УДК 512.57, 519.682.6

АБСТРАКТНО-АЛГЕБРАИЧЕСКИЙ ПОДХОД К ПОСТРОЕНИЮ ВЫЧИСЛИТЕЛЬНЫХ СРЕД ДЛЯ РЕШЕНИЯ ЗАДАЧ В ОБЪЕКТНЫХ ФОРМУЛИРОВКАХ

В. В. Суворов[1]

Предлагается метод решения математических и прикладных задач в объектных формулировках. Метод основан на представлении предметной среды и начальных условий задачи в терминах абстрактных алгебр и кодировании полученных формулировок средствами объектно-ориентированного языка программирования Java. Ключевое значение имеет формирование вычислительной среды, динамика которой управляется функциональными и предикатными компонентами объектного содержания задачи, в то время как действия управляющей программы сводятся к передаче сообщений и управления от объекта к объекту. Решение получается в результате самопреобразования постановки задачи в адекватное ей системное представление. Рассматривается возможность применения метода к решению алгебраических задач, условия которых заданы в матричной форме и в виде систем уравнений прежде всего для определения структурных и групповых характеристик. Работа выполнена при финансовой поддержке РФФИ (код проекта 09–07–12135 офи_м).

Ключевые слова: абстрактно-алгебраические системы, объектные формулировки задач, “самопреобразование данных”, системообразующий принцип.

1. Введение. Излагаемый метод и конкретная решаемая задача представляют общее решение и пример вычислительного подхода, в котором данные не только обрабатываются, но и управляют процессом решения задачи. Это обеспечивается реализацией в вычислительной среде функциональных и предикатных свойств содержания задачи в форме операций и функций, вызов которых производится объектами предметной среды задачи. Постановка задачи может инициировать направленность процесса, либо, в общем случае, происходит актуализация в структурах программных объектов свойств, неявно заключенных в условиях задачи. Результат решения содержится в явном виде в сформированной структуре данных.

“Данные” и “управление” — базовые понятия компьютерных технологий — имеют неоднозначную семантику. Исторически формировавшаяся терминология — dataflow (1969), data-driven processor (1980), data-driven system (системы, управляемые данными) и др. — демонстрирует, что данные в реальных компьютерных системах имеют де-факто значение не только пассивного (обрабатываемого) компонента. Современное развитие компьютерных технологий характеризуется “проблемой сложности”, которая проявляется в наличии принципиальных трудностей в достижении целостности постановок сложных задач, но уходит корнями в данные — “структурированные” и “неструктурированные”. В отличие от структурированных данных с установленными правилами, форматами, полями и т.д., наполняющих современные информационные системы, основная часть реальных накапливаемых данных, а также содержание задач в сложных постановках, относятся к категории неструктурированных данных, для которых нет заранее определенных правил и форматов [1].

Практическое значение приобретает, в связи с этим, задача интеграции и консолидации данных с целью получения единой целостной картины. Примеры экзотического и традиционного путей — single source of the truth (единый источник истины) и система управления знаниями (knowledge management). Используются онтологические, фреймовые, сетевые и другие вошедшие в практику решения, а также новые методы, например Report Mining — разработка отчетов. Усилия направляются при этом на преодоление (бывшего в прошлом достижением) разделения синтаксиса и семантики содержания задачи: “данные — в машине, контекст — в человеке” [1].

2. Метод. Разработанный метод “самопреобразования данных” (“data self-transforming”) имеет ассоциации с вышеназванными системами, управляемыми данными, с преобразованием неструктурированных данных в структурированные, а также с целостностью представления синтаксических и семантических характеристик содержания задач. Вместе с тем, имеется существенная особенность метода, состоящая в

[1] Научно-исследовательский вычислительный центр МГУ имени М. В. Ломоносова, Ленинские горы, д. 1, стр. 4, 119991, Москва; ученый секретарь НИВЦ МГУ, e-mail: svv@srcc.msu.ru

том, что решение заключается не в поиске способов преобразования исходных данных к требуемому результату, а в реализации определенным образом в программно-вычислительной среде содержания задачи.

Метод основан на синтезе абстрактного математического подхода, современных технологий программирования и определенного класса постановок задач, относящихся, помимо собственно математических формулировок, к областям научно-технических разработок, экономики и организации управления. Метод применим прежде всего к задачам, содержание которых в достаточной степени адекватно объектно-ориентированному подходу (ООП) и языкам программирования, реализующим ООП-парадигму.

Объектно-ориентированная парадигма служит в проводимом рассмотрении платформой для построения нового подхода к решению задач, содержательную основу которого составляет концепция интеллекта как системообразующего принципа. Аргументация данной трактовки интеллекта и соответствие ее современным знаниям об интеллекте изложено в [2–5]. Продуктивным следствием определяемого таким образом интеллекта является объективная идентификация интеллекта в содержании любой природы: непсихическом, неживом и искусственном, включая устройства искусственного интеллекта и компьютерные технологии, по критерию наличия в содержании потенциала к преобразованию его в системную форму. Дальнейшим результатом становится актуализация "интеллекта" предметного содержания решаемой задачи (в указанном смысле) в действия по получению результата решения посредством создания вычислительной среды для процесса "самопреобразования" содержания.

Представляемая работа демонстрирует "системообразующий" подход в форме метода "самопреобразования данных" на примере решения конкретной задачи.

Отличительной особенностью метода является то, что процесс решения происходит не по разработанному алгоритму последовательности действий, учитывающему специфику задачи, а в форме экспликации в вычислительной среде объектных свойств содержания задачи путем последовательной актуализации связей каждого объекта предметной среды с каждым другим объектом.

Содержание задачи преобразуется для ее решения средствами языка программирования в компоненты вычислительной среды:

а) структуру объектов-величин, фигурирующих в содержании задачи, снабженную идентификаторами объектов и ссылками на адреса памяти, зарезервированной для хранения значений величин;

б) атрибуты объектов и связи между ними в форме заносимых в память значений величин;

в) методы, реализующие действия по актуализации атрибутов одних объектов в отношении других;

г) алгоритм инициализации вычислительной среды и передачи управления от объекта к объекту в процессе решения.

