranium mineral can be simply calculated.*

Количество полония, которое должно быть получено из урана, можно довольно просто подсчитать.

to be obtained → которое должно быть получено

Инфинитив в функции второго дополнения (глаголы *cause*, *get*, *lead*, *make* + инфинитив) (42% в письменных переводах и 58% при опросе респондентов – профессиональных переводчиков):

(7) *Pressure causes ice to melt.*

Давление приводит к таянию снега.

to melt → к таянию

Инфинитив, стоящий в начале предложения и выполняющий функцию подлежащего (31% в письменных переводах и 44% при опросе респондентов – профессиональных переводчиков):

(8) *To define exactly what is meant by the total heat in a body is at present still not possible.*

Точное определение того, что имеется в виду под общим нагревом тела, в настоящий момент все еще невозможно.

To define exactly → Точное определение

Оборот «for + существительное + инфинитив» выполняет функции различных членов предложения (в научной литературе чаще всего функции обстоятельства цели или следствия) — 32% в письменных переводах и 43% при опросе респондентов – профессиональных переводчиков:

(9) *The temperature was too low for the substance to decompose.*

Температура была слишком низкой для разложения этого вещества.

to decompose → для разложения

Система правил трансфера для англо-русского машинного перевода вначале была построена по принципу одновариантных правил, когда соответствие подбиралось как наиболее широкий способ перевода некоторой конструкции, пусть не всегда совершенно грамматичный, однако же обеспечивающий «понятность» перевода в наибольшем числе случаев [10]. При этом подходе предпочтение отдавалось всегда тому варианту, который был по форме ближе всего к исходной английской конструкции: для того, чтобы избежать трансформаций при переводе, которые всегда приводят к появлению «шумов» и резкому увеличению вычислительных затрат и, соответственно, программистских усилий. Однако в настоящее время нами разработана система многовариантных правил трансфера фразовых структур, при этом функциональные значения языковых единиц отражены в расширенной системе категориально-функциональных атрибутов. Языковые структуры представлены в виде иерархии правил КТГ, которая является разновидностью унификационно-порождающей грамматики. Структуры атрибутов и значений и их преобразования задаются в виде контекстно-свободных и мягко контекстно-зависимых продукционных правил. Отношения зависимости реализуются через механизм головных вершин фразовых структур, а сами фразовые структуры задают линейные последовательности языковых объектов [28].

5 Многовариантный перевод и композиционные категории

Инструментом задания категориальных и функциональных значений языковых структур служат композиционные категории. Метод построения композиционных категорий позволяет реализовать суперпозицию категориальных значений языковых объектов для осуществления многовариантного перевода. Так, *герундий* мы относим к категории "*VerbNounIng*", инфинитив в субъектной функции — к "*toPlusInfinitiveSubj*", инфинитив в объектной функции — к "*toPlusInfinitiveObj*", русское деепричастие и английское Participle I в адвербиальной функции — к категории "*ParticipleAdv*", личные формы глагола — к "*VerbFinit*", используются также и другие категории:
{[Category: *VerbNounIng*]: *asking questions*};
{[Category: *toPlusInfinitiveSubj*]: *She* is known *to be a skilled typist*};
{[Category: *toPlusInfinitiveObj*]: *We feel them to be sensitive readers*}.

Разработаны многовариантные правила функционально-семантического переноса, посредством которых задаются наиболее вероятные способы перевода рассматриваемых языковых структур. Например, инфинитивные конструкции в функции обстоятельства цели/следствия могут переводиться следующим образом:

(10) *Hydrogen and oxygen unite to form water.*

Водород и кислород соединяются, образуя воду.

Водород и кислород соединяются и образуют воду.

Водород и кислород соединяются с образованием воды.

Схема многовариантного англо-русского трансфера инфинитивной конструкции *to form water* будет выглядеть следующим образом.

[Category: *VerbInf*] → {*to form water*}

OR

{[Category: *ParticipleAdv*]; {*образуя воду*}

[Category: *VerbFinit*]; {*образуют воду*}

[Category: *VerbNounIng*]} {*с образованием воды*}

В следующем примере инфинитивный оборот соотнесен с ранее стоящим наречием *too* и выполняет функцию обстоятельства следствия:

(11) *The temperature was too low for the substance to* decompose.

Температура была слишком низкой для разложения этого вещества.

Температура была слишком низка, для того чтобы вещество разложилось.

Температура была слишком низка, для того чтобы вещество могло разложиться.

Температура была слишком низка, и разложения вещества не произошло.

Схему трансфера данной инфинитивной конструкции можно представить в виде следующих правил:
[Category: *VerbInf*] → OR {[Category: *VerbNounIng*];
[Category: *VerbFinit*]; [Category: *VerbModalPhrase*];
[Category: *SentenceNeg*]}.

При осуществлении анализа предложений и последующей свертки сегментов конструкций в узлы для последующего трансфера этим узлам будут присваиваться соответствующие метки (*VerbNounIng*, *SentenceNeg* и т.д.), которые запускают необходимые процедуры трансформаций при переводе.

6 Исследование транскатегориальных переносов в существующих системах машинного перевода

Мы рассмотрели примеры перевода предложений с герундиальными и инфинитивными оборотами с английского языка на русский существующими машинными переводчиками. В большинстве систем доминирует подход, состоящий в том, чтобы выбирать эквивалент, наиболее близкий по форме к исходной языковой конструкции. Однако отчетливо прослеживается тенденция к включению трансформаций, даже если это не всегда пока еще улучшает качество перевода. Следует отметить, что имеет место большое сходство переводов в системах, основанных на совершенно различных принципах, что указывает на ситуацию интенсивной диффузии систем машинного перевода и различных подходов к лингвистическому моделированию. Рассмотрим примеры перевода предложений с герундиями в системах, которые в настоящее время лидируют в области англо-русского и русско-английского машинного перевода. Исходное предложение и его переводы, сделанные человеком-переводчиком:

(12) *He could not help joining the discussion.*

Он не мог не принять участия в обсуждении.

Он не мог не выступить в прениях.

Переводы предложения (12) существующими системами машинного перевода:
Он не смог помочь соединить обсуждение. (Babelfish)
Он не смог помогать соединению дискуссии. (ЭТАП)
Он не мог сдержать присоединение к обсуждению. (ПроМт)

Он не мог помочь присоединяясь к обсуждению. (CognitiveTroll)
Он не мог сдержать присоединение к обсуждению. (SDL)

7 Вероятностные методы грамматического разбора предложений и выравнивания параллельных текстов

Истоки стохастической исследовательской парадигмы находятся в проектах разработки алгоритмов распознавания речи, символов, исправления орфографии. Основным методом решения многих задач, в частности определения и разметки частей речи, вероятностного грамматического разбора, является правило Байеса. В архитектуре стохастических систем, в основном, используется алгоритм динамического программирования.

Машинное обучение в значительной степени основано на стохастической исследовательской парадигме. Алгоритмы обучения могут быть двух типов: неуправляемые и управляемые. Неуправляемый алгоритм должен вывести модель, пригодную для обобщения новых данных, которые ему ранее не предъявлялись, и этот вывод должен быть основан только на данных. Управляемый же алгоритм обучается на множестве правильных ответов на данные из обучающей выборки таким образом, что выведенная модель даст более точные решения. Целью машинного обучения является автоматический вывод модели для некоторой области на основе данных из этой области, таким образом, система, обучаемая, например, синтаксическим правилам, должна быть обеспечена базовым набором правил фразовых структур. В последнее время больше внимания исследователей было уделено построению N-граммов, отражающих сложности синтаксических и семантических структур [29, 30], применению N-граммов переменной длины [31], включению семантической информации в N-граммы: в работе [32] дается детальное описание подхода к созданию статистического машинного перевода, основанного на N-граммах двуязычных единиц, называемых «кортежами», а также четырех специальных атрибутных функций. Авторы иллюстрируют свой метод примерами переводов материалов заседаний Европарламента с английского языка на испанский и с испанского на английский.

Статистические методы обработки естественного языка расширяют схему основных существующих подходов к машинному переводу — прямого перевода, переноса (трансфера) и подхода на основе языка-посредника (интерлингвы) [33, 34]. Машинный перевод на основе статистики был впервые предложен в [35, 36].

Значения вероятностей для каждого возможного варианта грамматического разбора (т.е. развертывания нетерминального узла) вычисляются на основе частот встречаемости таких вариантов разбора в существующих текстовых корпусах с синтаксической разметкой (treebanks). Значения вероятностей для вариантов разбора могут быть также получены и в виде лингвистических экспертных оценок.

Для любой системы обработки естественного языка необходимо проектирование модуля определения и разметки частей речи (тэггера). Стохастические тэггеры появились в 1980-е годы. Их общая идея заключается в выборе наиболее вероятного тэга (т.е. частеречной метки) для данного слова. Чаще всего для вероятностных тэггеров используются Марковские модели. Так, например, для некоторого данного предложения или последовательности слов выбирается последовательность тэгов, которая максимизирует следующую формулу:

$$P(\textit{слово}|\textit{тэг}) * P(\textit{тэг}|\textit{предыдущие } n \textit{ тэгов}) .$$

Еще один подход к машинному обучению, основанный на правилах и стохастическом тэггировании (разметке частей речи), известен как обучение, основанное на трансформациях (Transformation-Based Learning, TBL). TBL — это метод управляемого обучения с использованием заранее размеченного обучающего корпуса.

Для вероятностного грамматического разбора применяются стохастические грамматики.

– Вероятностная контекстно-свободная грамматика, ее определение:

$$G = (N, T, P, S, D) .$$

Здесь N — это множество нетерминальных символов; T — множество терминальных символов; P — множество продукций вида $A \to b$, где A — это нетерминальный символ, b — это цепочка символов; S — специальный исходный символ; D — функция, приписывающая значения вероятности каждому правилу из множества P. Как получить необходимые данные для вероятностной контекстно-свободной грамматики? Один из путей — использование корпуса синтаксически размеченных предложений. Такой корпус называется банком синтаксических деревьев (treebank). Например, Penn Treebank [37] содержит деревья разбора для ряда текстовых корпусов (Brown Corpus, Switchboard

corpus). Если задан банк деревьев разбора, то вероятность каждой развертки некоторого нетерминального узла может быть вычислена путем подсчета числа раз, когда данная развертка встречается, с последующей нормализацией:

$$P(\alpha \to \beta|\alpha) = \frac{Count(\alpha \to \beta)}{\sum\limits_{\gamma} Count(\alpha \to \gamma)} = \\ = \frac{Count(\alpha \to \beta)}{Count(\alpha)}\,.$$

– Вероятностная грамматика замещения деревьев: является обобщением вероятностной контекстно-свободной грамматики, при этом более мощной стохастически, поскольку можно приписывать значения вероятности фрагментам или даже целым схемам разбора.

