ISSN 2500-1752

Международный журнал информационных технологий и энергоэффективности

Том 2 Номер 3

2017

Международный журнал информационных технологий и энергоэффективности
Т.2 №3(5) 2017 г.

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

СОДЕРЖАНИЕ / CONTENT

Марголин М.С. Поляков М.В. Анализ подходов к представлению временных зависимостей в сложных технических системах // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т. 2 № 3(5) с. 2-13

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.824

АНАЛИЗ ПОДХОДОВ К ПРЕДСТАВЛЕНИЮ ВРЕМЕННЫХ ЗАВИСИМОСТЕЙ В СЛОЖНЫХ ТЕХНИЧЕСКИХ СИСТЕМАХ[1]

Марголин М.С., Поляков М.В.

Филиал ФГБОУ ВО "НИУ "МЭИ" в г. Смоленске, Россия (214013, г. Смоленск, Энергетический проезд, дом 1); e-mail: mikemarg@mail.ru

Статья посвящена обзору существующих подходов и методов к представлению временных зависимостей в сложных технических системах, основанных на явном и неявном моделировании. В статье рассмотрены STRIPS-системы, сети Петри, такие логики как логика линейного времени, окамовская логика, логика деревьев вычислений. Выявлены основные особенности данных подходов. Определены сильные и слабые стороны рассмотренных логик.

Ключевые слова: явное моделирование времени, моделирование изменений, бизнес-процесс.

ANALYSIS OF APPROACHES TO INTRODUCING TIME DEPENDENCE IN COMPLEX TECHNICAL SYSTEMS

Margolin M.S., Polyakov M.V.

Smolensk Branch of the National Research University "Moscow Power Engineering Institute", Russia (214013, Smolensk, street Ehnergeticheskij, 1); e-mail: mikemarg@mail.ru

The article is devoted to the review of existing approaches and methods to the representation of temporal dependencies in complex technical systems based on explicit and implicit modeling. In the article STRIPS-systems, Petri nets, linear time logic, ockhamist logic, logic of calculation trees are considered. The main features of these approaches are revealed. The strengths and weaknesses of the logics examined are determined.

Key words: explicit time modeling, change modeling, business process.

Немецкий философ и логик Людвиг Витгенштейн полагал [1], что человек не может думать ни о чем нелогичном, т.к. в ином случае ему пришлось бы нелогично думать. Данный тезис можно считать одной из основ развития логики, особенно в связи с развитием большого числа различных неклассических логик.

[1] *Работа выполнена при поддержке Совета по грантам Президента РФ в рамках научного проекта МК-6184.2016.8.*

Классическая логика подразумевает всего два значения истинности – истина и ложь, поэтому она не может быть применима к категории времени. В классической логике не рассматривается возможность изменения значений истинности в зависимости от момента времени в различных сложных технических системах. Классическая логика базируется на законе исключенного третьего и не допускает появления событий, которые могут иметь истинное состояние, отличное от истины и лжи.

В процессе развития логики, ученые разработали различные модели неклассической логики, в рамках которых пытались найти выход из данного противоречия. С развитием логики как науки, сложилось несколько подходов к представлению временных зависимостей в сложных технических системах. Так в [2] предложено рассматривать явное моделирование времени (явный подход) и моделирование изменений (неявный подход). Вести и воспринимать различные математические расчеты удобнее, применяя методы неявного моделирования, нежели явного, однако они имеют определенные достаточно существенные ограничения в представлении сложных временных зависимостей.

Среди методов неявного моделирования хорошо известна т.н. Система STRIPS (Stanford Research Institute Problem Solver) [3, 4]. Система представляет собой разработку ученых Центра искусственного интеллекта при Стэнфордском исследовательском институте (США, Калифорния) Ричарда Файкса и Нильса Нилсона. В 1971 году ими был предложен автоматический планировщик, представляющий изменения в мире в виде переходов из одного состояния в другое. Система порождает планы путем обобщении уже решенных задач. Система выступала в роли модуля планирования для мобильного робота Shakey.

Задача планировщика заключалась в поиске такой последовательности действий (такого оператора), который позволял бы преобразовать исходную модель мира X в некую модель мира Y, для которой было достижимым некое целевое условие, заданное ранее. Иначе говоря, данная система позволяла определить команду, которую необходимо отдать роботу для выполнения цепочки последовательных действий.

Мир такой системы состоит из комнат, дверей, ящиков, окон и источников света. В каждом частном случае Мир описывается множеством утверждений, выраженных в форме предложений исчисления предикатов первого порядка.

Состояния в системе можно представить в виде множества фактов F = {F1…Fn}. Действия задаются с помощью базы правил перехода R = {R1…Rk}. Условия применимости действия (предусловие - precondition), задаются при помощи формулы. Для выяснения применимости некоторого действия в некотором заданном состоянии, необходимо проверить истинность предусловия, т.е. доказать, что предусловие является логическим следствием множества аксиом данного состояния. Действие применимо только в том случае, если истинно предусловие.

Система STRIPS начинает свою работу с попытки доказать, что цель является следствием множества формул, описывающих начальное состояние. Задача решена, если цель следует из начального состояния. Схем действий, удовлетворяющих решению задачи, может быть несколько. В таком случае формируются ветви поиска, в каждой из которых прорабатывается один способ. После того, как схема действия выбрана, ее условие применимости (предусловие) рассматривается в качестве новой подцели.

STRIPS система обладает рядом недостатков. Описание определенной модели мира может быть достаточно большим, копии модели мира неудобно хранить в системе. Одна и та

же модель мира может использоваться в описании нескольких узлов. Большинство утверждений в описании модели мира остаются неизменными. Поэтому все формулы моделей мира хранятся в единой структуре памяти.

Так же широко распространенным методом неявного моделирования считаются сети Петри. Сети Петри были предложены Карлом Адамом Петри в качестве механизма для моделирования динамических дискретных систем, которые содержат взаймодействущие параллельные компоненты.

В классическом представлении Сеть Петри представляет собой двудольный ориентированный граф, состоящий из вершин двух типов — позиций и переходов, соединённых между собой дугами. Вершины одного типа не могут быть соединены непосредственно. В позициях могут размещаться специальные метки (маркеры), способные перемещаться по сети.

Под событием понимается срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно, либо разновременно, при выполнении определенных условий.

Сеть Петри описывается моделью вида N = (P, T, E, L), где N – двудольный ориентированный граф:

- P и T – два непересекающихся множества вершин $(P \cap T = 0)$, называемых позициями и переходами соответственно,
- F – множество направленных дуг, каждая из которых соединяет позицию $p \in P$ с переходом $t \in T$,
- L – выходная функция, назначение меток для дуг.

Для каждого перехода $t \in T$ может быть задано множество всех дуг сети E + (t) и E – (t), входящих в вершину t, и множество всех дуг сети, выходящих из вершины t.

Произвольный переход t может быть выполнен, если выполняется предусловие для его исполнения.

Процесс функционирования Сети Петри может быть наглядно представлен графом достижимых маркировок. Состояние сети однозначно определяется её маркировкой, т.е. распределением фишек по позициям. Вершинами графа являются допустимые маркировки сети Петри, дуги помечены символом срабатывающего перехода. Дуга строится для каждого возбуждённого перехода. Построение прекращается поле того, как будут получены маркировки, в которых не возбуждён ни один переход, либо маркировки, содержащиеся в графе.

Недостатком Сетей Петри можно считать невозможность различить фишки в таких сетях. Содержание площадки можно представить с помощью указания числа фишек. В некоторых разновидностях Сетей Петри фишки имеют такую структуру, которая позволяет отличать их друг от друга. В Сетях Петри нет строго понятия процесса, который можно было бы выполнять на указанном процессоре. Различные Сети Петри можно использовать для анализа данных или моделирования бизнес-процессов. В [6] предлагается проводить анализ бизнес-процессов, представленных в нотации ARIS eEPC с использованием аппарата сетей Петри. Для этого необходимо интерпретировать бизнес-процесс в сеть Петри. В соответствие каждой функции бизнес-процесса в нотации ARIS eEPC ставится позиция сети Петри, а каждому событию – переход. Логические элементы, при этом, скрываются в позициях сети Петри. Такой подход позволяет интерпретировать бизнес-процесс в сеть Петри. Так же

подход по идентификации рисков в бизнес-процессах сложных технических систем и моделированию бизнес-процессов предложен в [7].

Среди недостатков таких подходов можно выделить следующие:

1. не ясным является учет логических элементов, имеющихся в бизнес-процессе, логика работы сети Петри в этом случае не раскрыта;
2. не отражен учет значимых элементов бизнес-процесса: ролей, документов, кластеров информации, информационных систем и др.

К явному подходу моделирования времени можно отнести множество временных логик. Время в таких логиках может быть представлено двумя способами: синтаксически (с помощью различных средств представления времени) или семантически (с помощью средств модальной логики).

Различные временные (темпоральные) логики являются более мощным средством и обладают широкими выразительными возможностями по представлению реальных временных конструкций, чем вышеупомянутые системы на основе моделирования изменений.

В [8] среди подходов к явному моделированию выделяют модальные темпоральные логики и темпоральные расширения классической логики.

В таких логиках в роли временных примитивов принимаются моменты или интервалы времени. Если за основу принимаются моменты времени, то интервал времени можно представить в качестве упорядоченной пары моментов, которые соответствуют началу и концу временного интервала. В ином случае момент времени можно рассматривать как интервал нулевой длины. Так же разработаны системы, которые допускают одновременное использование обоих временных примитивов.

В [8] отмечается, что структура времени в темпоральных логиках принимается исходя из проблемной области. При этом необходимо учитывать следующие аспекты:

- дискретность или непрерывность времени. Определяется ли минимальный шаг изменения времени, допустимо ли указание сколь угодно малого интервала времени.
- полнота или неполнота времени. Существует ли для определенной последовательности, принадлежащей заданной области интерпретации, предел, принадлежащей этой же области, или нет.
- ограниченность или неограниченность времени. Рассматривается ли достаточно большой интервал времени, но ограниченный или интервал не ограничен и возможны события, длящиеся бесконечное время.
- линейность, ветвистость или цикличность времени. Существует ли определенное отношение предшествования во времени для временных примитивов или нет.

Модальная логика строится на основе логики высказываний за счет добавления новых атрибутов, которые позволяют выражать отношение тех или иных высказываний к окружающей действительности. Как правило, к таким высказываниям относятся суждения о необходимости или возможности чего-либо.

Классическая логика рассматривает высказывания, которые являются ассерторическими – т.е. высказываниями, которые утверждают наличие или отсутствие той или иной ситуации. Однако, зачастую возникают ситуации, которые описывают высказывания, содержащие указания на возможность или необходимость. Появление таких высказываний связано с необходимостью описания событий, которые могут произойти в

будущем или имели место в прошлом, но которых нет в данный момент времени. Такие высказывания называются модальными (модальностями), а логики, их описывающие, - модальными логиками.

В [9] выделяют три вида модальностей.

- Алетические модальности. К данной группе относятся высказывания в терминах необходимости-случайности или возможности-невозможности. Такие суждения принимаются как логически значимые – истинные или ложные, т.е. не произвольно. А в силу каких-либо оснований.
- Деонтические модальности. Высказывания данной группы выражены в форме в форме совета, пожелания, правила поведения или приказа, побуждающего человека к конкретным действиям. К таким высказываниям можно отнести выражения: «обязательно», «запрещено», «разрешено».
- Эпистемические модальности. К данной группе относятся высказывания, выраженные в суждении информацией об основаниях его принятия и обоснованности, т.е. представляет собой характеристики знаний - «доказано», «опровергнуто», «знает», «верит» ит.д.

При изучении модальных логик в первую очередь стоит рассмотреть логику Прайора [10-11]. Данная логика пересматривает понятие истинности. В ней вводится система т.н. возможных миров, а так же операторы возможности (◊) и необходимости (□). Такие операторы позволяют характеризовать в рассуждениях фактор времени - с их помощью устанавливаются отношения во временных рядах. В логике Прайора некое высказывание p является истинным, если оно истинно в данном возможном мире и во всех остальных возможных мирах, достижимых для данного мира.

В модель Прайора входят:

- переменные p, q, r и т.д. – суждения, истинностные значения которых могут быть различными в различные моменты времени;
- ∧, ∨, ¬ и т.д. – логические связки;
- темпоральные операторы:
  - P, означающий «было так, что»;
  - H, означающий «всегда было так, что»;
  - F, означающий «будет так, что»;
  - G, означающий «всегда будет так, что».

P и F называют слабыми временными операторами, а H и G – сильными.

Продолжением развития логики Прайора стали работы Ганса Кампа [12, 13], в которых автор предложил расширить логику Прайора. Для этого он ввел новые операторы S (Since – с тех пор) и U (Until – до тех пор). Для данных операторов семантика задается следующим образом: υSψ – φ истинно с момента, когда ψ истинно; υUψ – υ будет истинно до тех пор, пока ψ ложно. Прайорские базовые операторы могут быть выражены с помощью операторов S и U: Fυ ≡ ⊤Uυ и Pυ ≡ ⊤Sυ. В логике Кампа операторы S и U позволяют представить большее число временных зависимостей, нежели в логике Прайора, например, последовательность событий. Позже были предложены одноместные операторы X (neXttime) и Y (Yesterday). Схема формул Xυ означает, что υ будет истинно в следующий момент времени, а Yυ – υ было истинно в прошлый момент времени, при этом время считается дискретным.

В [14] Фон Вригт предлагает ввести к логике высказываний бинарную связку Т, аргументами которой являются состояния дел. Под выражением АТВ понимается «сейчас происходит А, а в следующий момент времени происходит В. Использование бинарной связки Т позволяет строить сложные выражения вида -Т(-Т(-Т))), которые описывают состояния, выполняемые последовательно. Таким образом, в различные моменты времени некоторого отрезка проходит мир.

Относительно отдельного состояния А имеются четыре взаимоисключающие и совместно исчерпывающие возможности, а именно:

- АТА – имеет место состояние А, которое продолжает оставаться;
- АТ¬А – имеет место состояние А, но оно перестает существовать;
- ¬АТА– состояние А не имеет места, но возникает;
- ¬АТ¬А – состояние А не возникало и не возникнет.

В логике Фон Вригта время является дискретным. Оно представляет собой линейное течение последовательных случаев. Если число полных состояний мира равно 2^n, то число возможных историй в m последовательных моментах равно 2^{mn}.

Логика Пнуели представляет собой теорию, с помощью которой можно доказать наличие у высказывания свойств, характеризующих его правильное вычислительное поведение. Такая логика применяется с целью верификации компьютерных программ.

По мнению Пнуели для последовательных программ темпоральность не является существенным свойством. Зная точку останова и значения программных переменных, можно определить, на каком этапе выполнения программы находится оператор. Направленность последовательных миров от прошлого к будущему позволяет проводить рассуждения о времени в терминах «до» и «после». Для таких задач существует единственное прошлое и единственное будущее.

Однако для множества различных параллельных процессов этого не достаточно. Необходимо различать термины «где», «когда» и сохранять абсолютную временную шкалу, независимую от выполнения.

По мнению Пнуели поведение в прошлом для сложных технических систем менее важно, чем поведение в будущем, которое начинается с момента запуска систем. Поэтому в LTL логике отсутствуют какие-либо темпоральные операторы прошлого.

Данной логике устанавливается инвариантное правило, согласно которому, для установки неизменности некого свойства А в данном алгоритме, надо установить следующее:

- свойство имеет место в начале алгоритма;
- свойство сохраняется после выполнения каждой команды этого алгоритма.

С помощью данных правила в логике Пнуели можно доказать наличие различных инвариантных свойств. Например, к таким свойствам можно отнести свойство исключения критических секций. Данное свойство основывается на организации параллельных вычислений. Параллельные процессы могут включать в себя блоки, содержащие критические команды. Пока один процесс находится в своем критическом блоке, другой процесс ни при каких условиях не должен попасть в свой критический блок, иначе изменения в ячейках памяти при выполнении первого процесса исказят вычисления, используемый в рамках выполнения второго процесса. Второй процесс должен ждать, пока содержимое используемых им ячеек памяти не будет соответствовать требуемым значениям.

Ожидание обеспечивается благодаря выполнению специальных команд, называемых семафорными инструкциями.

Синтаксис логики LTL включает в себя множество пропозициональных переменных, логические связки (¬ и ∨), и временные модальные операторы (X и U). Остальные операторы могут быть выражены через базовые. Для упрощения формул логики LTL используют логические – ∧, →, ↔, ⊤, ⊥; и временные – G, F, R, W операторы. Смысл модальных операторов трактуется следующим образом:

- X φ – в следующий момент будет выполнено φ; (1)
- φ Uψ – существует состояние, в котором выполнено ψ и до него во всех предыдущих выполняется φ; (2)
- G φ – всегда будет выполнено φ; (3)
- F φ – будет выполнен φ; (4)
- φ Rψ – либо во всех состояниях выполняется ψ, либо существует состояние, в котором выполняется φ, а во всех предыдущих выполнено ψ; (5)
- φ Wψ – тоже самое, что и (5), но не гарантируется выполнение ψ в некоторый будущий момент. (6)

Полная аксиоматическая система для LTL, расширяющая классическую пропозициональную логику со стандартными K-аксиомами для G и X и правилами вывода modus ponens (MP) и необходимости (N) плюс аксиомы, выражающие неподвижные точки G и U выглядит следующим образом [20]:

(KG) $G(\varphi\to\psi)\to(G\varphi\to G\psi)$
(KX) $X(\varphi\to\psi)\to(X\varphi\to X\psi)$
(FUNC) $X\neg\varphi\leftrightarrow\neg X\varphi$
(FPG) $G\varphi\leftrightarrow(\varphi\wedge XG\varphi)$ (7)
(GFPG) $\psi\wedge G(\psi\to(\varphi\wedge X\psi))\to G\varphi$
(FPU) $\varphi U\psi\leftrightarrow(\psi\vee(\varphi\wedge X(\varphi U\psi)))$
(LFPU) $G((\psi\vee(\varphi\wedge X\theta))\to\theta)\to(\varphi U\psi\to\theta)$, где

(KG) $G(\varphi\to\psi)\to(G\varphi\to G\psi)$ - независимо от того, что всегда будет следовать из того, что всегда будет, всегда будет;

(KX) $X(\varphi\to\psi)\to(X\varphi\to X\psi)$ - независимо от того, что будет выполняться в следующий момент из того, что будет выполнено в следующий момент, будет выполнено в следующий момент;

(FUNC) X¬φ↔¬Xφ – аксиома функциональности.

Фактически аксиома FPG говорит, что $G\varphi$ является *фиксированной точкой* оператора ΓG, определяемого как $\Gamma G(\theta)=\varphi\wedge X\theta$, тогда как GFP$G$ говорит, что $G\varphi$ является (теоретически, с точки зрения его расширения) самой *высокой пост фиксированной точкой* ΓG. Точно так же аксиома FPU говорит, что $\varphi U\psi$ является *неподвижной точкой* оператора ΓU, определяемого как $\Gamma U(\theta)=\psi\vee(\varphi\wedge X\theta)$, тогда как LFP$U$ говорит, что $\varphi U\psi$- *самой низкой фиксированной точкой* $\varphi U\psi$. Заметим, что GFPG обобщает индукционную аксиому (IND): $\varphi\wedge G(\varphi\to X\varphi)\to G\varphi$.

Доказательства полноты вариаций приведенной выше аксиоматической системы можно найти в [22].

Система аксиом для данной логики является непротиворечивой и полной. ППФ доказуема в рамках PTL тогда и только тогда, когда она обозначима в PTL [15]. Недостатком такой логики можно считать то, что она не позволяет выстроить такую модель сложной технической системы, для которой будет допустимо наличие нескольких возможных будущих. Следовательно, такая модель ограничит возможность прогнозирования всех возможных вариантов развития и оценки риска выполнения каких-либо из них.

Помимо PTL выделяют пропозициональную темпоральную логику ветвящегося времени (BPTL). Такая логика является расширением пропозициональной темпоральной логики.

В отличие от линейной PTL, в логике BPTL каждое состояние может иметь более одного приемника и возможно несколько путей из текущего состояния. Модель времени в ветвящейся логике представляет собой бесконечное дерево, каждая вершина которого имеет конечное целое ненулевое число приемников.