Процесс решения управляется данными (условиями задачи) согласно понимаемой содержательно логике, в то время как внешние к задаче математические и программные средства имеют общий характер среды вычислений и не составляют собственно метода решения.

Для пояснения смысла решения приведем образную аналогию. Пусть имеется ограниченный объем пространства и в нем расположены объекты нескольких сортов, причем с каждым объектом связывается некоторое свойство, которое в отношении каждого другого объекта с другим свойством проявляется как притяжение или отталкивание. Пусть известно, что объекты, относящиеся к одному и тому же сорту, отталкиваются друг от друга, а притяжение проявляется селективно в отношении пар объектов. Путем внешнего воздействия создается динамика, вызывающая перемещение объектов в пространстве случайным образом. По истечении некоторого времени притягивающиеся объекты сгруппируются в кластеры, а кластеры, взаимно отталкиваясь, расположатся в удалении друг от друга. Результатом процесса станет полная структуризация данных, из которой следует, в частности, ответ на любой вопрос о принадлежности каждой пары объектов к одному или разным кластерам.

Следует отметить, что системообразующее преобразование, в отличие от статистического кластерного анализа (Data clustering), направленного на сортировку и упорядочивание объектов, имеет определяющим действием активное формирование значений величин и связей объектов. О последнем действии можно сказать упрощенно как о процедуре вычисления, но в действительности семантика реально управляет процессом решения. Экспликация свойств одного объекта в отношении других происходит путем симметричного "прописывания" этих свойств в структурах других объектов, что и приводит, в конечном счете, к получению требуемого решения.

3. Математический аппарат. С точки зрения математики содержание задач данного типа представляется в терминах алгебраических систем. Имея в виду математический и программный аспекты решения, следует отметить, что понятие "объект" в программировании и математике трактуется широко. В теории категорий, наиболее абстрактной формулировке математических структур, деление элементов на объекты и морфизмы имеет смысл только в пределах фиксированной категории. Объекты одной ка-

тегории могут быть морфизмами другой, и наоборот. В языке программирования Java объект является экземпляром класса. В свою очередь, класс — абстрактный тип данных, включающий в себя конструктор, поля, методы. Объектами в решаемой задаче становятся не собственно объекты в содержательной трактовке и не атрибуты объектов, а значения атрибутов. Причем именно последние оказываются активными "действующими лицами" в процессе решения.

В формировании алгебраической системы, служащей теорией для содержания задачи, исходим из сигнатуры $\Xi = (F, P, \mu)$, где F — множество функциональных символов; P — множество символов предикатов (отношений) и μ — отображение множества $F \cup P$ в множество натуральных чисел ω (арность, местность); $\mu(p) = n$ означает, что предикат p имеет n аргументов. При решении прикладной задачи первичным является содержание задачи, соответственно которому строится алгебраическая система $U = \big(W, \nu^U\big)$, где: W — основное множество (носитель), служащее формальным представлением для множества объектов предметной области задачи, $W = \big\{a_i \mid i \in N_A\big\}$, где N_A — множество индексов объектов; ν^U — отображение множества $F \cup P$ в множество отношений и операций на множестве W (интерпретация ν^U сигнатуры Ξ в множестве W). Алгебраическая система записывается, таким образом, в форме $U = (W, F, P, \mu)$. Тип системы образуется парой наборов $\big(n(f_1), \ldots, n(f_{Nf});\, n(p_1), \ldots, n(p_{Np})\big)$, состоящих из арностей (местностей) функций и предикатов, где N_f — количество функций, N_p — количество предикатов.

Задача, на примере которой излагается метод, имеет следующую формулировку.

1. Рассматриваются объекты предметной среды, называемые "комплексами", образующие множество $\big\{S_k \mid k \in N_S\big\}$, где N_S — множество индексов комплексов и $\big|N_S\big| = 5$ — число комплексов. Комплексы представляются совокупностями атрибутов и могут идентифицироваться по значениям одного из атрибутов.

2. Основное множество (носитель) является структурированным, на верхнем уровне оно представлено множеством доменов $W = \big\{A_i \,|\, i \in N_A\big\}$, где N_A — множество индексов доменов, в данной задаче $\big|N_A\big| = 6$ (число атрибутов, фигурирующих в задаче).

3. Каждый домен A_i является скалярной величиной, имеющей множество значений $\big\{V_{ij} \mid j \in N_{Vi}\big\}$, где N_{Vi} — множество индексов значений домена A_i. Для данной задачи $\big|N_{Vi}\big| = 5$ для каждого домена, вследствие чего вводится упрощение обозначения $N_{Vi} \equiv N_V$.

4. Множество-универсум $W^* = \big\{V_{ij} \mid i \in N_{Ai},\, j \in N_V\big\}$ имеет элементами каждое из значений каждого атрибута.

5. В каждом из комплексов S_k представлены значения каждого из атрибутов $\big\{A_i\big\}$. В свою очередь, каждый из атрибутов представлен в каждом комплексе одним определенным значением V_{ij}. Распределение значений по комплексам таково, что все значения полностью распределяются по комплексам, каждый комплекс содержит по одному значению каждого атрибута и при этом каждое значение одного атрибута принадлежит единственному комплексу.

6. Имеется N_{Fact} условий-фактов, фиксирующих принадлежность некоторых пар значений одному комплексу либо двум соседним.

7. Требуется: для заданного комплекса S_{ij}, идентифицируемого по значению V_{ij} атрибута A_i, указать принадлежащее этому комплексу значение V_{km} другого атрибута A_k $(k \neq i)$.

Функциональная компонента алгебраической системы регламентирует конструирование термов t. Термом является, прежде всего, каждый элемент множества W^*. Ввиду индексного именования значений по принадлежности к атрибутам вводится функция $\mathrm{Arg}(\mathrm{A_i}, \mathrm{V_j})$, значениями которой являются термы $AV_{ij} = \mathrm{Arg}(\mathrm{A_i}, \mathrm{V_j})$. В программной реализации этим термам соответствуют элементы массива $AV[i][j]$, составляющего вычислительное пространство задачи. Обозначение значения с одним индексом — V_j используется для отнесения значения к атрибуту с указанным номером — A_i.