Статистические подходы к выравниванию параллельных текстов направлены на то, чтобы найти наиболее вероятный вариант выравнивания A для двух заданных параллельных текстов S и T:

$$\arg\max_A P(A|S,\,T) = \arg\max_A P(A,S,T)\,.$$

Для того чтобы оценить значения вероятностей, которые указаны в этом выражении, чаще всего применяются методы, которые представляют параллельные тексты в виде последовательности выравниваемых цепочек предложений ($B_1,\ldots,B_K$). При этом предполагается, что вероятность одной цепочки не зависит от вероятностей других цепочек, а зависит только от предложений в данной цепочке [38]. Тогда

$$P(A,S,T) \approx \prod_{k=1}^{K} P(B_k)\,.$$

Этот метод просто учитывает длину предложения на исходном языке и на языке перевода, измеренную в символах. Предполагается, что более длинное предложение одного языка будет соответствовать более длинному предложению другого языка. Такой подход дает вполне устойчивые результаты для сходных языков и буквального перевода.

Более тонкие механизмы сопоставления обеспечиваются методами лексического выравнивания. Так, в работе [39] представлен метод выравнивания посредством создания модели последовательного пословного перевода. Наилучшим результатом выравнивания будет тот, который максимизирует вероятность порождения корпуса при заданной модели перевода. Для выравнивания двух текстов S и T необходимо разбить их и представить в виде последовательности цепочек предложений. Цепочка содержит ноль или более предложений на каждом из языков, а последовательность цепочек покрывает весь корпус:

$$B_k = (S_{a_k},\ldots,S_{b_k};\; t_{c_k},\ldots,t_{d_k})\,.$$

Затем наиболее вероятное выравнивание $A = \\ = B_1,\ldots,B_{m_A}$ данного корпуса определяется следующим выражением (при этом цепочки предложений не зависят друг от друга):

$$\arg\max_A P(S,T,A) = \arg\max_A P(L)\prod_{k=1}^{m_A} P(B_k)\,,$$

где $P(L)$ означает вероятность того, что порождается выравнивание L цепочек. Модель перевода, используемая при этом подходе, предельно упрощена и не учитывает фактор порядка слов в предложении и возможность того, что слову в исходном тексте может соответствовать более, чем одно слово в тексте перевода. В этой модели используются цепочки слов, при этом они ограничены соответствиями 1:1, 0:1 и 1:0. Суть модели заключается в том, что, если некоторое слово обычно переводится словом другого языка, то вероятность соответствия цепочек слов 1:1 будет высокой — значительно выше, чем произведение вероятностей соответствий 1:0 и 0:1 цепочек слов, использующих это рассматриваемое слово. При этом программа выбирает наиболее вероятный вариант выравнивания.

Модель перевода, основанная на пословном выравнивании (например, русского и английского параллельных текстов) будет выглядеть следующим образом:

$$P(r|e) = \frac{1}{Z}\sum_{a_1=0}^{l}\ldots\sum_{a_m=0}^{l}\prod_{j=1}^{m} P(r_j|e_{a_j})\,,$$

где e — предложение на английском языке; l — длина e, выраженная в словах; r — предложение на русском языке; m — длина r; r_j — j-тое слово в r; a_j — позиция в e, с которой выравнивается r_j; $P(w_r|w_e)$ — ***вероятность перевода***, т.е. вероятность того, что мы увидим w_r в предложении на русском языке, если мы встретим соответствующее w_e в английском предложении; Z — константа нормализации.

Для конкретного выравнивания перемножаются m вероятностей переводов, при этом отдельные переводы не зависят один от другого. Так, например, если необходимо вычислить вероятность:

$$P(\textit{Анна}|\textit{Ann}) \times P(\textit{видела}|\textit{saw}) \times P(\textit{Париж}|\textit{Paris}),$$

для выравнивания (*Анна*|*Ann*), (*видела*|*saw*), (*Париж*|*Paris*) следует перемножить вероятности этих трех переводных соответствий. При этом для каждого выравнивания делаются два упрощающих допущения: каждое русское слово порождается ровно одним английским словом (которое может быть нулевым, т.е. отсутствовать), и порождение каждого русского слова не зависит от других порождаемых слов в русском предложении.

Однако выше описанный подход, основанный на пословном сопоставлении и никак не учитывающий связи между словами и фразами, не дает оптимальных результатов при выравнивании русскоязычных и англоязычных текстов, поскольку между этими языками имеют место определенные структурные различия и при переводе могут осуществляться значительные трансформации.

Если рассматриваемые языки структурно отличаются, применяются методы, ориентированные на привлечение грамматических знаний, например используются методы выравнивания по словам, относящимся к значимым частям речи [40]. При этом служебные слова не учитываются. Для использования этих методов необходимо произвести разметку параллельных текстов по частям речи.

На современном этапе развития исследований и разработок в области машинного перевода и обработки текстовых знаний все больше назревает потребность в подходах, основанных на лингвистических знаниях. Это также осознается и сторонниками статистических подходов. Так, например, в работе [41] описан двухступенчатый метод автоматического извлечения переводных шаблонов из параллельных текстов на английском и китайском языках. Этот метод основан на алгоритме индукции грамматики и алгоритме выравнивания с использованием скобочной трансдукционной грамматики. Однако сами авторы указывают, что, поскольку в данной модели не заложены предварительные синтаксические знания, грамматическая правильность результата не может быть гарантирована.

8 Многовариантная вероятностная модель когнитивного трансфера

8.1 Вероятностная грамматика замещения деревьев

Рассмотрим, каким образом значения вероятности используются в процессе грамматического разбора. Например, вероятностная контекстно-свободная грамматика (Probabilistic Context Free Grammar, PCFG) и вероятностная грамматика подстановки деревьев (Probabilistic Tree Substitution Grammar, PTSG) присваивают вероятность P каждому дереву разбора T (т.е. каждому деривату) предложения S. Эта информация является ключевой для разрешения неоднозначности синтаксических структур. Вероятность каждого возможного дерева разбора T определяется как произведение вероятностей всех правил r, используемых для развертывания каждого узла n в дереве разбора:

$$P(T,S) = \prod_{n \in T} p(r(n))\,.$$

Вероятность однозначного предложения (т.е. предложения, где нам не надо разрешать неоднозначность) равна вероятности единственного дерева разбора для этого предложения, т.е. $P(T,S) = P(T)$. Вероятность же неоднозначного предложения равна сумме вероятностей всех возможных деревьев разбора ($\tau(S)$) данного предложения:

$$P(S) = \sum_{T \in \tau(S)} P(T,S) = \sum_{T \in \tau(S)} P(T)\,.$$

Вероятность полного разбора предложения вычисляется с учетом категориальной информации для каждой головной вершины каждого узла. Пусть n — синтаксическая категория некоторого узла n, $h(n)$ — головная вершина узла n, а $m(n)$ — материнский узел для узла n. Таким образом, мы будем вычислять вероятность $p(r(n)|n,\, h(n))$. Для этого мы преобразовываем выражение (1) таким образом, что каждое правило становится обусловленным своей головной вершиной:

$$P(T,S) = \prod_{n \in T} p(r(n)|n,\, h(n)) \times p(h(n)|n,\, h(m(n)))\,.$$

В нашей системе грамматики функциональные значения языковых структур определяются категориальными значениями головных вершин. Вероятностные характеристики вводятся в правила унификационной грамматики в виде весов, присваиваемых деревьям разбора. Неоднозначные и многозначные синтаксические структуры учитываются в многовариантной грамматике когнитивного трансфера (переноса). Неоднозначность является коренным свойством естественного языка и вызывает основные затруднения при создании систем машинного перевода.

8.2 Многовариантные правила когнитивного трансфера

В разрабатываемой нами системе грамматики *функциональные* значения языковых структур опре-

деляются *категориальными значениями головных вершин*. Вероятностные характеристики вводятся в правила унификационной грамматики в виде весов, присваиваемых деревьям разбора. Неоднозначные и многозначные синтаксические структуры учитываются в многовариантной КТГ (МКТГ).

Синтаксис МКТГ-структуры (МКТГС) может быть представлен в общем виде следующим образом:
МКТГС → МКТГС *<идентификатор> МКТГС <вес> МКТГС <метка> <Входная фразовая структура & набор атрибутов-значений> <Схема трансфера, управляемого головной вершиной> <Генерируемая фразовая структура & набор атрибутов-значений 1> <вес1> <Генерируемая фразовая структура & набор атрибутов-значений 2> <вес2> . . . <Генерируемая фразовая структура & набор атрибутов-значений N > <весN >*.

В новом варианте МКТГ отражено явление многозначности синтаксических структур и предусмотрены некоторые механизмы разрешения неоднозначности посредством включения в систему правил разбора и перевода статистической информации о возможных контекстах языковых структур. Многовариантная грамматика когнитивного переноса обеспечивает расширяемую платформу для разработки систем машинного перевода и извлечения знаний из текста: может применяться для большего числа языков. Новый синтетический подход к лингвистическому моделированию для систем машинного перевода и обработки знаний предполагает семантическое выравнивание структур для ряда европейских языков; установление кросс-лингвистических пространств когнитивных соответствий; разрешение неоднозначности с использованием стохастических методов, подготовкой языковых структур и шаблонов для машинного обучения систем.

Продолжают формироваться обучающие наборы данных для расширения и модификации правил. Для сокращения числа избыточных правил (которые будут неизбежно возникать на этапе обучения) в систему закладываются не только шаблоны языковых структур, по аналогии с которыми будут порождаться правила, но и правила-фильтры на основе ПКТ.

9 Заключение

Актуальность проблемы создания новых гибридных методов представления языковых объектов обусловлена тем, что на современном этапе исследований встает задача оптимального сочетания сильных сторон двух исследовательских парадигм: логико-лингвистического моделирования, использующего правила, и стохастического подхода. Особое значение предлагаемая работа имеет для решения проблемы структурного анализа и компьютерного моделирования полнотекстовых научных и патентных документов. Необходимость моделирования языковых трансформаций для систем машинного перевода и извлечения знаний из текстов обусловлена тем, что до сих пор эти явления мало исследованы с точки зрения возможностей их компьютерных реализаций и, соответственно, недостаточно учтены в действующих системах машинного перевода, а правила, задающие функциональную синонимию языковых конструкций, позволяют извлечь необходимую информацию из параллельных текстов и избежать формирования избыточных правил и «шумов».

При формировании процедур снятия неоднозначности мы используем статистические данные, полученные при изучении параллельных текстов научных и патентных документов нашего экспериментального корпуса. Так, например, при ранжировании предпочтительности вариантов трансфера нами используется статистика соотношения «глагольности-номинативности» русского и английского научного дискурса.

