УДК 519.7

М.Ф. Бондаренко, Ю.П. Шабанов-Кушнаренко
ХНУРЭ, г. Харьков, Украина

ОБ АЛГЕБРЕ КОНЕЧНЫХ ПРЕДИКАТОВ

Рассмотрены проблемы построения эффективного математического языка описания структур и функций систем естественного интеллекта. В качестве формального языка, на котором можно было бы математически описывать структуры и функции естественного интеллекта, используется алгебра конечных предикатов.

ТЕОРИЯ ИНТЕЛЛЕКТА, АЛФАВИТНЫЙ ОПЕРАТОР, ПРЕДИКАТ, АЛГЕБРА КОНЕЧНЫХ ПРЕДИКАТОВ

Введение

Теория интеллекта обслуживает *технику интеллекта* (то есть компьютеризацию и информатизацию), является ее математическим и физическим фундаментом. В соответствии с этим она делится на *математику* и *физику интеллекта*. Иногда говорят о теории искусственного интеллекта. Нам представляется это не вполне оправданным. Хотя интеллект невозможен без материального носителя, однако он может быть реализован как на естественной (человек), так и на искусственной (машина) физической основе. Механизмы разума можно изучать с общих позиций в рамках единой теории интеллекта, не привязываясь при этом к конкретному способу материальной реализации интеллектуальной системы.

Компьютер – машина математическая, он может воспроизвести лишь те структуры и функции, которые описаны в строго формализованном виде. Чтобы иметь возможность получать такие описания, необходимы математический язык описания и неформализованный материал о строении и функциях систем естественного интеллекта, прежде всего интеллекта человеческого. Теория интеллекта имеет многовековую предысторию. Существенный вклад в нее внесли многие выдающиеся мыслители. Особо следует отметить вклад Пифагора, Парменида, Платона, Аристотеля, Декарта, Лейбница, Ньютона, Буля, Фреге, Гильберта и Рассела.

Цифровая вычислительная техника в настоящее время стала одним из главных рычагов дальнейшего научно-технического прогресса, основой автоматизации процессов управления экономикой, производственных процессов, проектно-конструкторских и научно-исследовательских работ, важным источником повышения производительности труда и роста благосостояния народа, необходимым звеном в системе обороны страны. Электронные цифровые вычислительные машины представляют собой универсальное средство переработки информации, с их помощью, в принципе, можно автоматизировать любые виды умственного и физического труда [1]. Однако огромные потенциальные возможности вычислительных машин фактически используются далеко не в полной мере. Многие виды работ пока не поддаются автоматизации, а это серьезно сдерживает рост производительности труда, темпы прогресса. В чем причина такого парадоксального положения? Очевидно, в том, что многие работы, успешно выполняемые людьми, пока не под силу цифровой вычислительной машине: слишком еще слаб ее интеллект.

Как же повысить уровень машинного интеллекта? Для облегчения решения этой проблемы имеет смысл попытаться получить подсказку у природы, то есть пойти по бионическому [2] пути и обратиться к изучению человеческого интеллекта: ведь он способен выполнять информационные работы, недоступные пока вычислительной машине. Можно изучать две стороны человеческого интеллекта: 1) материальную сторону интеллекта – мозг, нервную систему, организм человека; 2) деятельность интеллекта, его функции, выражающиеся в поведении и действиях человека. Научная область, нацеленная на разработку описаний структуры и функций человеческого интеллекта, которые можно было бы использовать в деле совершенствования цифровых вычислительных машин, называется *теорией интеллекта*. При таком определении теории интеллекта ее успехи будут оцениваться не столько тем, как далеко эта теория продвинулась в познании интеллекта человеческого, но, главным образом, тем, какой уровень совершенства интеллекта машинного она смогла обеспечить.