Функция второго уровня Fact применяется для конструирования связей между значениями атрибутов, принадлежащих одному и тому же либо смежным комплексам:

$$\mathrm{Fact}(\mathrm{Arg_1}, \mathrm{Arg_2}, \mathrm{k}) \equiv \mathrm{Fact}\big(\mathrm{Arg}(\mathrm{A_{i1}}, \mathrm{V_{j1}}), \mathrm{Arg}(\mathrm{A_{i2}}, \mathrm{V_{j2}}), \mathrm{k}\big) \equiv \mathrm{Fact}(\mathrm{AV_{i1j1}}, \mathrm{AV_{i2j2}}, \mathrm{k}),$$

где $k \in \mathrm{Kind}$ — идентификатор типа условия, указывающий на отнесение аргументов к одному и тому же либо смежным комплексам. Предикатная составляющая алгебраической системы образуется введением на множестве-универсуме W^* двух видов эквивалентностей, соответствующих двум видам разбиений множества W^* на непересекающиеся подмножества:

$V_i \sim_A V_j$ — эквивалентность по признаку принадлежности значений V_i и V_j атрибуту одного типа;

$V_i \sim_S V_j$ — эквивалентность по признаку принадлежности значений V_i и V_j одному комплексу.

Предикат принадлежности значений V_{k1} и V_{k2} атрибуту A_i определяется выражением:

$$PA_i(V_{k1}, V_{k2}) \Longleftrightarrow \forall (V_{k1} \in A_i)\Big\{\big((V_{k2} \in A_i) \longrightarrow (V_{k1} \sim_A V_{k2})\big)\&\big((V_{k2} \notin A_i) \longrightarrow \neg(V_{k1} \sim_A V_{k2})\big)\Big\}.$$

Предикат принадлежности значений V_{k1} и V_{k2} комплексу S_i определяется выражением:

$$PS_i(V_{k1}, V_{k2}) \Longleftrightarrow \forall(V_{k1} \in S_i)\Big\{\big((V_{k2} \in S_i) \longrightarrow (V_{k1} \sim_S V_{k2})\big)\&\big((V_{k2} \notin S_i) \longrightarrow \neg(V_{k1} \sim_S V_{k2})\big)\Big\}.$$

Из п. 5 следует, что также выполняется отношение антиэквивалентности "$\oplus$" (исключающей дизъюнкции) по признакам принадлежности одному атрибуту и одному комплексу:

$$\big\{(V_i \sim_A V_k) \Longrightarrow (V_i \oplus_S V_k)\big\}\&\big\{(V_i \sim_S V_k) \Longrightarrow (V_i \oplus_A V_k)\big\}.$$

В кванторной форме записи

$$\Big\{(\forall A_a)(\forall S_b)\Big(\forall_{V_i}\big((V_i \in A_a)\&(V_i \in S_b)\big)\Big)(\forall V_k \in W^*) : \big[(V_k \in A_a)\&(V_k \neq V_i) \longrightarrow V_k \notin S_b\big]\Big\}\&$$
$$\&\Big\{(\forall A_a)(\forall S_b)\Big(\forall_{V_i}\big((V_i \in A_a)\&(V_i \in S_b)\big)\Big)(\forall V_k \in W^*) : \big[(V_k \in S_b)\&(V_k \neq V_i) \longrightarrow V_k \notin A_a\big]\Big\}.$$

При вычислениях используются свойства:

а) эквивалентности — рефлексивность, симметричность и транзитивность,

б) антиэквивалентности — коммутативность, ассоциативность, дистрибутивность.

Дистрибутивность антиэквивалентности относительно эквивалентности имеет, в частности, следствия: $(a \sim b)\&(a \oplus c) \Longrightarrow (b \oplus c)$ и $(a \oplus b)\&(a \sim c)\&(b \sim d) \Longrightarrow (c \oplus d)$.

4. Программно-вычислительная среда. Объекты вычислительной среды подобно объектам реальной физической среды заключают в своих структурах информацию о свойствах в отношении внешних им объектов. Применительно к решаемой задаче представителями объектов являются структуры, идентифицируемые по именам значений атрибутов. Имя объекта имеет ссылку на список имен атрибутов. В свою очередь, имя каждого атрибута имеет ссылку на множество значений данного атрибута. Среду вычислений образует множество объектов, идентифицируемых по именам всех значений всех атрибутов: для каждого имени — один объект. В табл. 1 приведена структура объекта AV_{ij}.

Таблица 1

Структура объекта AV_{ij}

AV_{ij} (идентификатор объекта)	
Тип атрибута	Множество значений
A_1	$\{av_{11}, av_{12}, av_{13}, av_{14}, av_{15}\}$
A_2	$\{av_{21}, av_{22}, av_{23}, av_{24}, av_{25}\}$
A_3	$\{av_{31}, av_{32}, av_{33}, av_{34}, av_{35}\}$
A_4	$\{av_{41}, av_{42}, av_{43}, av_{44}, av_{45}\}$
A_5	$\{av_{51}, av_{52}, av_{53}, av_{54}, av_{55}\}$
A_6	$\{av_{61}, av_{62}, av_{63}, av_{64}, av_{65}\}$