Дальнейшие наши исследования связаны с расширением числа типов трансформаций в англо-русском и русско-английском переводах и построением лингвистических представлений для многоязычной ситуации. Разрабатывается многоязычная лингвистическая база знаний, которая основана на сочетании методов семантического анализа фразовых структур (с акцентом на функциональной семантике) и статистических характеристиках языковых объектов.

Кроме компьютерных реализаций полученные результаты используются для выработки инновационных подходов к обучению иностранным языкам на основе семантической грамматики и созданию образовательных программ по информационным технологиям и компьютерной лингвистике.

Литература

1. *Brown R. D.* Example-based machine translation in the Pangloss system // 16th Conference (International) on Computational Linguistics, 1996. Copenhagen: Center for Sprogteknologi. P. 169–174.
2. *Knight K.* Automating knowledge acquisition for machine translation // Artificial Intelligence Magazine, 1997. Vol. 18. P. 81–96.
3. *Козеренко Е. Б.* Логико-статистические методы представления языковых структур в машинном пере-

воде // Труды Международной конференции Диалог'2005 «Компьютерная лингвистика и интеллектуальные технологии». М.: Наука, 2005.

4. *Wang Ye-Yi, Waibel A.* Modelling with structures in statistical machine translation // 36th Annual Meeting of the Association for Computational Linguistics / 17th Conference (International) on Computational Linguistics, 1998. P. 1357–1363.
5. Hutchins J., Hartmann W., Ito E., eds. Compendium of translation software // 7th Ed. European Association for Machine Translation, 2003.
6. *Lagoudaki E.* Translation memories survey 2006: User's perceptions around TM usage // 28th Translating and the Computer 28 Conference Proceedings. London: Aslib/IMI, 2006. P. 1–29.
7. *Alshawi H., Buchsbaum A. L., Fei Xia*. A comparison of head transducers and transfer for a limited domain translation application // 35th Annual Meeting of the Association for Computational Linguistics / 8th European Chapter of the Association for Computational Linguistics, 1997. P. 360–365.
8. *Plana E.* SIMILIS second-generation translation memory software // 28th Translating and the Computer Conference Proceedings. London: Aslib/IMI, 2005.
9. *Mel'cuk I. A.* Dependency syntax: Theory and practice. Albany: State University of New York, 1988.
10. *Pollard C., Sag I. A.* Head-driven phrase structure grammar. Chicago: University of Chicago Press, 1994.
11. *Kozerenko E. B.* Cognitive approach to language structure segmentation for machine translation algorithms // International Conference on Machine Learning, Models, Technologies and Applications Proceedings. Las Vegas, USA: CSREA Press, 2003. P. 49–55.
12. *Козеренко Е. Б.* Моделирование переноса функциональных значений для англо-русского машинного перевода // Труды Международной конференции Диалог'2004 «Компьютерная лингвистика и интеллектуальные технологии». М.: Наука, 2004.
13. *Якобсон Р. О.* Разработка целевой модели языка в европейской лингвистике в период между двумя войнами // Новое в лингвистике. М., 1965. Вып. 4. С. 372–377.
14. *Якобсон Р. О.* Шифтеры, глагольные категории и русский глагол // Принципы типологического анализа языков различного строя. М., 1972.
15. *Halliday M. A. K.* System and function in language // In: Halliday M. A. K. Selected Papers. London, 1976.
16. *Звегинцев В. А.* Функция и цель в лингвистической теории // Проблемы теоретической и экспериментальной лингвистики. М., 1977. С. 120–146.
17. *Слюсарева Н. А.* Проблемы функционального синтаксиса современного английского языка. М., 1981. 206 с.
18. *Halliday M. A. K.* An introduction to functional grammar. Ed. Arnold E. London, 1985.
19. *Шведова Н. Ю.* Один из возможных путей построения функциональной грамматики русского языка // Проблемы функциональной грамматики. М., 1985. С. 30–37.
20. *Гак В. Г.* К типологии функциональных подходов к изучению языка // Проблемы функциональной грамматики. М., 1985. С. 5–15.
21. *Степанов Ю. С.* Имена, предикаты, предложения (семиологическая грамматика). М.: Едиториал УРСС, 2004.
22. *Блумфилд Л.* Язык. Изд. 2-е, стереотипное. М.: Едиториал УРСС, 2002.
23. *Щерба Л. В.* Языковая система и речевая деятельность. Л., 1974.
24. *Золотова Г. А., Онипенко Н. К., Сидорова М. Ю.* Коммуникативная грамматика русского языка. М., 2004.
25. *Шаумян С. К.* Семиотическая лингвистика как объяснительная наука // Труды Международной конференции Диалог'2005 «Компьютерная лингвистика и интеллектуальные технологии». М.: Наука, 2005. С. 507–513.
26. *Shaumyan S.* A semiotic theory of language. Indiana University Press, 1987.
27. *Shaumyan S.* Signs, mind, and reality. USA: John Benjamins Publishing Company, 2006.
28. *Козеренко Е. Б.* Лингвистические аспекты информатики // Системы и средства информатики. Специальный выпуск «Научно-методологические проблемы информатики». М.: ИПИРАН, 2006. С. 88–111.
29. *Rosenfeld R.* A maximum entropy approach to adaptive statistical language modeling // Computer Speech and Language, 1996. Vol. 10. P. 187–228.
30. *Niesler T. R., Woodland P. C.* Modelling word-pair relations in a category-based language model // IEEE Conference (International) on Acoustics, Speech, and Signal Processing, 1999. P. 795–798.
31. *Ney H., Essen U., Kneser R.* On structuring probabilistic dependencies in stochastic language modeling // Computer Speech and Language, 1994. Vol. 8. P. 1–38.
32. *Marino J. B., Banchs R. E., Crego J. M., Adria de Gispert, Lambert P., Fonollosa J. A. R., Costa-jussa M. R.* N-gram-based machine translation // Computational Linguistics, 2006. Vol. 32. No. 4. P. 527–549.
33. *Voss C., Dorr B. J.* Toward a lexicalized grammar for interlinguas // Machine Translation, 1995. Vol. 10. No. 1–2. P. 139–180.
34. *Dorr B., Nizar H.* Interlingua approximation: A generation-heavy approach // AMTA-2002 Interlingua Reliability Workshop. Tiburon, California, USA, 2002.
35. *Brown P. F., Cocke J., Della Pietra S. A., Della Pietra V. J., Jelinek F., Lafferty J. D., Mercer R. L., Roossin P. S.* A statistical approach to machine translation // Computational Linguistics, 1990. Vol. 16. P. 79–85.
36. *Brown P. F., Della Pietra S. A., Della Pietra V. J., Mercer R. L.* The mathematics of statistical machine translation: Parameter estimation // Computational Linguistics, 1993. Vol. 19. No. 2. P. 263–311.

37. *Marcus M. P., Santorini B., Marcinkiewicz M. A.* Building a large annotated corpus of English: The Penn Treebank // Computational Linguistics, 1993. Vol. 19. No. 2. P. 313–330.
38. *Gale W. A., Church K. W.* A program for aligning sentences in bilingual corpora // Computational Linguistics, 1993. Vol. 19. P. 75–102.
39. *Chen S. F.* Aligning sentences in bilingual corpora using lexical information // 31st Annual Meeting of the Association for Computational Linguistics, 1993. P. 9–16.
40. *Haruno M., Yamazaki T.* High-performance bilingual text alignment using statistical and dictionary information // 34th Annual Meeting of the Association for Computational Linguistics, 1996. P. 131–138.
41. *Hu Rile, Zong Chengqing, Xu Bo.* An approach to automatic acquisition of translation templates based on phrase structure extraction and alignment // IEEE Transactions on Audio, Speech and Language Processing, 2006. Vol. 14. No. 5. P. 1656–1663.

ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2007, т. 1, №1: 66–78

СИМВОЛЬНАЯ МОДЕЛЬ СИСТЕМЫ ЗНАНИЙ ИНФОРМАТИКИ В ЧЕЛОВЕКО-АВТОМАТНОЙ СРЕДЕ

В. Д. Ильин[1], И. А. Соколов[2]

Аннотация: Предложен подход к построению системы знаний информатики в человеко-автоматной среде как средству формализованного представления научных результатов. Изложены основы концепции методологии построения и применения этой системы. Методология изучается как научное основание технологий автоматизации научных исследований, проектирования и образовательных процессов.

Ключевые слова: информатика; символ; символьная модель; человеко-автоматная среда; система знаний; символьное моделирование в человеко-автоматной среде

1 Введение

Символьные модели в познании. Совершенствование инструментов познания связано с изобретением и применением символов и символьных моделей изучаемых объектов. При этом **символ** рассматривается как заменитель некоторого объекта, включая другие символы. В наши дни наиболее употребляемыми **типами символов** являются аудио, текстовые, графические и видео (например, в гипермедийных документах, выложенных на веб-сайтах). Моделируемые объекты могут иметь любую физическую сущность, размещение, происхождение и назначение. Это могут быть объекты, наблюдаемые человеком непосредственно или с помощью технических устройств, порожденные его разумом и построенные с помощью машин. Другими словами, объектами символьного моделирования могут быть **произвольные объекты**: проектирование машин, программирование или что-то еще.

Результаты символьного моделирования размещаются в памяти человека и во внешней среде (в частности, в памяти машин). Затраты на построение, копирование, передачу, сохранение и накопление символьных моделей, как правило, значительно меньше, чем аналогичные затраты, связанные с несимвольными моделями (например, макетами судов, зданий и др.).

Основное достоинство символьного моделирования заключается в том, что оно не только повышает продуктивность познания, но, что не менее важно, расширяет его пространство. Достаточно вспомнить, какое значение на заре символьного моделирования имело изобретение **графических моделей языков сообщений** (основания письменности). Это изобретение сыграло выдающуюся роль в формировании инструментария научного познания.

Символьные модели систем понятий, построенных на их основе систем знаний и других объектов, методы их построения, сохранения и передачи — все это вызывает неуклонно растущий интерес с тех давних пор, как люди убедились в преимуществах, которые дают способности видеть актуальные задачи, формулировать их и находить методы решения. Этот интерес привел к идее применения машин, помогающих решать задачи символьного моделирования. На пути от рукописных текстов, рисунков и схем к книгопечатанию и графическим моделям в проектировании, от звукозаписи, фотографии и радио к кино и телевидению, от компьютеров и локальных сетей к глобальной сети, виртуальным лабораториям и дистанционному образованию постоянно растет роль символьных моделей, которые человек создает с помощью машин. Сначала решаемые с помощью машин задачи были математическими и имели вычислительный характер (отсюда и название — вычислительная машина (computer)). За «деревьями» проблем построения и применения машин для решения задач не сразу удалось разглядеть их принадлежность «лесу» проблем символьного моделирования произвольных объектов в человеко-автоматной среде.