Какие структуры и функции человеческого интеллекта должна изучать теория интеллекта? Очевидно, те и только те, которые, в принципе, доступны интеллекту машинному. Машинный же интеллект, то есть цифровая вычислительная машина, может действовать только механически, он способен воспроизводить лишь детерминированные, дискретные и конечные информационные процессы. *Детерминированные процессы* – это процессы с однозначным исходом, в них отсутствует фактор случайности. *Дискретные процессы* – это процессы, в которых информация имеет вид отдельных порций или квантов – цифр, букв, слов,

формул и т.д., в них отсутствует фактор непрерывности. *Конечные процессы* – это такие процессы, в которых может участвовать лишь конечное число единиц информации, в них отсутствует фактор бесконечности. Таким образом, *теория интеллекта* представляет собой науку о математическом описании детерминированных, дискретных и конечных интеллектуальных процессов, воспроизводимых человеческим разумом, и структур, обеспечивающих реализацию таких процессов, которая ориентирована на совершенствование цифровой вычислительной техники и ее практическое использование.

1. Алфавитные операторы

Для того чтобы иметь возможность развивать теорию интеллекта, прежде всего необходимо располагать формальным языком, на котором можно было бы математически описывать интересующие нас структуры и функции человеческого интеллекта. В качестве такого языка мы используем *алгебру конечных предикатов*, описание которой начнем с введения понятия конечного алфавитного оператора. К этому понятию естественным образом приводит ознакомление с принципом действия цифровой вычислительной машины. Любая вычислительная система, будь то машина в целом или ее отдельный блок, представляет собой устройство, осуществляющее преобразование информации. Эта система имеет *вход*, через который в нее поступает информация, подлежащая обработке, и *выход*, через который выдается выходная информация, сформированная вычислительной системой в ответ на поступившую в нее входную информацию. Как входная, так и выходная информация имеет вид знаковой последовательности, называемой *словом*. Знаки, из которых составлено слово, называются *буквами*.

Любая вычислительная система подвержена следующим ограничениям: 1) алфавит букв, из которых строятся слова для любой конкретной вычислительной системы, всегда конечен; 2) длина слов, которые способна воспринимать и формировать эта система, ограничена некоторым конечным наперед заданным числом букв, определяемым конструкцией системы, ее быстродействием и сроком службы; 3) реакции вычислительной системы строго детерминированы: повторное предъявление входного слова всегда приводит к формированию системой того же самого выходного слова.

Принципиально важно то, что человеческий интеллект в функциональном отношении весьма сходен с цифровой вычислительной машиной и подвержен тем же ограничениям, что и любая вычислительная система. Так же, как и вычислительная машина, человеческий интеллект способен воспринимать, преобразовывать и формировать информацию. Информация, которой оперирует человеческий интеллект, имеет вид слов, в роли которых выступают тексты, звучащая речь, чувственные образы предметов. Роль входа информации у человека выполняют органы чувств, роль выхода информации – органы движения и речи. Эти органы обладают конечной чувствительностью и конечной разрешающей способностью, поэтому в каждый момент времени человек может воспринимать сигналы (буквы) только из конечного множества (алфавита). Органы чувств, движения и речи обладают конечной полосой пропускания [1], поэтому они могут передать к интеллекту и от него лишь конечное число букв в единицу времени. Срок человеческой жизни ограничен, поэтому ограничена длина слов, которые могут обрабатываться интеллектом человека. Весьма вероятно, что реакции человека, формируемые его органами движения и речи, всецело определяются внешними воздействиями на этого человека, имевшими место в течение его предшествующей жизни. К числу этих воздействий на человека следует относить не только информацию, поступившую через органы чувств, но также и генетическую информацию, переданную его родителями. Важно отметить, что генетическая информация дискретна и конечна.

Рассмотрим конечное множество $A=\{a_1, a_2,..., a_k\}$, называемое *алфавитом*. Элементы $a_1, a_2,..., a_k$ этого множества называются *буквами* алфавита A. Число букв k в алфавите может быть произвольным. Примерами алфавитов могут служить: 1) русский алфавит, состоящий из 33 букв а, б,..., ю, я; 2) множество {0, 1,..., 9} всех десятичных цифр; 3) множество всех различных символов, использованных в этой статье (русские, латинские и греческие буквы различных шрифтов, знаки препинания, интервал между словами, математические знаки и т. п.); 4) множество символов, имеющихся на клавиатуре какой-либо вычислительной машины.