Фрагмент программы на языке Java иллюстрирует, как термы различного уровня — значения, атрибуты, объекты (AV[i,j]) и условия (Fact) — интегрируются в единую структуру вычислительного пространства.

```
class Argument {
    public int Attr; // индекс атрибута
    public int Val; // индекс значения атрибута

    public Argument(int Attr, int Val) {
        this.Attr = Attr;
        this.Val = Val;
    }

    public Argument() { }

    public boolean isSameType(Argument i)
        { return Attr == i.Attr; }

    public boolean equals(Object o) {
        Argument i = (Argument) o;
        return isSameType(i) && (Val == i.Val);
    }
} // class Argument
```

```
class Fact {
    public int kind;
    public Argument arg1 = new Argument();
    public Argument arg2 = new Argument();

    public Fact(int Attr1, int Val1, int Attr2, int Val2, int kind) {
        this.kind = kind;
        this.arg1.Attr = Attr1;
        this.arg1.Val = Val1;
        this.arg2.Attr = Attr2;
        this.arg2.Val = Val2;
    }

    public void init(int Attr1, int Val1, int Attr2, int Val2, int kind) {
        this.kind = kind;
        this.arg1.Attr = Attr1;
        this.arg1.Val = Val1;
        this.arg2.Attr = Attr2;
        this.arg2.Val = Val2;
    }

    public void init(Argument_a, Argument_b) {
        kind = 1;
        arg1 = _a;
        arg2 = _b;
    }

    public boolean equals(Object o) {
        Fact c = (Fact) o;
        return arg1.equals(c.arg1) && arg2.equals(c.arg2) && (kind == c.kind);
    }

    public Fact()
        { kind = 1; }

    public String toString()
        { return String.format("      } // class Fact

public class Task {
    public static char[][][][] AV12s;
    private static Fact[] ListOfFacts = new Fact[500];
    private static int ListOfFacts_n = 0;
    public static int nA = 6; // Количество атрибутов
    public static int nV = 5; // Количество значений у атрибутов
```

КОДЫ РЕАЛИЗАЦИИ ФУНКЦИЙ И ПРЕДИКАТОВ

```
    private static void init() {
        ListOfFacts_n = 0;
        ListOfFacts[ListOfFacts_n++] = new Fact(2,0,0,0,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(2,2,1,2,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(2,1,3,1,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(1,0,4,0,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(2,3,4,3,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(0,2,3,2,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(4,2,5,2,1);
```

```
        ListOfFacts[ListOfFacts_n++] = new Fact(2,4,5,4,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(5,1,1,1,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(4,4,3,4,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(1,3,3,3,1);
        ListOfFacts[ListOfFacts_n++] = new Fact(2,0,1,1,2);
        ListOfFacts[ListOfFacts_n++] = new Fact(1,3,1,4,3);
        ListOfFacts[ListOfFacts_n++] = new Fact(4,1,5,0,2);
        ListOfFacts[ListOfFacts_n++] = new Fact(4,1,3,0,2);

        AV12s = new char[nA][nV][nA][nV];

        for (int a1 = 0; a1 < nA; a1++)
            for (int v1 = 0; v1 < nV; v1++)
                for (int a2 = 0; a2 < nA; a2++)
                    for (int v2 = 0; v2 < nV; v2++)
                        AV12s[a1][v1][a2][v2] = '+'; // Инициализация
    } // init
} // class Task
```

Таблица 2

Графическое представление массива AV[Attr1][Val1][Attr2][Val2]

		A_1					A_2					A_3					A_4					A_5					A_6				
		V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	V_5	V_1	V_2	V_3	V_4	V_5
A_1	V_1	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_2	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_3	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
...	...	...					...					...					...					...					...				
A_6	V_1	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_2	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_3	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_4	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	V_5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Массив $AV12s$ = newchar[nA][nV][nA][nV] имеет первыми двумя индексами адрес объекта, которым является в алгебраической формулировке значение AV_{ij} из множества всех значений W^*. Третий и четвертый индекс являются соответственно ссылками на значение атрибута, с которым объект, идентифицированный первой парой индексов, может находиться в связи по признаку принадлежности одному комплексу или смежным.

В табл. 2 приведено графическое представление массива AV[Attr1][Val1][Attr2][Val2] с инициализацией значений символами "+". Каждая строка — значение определенного атрибута, с которым объект-столбец может находиться в связи по признаку принадлежности одному комплексу.

5. Тестирование метода на примере решения логической "задачи Эйнштейна". Задача имеет следующую формулировку [6].

— По улице в ряд стоят пять домов, каждый — своего цвета.

— В каждом доме живет человек национальности, отличной от национальностей жителей остальных домов.

— Каждый жилец предпочитает отличные от других: марку сигарет, напиток и домашнее животное.

— Кроме того:

1) норвежец живет в первом доме;

2) англичанин живет в красном доме;

3) зеленый дом находится слева от белого;

4) датчанин пьет чай;

5) тот, кто курит Marlboro, живет рядом с тем, кто выращивает кошек;

6) тот, кто живет в желтом доме, курит Dunhill;

7) немец курит Rothmans;
8) тот, кто живет в центре, пьет молоко;
9) сосед того, кто курит Marlboro, пьет воду;
10) тот, кто курит Pall Mall, выращивает птиц;
11) швед выращивает собак;
12) норвежец живет рядом с синим домом;
13) тот, кто выращивает лошадей, живет в синем доме;
14) тот, кто курит Winfield, пьет пиво;
15) в зеленом доме пьют кофе.
— Вопрос: кто разводит рыбок?

Алгебраическая система, определенная выше, и параметры программы вычислений соответствуют числу атрибутов и значений данной задачи. В табл. 3 приведены наименования атрибутов и их значений для “задачи Эйнштейна”.

Таблица 3

Наименования атрибутов и их значений

Атрибут	Тип атрибута	Списки наименований значений
1	“номер дома”	{“1”, “2”, “3”, “4”, “5”}
2	“цвет дома”	{“желтый”, “синий”, “красный”, “зеленый”, “белый”}
3	“национальность”	{“норвежец”, “датчанин”, “англичанин”, “немец”, “швед”}
4	“напиток”	{“вода”, “чай”, “молоко”, “кофе”, “пиво”}
5	“сигареты”	{“Pall Mall”, “Dunhill”, “Rothmans”, “Marlboro”, “Winfield”}
6	“животное”	{“кошка”, “лошадь”, “птицы”, “рыбки”, “собака”}

6. Действия в вычислительной среде заключаются в последовательной актуализации (“прописывании”) свойств каждого очередного объекта AV_{ij} путем конкретизации значений его атрибутов. Тем самым происходит “координация” комплексов (домов) по аналогии со средой физических объектов. Отдельное действие включает в себя:

а) считывание строки данных о свойствах объекта из списка фактов ListOfFacts, начинающегося с условий задачи (пп. 1–15) и расширяемого новыми фактами связи значений атрибутов по ходу решения;

б) актуализацию всех следствий из соотношений эквивалентности-антиэквивалентности свойств обрабатываемого объекта;

в) фиксацию новых фактов-условий, возникающих в результате последовательной конкретизации свойств объектов действиями актуализации;

г) дописывание новых фактов в конец списка;

д) вывод результатов в файл или на дисплей по исчерпании списка.