В наши дни трудно найти область деятельности, где бы не применялись результаты символьного моделирования в человеко-автоматной среде. Примеров удачной реализации символьных моделей в человеко-автоматной среде так много, что из них непросто выбрать. Это, конечно же, Интернет, электронная почта, Веб, САПРы и множество дру-

[1]ИПИ РАН, vdilyin@ipiran.ru
[2]ИПИ РАН, isokolov@ipiran.ru

гих, включая системы для игры в шахматы, лучшие из которых на равных соперничают с чемпионами мира.

Современный этап. На современном этапе в центре внимания — цифровое символьное моделирование. «Цифровое» — потому, что символы любого типа и построенные из них модели представлены в виде цифровых кодов, рассчитанных на манипулирование посредством электронных машин. В таких машинах основу аппаратной составляющей представляют микроэлектронные схемы, а цифровые коды реализуются такими схемами в виде двоичных кодов. Отсюда и слова «электронный» и «цифровой» в названиях понятий (электронный документ, электронная почта, цифровая технология и др.).

1.1 Цель исследований

Для науки особое значение имеют методологии символьного моделирования систем понятий и систем знаний [1]. Результаты изучения символьного моделирования в человеко-автоматной среде целесообразно представлять в виде символьных моделей систем понятий и систем знаний, удовлетворяющих требованиям реализации в человеко-автоматной среде.

В Институте проблем информатики РАН создается методология построения и применения **символьной модели системы знаний информатики**. Эта модель, получившая имя Синф (в формульных частях текста — Sinf), рассматривается как средство формализованного представления научных результатов, рассчитанных на применение **символьного моделирования произвольных объектов в человеко-автоматной среде**, причем не только исследователями (информатиками[1]). Методология построения и применения Синф изучается как основа технологий автоматизации научных исследований, образовательных процессов, проектирования и др. Результаты исследований, полученные при создании Синф-методологии, используются для научно-методологической поддержки создания Большой Российской Энциклопедии (части, отнесенной к разделу «Информатика») и в вузовском образовательном процессе (МИРЭА, базовая кафедра проблем информатики ИПИ РАН).

В статье представлены основы концепции создания Синф как гипермедийной системы знаний, имеющей сервис-ориентированную архитектуру.

1.2 Обозначения

Применяемые в статье обозначения введены для компактного представления часто повторяющихся устойчивых словосочетаний и выделения частей текста, к которым авторы хотят привлечь внимание читателя.

1. *Префикс s-* в сокращениях имен понятий введен как *обозначение принадлежности системе понятий символьного моделирования в человеко-автоматной среде*. Он может размещаться перед именем одного понятия (например, *s-моделирование*) или списком имен понятий, заключенных в скобки Например, s-(представление, конструирование, интерпретация). В определениях наиболее важных понятий сокращения не применяются.
2. К наиболее употребляемым сокращениям относятся:

   s-моделирование — символьное моделирование произвольных объектов в человеко-автоматной среде;

   s-модель — символьная модель произвольного объекта в человеко-автоматной среде;

   s-машина — машина для построения и применения s-моделей;

   s-среда — человеко-автоматная среда s-моделирования;

   Dn — обозначения;

   $\approx$ — заменяет.
3. Для выделения определений, замечаний, примеров, имен понятий и отдельных частей текста используются следующие средства:

   $\triangleright$ <текст> $\triangleleft$ — часть текста с фиксированными в ее пределах обозначениями, дополняющими приведенные здесь;

   $\square$ <текст> $\square$ — определение;

   $\diamond$ <текст> $\diamond$ — замечание;

   $\oplus$ <текст> $\oplus$ — пример;

   {S <*текст*><список> S} — здесь <*текст*> $\approx$ набранный *курсивом* текст (может быть пустым), который следует интерпретировать как расширенный префикс s-<*текст*> для выделенных *курсивом* элементов списка;

   $\oplus$ {S*модель*< список > S} — здесь расширенный префикс — *s-модель*; {S < список > S} — здесь префикс *s-*. $\oplus$

Разреженный шрифт, *курсив*, *разреженный курсив* применяются для названий понятий и частей текста, значение которых авторы выделяют.

[1] Информатика – информатик (так же, как математика – математик).

2 Символьное моделирование в человеко-автоматной среде: основные понятия

Изложение подхода к решению комплекса задач построения и применения системы знаний информатики в человеко-автоматной среде включает характеристику s-моделирования как объекта исследований, определение основных понятий, описание классов основных задач. Значительное место отведено мотивировке выбранного подхода и пояснениям предложенных определений.

С развитием техники, видимо, не могла не родиться идея решения задач с помощью автоматов. С тех пор, как эта идея стала постоянным стимулом для изобретателей машин, помогающих решать задачи, было предложено и построено немало автоматов такого назначения. Особое место среди них принадлежит *программируемым машинам со сменяемыми программами*. По мере роста производительности s-машин этого типа возрастал интерес к их применению. Настало время, когда в математике сформировался и стал интенсивно развиваться раздел вычислительной (машинной) математики. Изобретатели s-машин различных архитектур и программисты начального этапа s-моделирования, физики и инженеры, создававшие элементную базу, проектировавшие комплектующие изделия и периферийные устройства для первых s-машин — все они результатами доказали целесообразность серийного производства s-машин.

2.1 S-моделирование: новый тип символьного моделирования

Как инструмент познания этот тип символьного моделирования привлек внимание исследователей прежде всего своей высокой продуктивностью, не зависящей от природы моделируемых объектов. На определенном этапе стало ясно, что новая реальность, созданная людьми, заслуживает внимания организационно оформленных коллективов исследователей. Так, *изучение свойств и закономерностей символьного моделирования произвольных объектов в человеко-автоматной среде, состоящей из людей и управляемых ими машин для построения и применения символьных моделей*, привело к созданию новой науки.

Для краткости такое моделирование авторы назвали *s-моделированием*, среду его реализии — *s-средой*, получаемые модели — *s-моделями* [1], а машину для построения и применения символьных моделей — *s-машиной*.

Главной особенностью s-моделирования является то, что оно осуществляется с помощью s-машин, помогающих людям в построении и применении символьных моделей. Заметим, что и до того машины (печатные станки, типографские машины и др.) использовались людьми в символьном моделировании. Но это не была помощь в изобретении и конструировании символьных моделей. Такая помощь стала реальностью, когда появились программируемые машины со сменяемыми программами.

⊕ Исследования в области автоматизации программирования — пример изучения одного из типов s-моделирования, которым является программирование. ⊕

Продуктивность изучения свойств и закономерностей s-моделирования на каждом этапе развития информатики зависит от точности не только формулировки предмета исследований, но и основных классов задач, основы научного метода, содержания и формы представления научной продукции.

Какими свойствами обладает s-моделирование? Какие закономерности ему присущи? Как его типизировать? Что представляют собой s-модели произвольных объектов в s-среде, каковы их свойства, как строить такие модели и манипулировать ими, используя s-машины? От ответов на такие вопросы зависят ответы на более конкретные вопросы.

Какими должны быть правила конструирования s-моделей с помощью s-машин? Какой должна быть s-среда? Из поиска ответов на эти и связанные с ними вопросы состоит процесс исследования s-моделирования.

Какие задачи s-моделирования актуальны на современном этапе и как их решить? Какой должна стать s-среда, чтобы соответствовать требованиям к реализации задач s-моделирования ближайшего будущего? На разных этапах развития информатики ответы на поставленные вопросы будут отличаться.

Формализация. Особое место в развитии символьного моделирования принадлежит идее его формализации, заключающейся в том, чтобы строить символьные модели по определенным правилам из заранее определенных элементов. Эта идея реализована в математических методах символьного моделирования. Однако метод формализации [2], применяемый в математике для получения формальных систем [3], нельзя перенести на s-моделирование, так как s-модели не являются формальными системами. Объясним подробнее это важное замечание.

Задача в s-моделировании имеет более широкий смысл, чем в математике: рассматриваемые

в разд. 2.3 задачи s-(представления, распознавания, преобразования, конструирования, интерпретации, обмена, сохранения, накопления, поиска, защиты) не являются математическими, хотя при их решении применяются и математические методы. Но математический арсенал недостаточен для того, чтобы каждую из указанных задач можно было сформулировать и решить как математическую задачу. Дело здесь не только в том, что в классической математике царствует формальное доказательство (существования, единственности решения), а в s-моделировании — конструктивное доказательство (существования s-модели; а о единственности вообще речь не идет). Важно другое: неформальность s-моделей — это их полезное отличие, связанное с возможностью привлечения неформализованного знания человека-эксперта для управления ходом решения (примером может служить разработанная в ИПИ РАН методология интерактивного преобразования ресурсов [4]).

◇ *S-моделирование предполагает представление символов и построенных из них s-моделей в двух формах, одна из которых рассчитана на интерпретацию человеком, другая (в форме кодов символов и s-моделей) — на интерпретацию программой, выполняемой s-машиной.* Множество символов, применимых для построения s-моделей — это множество элементарных конструктивных объектов, каждый из которых наделен набором атрибутов и совокупностью допустимых операций. Построение конструкций из элементов этого множества определено системой правил конструирования s-моделей. ◇

2.2 Символы и коды в s-среде

Исходим из того, что произвольному объекту можно поставить в соответствие символ или символьную модель. Из этого следует, что и символу можно поставить в соответствие другой символ того же или другого типа. Символы, рассчитанные на s-машину, называют кодами, процесс преобразования символов в коды — кодированием, а обратный ему процесс — декодированием.

□ S-символ — это заменитель некоторого объекта, удовлетворяющий требованиям представления в s-среде. □

Не накладывается никаких ограничений на типы заменяемых объектов. В частности, одни символы могут служить заменителями других. Единственным ограничением, определяющим существование того или иного типа[1] символов, служит реализуемость этого типа в s-среде.

⊕ Примеры s-символов различных типов: буквы текста при работе с текстовым редактором (текстовый); гиперссылки веб-страницы (гипертекстовый); неподвижные изображения, полученные с помощью цифровой фотокамеры (графический); звуковые сигналы вызова мобильного телефона (аудио); видеоклипы (видео); вибровызов мобильного телефона (механический). ⊕

Базовые типы символов. Для представления символьных моделей используются следующие *базовые типы символов*:

- *аудио* (для представления звукового сообщения);
- *графический* (для представления сообщения в форме неподвижного изображения);
- *текстовый* (специализация типа *графический*);
- *числовой* (специализация типа *текстовый*);
- *видео* (для представления сообщения в форме движущихся изображений);
- *гипертекстовый* (для представления сообщения в форме гипертекста [5];
- *механический* (для представления сообщения в форме механических воздействий[2]).