Наличие последовательных миров в линейном времени не допускает наличия параллельной ветвящейся структуры. Ветвящаяся структура времени в отличие от линейной допускает ветвление будущего, т.е. допускается наличие единственного допустимого прошлого и нескольких вариантов будущего. Такое время соответствует концепции «возможных миров». Параллельная структура времени определяет различные параллельные миры. Таким образом, логика ветвящегося времени может использоваться при моделировании или прогнозировании. Для корректной интерпретации (определения истинности) утверждений о будущем должны заключать в себе два вида модальностей: с одной стороны, это модальность времени, а с другой – алетическая модальность. В [16] предложен анализ существующих подходов к темпоральным высказываниям, представляющим собой рассуждения о будущем. Такие утверждения могут быть рассмотрены подобно утверждениям о прошлом и настоящем времени как констатация существующего положения (ассерторические утверждения), т.е. содержать лишь модальность времени. Примерами таких утверждений могут служить «когда-нибудь будет p», «всегда будет q». Такие утверждения могут быть корректно интерпретированы в случае однозначности будущего, однако, в рамках концепции ветвящегося времени они не могут быть интерпретированы однозначно.

Оккамовская временная логика [20] Оккама и Прайора основана на предположении, что, хотя прошлое не может быть изменено, будущее может расходиться в разных направлениях с настоящего момента; однако в каждый момент времени одно возможное будущее считается актуальным, и именно в отношении этого будущего оцениваются будущие заявления. Таким образом, фразы «всегда / когда-нибудь в будущем» теперь означают «всегда / когда-то в будущем считаются актуальными». Более того, это фактическое будущее считается определяемым. Формально это означает, во-первых, что естественные потоки времени являются скорее древовидными, чем линейными, а во-вторых, что значение истинности любой формулы оцениваются не только относительно момента времени, но и истории, содержащей этот момент.

Так как в [21] каждая ветвь через t представляет собой возможный ход событий (для точки t и ветви b, мы говорим, что t лежит на b или b проходит через t, если t принадлежит b). Таким образом, мы можем представить возможное будущее t как множество всех

последующих точек на некоторой фиксированной ветви b на t; при этом каждая точка будет иметь уникальное прошлое.

Истинность формулы φ в модели M определяется так:

$$
\begin{aligned}
&M, t, b \models q \text{ если } \pi(t)(q) = 1,\\
&M, t, b \models \neg\varphi \text{ если не } M, t, b \models \varphi\\
&M, t, b \models \varphi \wedge \psi \text{ если } M, t, b \models \varphi \text{ и } M, t, b \models \psi,\\
&M, t, b \models G\,\varphi \text{ если } M, s, b \models \varphi \text{ для всех } s \text{ on } b \text{ с } t < s,\\
&M, t, b \models H\,\varphi \text{ if } M, s, b \models \varphi \text{ для всех } s \text{ на } b \text{ с } t > s,\\
&M, t, b \models \Box\,\varphi \text{ if } M, t, c \models \varphi \text{ для всех ветвей } c \text{ через } t,
\end{aligned}
\qquad (8)
$$

где π – это путь в M [23].

Но наиболее распространенной логикой ветвящегося времени является CTL логика. Она была предложена Эмерсоном и Кларком для формального описания требований корректного поведения, которые предъявляются к реагирующим вычислительным системам [17].

К таким системам можно отнести различные интерактивные, многонитевые, распределенные программы, встроенные информационные системы. Реагирующим системам присущи две характерные особенности – недетерминизм и незавершаемость вычислений. Поведение реагирующей системы определяется множеством ее бесконечных вычислений, организованном в виде дерева. В каждом состоянии вычисления могут быть зарегистрированы те или иные события, например, отправление запроса, открытие доступа к ресурсу, изменение значения переменной и др. Каждому из таких событий сопоставляется атомарная формула, принимающая логическое значение true только в том случае, когда по ходу вычисления реагирующей системы осуществляется соответствующее событие. Логика CTL позволяет описывать причинно-следственные зависимости между событиями, происходящими в различные моменты времени в процессе функционирования реагирующей системы.

Язык логики содержит временные операторы X, G, U. В CTL логике в качестве модальных операторов могут применяться кванторы пути A и E.

Под квантором A понимается выражение «на всех путях…», а под квантором E – «существует такой путь, что…». Квантор пути A понимается как «неизбежное» выполнение следующей за ним темпоральной формулы пути при всех вариантах развития событий из данного состояния. Квантор пути E можно понимать как «возможное» выполнение следующей за ним темпоральной формулы пути: эта формула пути выполняется, по крайней мере, на одном из вычислений, начинающих из данного состояния. В логике CTL временные операторы могут применяться только совместно с кванторами A и E

При помощи формул CTL можно задавать требования корректности, определяющие желаемое поведение системы, а затем для заданного формального описания (программы, схемы, спецификации и др.) реагирующей системы проверять, используя методы, алгоритмы и инструментальные средства верификации моделей программ, выполнимость этих требований.

Для увеличения выразительной возможности представления времени в искусственных системах были предложенные различные варианты расширения базовой CTL, например, логика с метрическими модализированными темпоральными операторами прошлого и будущего [20], пропозициональная темпоральная логика ветвящегося времени и т.д.

В [18,19] была предложена темпоральная интервальная логика LA. Данная логика была предложена в начале 80-х годов Дж. Алленом. В качестве примитивов интервальные логики используют темпоральные интервалы, которые вырождаются в точки.

Логика Аллена рассматривает 7 базовых интервальных отношений: b (be fore – рань ше), m (meets – встречает), o (overlaps – перекрывает), f (finishes – заканчивает), s (starts – начинает), d (during – в течение), e (equal - равняется). Каждое отношение имеет инверсию, которая обозначается «*».

Предложения в логике Аллена представляются в качестве выражения вида $\Omega = \{b, b^*, m, m^*, o, o^*, d, d^*, s, s^*, f, f^*, e\}$. При фиксированных интервалах A и B имеется ровно 2^{13} различных предложений логики Аллена, включая пустое и универсальное множества. Атомарное предложение логики Аллена имеет вид A ω B, где ω – одноэлементное подмножество (A {α} B, α ∈ Ω). Так же предложение A ω B может быть интерпретируемо как дизъюнкция $\bigvee\{A\ \alpha\ B \mid \alpha \in \omega\}$. Онтология в логике LA определяется как конечное множество O предложений логики LA.

В логике Аллена имеется отношение |= логического следования. Пусть O – онтология в LA и A ω B – произвольное предложение. Тогда мы говорим, что A ω B логически следует из O (в записи: O |= A ω B), если не существует интерпретации, при которой все предложения из O истинны, а предложение A ω B ложно.

Для построения модели, позволяющей анализировать выполнение бизнес-процесса сложной технической системы или моделировать этот процесс в будущем может использоваться CTL логика.

Логика Аллена широко применяется для решения многих задач искусственного интеллекта благодаря высокой выразительности для описания взаимосвязи между событиями и наличию полиномиального алгоритма вывода[19]. Например, она может использоваться для планирования действий агентов и расписания. Однако, алгоритм Аллена не является полным, т.е можно построить онтологию O и предложение A ω B такие, что O |= A ω B, но алгоритм выдает множество следствий из O, которое не содержит предложения A ω B.

В данной статье кратко рассмотрены некоторые темпоральные логики которые практически используются для представления темпоральной информации в интеллектуальных информационных системах.

Таким образом, временная логика включает в себя необходимые инструменты для получения и поиска ответов на основании рассуждения о времени.

Список литературы

1. Витгенштейн Л. Логико-философский трактат / Перевод и параллельный философско-семиотический комментарий В. П. Руднева // Логос. — 1999. — № 1, 3, 8.
2. Еремеев А.П., Троицкий В.В. Модели представления временных зависимостей в интеллектуальных системах поддержки принятия решений // Известия РАН. Теория и системы управления, 2003. - № 5. - С. 75-88.
3. R. Fikes and N. Nilsson (1971). STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2:189-208
4. Хант Э. Искусственный интеллект. М: Издательство «Мир». Редакция литературы по математическим наукам, 1978 г. - 558 с.

5. Питерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984.— 264 с, ил. Пер. с англ.
6. Елиферов, В.Г., Репин В.В. Бизнес-процессы: регламентация и управление: учеб. пособ. для слушателей образоват. учрежд., обуч. по MBA и др. программам подготовки управленческих кадров // Ин-т экономики и финансов "Синергия". — М.: Инфра-М, 2011.
7. Марголин М.С, Сеньков А.В. Подход к идентификации рисков бизнес-процессов в нотации ARIS eEPC на основе сетей Петри. XV национальная конференция по искусственному интеллекту с международным участием – Смоленск: Универсум, 2016. с 265-273.
8. Еремеев А.П. Темпоральные модели в интеллектуальных системах. IV Поспеловские чтения «Искусственный интеллект сегодня. Проблемы и перспективы», 2009.
9. Войшнилло Е.К. Дегтярев М.Г. Логика. М.: Владос-Пресс, 2001. – 528 с.
10. Prior, A.N. Time and Modality [Text] / A.N. Prior. – Oxford: Oxford University Press, 1957. – 160 p.
11. Prior, A.N. Past, present and future [Text] / A.N. Prior. – Oxford: Clarendon Press, 1967. – 228 p.
12. Kamp, H., Tense Logic and the Theory of Linear Orders , Ph.D. thesis, University of California, Los Angeles, 1968
13. Kamp, H., “Formal properties of ‘now’,”Theoria, vol. 37 (1971), pp. 227–73
14. Фон Вригт Г.Х. Логико-философские исследования. Избранные труды. М.: Прогресс 1986. – 593 с.
15. Torsun I.S. Foundations of Intelligent Knowledge-Based Systems // ACADEMIC PRESS, London, 1998
16. Смирнов В.А. Логические системы с модальными временными операторами // Материалы II Советско-финского коллоквиума по логике «Модальные и временные логики». – М.: Институт философии АН СССР, 1979. – С. 89-98.
17. Захаров П.Е., Булычев В.А. О верификации конечных параметризованных моделей распределенных программ. Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика, 2009, том 9, № 11, с. 116-123.
18. Allen, J.F. Maintaining Knowledge about Temporal Intervals [Text] / J.F. Allen // Communications of the ACM. – 1983. – Vol. 26, № 11. – P. 832-843.
19. Allen, J.F. Planning as temporal reasoning [Text] / J.F. Allen // Proc. of the 2 nd Intern. Conf. on Principles of Knowledge Representation and Reasoning. – 1991. – P. 3-14
20. V.Goranko and A.Galton, "Temporal Logic", The Stanford Encyclopedia of Philosophy (Winter 2015 Edition), Edward N. Zalta (ed.)
21. Y.Venema. Temporal logic. In L. Goble, editor, The Blackwell Guide to Philosophical Logic, pages 203–223. Blackwell Publishers, 2001
22. Goldblatt, R., 1980, “Diodorean Modality in Minkowski Spacetime”, Studia Logica, 39: 219–236. Reprinted in Mathematics of Modality (CSLI Lecture Notes 43), Stanford: CSLI Publications, 1993.
23. Кларк Э. М., Грамберг О., Пелед Д. Верификация моделей программ. Model Checking. М.: МЦНМО. 2002.

References

1. Vitgenshteyn L. Logiko-filosofskiy traktat / Perevod i parallel'nyy filosofsko-semioticheskiy kommentariy V. P. Rudneva // Logos. — 1999. — № 1, 3, 8.
2. Yeremeyev A.P., Troitskiy V.V. Modeli predstavleniya vremennykh zavisimostey v intellektual'nykh sistemakh podderzhki prinyatiya resheniy // Izvestiya RAN. Teoriya i sistemy upravleniya, 2003. - № 5. - S. 75-88.
3. R. Fikes and N. Nilsson (1971). STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2:189-208

4. Khant E. Iskusstvennyy intellekt. M: Izdatel'stvo «Mir». Redaktsiya literatury po matematicheskim naukam, 1978 g. - 558 s.
5. Piterson Dzh. Teoriya setey Petri i modelirovaniye sistem. M.: Mir, 1984.— 264 s, il. Per. s angl.
6. Yeliferov, V.G., Repin V.V. Biznes-protsessy: reglamentatsiya i upravleniye: ucheb. posob. dlya slushateley obrazovat. uchrezhd., obuch. po MVA i dr. programmam podgotovki upravlencheskikh kadrov // In-t ekonomiki i finansov "Sinergiya". — M.: Infra-M, 2011.
7. Margolin M.S., Sen'kov A.V. Podkhod k identifikatsii riskov biznes-protsessov v notatsii ARIS eEPC na osnove setey Petri. XV natsional'naya konferentsiya po iskusstvennomu intellektu s mezhdunarodnym uchastiyem – Smolensk: Universum, 2016. s 265-273.
8. Yeremeyev A.P. Temporal'nyye modeli v intellektual'nykh sistemakh. IV Pospelovskiye chteniya «Iskusstvennyy intellekt segodnya. Problemy i perspektivy», 2009.
9. Voyshnillo Ye.K. Degtyarev M.G. Logika. M.: Vlados-Press, 2001. – 528 s.
10. Prior, A.N. Time and Modality [Text] / A.N. Prior. – Oxford: Oxford University Press, 1957. – 160 p.
11. Prior, A.N. Past, present and future [Text] / A.N. Prior. – Oxford: Clarendon Press, 1967. – 228 p.
12. Kamp, H., Tense Logic and the Theory of Linear Orders , Ph.D. thesis, University of California, Los Angeles, 1968
13. Kamp, H., “Formal properties of ‘now’,”Theoria, vol. 37 (1971), pp. 227–73
14. Fon Vrigt G.KH. Logiko-filosofskiye issledovaniya. Izbrannyye trudy. M.: Progress 1986. – 593 s.
15. Torsun I.S. Foundations of Intelligent Knowledge-Based Systems // ACADEMIC PRESS, London, 1998
16. Smirnov V.A. Logicheskiye sistemy s modal'nymi vremennymi operatorami // Materialy II Sovetsko-finskogo kollokviuma po logike «Modal'nyye i vremennyye logiki». – M.: Institut filosofii AN SSSR, 1979. – S. 89-98.
17. Zakharov P.Ye., Bulychev V.A. O verifikatsii konechnykh parametrizovannykh modeley raspredelennykh programm. Nauchnyye vedomosti Belgorodskogo gosudarstvennogo universiteta. Seriya: Ekonomika. Informatika, 2009, tom 9, № 11, s. 116-123.
18. Allen, J.F. Maintaining Knowledge about Temporal Intervals [Text] / J.F. Allen // Communications of the ACM. – 1983. – Vol. 26, № 11. – P. 832-843.
19. Allen, J.F. Planning as temporal reasoning [Text] / J.F. Allen // Proc. of the 2 nd Intern. Conf. on Principles of Knowledge Representation and Reasoning. – 1991. – P. 3-14
20. V.Goranko and A.Galton, "Temporal Logic", The Stanford Encyclopedia of Philosophy (Winter 2015 Edition), Edward N. Zalta (ed.)
21. Venema. Temporal logic. In L. Goble, editor, The Blackwell Guide to Philosophical Logic, pages 203–223. Blackwell Publishers, 2001
22. Goldblatt, R., 1980, “Diodorean Modality in Minkowski Spacetime”, Studia Logica, 39: 219–236. Reprinted in Mathematics of Modality (CSLI Lecture Notes 43), Stanford: CSLI Publications, 1993.
23. Klark E. M., Gramberg O., Peled D. Verifikatsiya modeley programm. Model Checking. M.: MTSNMO. 2002.

Букачев Д.С. О реализации абстрактной граф-машины // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т. 2 № 3(5) с. 14-22

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.42

О РЕАЛИЗАЦИИ АБСТРАКТНОЙ ГРАФ-МАШИНЫ

Букачев Д.С.

ФГБОУ ВО Смоленский государственный университет, Смоленск, Россия (21400, г. Смоленск, ул. Пржевальского, 4), e-mail: dsbuka@yandex.ru

Статья посвящена вопросам оптимизации разработки программного обеспечения. В работе предложена конструкция и описание интерфейса абстрактной граф-машины, которая позволяет строить динамические графовые модели в любой предметной области независимо от типов данных, сопоставляемых элементам графа. В качестве примера в статье приведена типонезависимая реализация алгоритма Дейкстры построения оптимального маршрута между вершинами нагруженного графа.

Ключевые слова: универсальная алгебраическая система, абстрактная граф-машина, объектно-ориентированный подход, алгоритм Дейкстры.

ABOUT IMPLEMENTATION OF THE ABSTRACT GRAPH MACHINE

Bukachev D.S.

Federal State Educational Institution of Higher Education Smolensk State University, Smolensk, Russia (21400, Smolensk, street Przewalski, 4), e-mail: dsbuka@yandex.ru

Article is devoted to questions of optimization of software development. In operation construction and interface description of graphs machine by the abstract which allows to build dynamic graph models in any data domain irrespective of the data types compared to elements of a graph is offered. Implementation of an algorithm of Dijkstra of creation of an optimum route, independent of data type, between tops of the loaded graph is given as an example in article.

Key words: the universal algebraic system, the abstract graph machine, object-oriented approach, Dijkstra's algorithm.

Достаточно часто возникает ситуация, когда модели задач, решаемых в системах обработки информации и управления, принципиально отличаются лишь тем, что они оперируют понятиями (и, как следствие, типами данных) различных предметных областей. Это делает возможным использование единой фундаментальной модели. Иными словами, существует естественная необходимость использования так называемой абстрактной алгебраической системы [4], которая позволит решать определенный класс задач вне зависимости от их принадлежности к конкретной предметной области.

Обращение к алгебраическим моделям не случайно, так как они позволяют строить формализованные модели процессов обработки данных и решать задачи оптимизации этих процессов. В основу построения алгебраических моделей положены понятия универсальной

алгебры и универсальной алгебраической системы, известные определения которых приведены ниже [3, 4].

Пусть A – некоторое непустое множество. Частично определенная функция $y = F(x_1, \ldots, x_n), (y, x_1, \ldots, x_n) \in A$, называется *n-арной частичной операцией* на *A*. Если функция всюду определена, говорят просто об *n*-арной операции.

Система $U_A =< A, \Omega >$, состоящая из основного множества *A* и определенной на нем совокупности частичных операций $\Omega = \{F_s^{n_s}\}(s = 1, 2, \ldots)$, называется *частичной универсальной алгеброй* с сигнатурой Ω.

Универсальные алгебры U_A и U_B , в которых заданы соответственно сигнатуры Ω и Ω', называются *однотипными*, если можно установить такое взаимно-однозначное соответствие между сигнатурами Ω и Ω', при котором любая операция $\omega \in \Omega$ и соответствующая ей операция $\omega' \in \Omega'$ будут *n*-арными с одним и тем же *n*.

Пусть даны две однотипные универсальные алгебры $U_A =< A, \Omega >$ и $U_B =< B, \Omega >$ с основными множествами *A* и *B*.

Отображение φ: $A \to B$ называется *гомоморфным отображением* алгебры U_A в алгебру U_B, если для любых элементов $a_1, \ldots, a_n \in A$ и произвольной *n*-арной операции $F \in \Omega$ выполняется соотношение

$$\varphi(F(a_1, \ldots, a_n) = F(\varphi(a_1), \ldots, \varphi(a_n)),$$

где $\varphi(a_i) = b_i$ и $b_i \in B(i = 1,..., n)$. Алгебры U_A и U_B называются гомоморфными.

Пусть $\pi(x_1,..., x_n), x_1,..., x_n \in A$ – *n*-местный предикат, $\Pi = \{\pi_s^{n_s}\}(s = 1, 2, \ldots)$ – сигнатура предикатов. Система $\boldsymbol{U}_A = < A; \Omega; \Pi >$ называется *универсальной алгебраической системой*.

Случаи, когда можно обойтись единственным основным множеством, встречаются нечасто, поскольку объекты предметной области, как правило, являются сложными системами, для описания которых используется много разнотипных параметров. Для моделирования таких предметных областей используются многоосновные алгебры.