В процессе решения объекты в соответствии с порядком их встречаемости в списке фактов пошагово, все более точно, фиксируют свои и других объектов “координаты” (значения атрибутов). Процесс решения имеет содержательную семантику в отношении:

а) производимых действий — фиксации следствий эквивалентности-антиэквивалентности значений атрибутов;

б) направленности процесса — конкретизации данных;

в) результата решения — сгруппированных значений атрибутов, относимых к комплексам (домам).

Описание алгоритма вычисления. Принципиально важным является то, что компьютерный инструмент в методе “самопреобразования данных” не предполагает учета специфики задачи иначе, как в форме приспособленности к считыванию структуры объектов и наличия функций и предикатов, реализующих типы отношений объектов, т.е. без определения собственно алгоритма, регламентирующего определенный порядок действий по решению задачи. Применительно к данной задаче — это действия по актуализации (“прописыванию”) в вычислительной среде эквивалентности-антиэквивалентности объектов, заключенных в условиях задачи. Для этого, однако, необходимо, с одной стороны, представить соответствующим образом условия задачи и содержательно в некотором формате описать действия по “самопреобразованию”. С другой стороны, необходимо создать вычислительную среду, которая адекватно воспринимает данные в предлагаемом формате и реализует в вычислительной среде действия, предписываемые методом. Конкретизация метода и построение инструмента осуществляются посредством мате-

матической формализации постановки и решения задачи. Полномасштабная реализация метода означает создание развитой функциональной среды, не ограниченной описанными действиями. В рамках решаемой задачи алгоритм вычислений формулируется в терминах программы вычислений на алгоритмическом языке Java [7, 8].

Алгоритм решения задачи включает в себя выполнение следующих действий.

1. Инициализация среды объектов.

1.1. Формируется список фактов ListOfFacts, имеющий первыми 15 строками условия задачи. Элементы списка ListOfFacts[t] имеют формат $\big(\mathrm{Arg}_1(\mathrm{A}_1, \mathrm{V}_1), \mathrm{Arg}_2(\mathrm{A}_2, \mathrm{V}_2), \mathrm{k}\big)$, где t — порядковый номер элемента в списке, A (attribute) — тип атрибута; V (value) — значение атрибута; индексы 1 и 2 означают принадлежность типа и значения атрибута соответственно первому и второму аргументам Arg_1 и Arg_2; k (kind) — тип факта: 1 — эквивалентность (принадлежность значений аргументов одному комплексу); 2 — смежность (принадлежность значений аргументов соседним комплексам); 3 — порядок (расположение значений аргументов в соседних комплексах в определенном порядке следования).

1.2. Формируется пространство решений в форме массива программных объектов $AV_{ij}(A_k, V_m)$ в соответствии с табл. 2, где AV_{ij} — имя объекта, соответствующего типу атрибута A_i и значению V_j; A_k — атрибут объекта AV_{ij} и V_m — значение атрибута A_k. Элементы массива AV_{ij} при инициализации заполняются множествами из 5 потенциально возможных значений для каждого атрибута. Последующая конкретизация значений заключается в исключении всех, кроме одного, значений из каждого атрибута каждого объекта.

2. Последовательное считывание фактов из списка ListOfFacts до его исчерпания. Для очередного факта $\big(\mathrm{Arg}_1(\mathrm{A}_{10}, \mathrm{V}_{10}), \mathrm{Arg}_2(\mathrm{A}_{20}, \mathrm{V}_{20}), \mathrm{K}\big)$ выполняются следующие действия. Первый аргумент считанного факта $\mathrm{Arg}_1(\mathrm{A}_{10}, \mathrm{V}_{10})$ интерпретируется в качестве адреса (имени) объекта AV_{ij}, второй $\mathrm{Arg}_2(\mathrm{A}_{20}, \mathrm{V}_{20})$ — в качестве свойства этого объекта. Идентифицируется тип факта: K — эквивалентность (“живет в”, “предпочитает” и т.п.), смежность (“рядом”, “сосед”) или “порядок следования” (“слева”).

2.1. При значении $k = 1$ (эквивалентность по критерию принадлежности одному комплексу) аргументы факта находятся в отношении $\mathrm{Arg}_1 \sim_{\mathrm{C}} \mathrm{Arg}_2$ и выполняются действия:

а) актуализуется эквивалентность в непосредственном значении — в объекте AV_{ij}, расположенном по адресу $\mathrm{Arg}_1(\mathrm{A}_{10}, \mathrm{V}_{10})$; из атрибута-свойства A_{20} исключаются все значения V_2^*, кроме одного — V_{20};

б) актуализируются следствия из “а”:

— транзитивность эквивалентности: для объекта AV_{ij} по адресу $\mathrm{Arg}_1(\mathrm{A}_{10}, \mathrm{V}_{10})$ в цикле по A_2^* проверяется единственность соответствующего значения V_{20}^*. При обнаружении — фиксируется и дописывается в ListOfFacts новый факт $\big(\mathrm{Arg}_1(\mathrm{A}_{10}, \mathrm{V}_{10}), \mathrm{Arg}_2(\mathrm{A}_2^*, \mathrm{V}_{20}^*), 1\big)$;

— дистрибутивность антиэквивалентности значений одноименных атрибутов разных комплексов по критерию “$\sim_C$”: в цикле по значениям V_1^* атрибута A_{10} по соответствующим адресам объектов AV_{ij} в атрибутах A_{20} из множества значений исключается значение V_{20};

в) актуализируется свойство взаимно-однозначности отображения значений из распределения по атрибутам в распределение по комплексам: для каждого исключенного значения V_{20} проверяется единственность оставшегося объекта, у которого сохранилось данное значение. При обнаружении — фиксируется и дописывается в ListOfFacts новый факт $\big(\mathrm{Arg}_1(\mathrm{A}_1^*, \mathrm{V}_1^*), \mathrm{Arg}_2(\mathrm{A}_{20}, \mathrm{V}_{20}), 1\big)$.