Любая последовательность $\sigma_1\sigma_2...\sigma_m$ букв σ_1, σ_2,..., σ_m алфавита A называется *словом* в алфавите A. Число m букв в слове называется *длиной слова*. Длину слова ничем не ограничивают, поэтому m может быть любым натуральным числом. Заметим, что буквы в слове могут повторяться, так что на различных местах в слове могут встречаться одинаковые буквы. Последовательности, имеющие в своем составе только одну букву, и последовательности, не содержащие ни одной буквы, также считаются словами. В последнем случае говорят, что слово имеет *нулевую длину*. Такое слово называется *пустым* и обозначается нами символом *. Хотя введение пустого слова, быть может, выглядит неестественным, однако оно необходимо для логической завершенности понятия слова. Рассмотрим примеры слов: 1) русские слова *мальчик*, *лампа* и т. п. могут рассматриваться как слова в русском

алфавите (в только что определенном смысле); 2) десятичная запись любого натурального числа есть слово в алфавите {0, 1,..., 9}; 3) любое предложение этой статьи есть слово в алфавите, состоящим из всех различных символов, встречающихся в этой статье; 4) весь текст данной статьи можно считать словом в том же алфавите.

Пусть A – произвольно выбранный непустой алфавит. Рассмотрим множество M всех слов, составленных из букв алфавита A. Поскольку в этом множестве встречаются слова сколь угодно большой длины, то общее число слов в нем бесконечно. Формально множество M может быть представлено как объединение всех декартовых степеней алфавита A, таким образом, $M=A^0\cup A^1\cup A^2\cup...$. Любое подмножество [3] L множества M всех слов алфавита A называется *языком* над алфавитом A. Язык L может быть как *конечным*, так и *бесконечным*. Примерами языков могут служить: 1) множество всех русских слов, представляющее собой конечный язык над русским алфавитом; 2) множество десятичных кодов всех натуральных чисел, представляющее собой бесконечный язык над алфавитом десятичных цифр; 3) совокупность всевозможных предложений, которые можно построить из знаков, фигурирующих в этой статье. Поскольку длина предложений, используемых в естественном языке, ограничена, то в последнем случае мы имеем дело с конечным языком.

Пусть A – произвольно выбранный непустой алфавит, а M – множество всех слов этого алфавита. Любая функция $Y=FX$, отображающая слова X из множества M в слова Y множества M, называется *алфавитным оператором*, заданным в множестве M. Иными словами, алфавитным оператором называется всякое однозначное соответствие, которое сопоставляет словам какого-либо алфавита слова того же алфавита. Совокупность L_1 всех тех слов, для каждого из которых алфавитный оператор F ставит в соответствие некоторое вполне определенное слово, называется *областью определения* или *входным языком* алфавитного оператора F. Может случиться, что алфавитный оператор F некоторым словам множества M не ставит в соответствие никакого слова, так что входной язык не обязательно совпадает с множеством M. Множество L_2 всех слов, являющихся значениями алфавитного оператора F, называется его *областью значений* или *выходным языком*.

Слова входного языка называются *входными словами* алфавитного оператора, слова выходного языка – его *выходными словами*. Алфавитный оператор, входной язык которого совпадает с множеством M, называется *всюду определенным*; если же входной язык является лишь частью множества M, то алфавитный оператор называется *частичным*. Примером всюду определенного алфавитного оператора может служить оператор, заданный на множестве десятичных кодов всех натуральных чисел, который десятичному коду каждого натурального числа ставит в соответствие десятичный код квадрата этого числа. Примером частичного алфавитного оператора может служить оператор, заданный на том же множестве, который десятичному коду натурального числа ставит в соответствие десятичный код квадратного корня этого числа, но только при условии, что этот корень оказывается натуральным числом.

Описанное выше понятие алфавитного оператора неплохо подходит для того, чтобы служить средством математического описания деятельности интеллекта, то есть объективно наблюдаемых реакций интеллекта на внешние воздействия. Информация, воспринимаемая интеллектом, соответствует входным словам некоторого алфавита; информация формируемая интеллектом, соответствует выходным словам. Закономерности преобразования информации интеллектом, то есть *функции интеллекта*, соответствуют тем или иным алфавитным операторам, преобразующим входные слова в выходные. Задача математического описания функций интеллекта заключается в том, чтобы указать соответствующие этим функциям алфавитные операторы.