Композиции из базовых типов символов. Композицией из базовых типов символов будем называть любое сочетание из числа возможных сочетаний базовых типов. Вряд ли нуждается в пояснениях стремление использовать в сообщениях все типы базовых символов. Достаточно вспомнить, какую роль сыграла мультимедийная форма представления документов, которая с добавлением видео затем стала гипермедийной.

2.3 S-моделирование: основные составляющие и классы задач

Изучение свойств и закономерностей s-моделирования позволяет определить задачи s-моделирования и заняться поиском методов их решения. Говоря о задачах, имеем в виду задачи исследований. Поэтому названия задач далее представлены как названия тематических подпространств пространства s-моделирования.

Результат наших исследований *s-моделирования* как *технологии* (*комплекса методов*) *изготовления s-моделей произвольных объектов и манипулирования*

[1] Под типом понимается множество значений.

[2] Пример: вибровызов мобильного телефона.

ими представлен далее развернутым определением составляющих s-моделирования и соответствующих им классов задач. Форма определения может служить примером текстовой модели сообщения[1], рассчитанной на представление на бумажном носителе.

▷ [Dn: PrC ≈ «класс задач»; →≈ «связанный с предыдущим»]

□{S

1. *Представление моделей* на *языках сообщений*, рассчитанных на восприятие человеком и *машинами*:

   PrC языки как средство представления *моделей сообщений*: базовые типы *символов* и *кодов* → системы *символов* и *кодов*; *модели* человеко- и машинно-ориентированных *языков сообщений* (*языки* спецификации, программирования, запросов, системы команд машин); *модели* представления: данных [6], документов; представление *моделей* систем понятий, на которых интерпретируются *сообщения*, составленные на языках → *представление моделей* систем знаний; → *модели* технологий *представления*;

2. *Преобразование типов и форм представления моделей*.

   PrC *преобразование* типов представления *моделей* (⊕ аудио в текст и обратно ⊕);

   преобразование форм *представления моделей* (аналоговой в цифровую и обратно; несжатой в сжатую и обратно; незашифрованной в зашифрованную и обратно; несжатой и незашифрованной в сжатую и зашифрованную и обратно) (⊕ одной формы представления документа в другую: *.doc в *.pdf ⊕) → *модели* технологий *преобразования*.

3. *Распознавание моделей сообщений*:

   PrC сопоставление с *моделью-образцом*, сопоставление свойств распознаваемой *модели* со свойствами *модели-образца*; *распознавание содержимого документов* → *модели* технологий *распознавания*;

4. *Конструирование моделей*:

   PrC *конструирование моделей*: s-моделирования и его составляющих; s-среды; *языков*, систем понятий и систем знаний, *интерпретаторов сообщений* на *моделях* систем понятий, систем *символов*, систем *кодов*; *моделей* специфицирования, программирования, взаимодействия в s-среде; *моделей*: задач [7], алгоритмов, программ; моделей: архитектур машин, архитектур сетей, сервис-ориентированных архитектур. [⊕ В марте 2007 г. группа[2] участников консорциума W3C представила [8] спецификации языка SML моделирования сервисов (Service Modeling Language [9]) и формата обмена SML моделями (SML Interchange Format [10] Version 1.0). ⊕]; *моделей*: *сообщений* и средств их построения; документов и документооборота → модели технологий *конструирования*.

5. *Интерпретация моделей:*

   PrC *модели* интерпретации *сообщений на моделях* систем понятий; *модели* трансляции (компиляции, интерпретации, ассемблирования) → *модели* технологий *интерпретации*;

6. *Обмен моделями*:

   PrC *модели* взаимодействия в s-среде: человек–*машина*; *машина*–*машина*; *модели*: типов отправителей и получателей; средств отправки, передачи и получения сообщений; сред передачи *сообщений*; *модели*: архитектур сетей, сервис-ориентированных архитектур; *модели* систем правил обмена сообщениями (коммуникационных протоколов); *модели* документооборота; → *модели* технологий *обмена*;

7. *Сохранение, накопление и поиск моделей:*

   PrC *модели* процессов сохранения, накопления и поиска; типов памяти и накопителей; управления памятью и накопителями; форм сохранения и накопления; типов носителей; средств сохранения, накопления и поиска; *модели* баз данных, библиотек программ и др.; *модели* спецификации предмета поиска; *модели* поиска по образцу, по признакам, по описанию свойств; *модели* поисковых машин; → *модели* технологий сохранения, накопления и поиска.

8. *Защита моделей* от несанкционированного доступа и применения:

   PrC *модели* уязвимостей, контроля доступа, защиты от вторжений, вредоносных программ, перехвата сообщений, несанкционированного применения → *модели* технологий защиты.

S}□◁

Далее определения приведены с использованием названий основных классов задач s-моделирования.

□ *S-модель* — символьный конструктивный объект, s-представление которого удовлетворяет требованиям s-распознавания человеком и s-машиной. □

⊕ Веб-страница на экране монитора — это гипермедийная *s-модель* сообщения, где используются

[1] Эта форма применяется также в приложении.

[2] BEA, CA, Cisco, EMC (Documentum), HP, IBM, Intel, Microsoft and Sun Microsystems.

аудио-, текстовый, графический и видеотипы; сканируемая таблица — графическая s-модель; содержимое ячеек таблицы табличного процессора, для которых задан числовой формат (числовой тип); мелодия вызова мобильного телефона — аудио s-модель; схематический видеоклип из электронной энциклопедии, сопровождаемый пояснениями — s-модель, построенная с использованием аудио- и видеотипов. ⊕

□ *S-код* — символьный конструктивный объект, являющийся s-представлением s-модели, рассчитанным на s-машину. Удовлетворяет требованиям s-(распознавания, преобразования, конструирования, интерпретации, обмена, сохранения, накопления, поиска, защиты). □

⊕ *S-коды*: *код* исполняемой программы, *код* цифровой аудиозаписи. ⊕

◇ Разработка эффективных систем *символов* и *кодов*, методов *кодирования* и *декодирования* — важные составляющие проблемы построения s-среды. ◇

Цифровое представление символов. Символу любого типа можно поставить в соответствие некоторое число. Постановку числа в соответствие символу называют цифровым кодированием, а его результат — цифровым представлением символа. Для чисел, заменяющих символы, выбирают целесообразную систему счисления. Этот выбор направляется стремлением обеспечить наиболее эффективное манипулирование кодами символов и s-моделей, которое выполняет цифровая s-машина при решении различных задач. Выбор ограничен рядом условий, среди которых физико-техническая реализуемость используемых для построения s-машин элементов с количеством устойчивых состояний, равным основанию выбранной системы счисления. ⊕ Программы и данные в современных цифровых s-машинах (компьютерах, смартфонах, цифровых фото- и видеокамерах и др.) представлены в виде двоичных кодов. ⊕

Преобразование (АЦП) из аналоговой формы представления символов в цифровую при вводе и обратное преобразование (ЦАП) при выводе связывают цифровые s-машины с их аналоговым окружением в s-среде, к которому относятся и управляющие ими люди.

2.4 S-моделирование: общий метод и условие реализации

□ *Общий метод s-моделирования* — конструктивное доказательство существования s-модели, представимой в двух формах, одна из которых рассчитана на интерпретацию человеком, а другая — s-машиной. □

Необходимое условие реализации s-моделирования предполагает существование удовлетворяющих требованиям s-(представления, распознавания, преобразования, конструирования, интерпретации, обмена, сохранения, накопления, поиска и защиты):

(1) языка описания s-моделей, рассчитанного на человека;

(2) языка команд s-машины;

(3) программ s-преобразования s-моделей на языке для человека в описания на языке команд s-машины.

◇ Формальное символьное моделирование в математике не стеснено требованиями (1)–(3). Конечно, языку математического моделирования можно поставить в соответствие язык описания s-моделей. [⊕ Пролог (логика предикатов первого порядка), Лисп (λ-исчисление). ⊕]

Развитие языков s-моделирования, рассчитанных на человека, направляется стремлением использовать полную композицию базовых символов для построения символьных систем языков, библиотеки и средства конструирования спецификаций задач, а по полученным спецификациям — программ решения задач. ◇

В Приложении приведены этапные результаты, полученные в процессе становления и развития символьного моделирования.

3 S-моделирование системы знаний информатики: основания

Системы понятий и системы знаний являются наиболее важными для науки объектами символьного моделирования. Рассмотрение задачи построения s-модели системы *S-ics* понятий и системы *Sinf* знаний информатики начнем с формулирования требований, которым должны удовлетворять определения систем понятий информатики, чтобы быть применимыми в *Sinf*.

В создаваемой *Синф*-методологии система *S-ics* понятий рассматривается как пара: множество *set[S-ics]* систем понятий и семейство *rel[set[S-ics]]* связей, заданных на множестве *set[S-ics]* (где *[S-ics]* и *[set[S-ics]]* — пометы)[1]. Каждая система понятий из *set[S-ics*] представлена аналогично. Не

[1] «Одноэтажная» форма записи та же, что и разработанная в [7] для языка спецификации задачных конструктивных объектов. В *Синф* эта форма применяется в языках спецификации s-моделей систем понятий и систем знаний.

устанавливается никаких ограничений на добавление новых систем понятий, детализацию и коррекцию существующих. При описании связей между s-моделями понятийных систем за основу взят подход, разработанный в ИПИ РАН в рамках исследований, посвященных методологии порождения программ [7].
⊕ В частности, две системы понятий считаются связанными, если их пересечение по памяти непусто (память каждой системы представлена элементами множества входящих в нее понятий). ⊕

Язык спецификации s-моделей систем понятий и систем знаний, разрабатываемый применительно к *Синф*, является развитием языка спецификации задачных конструктивных объектов, разработанного в [7].

Естественно, рамки статьи не позволяют рассуждать о всей системе понятий информатики. Поэтому изложение сосредоточено на описании отдельных составляющих *S-ics*, среди которых *символьная модель системы понятий, символьная модель системы знаний, сообщение, данные, информация*.

3.1 Конструктивность определений

Конструктивность — это свойство объектов, заключающееся в том, что они могут использоваться для построения конструкций по определенной системе правил. При этом для каждого типа объектов должны быть заданы набор атрибутов и совокупность допустимых операций.