2.2. В считанном факте инвертируются аргументы $\big(\mathrm{Arg}_2(\mathrm{A}_2, \mathrm{V}_2), \mathrm{Arg}_1(\mathrm{A}_1, \mathrm{V}_1), 1\big)$, и для сформированного таким образом факта выполняются действия согласно п. 2.1.1.

2.3. При идентификации по параметру K отношения типа “смежность” или “следование” проверяется выполнимость этого отношения для комплексов для типа атрибута “номер”. При установлении единственного варианта “встраиваемости” условия формируются два новых факта: $\big($(тип атрибута — “номер”, “первый установленный номер”), $(A_{10}, V_{10}), 1\big)$ и $\big($(тип атрибута — “номер”, “второй установленный номер”), $(A_{20}, V_{20}), 1\big)$.

3. Вывод результатов.

Разработка, отладка и выполнение программы производилось в среде платформы Eclipse [8].

В табл. 4 изображены все элементы массива AV, имеющего значение пространства решений. Каждое элементарное действие по преобразованию (“самопреобразованию”) данных находит отражение в виде изменения значений изображенных элементов:

а) инициализация — 0 итераций;

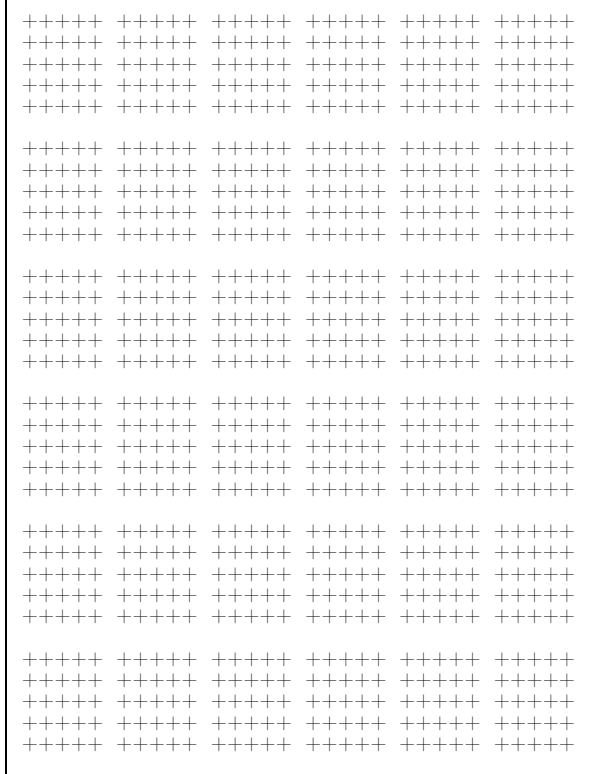
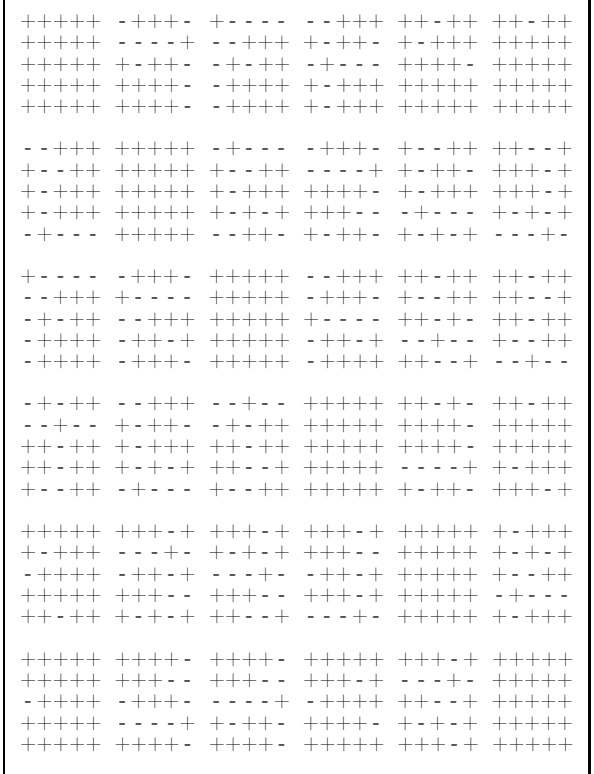
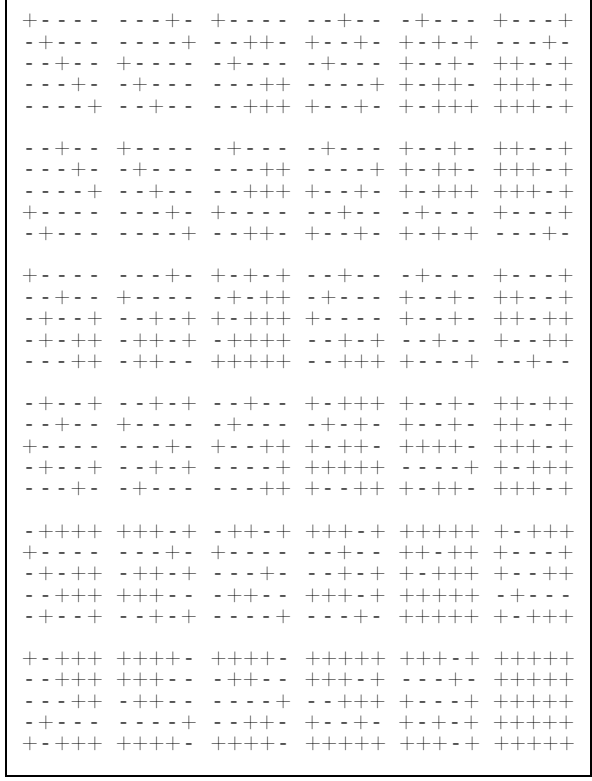
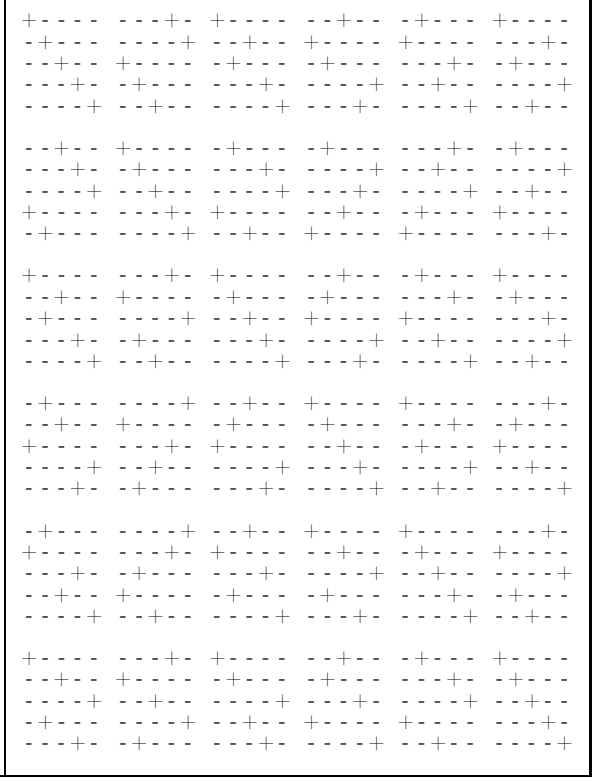
б) ввод исходных данных — 20 итераций;