Система $U_{\boldsymbol{M}} =< \boldsymbol{M}; \Omega >$, состоящая из семейства основных множеств $\boldsymbol{M} = \{A_\alpha\}$ ($\alpha = 1, 2, \ldots$) и сигнатуры Ω операций, определенных на семействе $\boldsymbol{M}$ так, что каждая *n*-арная операция из Ω является отображением декартова произведения *n* множеств из семейства $\boldsymbol{M}$ в множество из того же семейства $A_{\alpha_1} \times \ldots \times A_{\alpha_n} \to A_{\alpha_r}$ называется *многоосновной алгеброй*.

Система $\boldsymbol{U}_{\boldsymbol{M}} =< \boldsymbol{M}; \Omega; \Pi >$, где Π – сигнатура *n*-местных предикатов $\pi : A_{\alpha_1} \times \ldots \times A_{\alpha_n} \to \{0,1\}$, называется многоосновной алгебраической системой.

Многоосновные алгебры и алгебраические системы играют важную роль в программировании. Они служат основой для создания пользовательских типов данных в современных языках программирования, ими также являются некоторые встроенные в языки типы данных. В частности, в [4] показано, что строковый тип данных, системы числовых матриц и нагруженных графов являются многоосновными алгебраическими системами.

На многоосновных алгебраических системах базируется современное теоретическое и практическое программирования. *Абстрактные типы данных* (они же *объекты* или *классы*), сокращенно АТД, определяются как многоосновные алгебраические системы.

Среди множества произвольных абстрактных типов данных выделяется один специфический вид. Эти АТД представляют собой двухосновные алгебраические системы вида $\boldsymbol{E}=<S, T; \Omega; \Pi>$. Основу S назовем *структурой*, а T – *типом*.

Важная особенность этих АТД состоит в том, что суть операций над элементами структуры S, не изменяется при изменении сути операций над элементами типа T. Такие АТД будем называть *абстрактными алгебраическими машинами* (ААМ). В работе [4] предлагается механизм построения абстрактной матричной машины, реализующей типовые операции над матрицами вне зависимости от типов данных элементов матриц, а также приводится листинг класса *TAbstractMatrixMachine* с подробным описанием его методов. На основе класса *TAbstractMatrixMachine* строится класс-наследник *TBooleanMatrixMachine*, определяющий тип элементов матриц и описывающий операции над элементами.

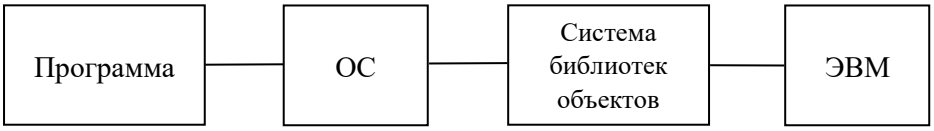
Абстрактные алгебраические машины составляют основу для решения задач оптимизации алгоритмов, реализующих процессы обработки данных. Оптимизация становится возможной благодаря отделению операций структуры от операций над элементами типа. При этом качество алгоритмов, реализующих операции структуры, не зависит от качества алгоритмов, реализующих операции типа.

Многие современные прикладные задачи часто бывает удобно формализовать с использованием теории графов. Такой подход позволяет, оперируя терминами этой теории, применять в ходе исследования известные алгоритмы на графах (см., например, [1]). Решая конкретную задачу, программист каждый раз должен:

1) описать структуры данных, которые он собирается использовать применительно к конкретной ситуации;
2) реализовать стандартные операции добавления/удаления элементов графа;
3) при необходимости модифицировать и реализовать стандартные алгоритмы на графах, используемые в процессе решения задачи.

Поскольку базовые операции над графами (такие как добавление/удаление элементов), а также реализация основных алгоритмов на графах фактически не зависят от характера данных, сопоставляемых элементам графа, и операций над этими данными, можно говорить о возможности создания такой алгебраической структуры, которая позволит работать с любыми графовыми моделями [2, 4].

Под абстрактной граф-машиной (АГМ) будем понимать ЭВМ и связанную с ней систему библиотек объектов:



В работе [2] рассмотрены основные принципы построения АГМ. В настоящей статье предлагается один из способов реализации АГМ на базе связных списков (см., например, [1]). Такой подход позволит использовать АГМ для создания динамических графовых моделей.

Листинг 1. Описание класса TVGraph (АГМ)

```
type
  TGrNodePtr=^TGrNode;                  //указатель на элемент списка вершин
  TListAdr=^TListNode;                  //указатель на элемент списка связей (рёбер)
  TGrNode=record
```

```
          N:integer;                //номер вершины в графе
          VData:pointer;            //указатель на данные в вершине
          Next:TGrNodePtr;          //указатель на следующую вершину
          Links:TListAdr;           //список связей
        end;

TListNode=record
           Node:TGrNodePtr;         //связь
           VWeight:pointer;         //ссылка на вес ребра
           Next:TListAdr;           //следующая связь
           VMark:pointer;           //ссылка на метку
         end;
 TRegIndex=1..3;                    //индексы регистров АГМ
 TObjectProc=procedure of object;
 TBoolFunc=function:boolean of object;
 TVGraph=class (TObject)
          private
            Graph:TGrNodePtr;       //список вершин
            FCount:integer;         //счетчик вершин
            FDVD:boolean;           //значение DestroyVirtualData
            //вспомогательные методы
            function GetCount:integer;
            function GetDVD:boolean;
            procedure SetDVD(Value: boolean);
          public
            RegResult:TRegIndex; //индекс регистра результата
            Regs:array[TRegIndex] of pointer;          //регистры данных
            VDCreate,VWCreate,VMCreate:TObjectProc;    //конструкторы данных
            VDDestroy,VWDestroy,VMDestroy:TObjectProc; //деструкторы данных
            //алгебра "абстрактных данных" вершин
            VDZero,VDAdd:TObjectProc;
            //алгебра "абстрактных весов"
            VWZero,VWInfinity,VWAdd:TObjectProc;
            VWOrder:TBoolFunc; //абстрактное отношение порядка
            //алгебра "абстрактных меток"
            VMZero,VMAdd:TObjectProc;
            property Count:integer read GetCount; //счетчик вершин
            //уничтожение (да/нет) связанных данных при удалении
            //элементов графа (вершин, рёбер)
            property DestroyVirtualData:boolean read GetDVD
                                        write SetDVD default true;
            //создание и уничтожение экземпляра АГМ
            constructor Create;
            destructor Destroy;
            procedure Reset; //сброс: p(G)->0, q(G)->0
            //инициализация деструкторов
            procedure InitConstructors(iVDCreate,iVWCreate,
                                       iVMCreate:TObjectProc);
            procedure InitDestructors(iVDDestroy,iVWDestroy,
                                      iVMDestroy:TObjectProc);
            //инициализация операций над данными вершин, рёбер(дуг), меток
            procedure InitVDataAlgebr(iVDZero,iVDAdd:TObjectProc);
            procedure InitVWeightAlgebr(iVWZero,iVWInf,iVWAdd:TObjectProc;
                                        iVWOrder:TBoolFunc);
            procedure InitVMarkAlgebr(iVMZero,iVMAdd:TObjectProc);
```

```
    //методы АГМ
    //операции с вершинами
    procedure AddNode(iData:pointer);
    procedure DelNode(Num:integer);
    function  GetNode(iNum:integer):TGrNodePtr;
    function  GetNodeProp(iNum:integer):pointer;
    procedure SetNodeProp(iNum:integer; iData:pointer);
    //операции с рёбрами/дугами
    function  YesLink(inN,toN:integer):boolean;
    procedure AddLink(inNum,toNum:integer; Double:boolean);
    procedure DelLink(inNum,toNum:integer; Double:boolean);
    function  GetLink(inNum,toNum:integer):TListAdr;
    function  GetLinkList(iNum:integer):TListAdr;
    function  GetLinkProp(inNum,toNum:integer):pointer;
    function  GetLinkMarker(inNum,toNum:integer):pointer;
    procedure SetLinkProp(inNum,toNum:integer; iWeight:pointer);
    //операции с метками
    procedure SetLinkMarker(inNum,toNum:integer; iMark:pointer);
    procedure ResetLinkMarker(ResetValue:pointer=nil);
    procedure ResetLink;
    //вычисление степени вершины графа
    function  DegIn(iNode:integer):integer;
    function  DegOut(iNode:integer):integer;
    function  DegDouble(iNode:integer):integer;
    function  Deg(iNode:integer):real;
  end;{TVGraph}
```

Класс не хранит и не обрабатывает данные, сопоставляемые элементам графа, в явном виде. Для работы с абстрактными данными класс содержит три регистра-указателя, абстрактные конструкторы, деструкторы, абстрактные операции сложения, умножения, присваивания нейтрала, абстрактное отношение порядка на множестве весов.

Описание классических алгоритмов на графах (волновой алгоритм, алгоритм Дейкстры, поиск циклических маршрутов, алгоритм Прима-Краскала, тест на эйлеровость) с абстракцией относительно типов данных элементов графа предлагается реализовать в виде отдельных классов. В работе [4] в качестве примера типонезависимого алгоритма приводится алгоритм Дейкстры поиска кратчайшего расстояния между вершинами нагруженного графа. В данной статье рассмотрим реализацию метода *Way* класса *TDijkstra*, реализующего данный алгоритм на базе класса *TVGraph* (АГМ).

Листинг 2. Описание метода Way класса TDijkstra

```
function TDijkstra.Way(inNode, toNode: integer; var WayArr: TIntArray): pointer;
type
  TDRec=record
          outNode:integer;  //номер вершины
          Len:pointer;      //указатель на длину пути до текущей вершины
          ConstLb:boolean;  //метка
        end;
var
  YL:boolean;
  DArr:array of TDRec;
  i,k,Lb:integer;
  e:pointer;
begin
```

```
WayArr:=nil;
//начальное заполнение массива DArr
if G.Count>0 then
  begin
    SetLength(DArr,G.Count);
    for i:=1 to G.Count do
      if i=inNode then       //если текущая вершина = стартовая
        with DArr[i-1] do
          begin
            outNode:=inNode; //запоминаем номер
            G.RegResult:=1;  //устанавливаем номер регистра-результата
            G.VWCreate;      //выделяем место для веса в регистре
            G.VWZero;        //обнуляем вес в регистре
            Len:=G.Regs[1];  //запоминаем указатель на длину пути
            ConstLb:=true;   //помечаем вершину
          end
      else                   //в противном случае
        with DArr[i-1] do
          begin
            outNode:=inNode; //запоминаем номер
            G.RegResult:=2;  //устанавливаем номер регистра-результата
            G.VWCreate;      //выделяем место для веса в регистре
            if G.YesLink(inNode,i) then //если есть ребро (дуга) inNode->i
              begin
                G.VWZero;    //обнуляем вес в регистре
                //получаем указатель на вес ребра (дуги) inNode->i
                G.Regs[1]:=G.GetLinkProp(inNode,i);
                if G.Regs[1]<>nil then
                  G.VWAdd; //складываем значения в регистрах 1 и 2
              end
            else //если нет ребра (дуги) inNode->i
              //устанавливаем в регистре-результате значение «бесконечность»
              G.VWInfinity;
            Len:=G.Regs[2]; //запоминаем указатель на длину пути
            ConstLb:=false; //деактивируем метку
          end;
    //вычисляем расстояние до финальной вершины
    while not DArr[toNode-1].ConstLb do
      begin
        Lb:=0;
        //находим первую непомеченную вершину в массиве DArr
        while DArr[Lb].ConstLb do
          Inc(Lb);
        //выбираем непомеченную вершину, наиболее близкую к стартовой
        for i:=Lb to (G.Count-1) do
          if not DArr[i].ConstLb then
            begin
              G.Regs[1]:=DArr[i].Len;
              G.Regs[2]:=DArr[Lb].Len;
              if ((G.Regs[1]=nil) and (G.Regs[2]<>nil)) or
                 ((G.Regs[1]<>nil) and (G.Regs[2]<>nil) and G.VWOrder) then
                Lb:=i;
            end;
        //активируем метку
        DArr[Lb].ConstLb:=true;
```

```
    Inc(Lb);
    //если не добрались до конечной вершины
    if Lb<>toNode then
      //пересчитываем расстояние до всех непомеченных вершин
      G.VWAdd;
      for i:=0 to (G.Count-1) do
        if not DArr[i].ConstLb then
          if G.YesLink(Lb,i+1) then
            begin
              e:=G.GetLinkProp(Lb,i+1);
              G.RegResult:=3;  G.VWCreate;
              G.Regs[1]:=DArr[Lb-1].Len;  G.Regs[2]:=e;
              if G.Regs[2]<>nil then
                G.VWAdd
              else
                begin
                  G.RegResult:=2; G.VWCreate;  G.VWZero;
                  G.RegResult:=3; G.VWAdd;
                  G.RegResult:=2; G.VWDestroy; G.RegResult:=3;
                end;
              G.Regs[1]:=G.Regs[3];  G.Regs[2]:=DArr[i].Len;
              //если пересчитанная длина через вершину Lb меньше текущей
              if G.VWOrder then
                //обновляем указатель на длину пути
                begin
                  DArr[i].outNode:=Lb;  G.RegResult:=1;
                  G.Regs[1]:=DArr[i].Len;
                  if G.Regs[1]<>nil then
                    G.VWDestroy;
                  DArr[i].Len:=G.Regs[3];
                end
              else
                G.VWDestroy;
            end;
  end;
G.RegResult:=2;  G.VWCreate;  G.VWInfinity;
G.Regs[1]:=DArr[toNode-1].Len;
//если маршрут до финальной вершины не построен
if not G.VWOrder then
  begin
    G.VWDestroy;  Result:=nil;
  end
else //если путь до финальной вершины существует
  begin
    //формируем массив, содержащий вершины маршрута
    G.VWDestroy; k:=0; i:=toNode;
    while i<>inNode do
      begin
        Inc(k); i:=DArr[i-1].outNode;
      end;
    if k>0 then
      begin
        Result:=DArr[toNode-1].Len;  SetLength(WayArr,k+1);  i:=toNode;
        repeat
          WayArr[k]:=i; i:=DArr[i-1].outNode; Dec(k);
```

```
            until (k<0);
          end
        else
          Result:=0;
      end;
    G.RegResult:=1;
    //освобождаем память
    for i:=1 to G.Count do
      if (i<>toNode) or (Result=nil) then
        begin
          G.Regs[1]:=DArr[i-1].Len;  G.VWDestroy;
        end;
    Finalize(DArr);
  end
 else
   Result:=nil;
end;{Way}
```

Предложенный вариант реализации алгоритма Дейкстры оперирует регистрами АГМ и абстрактными методами работы с регистрами, следовательно, не зависит от типа данных, используемых в качестве весовых коэффициентов рёбер (дуг) графа.

Процесс создания реальной граф-машины состоит в следующем: программисту необходимо построить класс-наследник, описать алгебры используемых данных, после чего инициализировать виртуальную машину уже описанными методами, реализующими алгебры данных (например, $< R_+ \cup \{0\}, \cdot >$ – алгебра неотрицательных действительных чисел с обычной операцией сложения, которая может быть использована для работы со взвешенными графами).

Предложенный вариант реализации абстрактной граф-машины на базе связных списков позволяет строить динамические графовые модели и решать типовые задачи теории графов независимо от типов данных, сопоставляемых элементам графа. Для работы в рамках определенной предметной области требуется лишь описать алгебру элементов и инициализировать АГМ. Данная оптимизация стала возможной благодаря отделению операций структуры от операций над элементами типа.

Список литературы

1. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Издательский дом «Вильямс», 2000. – 384 с.
2. Букачев Д.С., Мунерман В.И. О принципах реализации виртуальных алгебраических машин. //Системы компьютерной математики и их приложения: Материалы международной конференции. – Смоленск, СмолГУ, 2006. – с. 55-56.
3. Бурбаки Н. Теория множеств. – М.: Мир, 1968.
4. Левин Н.А., Мунерман В. И. Алгебраический подход к оптимизации обработки информации. – Системы и средства информатики. Спецвыпуск. Математические модели и методы информатики, стохастические технологии и системы. Москва: ИПИ РАН 2005. – с. 279-294.

References

1. Aho A., Hopkroft Dzh., Ulman Dzh. Struktury dannyh i algoritmy. – M.: Izdatel'skij dom «Vil'jams», 2000. – 384 s.
2. Bukachev D.S., Munerman V.I. O principah realizacii virtual'nyh algebraicheskih mashin. //Sistemy komp'juternoj matematiki i ih prilozhenija: Materialy mezhdunarodnoj konferencii. – Smolensk, SmolGU, 2006. – s. 55-56.
3. Burbaki N. Teorija mnozhestv. – M.: Mir, 1968.
4. Levin N.A., Munerman V. I. Algebraicheskij podhod k optimizacii obrabotki informacii. – Sistemy i sredstva informatiki. Specvypusk. Matematicheskie modeli i metody informatiki, stohasticheskie tehnologii i sistemy. Moskva: IPI RAN 2005. – c. 279-294.

Сеньков А.В. Формат базы знаний системы поддержки принятия решений для интеллектуального управления комплексными рисками в сложных организационно-технических системах // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т.2 №3(5) с. 23-34

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.89

ФОРМАТ БАЗЫ ЗНАНИЙ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ДЛЯ ИНТЕЛЛЕКТУЛАЬНОГО УПРАВЛЕНИЯ КОМПЛЕКСНЫМИ РИСКАМИ В СЛОЖНЫХ ОРГАНИЗАЦИОННО-ТЕХНИЧЕСКИХ СИСТЕМАХ[1]

Сеньков А.В.

Филиал ФГБОУ ВО "НИУ "МЭИ" в г. Смоленске, Россия (214013, г. Смоленск, Энергетический проезд, дом 1); e-mail: a.v.senkov@mail.ru

Статья посвящена разработке формата базы знаний системы поддержки принятия решений для интеллектуального управления комплексными рисками сложных организационно-технических системах. Предлагаемая структура обладает достаточной гибкостью, представлены варианты структуры для реализации в рамках базы данных, а также для реализации в виде открытой xsd-схемы, обеспечивающей возможность свободного обмена знаниями по интеллектуальному управлению рисками между всеми заинтересованными сторонами.

Ключевые слова: интеллектуальное управление рисками, база знаний, xsd.

KNOWLEDGE BASE FORMAT OF THE DECISION-MAKING SUPPORT SYSTEM FOR INTELLIGENT MANAGEMENT OF INTEGRATED RISKS IN COMPLEX ORGANIZATIONAL AND TECHNICAL SYSTEMS

Senkov A.V.

Smolensk Branch of the National Research University “Moscow Power Engineering Institute”, Russia (214013, Smolensk, street Ehnergeticheskij, 1); e-mail: a.v.senkov@mail.ru

The article is devoted to the development of the format of the knowledge base of the decision-making support system for the intelligent management of integrated risks of complex organizational and technical systems. The proposed structure has sufficient flexibility, provides options for the structure for implementation within the database, and for implementation in the form of an open xsd-scheme that provides the opportunity for free knowledge exchange on intelligent risk management among all stakeholders.

Key words: intelligent risk management, knowledge base, xsd.

В настоящее время ведутся активные работы по разработки методов, моделей и способов управления рисками в сложных организационно-технических системах [1, 2, 3]. Особенностью таких методов, моделей и способов является их ограниченность,

[1] Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-37-60059 мол_а_дк

нацеленность на решение строго определенной задачи управления рисками. Анализ баз цитирований Scopus и РИНЦ позволяет говорить о том, что до 80% интеллектуальных моделей, предлагаемых в таких материалах имеют своей целью оценивание рисков, при этом оценивание – далеко не самый значимый этап управления рисками.

В то же время в работе [4] предлагается комплексный подход к управлению рисками, основанный на рассмотрении как прелиминрного (заблаговременного) управления рисками, так и прецедентного управления рисками (управление последствиями). Кроме того, подход предполагает рассмотрение трёх аспектов управления рисками: процессного, структурного и системного. Указанные три аспекта позволяют наиболее полно идентифицировать, проанализировать, провести оценивание рисков и выработать решения по управлению ими.

В работе [5] предложена графическая нотация для представления процесса управления комплексными рисками. Такая нотация обеспечивает наглядное представление знаний о рисках, полученных в рамках разрабатываемого подхода. При этом следует учитывать, что знания как в форме фактов, так и в форме правил являются краеугольным камнем разработки систем поддержки принятия решений. Задачам разработки таких систем, функционирующих в различных условиях посвящено множество работ [6, 7], однако, в таких работах. как правило, мало внимания уделяется переносу знаний.