2. Конечные алфавитные операторы

И все же понятие алфавитного оператора обладает одним существенным недостатком: его задают на бесконечном множестве слов, содержащем слова сколь угодно большой длины. Это обстоятельство приводит к потенциальному присутствию бесконечности в понятии алфавитного оператора [4], что создает определенные неудобства при математическом описании функций интеллекта и практическом их применении. Этих неудобств можно избежать, заменив понятие алфавитного оператора близким ему понятием *конечного алфавитного оператора*. Отличие конечного алфавитного оператора от только что описанного алфавитного оператора состоит лишь в том, что он задается не на бесконечном множестве $A^0\cup A^1\cup A^2\cup...$ слов различной длины, а на конечном множестве A^m слов одинаковой длины m, составленных из букв алфавита A.

Понятие конечного алфавитного оператора очень удобно для того, чтобы служить средством математического описания функций интеллекта. Ограничение, наложенное на длину слов, не может служить препятствием для его использования в теории интеллекта, поскольку, как указано выше, интеллект человека подвержен такому же ограничению. В качестве числа m должно быть выбрано число, не меньшее максимальной длины слов, с которыми может оперировать изучаемый интел-

лект. Возникает, правда, затруднение, вызванное тем, что интеллект обычно оперирует словами различной длины, тогда как в понятии конечного алфавитного оператора фигурируют слова только одинаковой длины.

Затруднение это, однако, легко преодолевается за счет введения в алфавит A дополнительной буквы #, называемой *знаком пробела* или просто *пробелом*. Слово, имеющее длину, меньшую, чем m, при его математическом описании заменяется словом длины m, левая часть которого совпадает с исходным словом, а правая часть представляет собой последовательность пробелов (с тем же успехом можно было бы сделать и наоборот, поменяв местами левую и правую части). При чтении формальной записи слова знаки пробела, стоящие в слове правее всех остальных букв, не должны приниматься во внимание. Слово, составленное из одних пробелов, должно интерпретироваться как пустое слово. Например, если выбрано m=6, то слово *лист* будет формально представлено в виде слова *лист*##. Положение здесь такое же, как при машинной записи числовых кодов: длина всех кодов принята одинаковой, однако нули, стоящие в левой части кода, при его чтении во внимание не принимаются. Если бы мы определили понятие конечного алфавитного оператора таким образом, чтобы в его входной и выходной языки вошли и слова меньшей длины, чем m, то это существенно усложнило бы математический язык для записи таких операторов.

Множество M, на котором задан конечный алфавитный оператор, содержит $c=k^m$ слов, то есть ровно столько, сколько имеется всех различных m-разрядных k-ичных числовых кодов. Всего существует c^c различных конечных алфавитных операторов, заданных на множестве M. Для каждой изучаемой функции интеллекта, в принципе, можно выбрать число k букв, алфавит A и предельную длину слов m такие, что в множестве всевозможных конечных алфавитных операторов, заданных на A^m, всегда найдется такой конечный алфавитный оператор, который может быть принят в качестве адекватного математического описания этой функции интеллекта. Задача состоит в том, чтобы, сообразуясь с фактическими свойствами изучаемой функции интеллекта, суметь выделить этот оператор и записать его в виде формулы на некотором математическом языке. Реализуя полученную формулу средствами цифровой вычислительной техники, можно будет изученную функцию интеллекта затем искусственно воспроизвести.

3. Конечные предикаты

Для того чтобы иметь возможность математически описывать функции интеллекта, нам необходим формальный язык, на котором можно было бы вести такое описание. Формальный язык должен быть выбран с таким расчетом, чтобы на нем можно было в удобной форме записать любой конечный алфавитный оператор. Такой язык дает нам описываемая ниже алгебра конечных предикатов. Введем понятие *конечного предиката*. Пусть A – конечный алфавит, состоящий из k букв $a_1, a_2,..., a_k$, Σ – множество, состоящее из двух элементов, обозначаемых символами 0, 1 и называемых соответственно *ложью* и *истиной*. Переменную, заданную на множестве A, будем называть *буквенной*, переменную, заданную на множестве Σ, – *логической*. *Конечным n-местным предикатом* над алфавитом A называется любая функция $f(x_1, x_2,..., x_n)=t$ от n буквенных аргументов $x_1, x_2,..., x_n$, заданных на множестве A, и принимающая логические значения t. Иногда будем называть конечный предикат f *k-ичным*, подчеркивая тем самым, что его алфавит A состоит из k букв.