Определение системы понятий конструктивно, если оно может быть использовано для построения определений других систем понятий. Реализация конструирования определений систем понятий в s-среде накладывает ограничение, связанное с представлением определений в форме s-моделей.
□ Чтобы быть конструктивным, определение системы понятий информатики должно удовлетворять следующим необходимым требованиям:

1. Определение системы понятий должно быть представлено в виде пары <*описание области применимости*>, <*s-модель системы понятий*>.
2. В систему понятий, считающуюся определенной, не должны входить понятия, не имеющие определений (и при этом не относящиеся к понятиям-аксиомам).
3. Область применимости определяемой системы должна принадлежать информатике. □

Требование 1 связано с тем, что информатика имеет дело с моделями, которые рассчитаны на реализацию в s-среде (то есть с s-моделями).

□ Описание области применимости (точки зрения [7]) — это описание типов:

- *корреспондента* (кому адресовано определение);
- *цели*, в процессе достижения которой оно имеет смысл (классы задач, при изучении которых определение может быть полезно);
- *стадии*, на которой целесообразно использовать определение (концепция исследования проблемы, методология решения проблемы (постановки задач и методы их решения), проектирование (разработка программно-аппаратных средств, сервис-ориентированных архитектур), поддержка применения сервисов, применение сервисов и т.д.). □

Формулировка определения в виде пары <*описание области применимости*>, <*символьная модель системы понятий*> позволяет уменьшить неопределенность истолкования[1]. Когда по контексту ясно, что *описание области применимости* не изменилось, будем ограничиваться ссылкой на него (указанием идентификатора). Идентификатор имеет вид *as id*, где *a* — постоянная составляющая идентификатора (сокр. англ. applicability specification), а *id* — переменная, которая может быть представлена любой совокупностью символов.
◇ Немало дискуссий, разворачивающихся вокруг определений, посвящены выяснению области их применимости. Конечно, тому, кто предложил определение, не всегда удается увидеть сразу все классы задач, где определение может работать. Это объяснимо. По мере накопления знаний первоначально указанные классы задач могут быть изменены. ◇

Предлагаемые в статье определения имеют следующее описание области применимости:

{*as Sinf*}

Корреспондент — информатик.

Цель — создание *Синф*-методологии.

Стадия — концепция *Синф*; постановка задач и разработка методов решения.

Условимся в определениях, имеющих это описание области применимости, не воспроизводить его каждый раз, а только сопровождать указателем {*as Sinf*}.

Если определение системы понятий удовлетворяет требованиям 1–3, то ее s-модель рассчитана на применение при изучении задач, класс (или

[1] Большая часть недоразумений с употреблением определений не по назначению происходит из-за отсутствия описаний области применимости.

несколько классов) которых указан в строке *цель* описания области применимости.

В основание подхода к построению *Синф* положены следующие утверждения.

А1. Любые системы знаний представлены моделями (символьными и несимвольными).

А2. Интеллектуальная деятельность связана с построением символьных моделей и манипулированием ими.

А3. □ *S-моделирование в реализации* — это автоматизированное конструирование символьных моделей произвольных объектов в s-среде, среди которых — s-модели систем понятий и систем знаний. □

Из утверждений А1–А3 следует, что в s-среде научную продукцию информатики целесообразно представлять в форме s-моделей систем знаний о свойствах и закономерностях s-моделирования.

3.2 Методологии s-моделирования: научная продукция информатики

□ *Научная продукция информатики* — методологии s-моделирования, представленные в виде s-моделей систем понятий и систем знаний.

Методологии s-моделирования — это комплексы методов s-(представления, распознавания, преобразования, конструирования, интерпретации, обмена, сохранения, накопления, поиска, защиты). Они предназначены для автоматизированного конструирования и применения s-моделей произвольных объектов в s-среде (включая модель этой среды). □

Среди множества моделируемых объектов особое внимание уделяется процессу s-моделирования, системам понятий, системам знаний, s-машинам и s-среде. Создаваемые методологии служат научным основанием для разработки информационных технологий, программных и аппаратных средств.

Информатика создает методологическое обеспечение для построения s-среды, повышающей продуктивность деятельности людей в различных областях. Особое значение имеет повышение продуктивности интеллектуальной деятельности. Применение s-моделирования вместо физического моделирования приносит не только экономию средств. Оно позволяет существенно быстрее, многостороннее и глубже изучить и проверить на состоятельность результаты, полученные при разработке программ, аппаратных средств, информационных технологий. То же можно сказать и относительно результатов в других областях: физике, биологии и др.

Итак, методы s-моделирования служат общим для всех наук арсеналом. Методы организации коллективного применения этого арсенала — также одна из важных задач информатики.

Применение методологий s-моделирования. Системы понятий и системы знаний — целевые объекты применения методологий s-моделирования.

◇ Задача s-моделирования становится относящейся к информатике, если она соответствует определениям ее предмета и основы научного метода:

- символы и построенные из них s-модели изучаются как элементарные и составные конструктивные объекты, представленные кодами, рассчитанными на манипулирование s-машинами;
- конструктивно доказать существование s-модели — значит построить ее из конструктивных объектов, реализуемых в s-среде.

Другими словами, *задачи s-моделирования, изучаемые информатикой, являются задачами автоматизированного конструирования s-моделей в s-среде, где люди и s-машины взаимодействуют между собой*. ◇

⊕ На практике: гипермедийные документы, получаемые от веб-сервера и интерпретируемые браузером, — это гипермедийные s-модели сообщений; текст, напечатанный на машинке, после сканирования, распознавания и интерпретации одним из текстовых процессоров может стать текстовой s-моделью сообщения (так как приобретет свойства составного символьного конструктивного объекта, удовлетворяющего требованиям манипулирования с помощью программы s-машины). ⊕

3.3 S-модель системы понятий

Начнем с пояснения, почему говорим о моделировании системы понятий, а не одного понятия. Дело в том, что любое определяемое понятие, которое будем называть *ядром системы понятий*, необходимо связать с другими ранее определенными понятиями.

□ {*as Sinf*} *S*-модель *s-cs* системы *Cs* понятий — это пара <*set[Cs]*, *rul[set[Cs]]*>, где *set[Cs]* — множество понятий, а *rul[set[Cs]]* — система связей, заданных на *set[Cs]*. □

⊕ В модели *произвольный треугольник* множество понятий включает стороны a, b, c, углы α, β, γ, площадь q, периметр p и др. Связи: $\alpha + \beta + \gamma = \pi$; $p = a + b + c$ и др. ⊕

S-модель системы метапонятий. Система понятий может рассматриваться как некоторое метапонятие. Чтобы определить систему метапонятий, также необходимо представить ее s-модель и описание области применимости. Подобное движение вверх логически не ограничено. То же можно заметить и относительно движения вниз: не существует никаких ограничений на детализирующие определения понятийных систем путем построения их s-моделей.

◇ S-модель *S-ics* системы понятий информатики может служить примером s-модели системы метапонятий. ◇

□ *S-модель предметной области* ≈ s-модель системы метапонятий. □

◇ *S-ics — это s-модель предметной области информатики.* ◇

3.4 S-модель системы знаний

□ {*as Sinf*} *S*-модель *s-kn* системы знаний *Kn* — это триада <*s-ca*, *set[s-l]*, *set[s-i]*>, где *s-ca* — модель системы *Sa* метапонятий, , *set[s-l]* — множество языков сообщений, а *set[s-i]* — множество интерпретаторов на модели *s-ca* сообщений, составленных на языках из *set[s-l]*. □

□ *Интерпретация сообщения* на модели *s-ca* системы *Sa* метапонятий — это построение выходного сообщения по заданному входному сообщению. □

⊕ Пример применения s-модели системы знаний дорожного движения был рассмотрен в [1]. В [7] приведены примеры применения s-моделей систем знаний, связанные с автоматизацией программирования. ⊕

◇ S-модель системы знаний произвольного назначения рассматривается как *s-развертка* (расширение *p-конкретизации* задачных конструктивных объектов [7]) определенной здесь s-модели системы знаний. ◇

S-язык сообщений: условия построения и реализации. □ *Необходимым условием построения s-модели языка сообщений* является существование s-моделей системы метапонятий, на которой предполагается интерпретировать сообщения, составленные на языке, и базовых типов символов, композиции которых предполагается использовать для построения системы символов языка. Эти модели играют роль исходных для построения языка. *Построение s-модели языка сообщений* — это разработка моделей:

(1) композиции базовых типов (разд. 2.1.) символов;

(2) системы символов языка, построенной на основе модели композиции базовых типов символов;

(3) системы правил конструирования сообщений с использованием модели системы символов. □

⊕ Примером относительно легко s-формализуемого языка служит язык шахматной игры, поскольку эта игра основана на однозначно определенной системе понятий, в которой семейство связей между понятиями задано правилами игры. ⊕

⊕ Сложность формализации произвольного языка общения связана с построением s-модели системы метапонятий, на которой должны интерпретироваться сообщения, являющиеся предложениями этого языка. Пункты (1)–(3) выполнить существенно легче. ⊕

Реализация s-языка предполагает существование реализованного в s-среде интерпретатора сообщений, построенных на этом языке.

S-интерпретатор сообщений: условие построения. □ Необходимым условием построения s-интерпретатора сообщений является существование s-моделей входного и выходного языков, а также системы метапонятий, на которой должны интерпретироваться сообщения, составленные на входном языке. Построение s-модели интерпретатора — это разработка моделей:

(1) s-распознавания сообщений на принадлежность входному языку;

(2) s-интерпретации распознанных сообщений на модели системы метапонятий;

(3) s-представления результата интерпретации в виде сообщения на выходном языке. □

3.5 Пример s-моделирования систем понятий

Построение s-моделей систем понятий, представляющих результаты изучения s-моделирования, ограничено сформулированными в разд. 2.1 требованиями конструктивности. В рассматриваемом примере приведены определения понятий *сообщение*, *данные* и *информация*, являющихся ядрами соответствующих понятийных систем, имеющих непустое пересечение по памяти [7].

□ {*as Sinf*} *Сообщение* — это s-модель произвольного объекта, представленная в форме, удовлетворяющей системе правил взаимодействия источника с получателем. □

В множество понятий этой системы кроме *сообщения* (*ядра* этой системы) входят *источник [сообщения]*, *получатель [сообщения]*, *передача [сооб-*

щения] и др., каждое из которых является ядром в своей системе. Например, *передача* — ядро в системе, включающей *метод [передачи]*, *среда [передачи]* и др. Таким образом, система понятий с ядром *сообщение* является системой метапонятий.

⊕ Все, что передается в s-среде — это сообщения, каждое из которых имеет отправителя и получателя: веб-страница, текст электронной книги, исходный текст программы, программа в исполняемом формате и др. ⊕

□ {*as Sinf*} *Данные* — это s-модель сообщения, получателем которого является решатель задач[1] (человек или программа s-машины). □

S-модель системы понятий с ядром *данные* связана *отношением специализации* [7] с s-моделью системы понятий, где ядром является *сообщение*. То есть *данные* — это *специализация сообщения* по типу *получателя*, которым здесь является *решатель задач*.