В качестве форматов переноса знаний в последнее время часто используются открытые форматы типа XML [8, 9]. Для разработки формата XML требуется разработать xsd-схему, основанную на понимании структуры передаваемых данных.

Структура знаний в сфере управления рисками

В рамках [4] выделяются следующие сущности в сфере управления рисками, знания о которых должны храниться, обрабатываться и передаваться.

Риск-ситуация – RS_i – *i*-я риск-ситуация $i \in 1, ..., i'$, i' – количество риск-ситуаций.

Риск-событие – RE_j – *j*-е риск-событие, $j \in 1, ..., j'$, j' – количество риск-событий.

Источник риска – SR_k – *m*-й источник риска, $k \in 1, ..., k'$, k' – количество источников рисков.

Событие – E_l – *l*-й источник риска, $l \in 1, ..., l'$, l' – количество событий.

Элемент системы – SE_m – *m*-й источник риска, $m \in 1, ..., m'$, m' – количество элементов системы.

Элемент внешней среды – EE_n – *n*-й источник риска, $n \in 1, ..., n'$, n' – количество элементов внешней среды.

Прелиминарное мероприятие – $BMes_o$ – *o*-е предиктивное мероприятие, $o \in 1, ..., o'$, o' – количество предиктивных мероприятий.

Прецедентное мероприятие – $AMes_p$ – *p*-е прецедентное мероприятие, $p \in 1, ..., p'$, p' – количество предиктивных мероприятий.

Знания об этих сущностях в соответствии с [10] представляют собой базу фактов. Базу же правил представляют непосредственно диаграммы, описанные в [5]:

1. диаграмма процессного аспекта рисков;
2. диаграмма структурного аспекта рисков;

3. диаграмма системного аспекта рисков;
4. диаграмма риск-ситуаций;
5. диаграмма прецедентного управления рисками;
6. диаграмм прелиминарного управления рисками.

Для хранения указанных фактов и правил в базе знаний предлагается использовать следующую её структуру (рисунок 1).

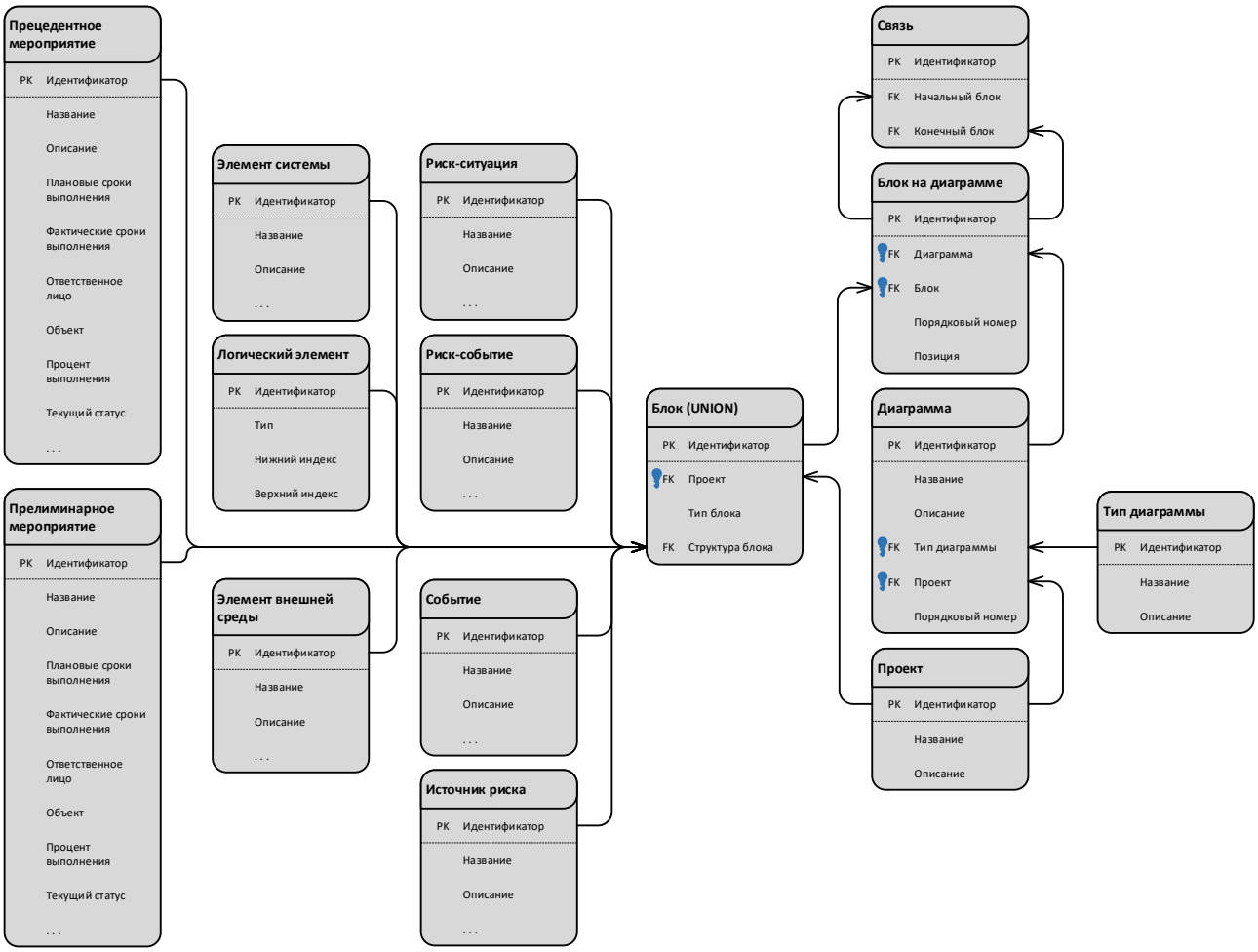


Рисунок 1 – Структура базы знаний для управления рисками

Представленная структура обладает следующими достоинствами:

1. она адаптирована для хранения в широко распространенных СУБД, поскольку представлена в формате ER-диаграммы;
2. полностью отражает все элементы нотации рисков, а значит содержит все необходимые для накопления, хранения и передачи факты и правила для управления рисками;
3. может быть легко конвертирована в открытый формат обмена данными, что обеспечит широкие возможностями по обмену знаниями между всеми заинтересованными сторонами.

XSD-схема для обмена знаниями между системами поддержки принятия решений

XSD-схема для обмена знаниями основывается на представленной на рисунке 1 структуре базы знаний и включает следующие типы элементов:

1. базовые типы – типы элементов, описывающие базовые типы данных и знаний, задающие форматы таких описаний;
2. справочные типы – типы элементов, позволяющие задать справочники и классификаторы, единые для всех пользователей открытого формата;
3. типы фактов – типы элементов, задающих факты базы знаний;
4. типы правил – типы элементов, задающих правила работы над фактами.

Рассмотрим перечисленные типы подробнее.

Базовые типы

К базовым типам относим:

- уникальный идентификатор (UUID);
- название (Name);
- описание (Description);
- положение (Position);
- точка (Point);
- массив точек (PointArray);
- срок (Period);
- процент (Percent);
- статус (Status).

Уникальный идентификатор используется для уникальной идентификации всех фактов и правил в рамках базы знаний, а также между базами знаний. Описывается в виде.

```
<xs:simpleType name="uuid">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9a-fA-F]{32}"/>
  </xs:restriction>
</xs:simpleType>
```

Название используется для задания названий элементам, представляет собой строку из 50 символов. Описывается в виде.

```
<xs:simpleType name="Name">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="50"/>
  </xs:restriction>
</xs:simpleType>
```

Описание используется для описания или комментирования объектов, представляет собой строку. Описывается в виде.

```
<xs:simpleType name="Description">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
  </xs:restriction>
</xs:simpleType>
```

Положение предназначено для задания координат блока на диаграмме. Представляет собой пару параметров «x» и «y», каждый из которых целое число, описывается в следующем виде.

```
<xs:complexType name="Position">
  <xs:sequence>
    <xs:element name="x" type="xs:integer"/>
    <xs:element name="y" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>
```

Точка используется для задания параметра типа точка. Представляет собой пару параметров «x» и «y», каждый из которых целое число, описывается в следующем виде.

```
<xs:complexType name="Point">
  <xs:sequence>
    <xs:element name="x" type="xs:integer"/>
    <xs:element name="y" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>
```

Массив точек используется для задания нечеткого значения. Представляет собой неограниченный массив точек, описывается в следующем виде.

```
<xs:complexType name="PointArray">
  <xs:sequence>
    <xs:element name="p" type="Point" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Период используется для задания сроков выполнения мероприятия. Представляет собой комбинацию из даты/времени начала и даты/времени окончания процесса, описывается в следующем виде.

```
<xs:complexType name="Period">
  <xs:sequence>
    <xs:element name="begin" type="xs:dateTime"/>
    <xs:element name="end" type="xs:dateTime"/>
  </xs:sequence>
</xs:complexType>
```

Процент используется для задания процента выполнения мероприятия. Представля-ет собой целое число от 0 до 100.

```
<xs:simpleType name="Percent">
  <xs:restriction base="xs:integer">
    <xs:minExclusive value="0"/>
    <xs:maxExclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
```

Статус используется для задания статуса выполнения мероприятия. Представляет собой список выбора: «new» (Новая), «inProcess» (В процессе), «complete» (За-вершена), «reject» (Отклонена), описывается следующим образом.

```
<xs:simpleType name="Status">
  <xs:restriction base="xs:string">
    <xs:enumeration value="new"/>
    <xs:enumeration value="inProcess"/>
    <xs:enumeration value="complete"/>
    <xs:enumeration value="reject"/>
  </xs:restriction>
</xs:simpleType>
```

Справочные типы

К справочным типам относим:

- тип диаграммы (DiagramType);
- тип блока (BlockType);
- тип логического элемента (LogicElementType).

Тип диаграммы используется для задания справочника типов диаграмм. Представляет собой структуру из полей: «id» (Идентификатор), «name» (Название), «description» (Описание). Поле «description» не является обязательным. Задается следующим образом.

```
<xs:element name="DiagramType">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Тип блока используется для задания типа блока на диаграмме. Представляет собой список выбора: «RiskSituation» (Риск-ситуация), «RiskEvent» (Риск-событие), «Event» (Событие), «RiskSource» (Источник риска), «CaseAction» (Прецедентное мероприятие), «PremilinarnyAction» (Прелиминарное мероприятие), «InternalElement» (Элемент системы), «ExternalElement» (Элемент внешней среды), «LogicElement» (Логический элемент). Задается следующим образом.

```
<xs:simpleType name="BlockType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="RiskSituation"/>
    <xs:enumeration value="RiskEvent"/>
    <xs:enumeration value="Event"/>
    <xs:enumeration value="RiskSource"/>
    <xs:enumeration value="CaseAction"/>
    <xs:enumeration value="PremilinarnyAction"/>
    <xs:enumeration value="InternalElement"/>
    <xs:enumeration value="ExternalElement"/>
    <xs:enumeration value="LogicElement"/>
  </xs:restriction>
```

```
</xs:simpleType>
```

Тип логического элемента используется для задания типа логического блока. Представляет собой список выбора. Описывается следующим образом.

```
<xs:simpleType name="LogicElementType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="And"/>
    <xs:enumeration value="Or"/>
    <xs:enumeration value="Not"/>
    <xs:enumeration value="Xor"/>
    <xs:enumeration value="Next"/>
    <xs:enumeration value="Last"/>
    <xs:enumeration value="AlwaysInFuture"/>
    <xs:enumeration value="AlwaysInPast"/>
    <xs:enumeration value="SometimeInFuture"/>
    <xs:enumeration value="SometimeInPast"/>
    <xs:enumeration value="Until"/>
    <xs:enumeration value="Unless"/>
    <xs:enumeration value="Since"/>
    <xs:enumeration value="Zince"/>
    <xs:enumeration value="Before"/>
    <xs:enumeration value="After"/>
    <xs:enumeration value="Meets"/>
    <xs:enumeration value="MetBy"/>
    <xs:enumeration value="Overlaps"/>
    <xs:enumeration value="OverlappedBy"/>
    <xs:enumeration value="During"/>
    <xs:enumeration value="Includes"/>
    <xs:enumeration value="Starts"/>
    <xs:enumeration value="StartedBy"/>
    <xs:enumeration value="Finishes"/>
    <xs:enumeration value="FinishedBy"/>
    <xs:enumeration value="Equals"/>
  </xs:restriction>
     </xs:simpleType>
```

Типы фактов

К типам фактов относим:

- риск-ситуацию (RiskSituatuation);
- риск-событие (RiskEvent);
- событие (Event);
- источник риска (RiskSource);
- прецедентное мероприятие (CaseAction);
- прелиминарное мероприятие (PreliminaryAction);
- элемент системы (InternalElement);
- элемент внешней среды (ExternalElement);
- логический элемент (LogicElement).

Все перечисленные блоки (кроме блока «Логический элемент») сами по себе являются фактами и используются в дальнейшем при построении правил. Описываются следующим образом.

```
<xs:element name="RiskSituation">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
```

```
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="RiskEvent">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Event">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="RiskSource">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="CaseAction">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
      <xs:element name="plan" type="Period"/>
      <xs:element name="actualPeriod" type="Period"/>
      <xs:element name="executor" type="Name"/>
      <xs:element name="object" type="Name"/>
      <xs:element name="percent" type="Percent"/>
      <xs:element name="status" type="Status"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="PremilinarnyAction">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
      <xs:element name="plan" type="Period"/>
```

```
      <xs:element name="actualPeriod" type="Period"/>
      <xs:element name="executor" type="Name"/>
      <xs:element name="object" type="Name"/>
      <xs:element name="percent" type="Percent"/>
      <xs:element name="status" type="Status"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="InternalElement">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="ExternalElement">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="LogicElement">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="type" type="LogicElementType"/>
      <xs:element name="minValue" type="xs:integer" minOccurs="0"/>
      <xs:element name="maxValue" type="xs:integer" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Типы правил

К типам правил относим:

- проект (Project);
- диаграмма (Diagram);
- соединение (Connection);
- блок в диаграмме (DiagramBlock);
- факт (Block).

Проект используется для описания проекта. Содержит в себе основные параметры проекта: «id» (Идентификатор), «name» (Название), «description» (Описание). А также следующие списки: «DiagramType» (Справочник типов диаграмм), «Block» (Список всех блоков проекта), «Diagram» (Список всех диаграмм проекта). Описывается следующим образом.

```
<xs:element name="Project">
```

```
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
      <xs:element ref="DiagramType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Block" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Diagram" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Диаграмма используется для описания диаграммы в проекте. Содержит в себе основные параметры диаграммы: «id» (идентификатор), «name» (название), «description» (описание), «number» (порядковый номер), «diagramTypeId» (идентификатор типа диаграммы из справочника). А также следующие списки: «DiagramBlock» (список блоков), «Connection» (список связей). Описывается следующим образом.

```
<xs:element name="Diagram">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="name" type="Name"/>
      <xs:element name="description" type="Description" minOccurs="0"/>
      <xs:element name="number" type="xs:integer"/>
      <xs:element name="diagramTypeId" type="uuid"/>
      <xs:element ref="DiagramBlock" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Connection" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Связь используется для описания связи на диаграмме. Представляет собой структуру из параметров объекта: «id» (Идентификатор), «beginDiagramBlockId» (Идентификатор блока на диаграмме, из которого выходит связь), «endDiagramBlockId» (Идентификатор блока на диаграмме, в который входит связь). Описывается следующим образом.

```
<xs:element name="Connection">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="beginDiagramBlockId" type="uuid"/>
      <xs:element name="endDiagramBlockId" type="uuid"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Блок в диаграмме используется для описания блока на диаграмме. Представляет собой структуру из параметров объекта: «id» (Идентификатор), «blockId» (Идентификатор блока из списка блоков проекта), «number» (Порядковый номер блока в диаграмме), «position» (Расположение элемента на диаграмме).

```
<xs:element name="DiagramBlock">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="blockId" type="uuid"/>
      <xs:element name="number" type="xs:integer"/>
      <xs:element name="position" type="Position"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Факт используется для описания факта в проекте. Содержит в основные параметры проекта: «id» (Идентификатор), «type» (Тип объекта). Описывается следующим образом.

```
<xs:element name="Block">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="uuid"/>
      <xs:element name="type" type="BlockType"/>
      <xs:element ref="RiskSituation" minOccurs="0"/>
      <xs:element ref="RiskEvent" minOccurs="0"/>
      <xs:element ref="Event" minOccurs="0"/>
      <xs:element ref="RiskSource" minOccurs="0"/>
      <xs:element ref="CaseAction" minOccurs="0"/>
      <xs:element ref="PremilinarnyAction" minOccurs="0"/>
      <xs:element ref="InternalElement" minOccurs="0"/>
      <xs:element ref="ExternalElement" minOccurs="0"/>
      <xs:element ref="LogicElement" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Предложенные в настоящей статьей структура базы знаний и XSD-схема для передачи знаний в открытом формате XML обладают следующими достоинствами. Структура базы знаний:

1. адаптирована для хранения в широко распространенных СУБД, поскольку представлена в формате ER-диаграммы;
2. полностью отражает все элементы нотации рисков, а значит содержит все необходимые для накопления, хранения и передачи факты и правила для управления рисками;
3. может быть легко конвертирована в открытый формат обмена данными, что обеспечит широкие возможностями по обмену знаниями между всеми заинтересованными сторонами.

В свою очередь, XSD-схема

1. позволяет выполнять обмен знаниями между различными базами знаний;
2. обеспечивает валидацию полученных знаний;
3. проста для использования с технологической точки зрения.

Список литературы

1. Борисов В.В., Сеньков А.В., Интеллектуальное управление рисками в топливно-энергетическом комплексе. – Смоленск: Универсум, 2015

2. Канев В.С., Шевцова Ю.В. Основы моделирования и управления операционными рисками в электронной коммерции и телекоммуникациях. – М.: Горячая линия – Телеком, 2015
3. Астахов А.М. Искусство управления информационными рисками.– М.: ДМК Пресс 2010
4. Сеньков А.В. Управление рисками: интеллектуальные модели, методы, средства. – Смоленск: Универсум, 2016
5. Сеньков А.В., Графическая нотация для представления процесса управления комплексными рисками // Современные наукоемкие технологии. – 2016. – № 12-1 . – С. 72-81
6. Vagin V.N., Eremeev A.P. Some Basic Principles of Design of Intelligent Systems for Supporting Real-Time Decision Making // J. of Computer and System Sciences International, Vol. 40, No. 6, 2001, pp. 953-961.
7. Eremeev A.P., Vagin V.N. A Real Time Decision Support System for Monitoring and Management of a Complex Object Using Parallel Processing // Proc. of the IEEE Int. Conf. Artificial Intelligent Systems IEEE AIS'02, Sept. 5-10, 2002, Divnomorskoe, Russia, pp. 139-144
8. W.M.P. van der Aalst, H.M.W. Verbeek, and A. Kumar. Verification of XRL: An XML-based Workflow Language. In W. Shen, Z. Lin, J.P. Barthes, and M. Kamel, editors, Proceedings of the 6th International Conference on CSCW in Design, pages 427-432. NRC Research Press, Ottawa, Canada, 2001
9. W.M.P. van der Aalst and A. Kumar. XML Based Schema Definition for Support of Inter-organizational Workflow. Information Systems Research, 14(1):23-46, 2003
10. Поспелов Д.А. Искусственный интеллект. Справочник. Книга 2. Модели и методы. М.: Радио и связь, 1990.