в) промежуточное состояние — 55 итераций;

г) результат процесса — 80 итераций.

Таблица 4

Состояние пространства решений — значения элементов массива AV

Инициализация — 0 итераций	Ввод исходных данных — 20 итераций
	
Промежуточное состояние — 55 итераций	**Результат решения — 80 итераций**
	

В табл. 5 показаны результаты решения, верхняя часть массива AV с поименованными значениями атрибутов. Из таблицы следует, что “рыбок” разводит “немец”, который пьет “кофе”, курит Rothmans и живет в “четвертом”, “зеленом” доме.

Таблица 5

Результаты решения

	Цвет					Национальн.					Напиток					Сигареты					Животное				
Номер дома	красный	зеленый	белый	желтый	синий	норвежец	англичанин	датчанин	немец	швед	чай	молоко	вода	пиво	кофе	Marlboro	Dunhill	Rothmans	Pall Mall	Winfield	кошка	птицы	собака	лошадь	рыбки
1	.	.	.	+	.	+	.	.	.	.	.	.	+	.	.	.	+	.	.	.	+	.	.	.	.
2	.	.	.	.	+	.	.	+	.	.	+	.	.	.	.	+	.	.	.	.	.	.	.	+	.
3	+	.	.	.	.	.	+	.	.	.	.	+	.	.	.	.	.	.	+	.	.	+	.	.	.
4	.	+	.	.	.	.	.	.	+	.	.	.	.	.	+	.	.	+	.	.	.	.	.	.	+
5	.	.	+	.	.	.	.	.	.	+	.	.	.	+	.	.	.	.	.	+	.	.	+	.	.

Таблица 6

Решение при исключении условий 5 и 13

	Цвет					Национальн.					Напиток					Сигареты					Животное				
Номер дома	красный	зеленый	белый	желтый	синий	норвежец	англичанин	датчанин	немец	швед	чай	молоко	вода	пиво	кофе	Marlboro	Dunhill	Rothmans	Pall Mall	Winfield	кошка	птицы	собака	лошадь	рыбки
1	.	.	.	+	.	+	.	.	.	.	.	.	+	.	.	.	+	.	.	.	+	.	.	+	+
2	.	.	.	.	+	.	.	+	.	.	+	.	.	.	.	+	.	.	.	.	+	.	.	+	+
3	+	.	.	.	.	.	+	.	.	.	.	+	.	.	.	.	.	.	+	.	.	+	.	.	.
4	.	+	.	.	.	.	.	.	+	.	.	.	.	.	+	.	.	+	.	.	+	.	.	+	+
5	.	.	+	.	.	.	.	.	.	+	.	.	.	+	.	.	.	.	.	+	.	.	+	.	.

7. Обсуждение решения. Объем вычислений оценивается по числу итераций (шагов), заключающихся в “прописывании” свойств одного объекта. В рассмотренной задаче оно составляет 80. Число элементарных действий по сравнениям и исключениям отдельных значений примерно на порядок больше 10^3. Это на семь десятичных порядков меньше числа вариантов комбинаторного перебора $(5!)^5 \sim 10^{10}$. Расширенное применение метода в сравнении с рассмотренным типом задачи возможно в двух направлениях. Во-первых, процесс решения может производиться для некорректно поставленных задач — неоднозначно определенных и содержащих противоречия. В случае неоднозначности будет получен спектр решений. Например, при исключении условий 5 и 13 результатом будет решение, представленное в табл. 6. Можно выбрать иные, отличные от 5 и 13, дополнительные условия для получения решения без вариантов. Заметим также, что анализ результатов показывает избыточность исходныч условий “источника” являются избыточными. Правильное решение может быть получено и при исключении п. 9 условий. При наличии противоречий в условиях с помощью предложенного инструмента можно производить решение до обнаружения факта противоречия и предъявлять его в качестве диагностики для отработки исходных данных или для корректирования условий в интерактивном режиме либо давать варианты решений в предположении истинности каждой альтернативы. Существенно то, что процесс решения может выполняться для некорректных постановок задач и давать конкретные результаты, содержащие указания на недостаточность или избыточность условий. Во-вторых, метод применим для решения не только задач, в которых фигурируют логические связи — эквивалентности, но также для интерпретаций сигнатур, в которых символам предикатов и функций ставятся в соответствие, в общем случае, произвольные свойства объектов предметной среды содержания задачи.