□ {*as Sinf*} *Информация* — это s-модель результата интерпретации сообщения на s-модели выбранной системы понятий. □

Для извлечения информации необходимо иметь: принятое сообщение; хранящиеся в памяти s-модели систем понятий, среди которых — необходимая для интерпретации принятого сообщения; механизмы поиска необходимой s-модели, интерпретации сообщения, представления результата интерпретации в виде s-модели и записи этой s-модели в память. S-модель системы понятий с ядром *информация* имеет непустое пересечение по памяти [7] с s-моделями систем, где ядрами являются *сообщение*, *s-модель системы понятий*, *интерпретация на s-модели системы понятий* и др.

◇ Если следовать предложенным определениям, то:

- более корректно говорить о передаче сообщений, а не информации или данных;
- не понять полученное сообщение — то же, что не суметь его интерпретировать;
- неправильная интерпретация принятого сообщения (из-за неправильного выбора системы понятий, на которой необходимо интерпретировать сообщение, или неправильной работы механизма интерпретации) приведет к получению некоторой информации. Но она будет ошибочной. ◇

⊕ Запрос веб-клиента — сообщение, интерпретируемое веб-сервером. Веб-страница, сформированная для отправки веб-клиенту — информация, полученная в результате интерпретации на s-модели. Отправленная веб-сервером веб-страница — отправленное сообщение. Она же принятая веб-клиентом — принятое сообщение. Результат интерпретации принятого веб-клиентом сообщения — экранное представление веб-страницы, рассчитанное на восприятие человеком. ⊕

Мы не затрагиваем здесь проблемы истинности извлеченной из сообщения информации, правильности s-модели, механизма интерпретации и др. Это отдельные важные задачи информатики.

Об определениях без указания области применимости. В работах К. Шеннона [11, 12] и А. Н. Колмогорова [13] «сообщение» и «информация» рассматриваются как составляющие систем понятий, имеющих другие (по сравнению с рассматриваемой в этой статье) области применимости. Явно эти области не указаны.

Употребление этих понятий связано там с задачами оценки объема кода некоторого сообщения или изменения предсказуемости исхода опыта. Так, для оценки изменения предсказуемости исхода опыта b в зависимости от исхода опыта a применяется разность энтропий

$$I(a,b) = H(b) - Ha(b)\,,$$

где $H(b)$ и $Ha(b)$ — энтропия исхода опыта b при неизвестном и известном исходе опыта a соответственно. При этом $I(a,b)$ рассматривается как приращение предсказуемости исхода опыта b, если известен исход опыта a. Заметим, что содержание опытов a и b и типы возможных исходов предполагаются заранее известными. Предполагается также, что знание исхода опыта a поможет в предсказании исхода опыта b. Другими словами, все известно, кроме исхода опыта.

Шеннон в [11] определил основную задачу связи как «точное или приближенное воспроизведение в некотором месте сообщения, которое было выбрано из некоторого множества возможных сообщений и отправлено из другого места». Он рассматривал эту работу именно как математическую теорию связи. В предложенной им коммуникационной модели определены основные элементы, присущие любой коммуникационной системе. Теория связи К. Шеннона представляет собой методологическое обеспечение технологий кодирования, передачи, декодирования и приема сообщений.

Шеннон разделяет задачи передачи сообщений и определения их смыслового значения: «семантические аспекты связи не имеют отношения к технической стороне вопроса»: «. . .часто сообщения имеют значение, т.е. находятся в соответствии с не-

[1] Задача в s-моделировании имеет более широкий смысл, чем в математике (см. разд. 2.1).

которой системой с определенной физической или умозрительной сущностью».
◇ Обратим внимание на следующее утверждение К. Шеннона: «Если множество возможных сообщений конечно, то число сообщений или любую монотонную функцию от этого числа можно рассматривать как меру информации, создаваемой выбором сообщения из этого множества, в предположении, что все сообщения равновероятны».

Однако сам по себе выбор сообщения не создает информацию. Нетрудно представить, что одно и то же сообщение, может иметь различающиеся результаты интерпретации. То есть одно и то же сообщение может порождать различную информацию (точнее, различные экземпляры информации). Полученные экземпляры зависят от того, какие будут применены модели систем понятий и методы интерпретации на выбранных моделях. ◇

В работах К. Шеннона [11] и А. Н. Колмогорова [13] говорится о «количестве информации» и рассматриваются задачи, связанные с этим понятием. Понятие «информация» там не определено. В этих и других работах этих авторов задача извлечения информации путем интерпретации сообщений на моделях систем понятий не изучалась[1].

4 Реализация и применение: общая характеристика

При том, что успех реализации в основном определяется тем, как построена s-модель системы метапонятий информатики, он не может быть достигнут, если неудачно построены и реализованы языки и интерпретаторы системы. Гипермедийные s-модели систем символов для построения входных и выходных языков системы *Синф*, включающие тип «интерактивное видео», обладают рядом важных для реализации *Синф* достоинств. Для науки и образования особого внимания заслуживают интерактивные видеоклипы, содержащие схематически изображенные движущиеся изображения, сопровождаемые аудио-, текстовыми и графическими пояснениями. Поэтому наиболее целесообразной выглядит реализация *Синф* как распределенной интерактивной гипермедийной системы знаний.

4.1 Применение в научной деятельности

Существование *Синф*, ее обновление и применение имеет смысл связать с деятельностью виртуальной лаборатории информатики. Деятельность по созданию, обновлению и применению *Синф* могла бы содействовать существенным изменениям в технологиях представления и апробации научных результатов.

Полагаем, что по правилам виртуальной лаборатории информатики научные результаты сначала могли бы выкладываться для обсуждения на семинарах, каждый из которых объединяет специалистов в определенном разделе информатики. Результаты, получившие признание на семинарах, могли бы быть оформлены в виде проектов обновлений *Синф*. Такие проекты имело бы смысл представлять для обсуждения на конференциях, каждая из которых объединяет специалистов нескольких смежных разделов. Результату, одобренному конференцией, целесообразно было бы присваивать статус рекомендуемого (для реализации) обновления *Синф*.

Таким образом, процесс апробации результатов был бы связан с построением *Синф*. Научный результат, изменивший состояние *Синф*, целесообразно считать *Синф-сертифицированным*, а справку о его авторе (или авторах) заносить в соответствую базу авторов *Синф*. Легко представить, как это изменило бы научный процесс.

4.2 Применение в проектировании

Уже сегодня, говоря о проектировании, прежде всего имеют в виду автоматизированное проектирование, представленное САПРами различного назначения, уровня сложности, совершенства и доступности. По существу мозговым центром любой САПР является система знаний, реализованная в той или иной форме. От того, насколько удачно построена s-модель системы знаний, положенная в основу реализации этой системы, зависит продуктивность проектирования. Пример *Синф* и здесь был бы небесполезен.
⊕ В частности, предложенный подход к построению s-моделей системы метапонятий, языков и интерпретаторов целесообразно использовать в проектировании систем машинного перевода, *особое внимание уделяя s-модели системы метапонятий*. ⊕

Информатика и инфотехника. □ {*as Sinf*} *Информационная технология* — это комплекс методов, предназначенный для решения некоторого класса задач s-моделирования. □

Каждая информационная технология может иметь различные реализации. Разработка и исследование информационных технологий относятся к информатике, а все, что связано с их реализацией — к *инфотехнике*.

[1] Насколько нам известно, эта задача не изучалась и в других работах, связанных с понятием «информация».

□ {*as Sinf*} *Предметом инфотехники* является построение s-среды как основы для поддержки деятельности в различных областях. □

Научным основанием для решения задач инфотехники служат результаты информатики.

4.3 Применение в образовании

Вполне естественно, что *Синф* могла бы стать профессионально изготовленным образцом для энциклопедий информатики разного уровня сложности[1] (от «для начинающих» до «для информатиков»[2]) и систем знаний дистанционного образования.

5 Заключение

1. Символьная модель системы знаний информатики, основы концепции построения и применения которой изложены в статье, рассматривается как средство формирования понятийного аппарата информатики коллективными усилиями исследователей. Методология создания этой системы изучается как составляющая методологического обеспечения разработок технологий автоматизации научных исследований, образовательных процессов и проектирования.

2. Построение системы знаний информатики, ее обновление и применение целесообразно связать с деятельностью виртуальной лаборатории информатики, что важно как для формирования понятийного аппарата информатики, так и для реализации полезных изменений в технологиях представления и апробации научных результатов.

3. Результаты исследований, полученные при создании методологии, используются в научно-исследовательском и образовательном процессах[3], включая процесс научно-методологической поддержки создания Большой Российской Энциклопедии (в части, отнесенной к разделу «Информатика»)[4].

Приложение

Схематически[5] представленные здесь результаты сгруппированы по типам символов, играющих роли основных в своих группах. Этот, конечно же, неисчерпывающий перечень не содержит комментариев и является своеобразной схемой очерка[6], посвященного становлению и развитию символьного моделирования, в котором все более значительную роль играет символьное моделирование в человеко-автоматной среде, названное нами *s-моделированием*.

▷[Dn: →≈ «связанный с предыдущим»]
{*S модели*

Аудио:

- звуки → речь;
- языки звуковых сообщений: языки общения, профессионально-ориентированные расширения языков общения;
- символьные модели звуковых сообщений: аналоговое кодирование → запись и воспроизведение, сохранение и накопление звуковых сообщений (грамзапись, запись на магнитных носителях → фонотеки); удаленная передача и прием (телефония, радиосвязь, радиовещание);
- *звуковых сообщений*: цифровое кодирование → цифровые технологии записи (на магнитных и оптических носителях), редактирования и воспроизведения, удаленной передачи и приема, синтеза и распознавания, сохранения и накопления звуковых сообщений → цифровые технологии мобильной и стационарной связи, спутниковой радиосвязи, Интернет-телефонии, радиовещания.

Графический → текстовый → числовой:

- рисунки → схемы;
- графические символьные модели звуковых сообщений: буквы, иероглифы, ноты (как элементарные графические символы для построения текстов и нотных записей) → текстовые символьные модели языков сообщений[7] → основание письменности;
- символьные модели системы понятий, включающей число → позиционные системы счисления;
- символьные модели систем понятий, включающих задачу и алгоритм: формулировка задачи → существование решения, единственность → методы решения → алгоритмы → оценки трудоемкости реализации алгоритмов → конструктивные доказательства существования алгоритмов;

[1] Это могло бы способствовать школьному и вузовскому образованию.