References

1. Borisov VV, Senkov AV, Intellectual Risk Management in the Fuel and Energy Complex. - Smolensk: Universum, 2015
2. Kanev VS, Shevtsova Yu.V. Basics of modeling and management of operational risks in e-commerce and telecommunications. - M .: Hot line - Telecom, 2015
3. Astakhov A.M. The art of managing information risks .- Moscow: DMK Press 2010
4. Senkov A.V. Risk management: intellectual models, methods, tools. - Smolensk: Universum, 2016
5. Senkov AV, Graphical notation for the presentation of the process of managing complex risks // Modern high technology. - 2016. - No. 12-1. - P. 72-81
6. Vagin V.N., Eremeev A.P. Some Basic Principles of Design of Intelligent Systems for Supporting Real-Time Decision Making // J. of Computer and System Sciences International, Vol. 40, No. 6, 2001, pp. 953-961.
7. Eremeev A.P., Vagin V.N. A Real Time Decision Support System for Monitoring and Management of a Complex Object Using Parallel Processing // Proc. of the IEEE Int. Conf. Artificial Intelligent Systems IEEE AIS'02, Sept. 5-10, 2002, Divnomorskoe, Russia, pp. 139-144
8. W.M.P. van der Aalst, H.M.W. Verbeek, and A. Kumar. Verification of XRL: An XML-based Workflow Language. In W. Shen, Z. Lin, J.P. Barthes, and M. Kamel, editors, Proceedings of the 6th International Conference on CSCW in Design, pages 427-432. NRC Research Press, Ottawa, Canada, 2001
9. W.M.P. van der Aalst and A. Kumar. XML Based Schema Definition for Support of Inter-organizational Workflow. Information Systems Research, 14 (1): 23-46, 2003
10. Pospelov DA Artificial Intelligence. Directory. Book 2. Models and methods. M .: Radio and Communication, 1990.

Зернов М.М., Зернова Т.О., Панкратова Е.А. Алгоритм каскадного нечёткого логического вывода типа Мамдани // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т.2 №3(5) с. 35-44

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.827

АЛГОРИТМ КАСКАДНОГО НЕЧЁТКОГО ЛОГИЧЕСКОГО ВЫВОДА ТИПА МАМДАНИ

Зернов М.М., Зернова Т.О., Панкратова Е.А.

Филиал ФГБОУ ВО "НИУ "МЭИ" в г. Смоленске, Россия (214013, г. Смоленск, Энергетический проезд, дом 1); e-mail: zmmioml@yandex.ru

В статье предложен общий алгоритм каскадного нечёткого логического вывода на структурах типа Мамдани, реализующий предложенный ранее способ, отличающийся учётом взаимной зависимости между сигналами на основе дискретизированного представления входных и выходных нечётких множеств и позволяющий уменьшить мощность результата каскадного вывода.

Разработан алгоритм анализа каскада FIS-структур, реализующий этап алгоритма каскадного нечёткого логического вывода, отличающийся представлением каскадной системы в виде информационного графа. Алгоритм позволяет выделить ветвящиеся сигналы со взаимодействующими потомками и определить последовательность срабатывания элементов каскада.

Рассмотрены вопросы оценки снижения неточности результата каскадного нечёткого логического вывода при использовании предложенного алгоритма. В частности, предложена методика оценки и представлены результаты вычислительного эксперимента для каскадов различной сложности.

Ключевые слова: каскадный нечёткий логический вывод, нечёткий логический вывод Мамдани, уменьшение неопределённости.

ALGORITHM OF A CASCSDED FUZZY INFERENCE MAMDANI-TYPE

Zernov M.M., Zernova T.O., Pankratova E.A.

Smolensk Branch of the National Research University "Moscow Power Engineering Institute", Russia (214013, Smolensk, street Ehnergeticheskij, 1); e-mail: zmmioml@yandex.ru

General algorithm of a cascaded fuzzy inference Mamdani-type structures is proposed. The algorithm implements the previously proposed method, characterized by taking into account mutual dependencies between signals based on sampled representations of the input and output fuzzy sets, which reduces the cardinality of the result of the cascade output.

The algorithm of the analysis of the cascade FIS-structures is developed. It implements the algorithm's stage of the cascaded fuzzy inference, wherein the cascaded system is represented in the form of information graph. The algorithm allows to identify branching signals with interacting descendants, and to determine the sequence of activation of elements of the cascade.

The questions of the uncertainty reduction evaluation of the result of cascaded fuzzy inference when using the proposed algorithm are considered. In particular, the techniques of evaluation is proposed and the results of numerical experiments for cascades of varying complexity are presented.

Key words: cascaded fuzzy inference, fuzzy inference Mamdani-type, reducing the uncertainty.

Одной из основных проблем каскадного нечёткого логического вывода типа Мамдани с исключённым этапом дефаззификации [1, 5] является существенное накопление нечёткости в выходных сигналах FIS-структур от первых элементов каскада к последним. Причинами накопления нечёткости являются как свойства процедуры нечёткого логического вывода как нечёткой функции от чётких и нечётких переменных, так и взаимная зависимость аргументов структур. Последний случай имеет место, когда сигналы, взаимодействующие в качестве аргументов одной структуры, имеют общего предка и, следовательно, должны рассматриваться в контексте значений их общего предка. Такие сигналы-предки будем называть ветвящимися сигналами со взаимодействующими потомками – ВСВП-сигналы (сюда также относим случай, когда сигнал-предок взаимодействует с собственным потомком).

В статье [3], авторы рассматривают подходы к уменьшению неопределённости результата каскадного нечёткого логического вывода типа Мамдани с исключённым этапом дефаззификации за счёт учёта взаимной зависимости сигналов. Там же предложен способ, основанный на дискретизации ВСВП-сигналов, нахождении всевозможных сочетаний их элементов и агрегации результатов, полученных для отдельных сочетаний.

В данной статье предлагается 2 алгоритма, реализующие как способ в целом, так и этап анализа каскада FIS-структур как информационного графа.

1. Общий алгоритм каскадного нечёткого логического вывода типа Мамдани

Введём ряд обозначений. Под $S = \{s_1, s_2, ..., s_n\}$ будем понимать множество всех сигналов каскада. В него включаются множества входных *X* и выходных *Y* сигналов каскада в целом:

$$X, Y \subset S, X \cap Y = \varnothing.$$

Множество всех структур нечёткого логического вывода обозначим как $FIS = \{ f_1, f_2, ..., f_m \}$.

Для каждой структуры $f \in FIS$ определены множества входов $Inputs(f)$ и выходов $Outputs(f)$, совокупность которых по всем *FIS* и даёт полное множество сигналов:

$$S = \bigcup_{f \in FIS} Inputs(f) \cup Outputs(f).$$

Можно более строго определить множества входов и выходов каскада:

$$X = S \setminus \bigcup_{f \in FIS} Outputs(f),$$

$$X = S \setminus \bigcup_{f \in FIS} Inputs(f).$$

При работе с массивами, как одномерными, так и двумерными индексы элементов будем писать в квадратных скобках после имени массива, например, $T(i, j)$. При этом в качестве значений индексов допускается использовать сразу множества/диапазоны, подразумевая что при присваивании, набору элементов с указанными диапазонами строк и столбцов сопоставляется массив аналогичного размера.

На рисунке 1 представлена схема общего алгоритма каскадного нечёткого логического вывода типа Мамдани, реализующего способ вывода, учитывающий взаимную зависимость между сигналами.

На схеме использованы следующие обозначения.

Answer(s_j) - результирующее нечёткое множество для выходного сигнала каскада s_j.

Combs(*Sets*) - функция, возвращающая всевозможные сочетания синглтонов – элементов входного набора нечётких множеств *Sets*.

NewData – наборы данных, по которым осуществляется расчёт, и в которые помещается результат на каждом такте – массив размерности $N_i \times n$, элементами которого являются нечёткие множества значений сигналов. При этом не все элементы могут быть заполнены.

N_i – число наборов данных на такте *i*.

OldData – наборы данных, сформированные на предыдущем такте работы каскада – массив множеств размерности $N_{i-1} \times n$.

FisTacts – массив-столбец $m \times 1$, где *FisTacts*(*f*) - № такта, на котором рассчитывается FIS *f*.

Аналогично, Tacts – массив-столбец $n \times 1$, где *Tacts*(*i*) – № такта, на котором рассчитывается сигнал *i*.

Size (*FS*) – число дискрет в нечётком множестве *FS*.

Val(*s*) – нечёткое значение сигнала *S*.

Br – список ВСВП-сигналов.

Алгоритм работает по следующему принципу. После этапа инициализации переменных и подготовки структур данных выполняется анализ каскада с применением вспомогательного алгоритма, который будет рассмотрен ниже. По результатам анализа формируется список ВСВП-сигналов, определяются такты, на которых срабатывают FIS, формируются сигналы, общее число тактов работы каскада.

Первоначально, алгоритм начинает работу с единственным набором данных, соответствующих входным данным каскада (условно, такт №0). Затем итерационно, на каждом такте, определяются ВСВП-сигналы, сформированные на предыдущей итерации. Если таких сигналов нет, то новые наборы данных для данного такта получаются простым переносом из предыдущих. В противном случае, число наборов данных увеличивается по следующему принципу.

Для каждого набора из предыдущего такта, для всех сформированных на этом такте ВСВП-сигналов, формируются всевозможные сочетания синглтонов – элементов множества значений ВСВП-сигналов. Исходный набор заменяется целым массивом наборов, в котором значения всех сигналов, кроме только что сформированных ВСВП-сигналов, копируются целиком. Для сформированных на предыдущем такте ВСВП-сигналов указываются сочетания синглтонов (по одному на каждый новый набор).

По окончании работы цикла по тактам, для всех выходных сигналов каскада проводится агрегация и (при необходимости), дефаззификация.

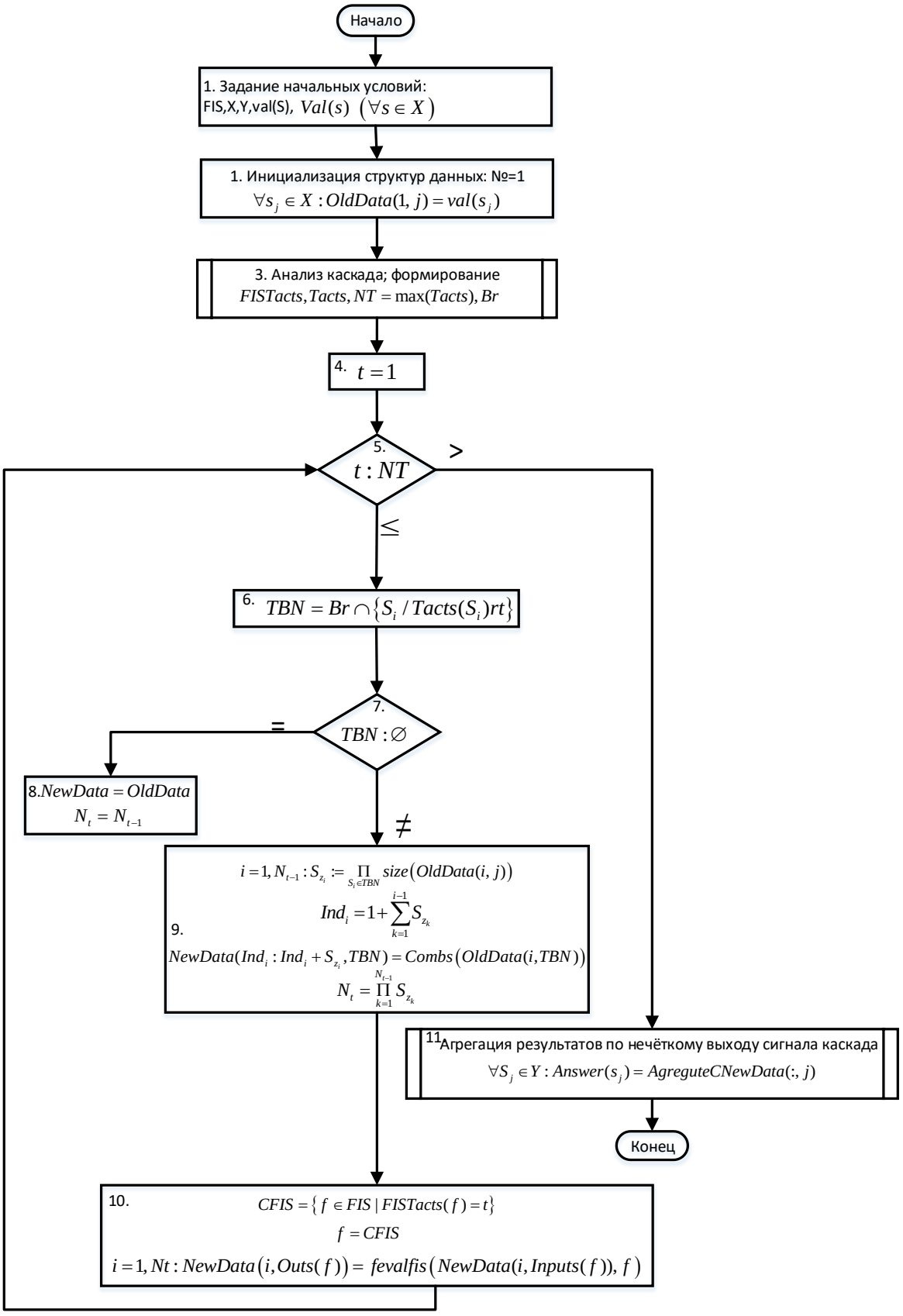


Рисунок 1 – Схема общего алгоритма каскадного нечёткого логического вывода

2. Алгоритм анализа каскада структур нечёткого логического вывода типа Мамдани

Важным этапом предложенного общего алгоритма нечёткого логического вывода типа Мамдани является анализ каскада. Рассмотрим алгоритм анализа на основе представления каскада в виде информационного графа.

Информационный граф образуют элементы потока информации, связанные отношениями вхождения и предшествования. В конечном итоге 2 элемента связаны, если для формирования второго элемента необходим первый [4]. Для заданной матрицы смежности информационного графа, основными инструментами анализа являются:

- матрица достижимости, определяемая как (для случая отсутствия контуров):

$$M_{\sum} = \sum_{\lambda=1}^{n-1} M^{\lambda} = \sum_{\lambda=1}^{\lambda max} M^{\lambda},$$

$$\lambda max = \max\left(\lambda < n \mid M^{\lambda} \neq \mathrm{O}\right), m.e. M^{\lambda max} \neq \mathrm{O}, M^{\lambda max+1} = \mathrm{O}.$$

- промежуточные матрицы $M^{\lambda}(\lambda \le \lambda max)$.

Среди прочих показателей, на их основе находится порядок вершин π_j, определяемый выражением:

$$\sigma_j(\lambda) = \sum_j M^{\lambda}(i, j),$$

$$\sigma_j(\pi_j) \neq 0, \sigma_j(\pi_j + 1) = 0.$$

Порядок вершины (элемента информационного потока) соответствует номеру такта, на котором получается рассматриваемый элемент.

Множества непосредственных и всех потомков для выбранной вершины находятся на основе матриц смежности и достижимости, как прямое отображение 1 порядка и прямое транзитивное замыкание (без учёта самой вершины) соответственно [2].

Итак, имеем информационный граф *G* построенный на множестве сигналов:

$$G = \langle S, E \rangle,$$

$$(s_1, s_2) \in E \leftrightarrow \exists! f \in FIS : s_1 \in Inputs(f), s_2 \in Outputs(f).$$

Признаком отнесения сигнала *s'* к ВСВП-сигналам является наличие среди его непосредственных потомков хотя бы 2-х, у которых пересекаются прямые транзитивные замыкания:

В результате, алгоритм анализа каскада состоит из следующих этапов.

1. Инициализация, выполнить 1.1-1.4:

1.1. Для всех $i = \overline{1,m} : Tacts(i) = n;$

1.2. $M_{\sum} = \mathbf{O};$

1.3. $TM = E(n);$

1.4. $Br = \varnothing.$

2. Для всех $i = \overline{1,n}$ выполнить 2.1-2.3:

2.1. $TM = TM \cdot \mathrm{M};$

2.2. Для всех $j = \overline{1,n}$ выполнить:

Если $\sum_{k=1}^{n} TM(k, j) = 0,$ то $Tacts(j) = \min\left(Tacts(j), i-1\right).$

2.3. $M_{\Sigma} = M_{\Sigma} + TM.$

3. Для всех $s_i \in S \setminus X : FISTacts(Outputs^{-1}(s_i)) = Tacts(i)$

4. $C = \left\{ s_i \in S \mid \sum_{j=1}^{n} M(i, j) \geq 2 \right\}.$

5. Для всех $s_i \in C$ выполнить 5.1-5.2:

5.1. $D = \left\{ s_j \in S \mid M(i, j) = 1 \right\};$

5.2. Для всех пар $(S_j, S_k), S_j, S_k \in D, j \neq k$ выполнить 5.2.1-5.2.3:

5.2.1. $\Gamma^{+}(s_j) = \left\{ s_r \in S \mid M_{\Sigma}(j, r) > 0 \right\};$

5.2.2. $\Gamma^{+}(s_k) = \left\{ s_r \in S \mid M_{\Sigma}(k, r) > 0 \right\};$

5.2.3. Если $\Gamma^{+}(S_j) \cap \Gamma^{+}(S_k) \neq \varnothing$, то $B_r = B_r \cup \left\{ S_i \right\}.$

Здесь $E(n)$ – единичная матрица размера $n \times n$, а $Outputs^{-1}(s_i)$ – функция обратная $Outputs$, возвращающая FIS-структуру по её выходному сигналу.

3. Оценка снижения неточности при применении алгоритма

Эффект от применения способа учёта взаимной зависимости аргументов оценивается как уменьшение неточности нечёткого результата вывода, при использовании предлагаемого способа в сравнении со способом без учёта взаимной зависимости аргументов.

В качестве меры неточности выбрана мера по Хигаши-Клиру (дискретный вариант) [6]:

$$Unc(A) = \sum_{\alpha=0}^{\alpha_{\max}} \log_2 \left| A_{\alpha} \right|$$

Методика включает в себя следующие этапы.

1. Задать требуемую сложность каскада FIS-структур:

- число структур NFIS;
- максимальное число входов MaxInputs;
- число ВСВП-сигналов NBranch.

2. Задать число опытов:

- число генерируемых каскадов NCasc;
- число наборов данных для каждого каскада NX;

3. Случайно сгенерировать матрицы связей каскадов:

- задать матрицу M1 размера NFISxNFIS с нулями по главной диагонали и ниже главной диагонали и случайной расстановкой единиц выше главной диагонали;
- взвесить ненулевые значения номерами столбцов (т.е. номерами структур);

- определить полученное число ВСВП-сигналов, если оно не равно NBranch – сгенерировать заново;

- дополнить матрицу входными сигналами;

4. Для каждой матрицы сгенерировать набор структур:

4.1 для структур от первого такта до последнего сгенерировать обучающие выборки:

- для каждой структуры сгенерировать полином 2-й степени от входных сигналов со случайными коэффициентами в диапазоне [-1,1];

- для входных структур берём случайные входные значения из диапазона [-1,1], дополняем крайними значениями;

- для остальных – из диапазона выходных значений сгенерированных структур;

- выходное значение рассчитываем в соответствии с заданным полиномом;

4.2. обучить ANFIS-структуры (Сугэно 0-го порядка);

4.3. размыть выходные константы обученных структур.

5. Для каждого каскада:

5.1. случайно задать *NX* наборов значений входных сигналов каскада: задаём чёткие значения и представляем их синглтонными множествами;

5.2. для каждого набора и каждого выхода Y_j каскада:

- рассчитать нечёткие выходы каскада обычным способом (Y_j') и с учётом взаимодействия (Y_j'');

- рассчитать оценки неточности выходов по Хигаши – Клиру: $Unc(Y_j')$и $Unc(Y_j'')$;

- рассчитать относительное уменьшение неточности: ($Unc(Y_j')$-$Unc(Y_j'')$)/ $Unc(Y_j')$.

6. Найти минимальное *minUncDiff*, среднее a*vgUncDiff* и максимальное *maxUncDiff* значение относительного уменьшения неточности.

На рисунке 2 представлен пример сгенерированного каскада из 5 структур с 3-мя ВСВП-сигналами. На рисунке 3 изображены результирующие нечёткие множества выхода каскада, полученные с учётом взаимодействия потомков ВСВП-сигналов и без для следующих чётких входных значений каскада: X_1 = 0.6294, X_2 = -0.8049, X_3 = -0.6848, X_4 = -0.7162, X_5 = 0.3115.

Сравнительная оценка неточности результатов представлена в таблице 1.