8. Применимость изложенного метода к решению задач линейной алгебры. Аппарат абстрактных алгебр является инструментом для описания и преобразования математических объектов произвольной структуры — от неупорядоченных множеств до решеток логико-математических исчислений. Более высокую степень абстракции имеет только математическая теория категорий. Примеры приложения абстрактно-алгебраического метода к объектам вычислительной алгебры: алгебраические группы

матриц, а также исторический прецедент обнаружения Галуа групповых свойств корней степенного уравнения без вычисления значений корней.

Вышеописанный подход, в котором процесс вычисления решения происходит в форме перманентной актуализации функциональных и предикатных свойств объектов предметной среды содержания задачи, применим к решению алгебраических задач в матричной форме и в виде систем уравнений. Иллюстрацией может служить алгебраическая интерпретация выполненного решения логической задачи.

Содержательно задача формулируется следующим образом.

1. W — множество значений, полученное в результате объединения подмножеств A_i — атрибутов, $W = \cup A_i$. Каждое из подмножеств образовано элементами V_{ij}, где i — индекс принадлежности атрибуту, j — идентификатор значения соответствующего атрибута.

2. $\{\mathrm{Fact_s}\}$ — список утверждений (фактов), фиксирующих некоторые взаимосвязи между значениями разных атрибутов, приводящий к разбиению множества W на подмножества (комплексы) Q_t.

3. Требуется по заданному значению одного произвольного значения V_{ij} атрибута A_i установить значение V_{km} другого указанного атрибута A_k, принадлежащего тому же подмножеству Q_s, что и V_{ij}.

Алгебраическое решение включает в себя следующие действия.

1. Построение 6-мерного пространства (по числу атрибутов) с дискретным конечным множеством значений по оси каждого измерения — 5 значений. Формирование матрицы 6-го порядка с присвоением каждому элементу матрицы, в качестве его значения, множества, элементы которого идентифицируют значения соответствующего атрибута.

2. Фиксация точек (областей), заданных условиями $\{\mathrm{Fact_s}\}$ путем исключения соответствующих условиям значений множеств-элементов матрицы.

3. Выполнение линейных преобразований по изменению порядка расположения значений по каждой оси таким образом, чтобы в результате перехода к пространству S^N с новым порядком следования значений зафиксированные точки оказались на диагонали пространства (диагонализация матрицы).

Обсуждение. Из исходных условий симметрии и антисимметрии значений атрибутов по признаку принадлежности их к подмножествам атрибутов и комплексов следует, что каждая точка на диагонали будет иметь координатами совокупность значений, принадлежащих одному комплексу Q_t. Это демонстрируют также полученные результаты решения.

Таким образом, во всяком случае, для задач данного класса полученный метод является альтернативой нахождения решения путем преобразований матричных объектов.

Вышесказанное не является строгим обоснованием. Приведено содержательное пояснение возможности решения алгебраических задач и установления системных свойств математических объектов методом, альтернативным традиционным алгебраическим приемам.

9. Заключение. Суть метода заключается в том, что действия по разработке алгоритма решения и составлению компьютерной программы ограничиваются построением в программно-вычислительной среде

а) модельных структур для объектов,

б) операций, реализующих предикатные и функциональные свойства объектов, фигурирующих в содержании задачи,

в) последующим запуском в вычислительной среде процессов прогрессирующей актуализации свойств “б” в структурах “а”.

Следует подчеркнуть, что в вычислительной среде отсутствует компонент, имеющий значение разработанного алгоритма решения. Процесс решения управляется самими данными, которые в конечном счете принимают явную форму результата решения. Это есть именно процесс “самопреобразования” данных. Каждое очередное действие в вычислительной среде происходит в результате передачи управления следующему объекту и заключается в “прописывании” этим объектом в атрибутах других объектов указаний на взаимосвязи, имеющиеся в его структуре. Число шагов не регламентируется заранее. Объект получает управление многократно по мере накопления изменений, производимых в нем другими объектами. Процесс завершается, когда не остается ни одного объекта, свойства которого не прописаны полностью в структурах всех других объектов. Результатом становится искомое решение в явном виде.

Формулировка рассмотренной задачи и соответствующей алгебраической системы фиксирует в функциональных и предикатных компонентах качественную специфику рассмотренной в качестве примера задачи — двух типов эквивалентностей и отношения антиэквивалентности для каждого типа в отношении другого. “Прописывание” свойств одного объекта в среде остальных объектов есть действие актуализации следствий из эквивалентностей и антиэквивалентности.

Метод “самопреобразования данных” служит реализацией более широко понимаемого системообра-

зующего подхода, согласно которому содержание задачи, помимо того, что оно является информацией для обработки, заключает неявным образом информацию о действиях, которые следует произвести для получения результата решения. Решение выполняется не путем извлечения необходимой информации и ее целенаправленной организации, а посредством создания вычислительной среды, в которой эта информация приобретает динамику и примет форму, представляющую в явном виде требуемое решение.

СПИСОК ЛИТЕРАТУРЫ

1. *Черняк Л.* Интеграция данных: синтаксис и семантика // Открытые системы. 2009. № 10.
2. *Рябов Г.Г., Суворов В.В.* Интеллект — системообразующий принцип // Искусственный интеллект. 2005. № 4. 36–42.
3. *Рябов Г.Г., Суворов В.В.* Комплексные фундаментальные исследования интеллекта — путь к созданию компьютерных технологий новых поколений // Информационные технологии. 2005. № 6. 2–7.
4. *Рябов Г.Г., Суворов В.В.* Что такое интеллект? // Тр. XXXII Междунар. конф. “Информационные технологии в науке, социологии, экономике и бизнесе”. Приложение к журналу “Открытое образование”. 2005. 40–43.
5. *Суворов В.В.* Интеллект и креативность в постнеклассической науке. Интеллект неискусственный. М.: Изд-во Моск. ун-та, 2006.
6. URL: http://en.wikipedia.org/wiki/Zebra_Puzzle
7. Компилятор языка Java на сайте фирмы Sun (http://java.sun.com/).
8. Бесплатная среда разработки на языке Java (http://www.eclipse.org/downloads/).

Поступила в редакцию
13.02.2010