[2] Поскольку знания, не относящиеся к своему разделу, конечно же, сначала лучше получить в адаптированной форме.

[3] Студентам МИРЭА, обучающимся на базовой кафедре проблем информатики ИПИ РАН, с 2008 г. будет читаться профильный курс «Символьное моделирование в информатике», в основу которого положены обсуждаемые в этой статье результаты.

[4] Эту работу ИПИ РАН выполняет совместно с редакцией «Техника» БРЭ (формирование словника раздела «Информатика», научное консультирование, написание и рецензирование статей).

[5] Без учета хронологической последовательности их появления.

[6] В обычной форме такой очерк потребовал бы как минимум отдельной статьи.

[7] Включая графический язык записи музыкальных композиций (нотной записи).

цифрового кодирования символов любого типа → предпосылка изобретения автоматов, манипулирующих кодами символов и построенных из них s-моделей → цифровые автоматы s-среды;

- *алгоритма* → программа;
- *s-машины* → компьютеры и компьютерные устройства (смартфоны, цифровые фотокамеры и др.) → формирование s-среды;
- *аппаратно реализуемых конструктивных элементов (микропроцессора, памяти и др.) и систем правил конструирования s-машин*: элементные базы → комплектующие → автоматизированное конструирование аппаратных средств;
- *архитектур s-машин* → персональные компьютеры, суперкомпьютеры и др.;
- *языков сообщений в s-среде*: языки программирования, запросов, спецификации;
- *процессов s-моделирования* различного типа (программирования, специфицирования задач и др.) → средства автоматизации программирования: языки, трансляторы (ассемблеры, компиляторы, интерпретаторы), библиотеки программ, инструментальные системы программирования → s-автоматизация проектирования;
- *систем понятий, включающих данные* → базы данных, системы управления базами данных;
- *процессов взаимодействия человека с окружением и решения им задач различного назначения в s-среде* → искусственный интеллект;
- *компьютерных сетей*: сетевые архитектуры, локальные и региональные сети, Интернет → службы, работающие на базе Интернета (электронная почта, Веб, поиск и др.);
- *методов решения задач объединениями s-машин s-среды* → Грид;
- *сервис-ориентированных архитектур (СОА)* → конструирование служб (сервисов) ◇ новый этап в развитии s-среды ◇;
- *конструирования изображений в s-среде (где пиксель — элементарный конструктивный объект)* → цифровая фотография и редактирование изображений;
- *конструирования документов в s-среде* → электронный документооборот;
- *механизма перемещения между документами и их составляющими* → гипертекст;
- *автоматизированного проектирования в электронной промышленности, машиностроении и др. областях* → САПР как инструмент проектирования аппаратных средств s-среды, изделий электроники, машиностроения и др.;
- *распознавания изображений в s-среде*: распознавание текстов, биометрическая идентификация.

***Видео*:**

- символьные модели процессов записи и воспроизведения движущихся изображений (видеосъемка, кино, телевидение);
- *цифрового кодирования движущихся изображений, записи и воспроизведения цифрового видео* → цифровая видеосъемка и монтаж, цифровые технологии видеоконференций, видеомобильной связи, телевидения, интерактивного видео и телевидения.

Механический:

- жесты → языки жестовых сообщений;
- *составляющей гипермедиа* → вибросигнализация в мобильной связи, игровые приложения.

***Композиции базовых типов символов*:**

различных объектов, для представления которых используются текстовые, гипертекстовые, графические, аудио-, видео- и механический типы символов: мультимедиа → гипермедиа → конструирование веб-сервисов → СОА, гипермедийные системы знаний, имеющие сервис-ориентированную архитектуру.
S}◁

Литература

1. *Ильин В. Д., Соколов И. А.* Информация как результат интерпретации сообщений на символьных моделях систем понятий // Информационные технологии и вычислительные системы, 2006. № 4. С. 74–82.
2. *Артемов С. Н.* Формализации метод // Математическая энциклопедия, 1985. Т. 5. С. 635.
3. *Гришин В. Н.* Формальная система // Математическая энциклопедия, 1985. Т. 5. С 639.
4. *Ильин А. В., Ильин В. Д.* Интерактивный преобразователь ресурсов с изменяемыми правилами поведения // Информационные технологии и вычислительные системы, 2004. № 2, С. 67–77.
5. *Ильин В. Д.* Гипертекст. Большая Российская Энциклопедия, 2007. Т. 7.
6. *Соколов И. А.* Данные в информатике. Большая Российская Энциклопедия, 2007. Т. 7.
7. *Ильин В. Д.* Система порождения программ. М.: Наука, 1989.
8. http://www.w3.org/Submission/2007/01/
9. http://www.w3.org/Submission/2007/SUBM-sml-20070321/.
10. http://www.w3.org/Submission/2007/SUBM-sml-if-20070321/.
11. *Shannon C. E.* A mathematical theory of communication // Bell System Technical J., 1948. July and October, Vol. 27. P. 379–423 and 623–656. http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
12. *Шеннон К.* Работы по теории информации и кибернетике. Пер. с англ. под ред. Р. Л. Добрушина и О. Б. Лупанова. С предисловием А. Н. Колмогорова. М.: Иностранная литература, 1963.
13. *Колмогоров А. Н.* Три подхода к определению понятия «Количество информации» // Проблемы передачи информации, 1965. Т. I. Вып. 1. С. 3–11.

ABSTRACTS

DEVELOPMENT OF PUGACHEV FILTERING FOR STOCHASTIC SYSTEMS

I. N. Sinitsyn

IPI RAS, sinitsin@dol.ru

Statistical methods of information processing (filtering, extrapolation, interpolation etc) in stochastic systems are the basis of modern statistical informatics. Analytical survey and main tendencies of nonlinear conditionally optimal Pugachev filters for stochastic systems are considered. Pugachev filters for regular and nonregular, continuous and discrete, Gaussian and non-Gaussian stochastic systems are discussed. Interconnection between Pugachev and Kalman filters including various statistical criteria is given. Problems of joint filtering and recognition and Pugachev filters trends are considered.

Keywords: stochastic system, conditionally optimal filtering, extrapolation, interpolation, Pugachev filter, online information processing, autocorrelated noise

MEANS PROVIDING APPLICATIONS FAULT TOLERANCE

V. Zakharov[1] and V. Kozmidiady[2]

[1]IPI RAS, vzakharov@ipiran.ru

[2]IPI RAS, kozmidiady_v@tochka.ru

The problems of fault tolerant servers creation, caused by nondeterminate applications behavior, are considered. A formal model based on resources and events and describing an application behavior is described. The algorithms of an application execution logging on the reserved cluster node, of restoring and continuation of running after main node fault are proposed. In the proposed approach, both fault and recovery are hidden from the client who experiences slight service quality degradation at worst.

Keywords: applications server; transparent fault tolerance; process; resource; event; check point; determinate

MULTICHANNEL QUEUING SYSTEM WITH FINITE BUFFER AND UNRELIABLE SERVERS

A. Pechinkin[1], I. Sokolov[2], and V. Chaplygin[3]

[1]IPI RAS, apechinkin@ipiran.ru

[2]IPI RAS, isokolov@ipiran.ru

[3]IPI RAS, vchaplygin@ipiran.ru

Multichannel queuing system with a semimarkovian input flow, a phase type distribution of the servicing, a finite buffer and unreliable servers refusing independently one from other and regargless of the whole process of refusals and restorations is considered. Mathematical relations allowing calculation of the main stationary characteristics of system functioning are found under some different variants of the process of refusals and restorations.

Keywords: queuing system; unreliable servers

A NEW METHOD FOR THE PROBABILISTIC AND STATISTICAL ANALYSIS OF INFORMATION FLOWS IN TELECOMMUNICATION NETWORKS

D. Batrakova[1], V. Korolev[2], and S.Shorgin[3]

[1]IPI RAS, daria.batrakova@gmail.com
[2]Faculty of Computational Mathematics and Cybernetics of Moscow State University; IPI RAS, vkorolev@comtv.ru
[3]IPI RAS, sshorgin@ipiran.ru

A new method is proposed for the analysis of the stochastic structure of chaotic information flows in convergent telecommunication networks. The proposed method is based on a stochastic model of a telecommunication network which has the form of a superposition of some simple sequential-parallel structures. This model quite naturally generates mixtures of gamma-distributions for the network operation execution time. The parameters of the mixture of gamma-distributions characterize the stochastic structure of chaotic information flows in the network. The problem of statistical estimation of the parameters of gamma-distributions is solved by the EM-algorithm. To trace the variation of the stochastic structure of information flows in time, the EM-algorithm is applied for the moving window. The software for the division of distribution mixtures is developed and documented. The real input data is used. The interpretation of results is given.

Keywords: telecommunication networks; information flows; division of mixes of distributions; method of a moving window; software for division of distribution mixtures

LINGUISTIC SIMULATION FOR MACHINE TRANSLATION AND KNOWLEDGE MANAGEMENT SYSTEMS

E. B. Kozerenko

IPI RAS, kozerenko@mail.ru

This paper is dedicated to the vital problems of creating semantic-syntactic presentations for the systems of machine translation and extraction of knowledge from natural language texts. The purpose of our studies is the construction of an integral linguistic model on the basis of a synergetic approach, which uses linguistic knowledge, statistical methods, and mechanisms of machine learning for the extraction of new grammar rules from text corpora and disambiguation of language structures. To formalize linguistic knowledge, we have developed a new Cognitive Transfer Grammar which is a semantically motivated version of a generative unification grammar. For the preparation of system training components and obtaining statistical data about language structures, a multilingual resource is being created, comprising a Treebank and a corpus of semantically aligned parallel texts in Russian, English, and a number of other European languages.

Keywords: machine translation; grammar formalisms; linguistic model; parallel texts alignment; semantics; syntax; phrase structure

THE SYMBOL MODEL OF INFORMATICS KNOWLEDGE SYSTEM IN HUMAN-AUTOMATON ENVIRONMENT

V. D. Ilyin[1] and I. A. Sokolov[2]

[1]IPI RAS, vdilyin@ipiran.ru.
[2]IPI RAS, isokolov@ipiran.ru

The paper describes the authors approach to the construction of informatics knowledge system in human-automaton environment. This system is investigated as means of formalized representation of scientific results. The bases of conception of the construction and application methodology of this system are described. This methodology is studied as scientific foundation for automation of research, design, and education.

Keywords: informatics; symbol; symbol model; human-automaton environment; knowledge system; symbol modeling in human-automaton environment

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, место 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, комн. 531
Тел.: +7 (495) 135-86-92 Факс: +7 (495) 930-4505. E-mail: rust@ipiran.ru; asorokin@ipiran.ru