Таблица 1 – Оценки неточности нечёткого значения выхода каскада

Результаты	Обычный способ	С учётом взаимодействия	Относительная разница
Неточность по Хигаши-Клиру	277.7246	252.5464	9,07%

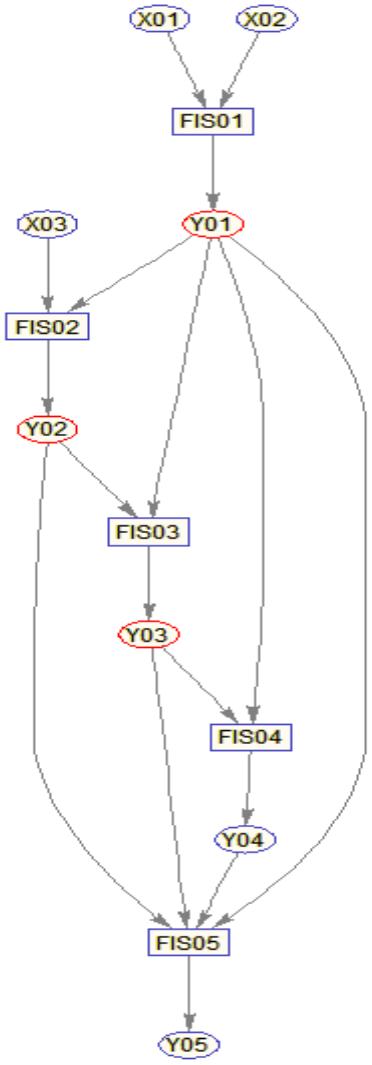


Рисунок 2 – Случайный каскад из 5 FIS-структур с 3-мя ВСВП-сигналами

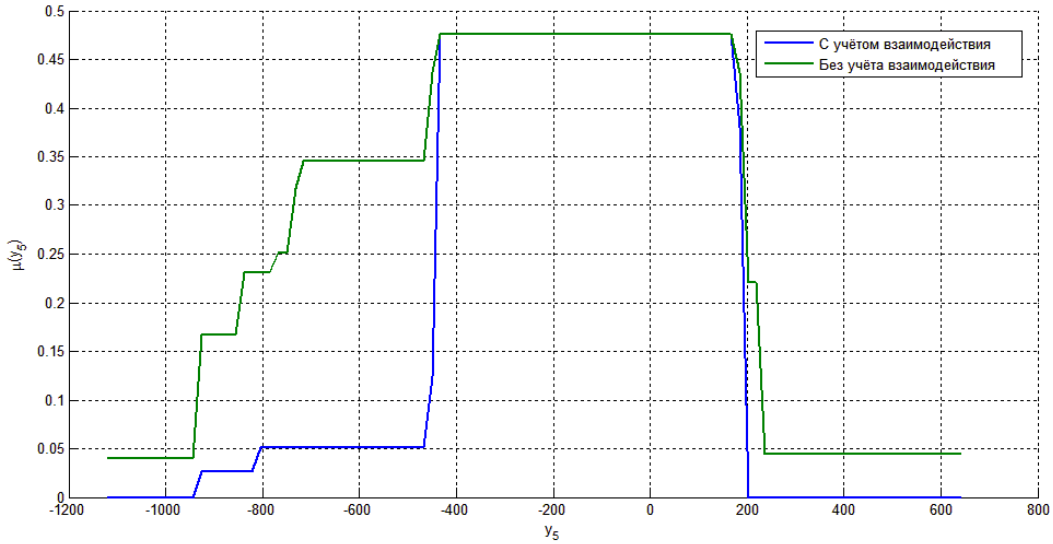


Рисунок 3 – Нечёткие множества выхода каскада, рассчитанные разными способами

В таблице 2 приведены результаты 100 опытов, проведённых в соответствии с

представленной методикой для каскадов из 5 FIS-структур с числом ВСВП-сигналов равным 2 и 3 (учитываются только выходы, зависящие от взаимодействующих потомков ВСВП-сигналов).

Таблица 2 – Результаты эксперимента по оценке уменьшения неточности

Сложность каскада			Число опытов		Относительное уменьшение неточности		
NFIS	MaxInputs	NBranch	NCasc	NX	minUncDiff	avgUncDiff	maxUncDiff
5	5	2	20	5	0.4%	5,41 %	14,34 %
5	5	3	20	5	1,38 %	8,51 %	23,52 %

Как видно, снижение неточности тем сильнее, чем выше число ВСВП-сигналов, при этом как средний, так и максимальный выигрыш от учёта взаимодействия достигает значительных величин, способных существенно повлиять на дефаззифицированный результат или дальнейшие выводы по результату анализа нечёткого выхода каскада. На нижнюю границу оценки снижения неточности оказывает влияние то, что не все выходы всех случайных каскадов являются результатом одновременного взаимодействия сразу всех потомков всех ВСВП-сигналов.

Предложен алгоритм каскадного нечёткого логического вывода на структурах типа Мамдани с исключенным этапом дефаззификации, реализующий предложенный ранее способ каскадного нечёткого логического вывода, отличающийся учётом взаимной зависимости между сигналами на основе дискретизированного представления входных и выходных нечётких множеств и позволяющий уменьшить мощность результата каскадного вывода за счет исключения из рассмотрения невозможных сочетаний элементов множеств взаимодействующих нечётких аргументов.

Разработан алгоритм анализа каскада FIS-структур, реализующий этап алгоритма каскадного нечёткого логического вывода, отличающийся представлением каскадной системы в виде информационного графа и позволяющий определить последовательность срабатывания структур и найти ветвящиеся сигналы со взаимодействующими потомками.

Предложена методика оценки снижения неточности результата каскадного нечёткого логического вывода при использовании предложенного алгоритма относительно обычного способа расчёта, не учитывающего взаимодействия сигналов.

По результатам вычислительного эксперимента, проведённого в соответствии с представленной методикой, показано существенное уменьшение неточности в среднем и лучшем случаях. Отмечен рост эффективности применения способа при увеличении числа ВСВП-сигналов. Что позволяет говорить о достижении цели исследования.

Список литературы

1. Борисов В. В., Круглов В. В., Федулов А. С. Нечеткие модели и сети. - 2-е изд., стереотип. - М.: Горячая линия - Телеком, 2012. - 284 с.
2. Волченская Т.В., Князьков В.С. Компьютерная математика: Часть 2 Теория графов/ Учебн. пособ. - Пенза: Изд-во Пенз. ун-та, 2002.- 101 с.

3. Зернов М.М., Зернова Т.О. Способ каскадного нечёткого логического вывода типа Мамдани//Международный журнал информационных технологий и энергоэффективности. – 2017. – Т.2 №2(4) с. 2-15
4. Меньков А.В. Теоретические основы автоматизированного управления / А.В. Меньков, В.А. Острейковский – М.: Издательство Оникс, 2005. – 640 с.
5. Babuška R. Fuzzy modeling for control. – Springer Science & Business Media, 2012. – Т. 12. - pp. 21-24.
6. Higashi M., Klir G. J. Measures of uncertainty and information based on possibility distributions //International Journal of General Systems. – 1982. – Т. 9. – №. 1. – С. 43-58.

References

1. Borisov V.V., Kruglov V.V., Fedulov A.S. “Nechetkie modeli i seti”(“Fuzzy models and networks”). Izd. 2 stereotip. - M.: Goryachaya liniya - Telekom, 2012. - 284 p.
2. Volchenskaya T.V., Knyaz’kov V.S. “Komp’yuternaya matematika”(“Computer mathematics”): Chast’ 2 Teoriya grafov/ Uchebn. Posob. – Penza: Izd-vo Penz. Un-ta, 2002. – 101 p.
3. Zernov M.M., Zernova T.O. “Sposob kaskadnogo nechyotkogo logicheskogo vyvoda tipa Mamdani” (“Method of a cascaded fuzzy inference Mamdani-type”)// Mezhdunarodnyj zhurnal informacionnyh tekhnologij i ehnergoehffektivnosti. – 2017. – vol.2 iss. 2(4) pp. 2-15
4. Men’kov A.V. “Teoreticheskie osnovy avtomatizirovannogo upravleniya”(“Theoretical foundations of automated control”) / A.V. Men’kov, V.A. Ostrejkovskij – M.: Izdatel’stvo Oniks, 2005. – 640 p.
5. Babuška R. Fuzzy modeling for control. – Springer Science & Business Media, 2012. – V. 12. - pp. 21-24.
6. Higashi M., Klir G. J. Measures of uncertainty and information based on possibility distributions //International Journal of General Systems. – 1982. – V. 9. – №. 1. – pp. 43-58.

Симоненков П.С., Свириденков К.И. Способ фрактального сжатия изображений с модифицированной схемой покрытия ранговых блоков доменными // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т.2 №3(5) с.45-50

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.627

СПОСОБ ФРАКТАЛЬНОГО СЖАТИЯ ИЗОБРАЖЕНИЙ С МОДИФИЦИРОВАННОЙ СХЕМОЙ ПОКРЫТИЯ РАНГОВЫХ БЛОКОВ ДОМЕННЫМИ

Симоненков П.С., Свириденков К.И.

Федеральное государственное бюджетное образовательное учреждение высшего образования Смоленский Филиал Московского Энергетического Института, Россия (214013, г. Смоленск, пр-д. Энергетический, 1); e-mail: c-mao@mail.ru

Статья посвящена применению дерева квадрантов в методе фрактального сжатия изображений. В ней разобрана схема покрытия ранговых блоков доменными на основе дерева квадрантов и особенности применения данного способа. В ходе эксперимента показано, что наилучших результатов с помощью данного способа можно достичь в случае с изображениями, имеющими большое число монотонных областей.

Ключевые слова: фрактальное сжатие изображений, схема покрытия ранговых блоков доменными, дерево квадрантов.

FRACTAL IMAGE COMPRESSION METHOD WITH MODIFIED SCHEME OF COVERING RANK BLOCKS BY DOMAIN

Simonenkov P.S., Sviridenkov K.I.

Federal State Educational Institution of Higher Education Smolensk Branch of Moscow Power-Engineering Institute, Russia (214013, Smolensk, Energetichesky pass., 1); e-mail: c-mao@mail.ru

The article is devoted to the use of quadtree in fractal image compression method. It includes review of the scheme of covering rank blocks by domain, based on a quadtree, and features of using fractal image compression method the given way. Experiments show that the best results with this method can be achieved in case of images that have a large number of monotonous regions.

Key words: fractal image compression, scheme of covering rank blocks by domain, quadtree.

Для сжатия графических файлов широко используется метод фрактального сжатия [1,2]. Однако, классический способ фрактального сжатия [3] имеет недостаток, связанный с тем, что при установке малых размеров ранговых блоков монотонные области кодируются большим количеством информации. Коэффициент сжатия изображения в таком случае относительно низок [4]. В то же время при установке больших размеров ранговых блоков мелкие детали изображения часто теряются. В данной статье для устранения указанного

недостатка предлагается модифицированный способ фрактального сжатия, основанный на использовании дерева квадрантов [5].

Основная идея модифицированного способа состоит в следующем: сначала пользователем задается допустимая разница между ранговым и доменным блоком. Затем, на первой итерации программы ранговые блоки выбираются максимально большими [6]. В случае если для них находится доменный блок, из которого их можно восстановить с потерями, не превышающими заданную допустимую разницу, то происходит переход к следующему ранговому блоку такого же размера. Если же такой блок не находится, то ранговый блок разбивается на блоки со стороной в 2 раза меньше начальной, и уже для них ищутся доменные блоки, из которых их можно восстановить. Таким образом, большие монотонные блоки будут кодироваться малым числом записей в результирующем файле, а высоко детализированные области будут кодироваться большим числом записей, позволяющим гарантировать высокое качество восстановленного изображения.

На рисунке 1 представлен пример разбиения изображения на ранговые блоки на различных итерациях (5 итерация на рисунке а), последняя, 16-ая – на рисунке б)) с учетом допустимой ошибки.

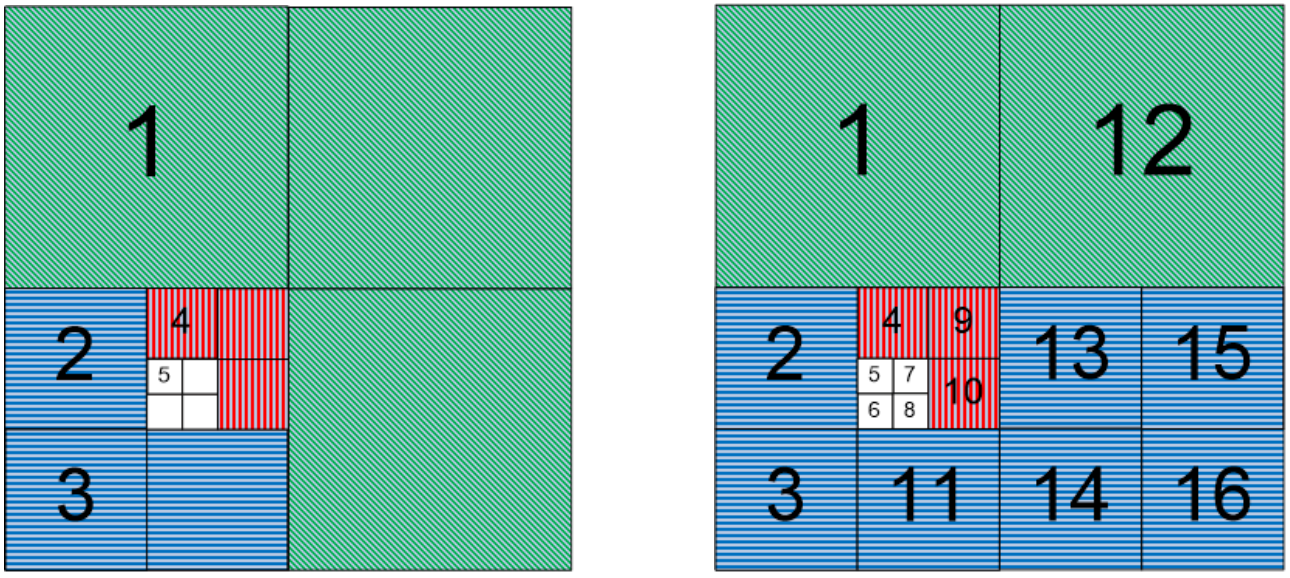


Рисунок 1 – Порядок разбиения изображения на ранговые блоки и занесения записей о ранговых блоках в файл

На рисунке показано, как меняется нумерация блоков при разбиении на меньшие с целью улучшения качества восстановленного изображения. На 5 итерации номера записей присвоены тем ранговым блокам, для которых уже удалось найти подходящий доменный блок и его преобразование, также показано разбиение изображения на ранговые блоки с учетом уже записанных. На последней итерации всем ранговым блокам присвоены номера записей, кроме того, в сравнении с рисунком, соответствующим 5 итерации, пришлось разбить правый нижний ранговый блок на 4 меньших блока.

Запись о ранговом блоке в классическом способе фрактального сжатия содержит следующую информацию (см. рисунок 2):

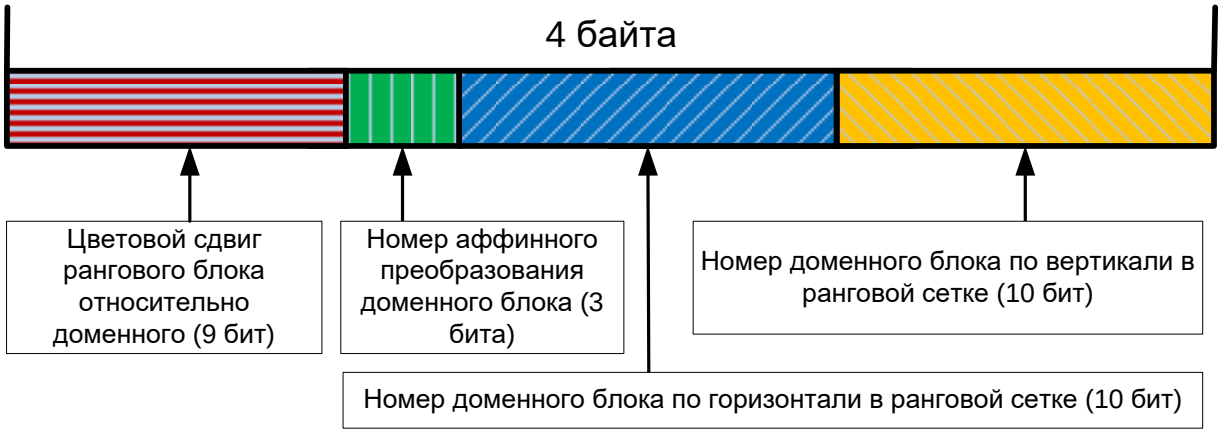


Рисунок 2 – Стандартный формат записи сжатого изображения

Однако при использовании модифицированного способа на основе дерева квадрантов, кроме информации, представленной на рисунке 2, также нужно запоминать и размер рангового блока. Это можно сделать, не увеличивая размер записи.

В файл заносится значение 1024/r (где r – сторона рангового блока). И это значение прибавляется к номеру доменного блока по строке в ранговой сетке. Например, для рангового блока со стороной 512 (максимальная) позиция доменного блока по строке может быть только одна: 0000000000. К ней прибавляется 0000000010 (1024/512=2, в двоичной системе 10). Для рангового блока со стороной 256 возможны уже варианты 00, 01 и 10. Но к ним будет прибавляться 100. Таким образом, в выходном 10-битном значении первый единичный бит всегда кодирует размер рангового блока, а номер доменного блока по строке в ранговой сетке – это все остальное значение, без первой единицы.

На рисунке 3 показан процесс восстановления изображения, сжатого с помощью классического способа фрактального сжатия, а на рисунке 4 - сжатого с помощью способа с применением дерева квадрантов.

Рисунок 3 – Процесс восстановления изображения, сжатого при помощи классического способа фрактального алгоритма сжатия

Рисунок 4 – Процесс восстановления изображения, сжатого при помощи способа фрактального алгоритма сжатия с измененной схемой покрытия ранговых блоков доменными

Был проведен сравнительный анализ классического способа с предложенным модифицированным способом. Метод фрактального сжатия относится к классу методов сжатия изображений с потерями [7]. Кроме размера сжатого файла важным показателем является качество восстановленного изображения [8]. Оценка данного показателя осуществляется с помощью показателя ошибки Δ. Расчёт данной ошибки осуществляется на основе показателей интенсивности для каждого из цветов RGB-палитры [9]. Показатель ошибки Δ рассчитывается как среднеквадратическое отклонение [10] интенсивности цветовой составляющей точек сжатого изображения от интенсивности точек исходного файла:

$$\Delta = \sqrt{\sum_{i=0}^{n}\sum_{c=1}^{3}(R_{1ci} - R_{2ci})^2} /(3 \cdot n),$$

где i – номер текущего пиксела,

n – количество пикселов в изображении,

c – номер цветовой составляющей,

R_{1i} – яркость i-го пиксела в оригинальном изображении,

R_{2i} – яркость i-го пиксела в сжатом изображении.

Для сравнения работы классического способа фрактального сжатия и способа с применением дерева квадрантов использовались пять графических образцов размера 256x256: портрет, пейзаж, натюрморт, градиентная заливка и монотонная заливка.

По данным, описывающим указанные графические образцы, с помощью формулы (1) была рассчитан показатель ошибки Δ, позволяющий оценить качество восстановленных изображений. Результаты расчёта представлены в таблице 1, где k – коэффициент сжатия,

СФА – классический способ фрактального сжатия, КФА – способ фрактального сжатия с применением дерева квадрантов.

Таблица 1 – Параметры сжатия образцов графических файлов

Класс изображения	**Коэффициент сжатия k**		Δ	
	СФА	**КФА**	**СФА**	**КФА**
Портрет	0,086	0,068	6,5	6,7
Пейзаж	0,086	0,085	5,2	5,5
Натюрморт	0,086	0,061	6,2	6,6
Градиентная заливка	0,086	0,021	1,2	1,7
Монотонная заливка	0,086	0,0001	0	0

Использование дерева квадрантов привело к некоторому уменьшению размера сжатых файлов, однако вместе с этим качество восстановленных изображений осталось практически тем же. На рисунке 5 приведен пример исходного изображения (натюрморт) и восстановленных после сжатия его с использованием разных способов. Стоит отметить, что потери качества наглядно практически не видны.

а) б) в)

Рисунок 5 – Оригинальное изображение (а); изображение, сжатое и восстановленное классическим способом фрактального сжатия (б); изображение, сжатое и восстановленное способом фрактального сжатия с использованием дерева квадрантов (в)

В статье экспериментально показано, что предложенный модифицированный способ ни в чем не уступает классическому. Наилучшие результаты применение метода квадратичного дерева дает для крупных изображений с большими монотонными областями. Главным положительным эффектом его применения является уменьшение размеров сжатых файлов относительно размеров файлов, сжатых с помощью классического способа фрактального сжатия.

Список литературы

1. S. Welstead. Fractal and Wavelet Image Compression Techniques. // Society of Photo-Optical Instrumentation Engineers (SPIE) Bellingham, WA, USA. – 1999
2. Шарабайко М.П., Осокин А.Н. Фрактальное сжатие изображений. Реализация и исследование алгоритмов // LAP Lambert Academic Publishing. – 2012
3. Д.С. Ватолин. Алгоритмы сжатия изображений. - Методическое пособие. Москва 1999
4. Симоненков П.С., студ.; рук. К.И. Свириденков. Анализ программных средств для сжатия графических файлов // Энергетика, информатика, инновации - 2016. Сб трудов VI -ой Межд. науч.-техн. конф. студентов и аспирантов. В 3 т. Т 1. – 2016. - с. 339-343
5. Raphael Finkel and J.L. Bentley (1974). «Quad Trees: A Data Structure for Retrieval on Composite Keys». Acta Informatica 4 (1): 1–9. DOI:10.1007/BF00288933
6. Симоненков П.С., студ.; рук. К.И. Свириденков. Фрактальное сжатие графических файлов с применением метода квадратичного дерева // Интеллектуальные информационные технологии, энергетика и экономика. Сб трудов XIV -ой Межд. науч.-техн. конф. студентов и аспирантов
7. Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. — Диалог-МИФИ, 2002.
8. Монич Ю. И., Старовойтов В. В. Оценки качества для анализа цифровых изображений // Искусственный интеллект. 2008. № 4. С. 376 — 386.
9. Синтез цвета // Фотокинотехника: Энциклопедия / Гл. ред. Е. А. Иофис. — М.: Советская энциклопедия, 1981.
10. Ивченко Г.И., Медведев Ю.И. Введение в математическую статистику. — М. : Издательство ЛКИ, 2010.

References

1. S. Welstead. Fractal and Wavelet Image Compression Techniques. // Society of Photo-Optical Instrumentation Engineers (SPIE) Bellingham, WA, USA. - 1999
2. Sharabayko M.P., Osokin A.N. Fractal compression of images. Realization and research of algorithms // LAP Lambert Academic Publishing. - 2012
3. D.S. Vatolin. Image compression algorithms. - Toolkit. Moscow 1999
4. Simonenkov, PS, student; hands. K.I. Sviridenkov. Analysis of software tools for the compression of graphic files. Energetika, Informatics, Innovations - 2016. Sat of the VIth Int. scientific-techn. Conf. students and graduate students. In 3 vol. T 1. - 2016. - p. 339-343
5. Raphael Finkel and J.L. Bentley (1974). "Quad Trees: A Data Structure for Retrieval on Composite Keys." Acta Informatica 4 (1): 1-9. DOI: 10.1007 / BF00288933
6. Simonenkov PS, student; hands. K.I. Sviridenkov. Fractal compression of graphic files using the quadratic tree method // Intelligent information technologies, energy and economics. Sat trudov XIV-th Int. scientific-techn. Conf. students and graduate students
7. D. Vatolin, A. Ratushnyak, M. Smirnov, V. Yukin. Methods of data compression. The device archivers, compression of images and video. - Dialogue-MEPhI, 2002.
8. Monich Yu. I., Starovoitov V. V. Estimates of quality for the analysis of digital images // Artificial intelligence. 2008. № 4. P. 376 - 386.
9. Synthesis of color / / Photo-technique: Encyclopedia / Ch. Ed. E. A. Iofis. - Moscow: Soviet Encyclopedia, 1981.
10. Ivchenko GI, Medvedev Yu.I. Introduction to mathematical statistics. - M.: Publishing house LCI, 2010.

Балашов О.В., Кондратова Н.В. Способ повышения согласованности экспертных данных без привлечения дополнительной информации // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т.2 №3(5) с. 51-58

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 519

СПОСОБ ПОВЫШЕНИЯ СОГЛАСОВАННОСТИ ЭКСПЕРТНЫХ ДАННЫХ БЕЗ ПРИВЛЕЧЕНИЯ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ

Балашов О.В., Кондратова Н.В.

Смоленский филиал АО «Радиозавод», Россия, (214027, г. Смоленск, улица Котовского, дом 2); e-mail: smradio@mail.ru

Рассматривается качество принимаемых решений в условиях неопределенности, основанных на экспертной информации. Предлагается способ коррекции несогласованных экспертных данных, реализующий метод определения доли относительной интенсивности альтернатив.

Ключевые слова: решение, неопределённость, данные, согласование оценок, отношение предпочтения.

A METHOD OF INCREASING THE CONSISTENCY OF EXPERT DATA WITHOUT ADDITIONAL INFORMATION

Balashov O.V., Kondratova N.V.

Smolensk branch of joint-stock company "Radio factory", Russia, (214027, Smolensk, street Kotovskogo, the house 2); e-mail: smradio@mail.ru

The quality of decisions taken in the context of uncertainty based on expert information is considered. A method for correcting uncoordinated expert data is proposed, which implements a method for determining the relative relative intensity of alternatives.

Key words: decision, uncertainty, data, agreement of estimates, preference relation.

Известно, что в организационных системах, очень трудно оценить качество принятых решений из-за большого субъективизма целевой направленности действий одного человека (ЛПР), а тем более коллективов людей. Принятые в таких системах решения часто основываются на личном опыте ЛПР и его субъективном мнении, формирующем в некоторых случаях необъяснимую даже им самим систему предпочтений. Такие решения могут проверить и оценить только эксперты, имеющие богатый опыт работы в предметной области решаемой задачи. Как вывод, оценка качества принятых решений в организационно-технических системах крайне субъективна.

Целевые свойства характеризуют степень соответствия полученного результата целям и задачам, поставленным при принятии решения. Степень соответствия может быть измерена посредством показателя адекватности характеристик полученного результата (свойства решения), тем требованиям, которые поставил ЛПР перед принятием решения в виде ограничений на каждую из характеристик. При этом допускается создание сложных оценочных моделей проверки адекватности на основе комплексного показателя, учитывающего важность каждого из частных показателей и их взаимосвязь. Свойство адекватности может быть описано через такой показатель, как точность характеристик

принятого решения. Точность может быть оценена посредством ограничений (пороговые характеристики), поставленных как на отдельные компоненты решения, так и на всё решение в целом.

Оценка качества принятых решений в организационно-технических системах, в отличие от оценки технических, проводится только по целевому свойству и сводится к тому, что само решение представляется как совокупность задач, требующих решения к определенному времени. Численным значением показателя качества принятого решения, состоящего из совокупности задач, в этом случае выступает отношение числа решенных задач к числу задач, которые требовалось решить в указанный период.

Такой подход, даже с учётом важности каждой из решаемых задач, отражает качество принятого решения только через его объём и не затрагивает «внутренних (смысловых)» характеристик каждой из частных задач. Не оцениваются характеристики частных задач и степень их соответствия требованиям предъявляемым к ним со стороны лица, принимающего решения. Следовательно, задачу оценки качества решений в организационно-технических системах необходимо решать с позиций системного подхода, то есть проводить исследования по последовательно убывающим уровням обобщения с учетом комплекса межуровневых и внутри уровневых взаимосвязей, рассматривая решение как систему, состоящую из связанных между собой подсистем – частных решений, каждое из которых имеет конечный набор свойств (характеристик), влияющих на качество принятого решения.

Способы повышения качества принимаемых решений могут быть классифицированы как по особенностям представления и преобразования информации, так и по месту их применения на разных этапах принятия решений. В зависимости от необходимости привлечения дополнительной информации для решения проблемы повышения качества и обоснованности принимаемых решений применяют два основных способа снижения уровня неопределённости:

- с привлечением дополнительной (вспомогательной) информации о свойствах альтернатив (экстенсивные меры снижения неопределённости);
- без привлечения дополнительной информации (интенсивные меры снижения неопределённости).

Реализация первого способа связана с отысканием новых источников информации о свойствах альтернатив, что в ряде случаев затруднительно и не дает уверенности в том, что дополнительные оценки снизят неопределённость уже имеющейся информации. Кроме того, с увеличением числа источников информации, возрастает количество сравниваемых вариантов свойств, что приводит к увеличению размерности решаемой задачи, усложнению алгоритмов обработки информации и повышению временных затрат на получение результата решения при постоянной производительности вычислителя.

Реализация второго способа связана с отысканием набора оценок свойств альтернатив, обладающих свойствами связности и транзитивности, на множестве элементов которых ЛПР может четко выразить суждения об отношении предпочтения.

В настоящей статье экстенсивные меры снижения неопределённости не рассматриваются, как практически неприменимые для обеспечения процессов подготовки и принятия решений в условиях реального масштаба времени, когда физически невозможно получить и обработать дополнительную информацию. Далее рассматриваются только те методы и способы уменьшения неопределённости, которые относятся к интенсивным мерам.

По месту применения в схеме принятия решений различают способы корректировки исходных данных на этапах:

- подготовки решений;
- непосредственного принятия решений.

Первая группа способов направлена на уменьшение влияния условий и причин возникновения неопределенности. Способы предназначены для уменьшения ошибок, возникающих непосредственно при:

- опросе ЛПР (ошибки, появляющиеся в результате преобразования данных и знаний ЛПР из его системы представления в систему оценок, навязанную ЛПР при опросе);
- преобразовании информации (ошибки, появляющиеся в результате преобразования оценок в систему показателей предпочтения альтернатив).

Вторая группа способов направлена на уменьшение влияния последствий проявления неопределённости на полученные промежуточные и окончательные результаты задачи принятия решений. Эти способы предназначены для коррекции оценок показателей или группы оценок (в соответствии с рассматриваемой иерархией показателей) альтернатив (возможных решений) до выполнения главной процедуры выбора на множестве альтернатив. Особая роль в этой группе отводится способам и методам согласования информации.

В зависимости от того, кто или что согласует информацию, традиционно выделяют два типа процедур согласования [1, 2]: «чисто переговорные», то есть без использования вычислительной техники и многоуровневые (итеративные) без личных контактов с контролируемой обратной связью, осуществляемой специальным программным обеспечением.

Способы согласования, как правило, включают методику согласования решений (оценок), в состав которой входят методы получения, представления и преобразования оценок свойств альтернатив и оценок самих альтернативных решении по каждому из свойств. В том случае, когда не требуется предварительное преобразование информации, её согласование производится на основе одного метода, а не совокупности методов, сам метод выступает в роли способа согласования. При этом под методикой согласования подразумеваются процедуры реализации рассматриваемого метода.

В зависимости от ситуации принятия решений за основу классификации моделей согласования данных и решений может быть выбрана схема (таблица 1) [3].

Таблица 1 – Классификация моделей в зависимости от ситуации принятия решений

Тип модели	Тип решения	Критериально-экспертный выбор	Целостный выбор
Объективная модель при многих критериях	Уникальные решения	А	Б
	Повторяющиеся решения	В	Г
Субъективная модель	Уникальные решения	Д	Е
	Повторяющиеся решения	К	М

Примечание – Ситуации принятия решений {А, Б, В, Г, Д, Е, К, М}:

А и В	–	задачи математического программирования;
Б	–	выбор конструкции механизмов при наличии многих критериев;
В	–	применение механизмов, использующих адаптивные алгоритмы;
Д	–	применение многокритериальных методов;
Е	–	при принятии личных решений;
К	–	модели, аппроксимирующие поведение человека;
М	–	модели, используемые при построении экспертных систем (для решения проблемы выявления знаний).

Первая группа способов решения задач ЛПР характеризуется, в основном, критериально-экспертным выбором на основе субъективной модели, использующей уникальные неповторяющиеся данные, полученные при групповом или единоличном опросе ЛПР.

Способы второй группы характеризуются целостным выбором и сводятся к процедурам нормирования характеристик показателей решений. Эти задачи не позволяют устранять

ошибки, полученные при экспертном опросе и при преобразовании информации из одной формы представления в другую.

Способы первой группы могут устранять внесенные и полученные ошибки на основе хорошо изученных методов согласования групповых оценок [1, 2]: идеальной точки; ранжирования по Парето; с использованием кусочно-линейной аппроксимации функции предпочтения ЛПР; с использованием «лямбда» коэффициентов и другие методы.

Однако при согласовании единоличных уникальных неповторяющиеся субъективных оценок, большинство из перечисленных выше способов невозможно использовать, поскольку трудно отыскать связь между этими оценками. Одним из способов согласования информации в указанных условиях является использование процедур получения экспертных знаний по Стивену и Галантеру [4], в основу которого положена гипотеза о согласованности экспертных оценок по каждому из показателей.

Такой подход, названный «психофизическим» шкалированием, предопределяет согласованность тем, что при опросе экспертам предлагается одновременно сравнивать каждое свойство со всеми остальными свойствами, получая только одну строку (столбец) матрицы **M, (**см. формулу (7). В работе [5] критикуется этот подход и утверждается, что гипотеза одномерности не может быть проверена непосредственно и что нет способа связать, одну шкалу с другой как в методе анализа иерархий. Кроме того, на практике доказано, что гипотеза о согласованности экспертных оценок по каждому из показателей очень часто не выполняется.

Другой альтернативный способ согласования экспертных оценок – применение методов транзитивного замыкания отношения предпочтения. Для реализации указанных методов необходим переход от суждений к отношениям и представление исходной субъективной информации в виде модельных отношений предпочтения.

Способ транзитивного замыкания отношения предпочтения. Отыскание набора оценок свойств альтернатив, обладающих свойствами связности и транзитивности, связано с операцией замены не транзитивного отношения R «ближайшим» к нему наименьшим транзитивным отношением R^*, включающим в себя R. Такая операция называется транзитивным замыканием отношения R. Транзитивное замыкание часто используют для уточнения экспертных данных на этапе подготовки принятия решения.

В зависимости от функционала, лежащего в основе процедуры транзитивного замыкания, различают минимаксную, максиминную, максимультипликативную и другие стратегии. Более подробно способы транзитивного замыкания отношения предпочтения на основе наиболее часто применяемых стратегий замыкания рассмотрены в [6]. Однако все представленные способы обладают рядом существенных недостатков, ограничивающих их использование в методах анализа иерархий при коррекции субъективно полученных данных. На примере одного из способов транзитивного замыкания проведем анализ достоинств и недостатков этого способа коррекции данных.

Известно, что свойство транзитивности – необходимое условие формирования системы предпочтении [6]. По определению, отношение R называется транзитивным, если для любой тройки элементов $d_1, d_2, d_3 \in D$ выполняется условие, такое, что из условия $(d_1, d_2) \in R$ и $(d_2, d_3) \in R$ следует, что $(d_1, d_3) \in R$.

В формализованном виде операция транзитивного замыкания может быть представлена в виде равенства

$$R^* = R \cup R^2 \cup R^3 \cup \ldots \cup R^n \ldots,$$

где $R^2 = R \otimes R$ – композиция отношения R;

$$R^3 = R^2 \otimes R,\ R^4 = R^3 \otimes R,\ \ldots,$$

а композиция R^2 определяется по правилу перемножения матриц смежности отношений R с заменой арифметических операций операциями булевой алгебры.

В качестве примера рассматривается операция транзитивного замыкания, проведенная с целью «восстановления» транзитивности у экспертных данных, полученных при опросе [6]. Пусть при парном сравнении установлено, что W_1 не менее предпочтительнее W_2, а W_2 не менее предпочтительнее W_3. Указанное отношение несвязно и не транзитивно. Матрица смежности этого отношения представлена в виде матрицы

$$R = \begin{array}{c|ccc} & W_1 & W_2 & W_3 \\ W_1 & 1 & 1 & 0 \\ W_2 & 0 & 1 & 1 \\ W_3 & 0 & 0 & 1 \end{array}, \quad (1)$$

а композиции отношения имеют вид

$$R^2 = \begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}, \qquad R^3 = \begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}. \quad (2)$$

Поскольку R^3 совпадает с R^2, то транзитивное замыкание совпадает с R^2.

По аналогии с транзитивным замыканием отношения R, транзитивное замыкание нечёткого отношения $\tilde{R}^*$ может быть представлено в виде равенства

$$\tilde{R}^* = \tilde{R} \cup \tilde{R}^2 \cup \tilde{R}^3 \cup \ldots \cup \tilde{R}^n, \quad (3)$$

где $\tilde{R}^2 = \tilde{R} \otimes \tilde{R}$ – композиция нечёткого отношения $\tilde{R}$, а значения функции принадлежности элементов композиции нечёткого отношения $\tilde{R}^2$ могут быть найдены на множестве элементов из выражения

$$\mu_{R^2}(d_1, d_3) = \max_{d_2} min\{m_R(d_1, d_2), m_R(d_2, d_3)\} \text{ для любых } d_1, d_2, d_3 \in D,$$

где $\mu_{R^2}(d_1, d_3)$ – уточнённая оценка.

Приведем пример транзитивного замыкания нечёткого отношения $\tilde{R}$.

Исходные данные представлены в виде обратно симметричной матрицы значений функции принадлежности μ_R.

$$\tilde{R} = \begin{vmatrix} 1 & 0{,}6 & 0 \\ 0{,}4 & 1 & 0{,}3 \\ 0 & 0{,}7 & 1 \end{vmatrix}, \quad (4)$$

а композиции отношения имеют вид

$$\tilde{R}^2 = \begin{vmatrix} 1 & 0{,}6 & 0{,}3 \\ 0{,}4 & 1 & 0{,}3 \\ 0{,}4 & 0{,}7 & 1 \end{vmatrix}, \qquad \tilde{R}^3 = \begin{vmatrix} 1 & 0{,}6 & 0{,}3 \\ 0{,}4 & 1 & 0{,}3 \\ 0{,}4 & 0{,}7 & 1 \end{vmatrix}. \quad (5)$$

Поскольку $\tilde{R}^3$ совпадает с $\tilde{R}^2$, то транзитивное замыкание совпадает с $\tilde{R}^2$. Уточненные оценки, полученные на основе выбора из исходного множества экспертных оценок, обладают рядом существенных недостатков:

- в них присутствует изначально заложенная экспертом погрешность суждений;
- значения уточнённых оценок ограничиваются множеством исходных значений, что по своей сути противоречит суждениям эксперта;
- после завершения процедуры транзитивного замыкания возникает «через диагональная несогласованность данных» (см. в полученной матрице $\tilde{R}^3$ (5) значения, расположенные в первой строке третьего столбца и третьей строке первого столбца).

В противоположность методу транзитивного замыкания, основанному на булевой алгебре, предлагается более точный (по показателю отношение согласованности [5]) и более простой (по количеству реализованных процедур вычислений и проверочных циклов, соответствующих числу итераций) способ аппроксимации экспертных оценок, названный способом корректировки несогласованных экспертных данных методом отношений.

Способ коррекции несогласованных экспертных данных, реализующий метод определения доли относительной интенсивности альтернатив. Физическая интерпретация предложенного способа заключается в том, что при выполнении условий связности и транзитивности все оценки должны быть связаны между собой. Эта связь особенно наглядно проявляется в том случае, когда экспертные оценки получены при парном сравнении рассматриваемых показателей свойств методом отношений. В этом случае по численным значениям оценок только одной строки матрицы **M** (см. (7) могут быть найдены все остальные значения оценок. Способ коррекции несогласованных экспертных данных использует результаты, полученные методом определения доли относительной интенсивности альтернатив (или отношений) и состоит из следующих процедур:

- преобразование экспертных оценок, полученных в виде численных значений функции принадлежности, в оценки функции отношения предпочтения при парном выражении предпочтения как доли относительной интенсивности (шкала отношений предпочтения);
- определение рангов показателей свойств как доли суммарной интенсивности всех оценок по каждому из свойств в интегральной оценке всех свойств (нормированная интервальная шкала оценок);
- обратное преобразование интервальных опенок в согласованные оценки по шкале отношений и заполнение полученными данными обратно симметричной матрицы отношений предпочтений.

Основное содержание способа коррекции данных состоит в преобразовании экспертных оценок, полученных в виде функции принадлежности $\mu_R\ (d_k, d_f)$, в оценки функции отношения предпочтения W_{kf}

$$W_{kf} = W(d_k, d_f) = \frac{\mu_R(d_k, d_f)}{\mu_R(d_f, d_k)} \qquad (6)$$

и представлении их в виде отношения предпочтения сравниваемых показателей *a, b, c. d,* объединенных в обратно симметричную матрицу (7).

$$M = \begin{vmatrix} 1 & W_{12} & W_{13} & W_{14} \\ W_{21} & 1 & W_{23} & W_{24} \\ W_{31} & W_{32} & 1 & W_{34} \\ W_{41} & W_{42} & W_{43} & 1 \end{vmatrix} = \begin{vmatrix} 1 & a/b & a/c & a/d \\ b/a & 1 & b/c & b/d \\ c/a & c/b & 1 & c/d \\ d/a & d/b & d/c & 1 \end{vmatrix}. \qquad (7)$$

Представление оценок в виде отношений позволяет отказаться от применения процедур булевой алгебры и перейти к альтернативным способам корректировки данных, основанным на методе отношений и методе определения наибольшего собственного значения матрицы отношений. Комплексное использование этих методов для решения поставленной задачи коррекции экспертных оценок позволяет заменить математическую модель

последовательной проверки на транзитивность отношении между выбранной оценкой и всеми парами связных с ней оценок моделью параллельной проверки на транзитивность отношения между выбранной оценкой и всеми связными с ней оценками.

Численные значения самих показателей *a, b, c, d* неизвестны. Однако, применяя процедуру отыскания рангов показателей свойств объекта, предложенную в работе [5], доля суммарной интенсивности каждого из свойств определяется по формуле

$$\lambda_f = \sqrt[n]{\prod_{i=1}^{n} W_{ji}} \times \frac{1}{\sum_{j=1}^{n} \sqrt[n]{\prod_{i=1}^{n} w_{ji}}}. \quad (8)$$

Для перехода от интервальной шкалы оценок к шкале отношений рассчитываются уточненные значения оценок

$$W_{kf}^{*} = \frac{\sqrt[n]{\prod_{i=1}^{n} W_{ki}}}{\sum_{j=1}^{n} \sqrt[n]{\prod_{i=1}^{n} w_{ji}}} \div \frac{\sqrt[n]{\prod_{i=1}^{n} W_{fi}}}{\sum_{j=1}^{n} \sqrt[n]{\prod_{i=1}^{n} w_{ji}}} = \sqrt[n]{\prod_{i=1}^{n} \frac{W_{ki}}{W_{fi}}} = \frac{\lambda_k}{\lambda_f}, \quad (9)$$

где $k, f = \overline{1, n}$.

Полученные значения заносятся в массив экспертных оценок, представленный в виде обратно симметричной матрицы отношений предпочтения.

Особенность предложенного способа состоит в том, что при любом уровне несогласованности экспертных оценок корректировка их численных значений проводится только за один период уточнения (для сравнения, при равных условиях эксперимента для корректировки экспертных оценок методом транзитивного замыкания требуется от 3 до 5 периодов уточнения). При этом степень несогласованности полученных данных предложенные в работе методом составляет порядка 10^{-8} (при использовании метода транзитивного замыкания – 10^{-2}). Для программной реализации процедуры корректировки несогласованных экспертных данных методом отношений могут быть использованы нейронные сети с обратным распространением сигнала.

Физическая сущность процедуры корректировки состоит в том, что одновременно корректируются значения всех экспертных оценок за счёт других оценок (оценок того ряда и столбца матрицы, которым принадлежит уточняемая оценка). Несогласованность отдельных оценок как бы «размывается» по всем оценкам равномерно. В результате этого с локальным перераспределением несогласованности появляется транзитивность.

Однако рассмотренный способ корректировки экспертных данных обладает одним существенным недостатком – ограничением на область применения. Использование способа возможно лишь в том случае, когда ЛПР согласно с полученными рангами альтернатив. При использовании предложенного способа ранги альтернатив сохраняются неизменными. Изменяются сами экспертные оценки в направлении улучшения свойств связности и транзитивности рассматриваемых отношений.

В том случае, когда численные значения рангов альтернатив не удовлетворяют требованиям системы предпочтений ЛПР, а сами экспертные оценки не согласованы и отношение не обладает свойством транзитивности, применение предложенного способа коррекции нецелесообразно по причине того, что вновь полученные (или уточненные) экспертные оценки, обладая свойствами связности и транзитивности, по-прежнему не будут удовлетворять системе предпочтений ЛПР. Это объясняется тем, что при коррекции численных значений экспертных оценок, вне зависимости от степени их согласованности по каждому из сравниваемых показателей свойств, проводится их аппроксимация на множестве близлежащих оценок. Это означает, что уточненная оценка получается на множестве оценок, принадлежащих строке и столбцу обратно симметричной матрицы отношений предпочтения, которым принадлежит и уточняемая оценка. В том случае, когда близлежащие оценки

получены с ошибкой, эта ошибка усредняется и передается в качестве уточняющего приращения корректируемой оценке.

Для устранения указанною недостатка и коррекции несогласованных экспертных данных при заранее неизвестном отношении ЛПР к получаемым рангам альтернатив должна применяться процедура уточнения наименее согласованных экспертных оценок на основе наиболее согласованных. Такой подход оправдан тем, что по своей природе экспертные знания неоднородны, а следовательно, уточнению должны быть подвергнуты не все оценки, а только те, в которых заложена ошибка, полученная при опросе.

Таким образом, анализ способов выражения предпочтений и методов коррекции данных, полученных на их основе, показал, что для получения надежных и обоснованных решений, удовлетворяющих системе предпочтений лица, принимающего решения, исходная информация должна быть проверена на согласованность и непротиворечивость, полноту и однозначность. Эти свойства информации определяют выбор на множестве альтернатив. Такой выбор возможен только при наличии отношений связности и транзитивности в системе предпочтений ЛПР. Следовательно, главная цель этапа подготовки принятия обоснованных решений – определение показателя уровня согласованности исходной информации и сравнение его численного значения с допустимым значением. Повышение уровня согласованности исходных данных возможно на основе транзитивного замыкания отношения предпочтения или коррекцией исходных данных в направлении получения отношения связности и транзитивности между ними.

Список литературы

1. Трахтенгерц Э.А. Компьютерные методы реализации экономических и информационных управленческих решений. В 2-х томах. Том 1. Методы и средства. – М.: СИНТЕГ, 2009.
2. Мулен Э. Кооперативное принятие решений: Аксиомы и модели /Пер. с англ. – М.: Мир, 1991.
3. Ларичев О.И. Теория и методы принятия решений. – М.: Логос, 2002.
4. Балашов О.В., Кондратова Н.В., Ошеров А.Я. Советующие системы для принятия решений при управлении организационно-техническими системами. – Смоленск: изд-во Смоленского филиала АНО ВПО ЦС РФ «Российский университет кооперации», 2012.
5. Саати Т. Л., Кернс К. Аналитическое планирование. Организация систем. – М.: Радио и связь, 1991.
6. Кофман А. Введение в теорию нечётких множеств /Под ред. С. И. Травкина. – М.: Радио и связь», 1982.

References

1. Trakhtengerts E. A. Computer methods of realisation of economic and information administrative decisions. In 2 volumes. Volume 1. Methods and means. – M.: SINTEG, 2009. (in Russian)
2. Moulin E. Co-operative decision-making: Axioms and models / Trans. with English. – Moscow: The World, 1991. (in Russian)
3. O.I. Larichev. Theory and methods of decision-making. – M.: Logos, 2002. (in Russian)
4. Balashov O.V., Kondratova N.V., Osherov A.Ya. Advising systems for decision-making in the management of organizational and technical systems. – Smolensk: publishing house of the Smolensk branch of the ANO VPO of the RF Central Committee "Russian University of Cooperation", 2012. (in Russian)
5. Saati TL, Kerns K. Analytical planning. Organization of systems. – M.: Radio and Communication, 1991. (in Russian)
6. Kofman, A., Introduction to the theory of fuzzy sets, Ed. SI Travkina. – M.: Radio and Communication, 1982. (in Russian)

Букачев Д.С. О решении задач дидактометрии средствами абстрактной граф-машины// Международный журнал информационных технологий и энергоэффективности. – 2017. – Т. 2 № 3(5) с. 59-63

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.42

О РЕШЕНИИ ЗАДАЧ ДИДАКТОМЕТРИИ СРЕДСТВАМИ АБСТРАКТНОЙ ГРАФ-МАШИНЫ

Букачев Д.С.

ФГБОУ ВО Смоленский государственный университет, Смоленск, Россия (21400, г. Смоленск, ул. Пржевальского, 4), e-mail: dsbuka@yandex.ru

Статья посвящена вопросам разработки программного обеспечения в сфере математического моделирования психолого-педагогических процессов и явлений. В работе предложена концепция программного инструмента для решения задач дидактометрии на базе абстрактной граф-машины, реализующей универсальную графовую алгебраическую систему. Дидактометрическая граф-машина позволяет строить модели учебной темы, находить веса графа, его инварианты, оценивать сложность темы, определять количество ассоциативных связей между ее элементами, находить оптимальную траекторию обучения.

Ключевые слова: универсальная алгебраическая система, абстрактная граф-машина, дидактометрия, объектно-ориентированный подход.

ABOUT THE SOLUTION OF DIDACTOMETRY PROBLEMS BY MEANS OF THE ABSTRACT GRAPH MACHINE

Bukachev D.S.

Federal State Educational Institution of Higher Education Smolensk State University, Smolensk, Russia (21400, Smolensk, street Przewalski, 4), e-mail: dsbuka@yandex.ru

Article is devoted to questions of software development in the sphere of mathematical simulation of psychology and pedagogical processes and the phenomena. In operation the concept of the software tool for the decision of tasks of a didaktometry the basis of the abstract graph machine realizing the universal graph algebraic system is offered. Didactometrical graph machine allows to build models of an educational subject, to find weights of a graph, its invariants, to estimate complexity of a subject, to define the number of the associative communications between its elements, to find an optimum path of training.

Key words: the universal algebraic system, the abstract graph machine, didaktometry, object-oriented approach.

Использование графовых моделей при решении современных прикладных задач весьма обусловлено. Благодаря своей наглядности и достаточно разработанной теории, часто бывает удобно описать структуру и свойства исследуемого объекта или процесса средствами дискретной математики. Такая формализация прикладной задачи дает возможность оперировать терминами теории графов и использовать в ходе исследования известные

алгоритмы. Методы математического исследования могут в определённой степени дополнить и развить традиционные методы психолого-педагогического анализа [2, 4, 7]. Это позволяет ставить вопрос о введении обоснованных количественных методов изучения педагогических явлений и объектов, установлении содержательных критериев для определения величин, характеризующих те или иные стороны этих явлений и процессов. При этом перспективной с точки зрения решения задач дидактометрии представляется формализация содержания предметной области. Использование графовых моделей в педагогике к настоящему моменту предлагалось многими авторами. Можно говорить о существовании определенных традиций графового моделирования в этой области [2, 5].

Модели, полученные при формализации разделов изучаемых дисциплин, часто характеризуются значительным объёмом, сложной структурой и динамикой. При решении конкретной задачи требуется выбрать оптимальный способ машинного представления графов: матрица смежности, матрица инцидентности, списковые структуры, отличающиеся друг от друга простотой доступа к данным, скоростью их обработки, “гибкостью” в использовании. Выбор структуры порождает необходимость модификации реализации операций над графами и базовых алгоритмов (волновой алгоритм, алгоритм Дейкстры, поиск циклических маршрутов, алгоритм Прима-Краскала). Несомненно, это вызывает ряд технических трудностей при программной реализации модели, поскольку разработчику зачастую приходится отвлекаться от рассматриваемой прикладной задачи и тратить время на реализацию давно известных алгоритмов.

Весьма актуальным является создание универсальных инструментов работы с графами, инвариантных относительно конкретных приложений и типов данных, сопоставляемых элементам графа. В работах [3, 6] предлагается оригинальный подход на основе алгебраизации теории графов и объектно-ориентированного подхода в программировании. Исследование многоосновных графовых алгебр, где множество-носитель – множество всех взвешенных графов, а роль операций выполняют известные алгоритмы на графах, позволяет говорить о возможности создания такой универсальной алгебраической системы, которая позволит работать с любыми графовыми моделями благодаря отделению операций структуры от операций над элементами типа.

Поскольку операции над графами (такие, как добавление/удаление элементов), а также реализация основных алгоритмов на графах фактически не зависят от характера данных, сопоставляемых элементам графа, и операций над этими данными, авторам удалось создать так называемую абстрактную граф-машину (АГМ) на базе списковых структур, работающую с абстрактными данными и позволяющую создавать весьма сложные и динамические графовые модели [3].

Для решения прикладной задачи с помощью АГМ разработчик должен создать реальную граф-машину, то есть построить класс-наследник, описать алгебры используемых данных, при необходимости задать порядок на множестве весов, после чего инициализировать виртуальную машину описанными им методами. Реальная граф-машина при таком построении будет оперировать уже понятиями той предметной области, в которой решается прикладная задача.

В настоящее время АГМ способна решать следующие задачи:

- создание динамических графовых моделей практически неограниченной сложности независимо от характера и алгебры данных, сопоставляемых вершинам графа (благодаря использованию списковых структур);

- нахождение минимального остова на неориентированных графах (алгоритм Прима-Краскала) и на графах смешанного типа (алгоритм комбинаторного характера);
- поиск минимального маршрута как по числу ребер (волновой алгоритм), так и исходя из весовых коэффициентов ребер (алгоритм Дейкстры);
- анализ модели и определение ее характеристик (нахождение инвариантов графа).

Последняя задача особенно актуальна при решении задач дидактометрии. В данной области при исследовании различных характеристик педагогического процесса, оценки трудности и сложности учебных задач наиболее естественным является использование именно графовых моделей в силу своей наглядности и простоты интерпретации. При построении модели конкретной учебной темы с вершинами графа ассоциируются элементы знаний по данной теме, а ребрам сопоставляется некоторая оценка сложности учебной деятельности, которую должны осуществить учащиеся при переходе от одного понятия к другому [1, 2]. Оценивая вес графа, определяя его инварианты, можно сделать вывод о сложности темы, количестве ассоциативных связей между ее элементами, построить оптимальную траекторию обучения [2].

Рассмотрим реализацию дидактометрической граф-машины (ДГМ) на базе АГМ.

Листинг 1. Описание класса-наследника АГМ

```
  //описание типов данных, сопоставляемых элементам графа
  TObjType=(Intermediate,Source,Final);
  TData=record   //тип данных вершины
          Obj:integer;
          ObjType:TObjType;
        end;
  TWeight=record //тип данных весового коэффициента
            Operation:integer;
            Difficulty:real;
          end;
  TMark=string;  //тип данных метки
//типизированные указатели на данные вершины, весового коэффициента и метки
  TDataPointer=^TData;
  TWeightPointer=^TWeight;
  TMarkPointer=^TMark;
//нейтралы по операции "+"
var
  DataNull:TData;
  WeightNull:TWeight;
  MarkNull:TMark;
type
  TGraph=class (TVGraph)
           private
             //конструкторы реальных данных
             procedure DataCreate;
             procedure WeightCreate;
             procedure MarkCreate;
             //деструкторы реальных данных
             procedure DataDestroy;
             procedure WeightDestroy;
             procedure MarkDestroy;
           public
             constructor Create;
```

```
        //алгебры реальных данных
        //создание нейтрального элемента
        procedure RDataZero;
        procedure RWeightZero;
        procedure RMarkZero;
        procedure RWeightInfinity;      //создание «бесконечности»
        procedure RWeightAdd;           //операция «+»
        function  RWeightOrder:boolean; //функция порядка
    end;{TGraph}
```

Для создания экземпляра реальной (дидактометрической) граф-машины требуется инициализация АГМ реальными конструкторами, деструкторами данных, сопоставляемых элементам графа, алгебраическими операциями над этими данными и отношением порядка на множестве весов.

Листинг 2. Конструктор ДГМ

```
constructor TGraph.Create;
begin
  inherited Create;
  //замена абстрактных конструкторов и деструкторов реальными
  InitConstructors(DataCreate,WeightCreate,MarkCreate);
  InitDestructors(DataDestroy,WeightDestroy,MarkDestroy);
  //инициализация АГМ алгебрами элементов
  InitVDataAlgebr(RDataZero,nil);
  InitVWeightAlgebr(RWeightZero,RWeightInfinity,RWeightAdd,RWeightOrder);
  InitVMarkAlgebr(RMarkZero,nil);
end;{Create}
```

После создания экземпляра класса-наследника АГМ реальная граф-машина во всех унаследованных алгоритмах будет оперировать реальными данными и операциями над ними конкретной предметной области. В качестве примера рассмотрим описание реальной алгебры весовых коэффициентов, использующейся при инициализации дидактометрической граф-машины.

Листинг 3. Реализация алгебры весовых коэффициентов ДГМ

```
procedure TGraph.WeightCreate;     //конструктор
begin
  New(TWeightPointer(Regs[RegResult]));
end;{WeightCreate}

procedure TGraph.WeightDestroy;    //деструктор
begin
  Dispose(TWeightPointer(Regs[RegResult]));
end;{WeightDestroy}

procedure TGraph.RWeightZero;      //нейтрал по операции «+»
begin
  TWeightPointer(Regs[RegResult])^:=WeightNull;
end;{RWeightZero}

procedure TGraph.RWeightInfinity; //бесконечность
begin
  TWeightPointer(Regs[RegResult])^.Difficulty:=Infinity;
```

```
end;{RWeightInfinity}

procedure TGraph.RWeightAdd;       //операция «+»
begin
  if (TWeightPointer(Regs[1])^.Difficulty/2+
      TWeightPointer(Regs[2])^.Difficulty/2)>(Infinity/2) then
    TWeightPointer(Regs[RegResult])^.Difficulty:=Infinity
  else
    TWeightPointer(Regs[RegResult])^.Difficulty:=
      TWeightPointer(Regs[1])^.Difficulty+TWeightPointer(Regs[2])^.Difficulty;
end;{RWeightAdd}

function TGraph.RWeightOrder: boolean; //отношение порядка на множестве весов
begin
  if TWeightPointer(Regs[1])^.Difficulty<
      TWeightPointer(Regs[2])^.Difficulty then
    Result:=true
  else
    Result:=false;
end;{RWeightOrder}
```

Достаточно простая схема инициализации абстрактной машины реальными данными и операциями над ними даёт возможность использования АГМ в качестве универсального инструмента работы с моделями из различных предметных областей. Решение задач дидактометрии является одним из актуальных приложений абстрактной граф-машины.

Список литературы

1. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Издательский дом «Вильямс», 2000. – 384 с.
2. Букачев Д.С., Мунерман В.И. О принципах реализации виртуальных алгебраических машин. //Системы компьютерной математики и их приложения: Материалы международной конференции. – Смоленск, СмолГУ, 2006. – с. 55-56.
3. Бурбаки Н. Теория множеств. – М.: Мир, 1968.
4. Левин Н.А., Мунерман В. И. Алгебраический подход к оптимизации обработки информации. – Системы и средства информатики. Спецвыпуск. Математические модели и методы информатики, стохастические технологии и системы. Москва: ИПИ РАН 2005. – с. 279-294.

References

1. Aho A., Hopkroft Dzh., Ulman Dzh. Struktury dannyh i algoritmy. – M.: Izdatel'skij dom «Vil'jams», 2000. – 384 s.
2. Bukachev D.S., Munerman V.I. O principah realizacii virtual'nyh algebraicheskih mashin. //Sistemy komp'juternoj matematiki i ih prilozhenija: Materialy mezhdunarodnoj konferencii. – Smolensk, SmolGU, 2006. – s. 55-56.
3. Burbaki N. Teorija mnozhestv. – M.: Mir, 1968.
4. Levin N.A., Munerman V. I. Algebraicheskij podhod k optimizacii obrabotki informacii. – Sistemy i sredstva informatiki. Specvypusk. Matematicheskie modeli i metody informatiki, stohasticheskie tehnologii i sistemy. Moskva: IPI RAN 2005. – c. 279-294.