Любой конечный предикат f можно, в принципе, задать с помощью таблицы его значений. В этой таблице каждому набору значений аргументов (x_1, $x_2,..., x_n$) ставится в соответствие значение t предиката f. В виде примера таблицей 1 задан двуместный предикат $t=f(x_1, x_2)$, определенный над алфавитом A={а, м, п}. По таблице находим, что, например, на наборе (x_1, x_2) значений аргументов x_1, x_2, равном (а, м), предикат f принимает значение 0, на наборе (п, м) – значение 1. Аналогично находим значения предиката и для остальных наборов. Расставив в последней строке таблицы логические значения каким-либо иным способом, получим другой двуместный предикат над тем же алфавитом.

Таблица 1

x_1	а	а	а	м	м	м	п	п	п
x_2	а	м	п	а	м	п	а	м	п
t	0	0	1	1	1	0	0	1	1

Для упрощения дальнейшего изложения сейчас нам будет полезно познакомиться с некоторыми сведениями об *n-разрядных k-ичных числовых кодах*. При построении k-ичных кодов используют k знаков 0, 1,..., k-1, называемых *k-ичными цифрами*. Число k называется *основанием* k-ичной *системы счисления*. Цифры в k-ичном коде располагаются в определенном порядке, значение каждой цифры определяется ее положением в коде. Так, например, в десятичной системе счисления в записи 179 цифра 1 обозначает число 100 или $1\cdot10^2$, цифра 7 – число 70 или $7\cdot10^1$, цифра 9 – число 9 или $9\cdot10^0$. Сумма этих чисел дает значение кода:

$$179=1\cdot10^2+7\cdot10^1+9\cdot10^0.$$

В общем виде десятичный код $a_1a_2...a_i...a_{n-1}a_n$, где $a_1, a_2,..., a_i,..., a_{n-1}, a_n$ – десятичные цифры; n – число разрядов в коде; i – номер разряда кода, обозначает число:

$$a_1a_2...a_i...a_{n-1}a_n=a_1 10^{n-1}+a_2 10^{n-2}+...+a_i 10^{n-i}+...+$$
$$+a_{n-1}10^1+a_n10^0.$$

Как узнать, какое число представляет собой тот или иной k-ичный код? Для этого достаточно его перевести в привычный нам десятичный код. Этот перевод можно осуществить, воспользовавшись определением k-ичного кода. Пусть $b_1b_2...b_i...b_{n-1}b_n$ – k-ичный код, где b_1, b_2,..., b_i,..., b_{n-1}, b_n – какие-нибудь произвольно выбранные k-ичные цифры. По определению этот код представляет собой следующее число:

$$b_1b_2...b_i...b_{n-1}b_n=b_1k^{n-1}+b_2k^{n-2}+...$$
$$...+b_ik^{n-i}+...+b_{n-1}k^1+b_nk^0.$$

В качестве примера переведем двоичный код 10110011 в десятичный:

$$10110011_2=$$
$$=1\cdot2^7+0\cdot2^6+1\cdot2^5+1\cdot2^4+0\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0=179_{10}.$$

При одновременном рассмотрении кодов с различными основаниями, во избежание путаницы, справа внизу у кода будем указывать его основание.

Рассмотрим теперь способ обратного перевода десятичного кода числа в k-ичный. Пусть задано некоторое число N в десятичном коде, которое мы хотим перевести в k-ичный код. Предположим, что $b_1b_2...b_n$ есть k-ичный код числа N. Тогда:

$$N=b_1k^{n-1}+b_2k^{n-2}+...+b_{n-1}k^1+b_nk^0.$$

Разделим число N на k:

$$\frac{N}{k}=b_1k^{n-2}+b_1k^{n-3}+...+b_{n-2}k^0+\frac{b_n}{k}.$$

Целая часть частного N_1 этого деления равна:

$$N_1=b_1k^{n-2}+b_2k^{n-3}+...+b_{n-2}k^1+b_{n-1}k^0.$$

В остатке получаем b_n. Итак, в результате деления числа N на k в остатке получаем последний разряд k-ичного кода этого числа. Выполним второе деление, разделив число N_1 на k. В целой части частного получаем число

$$N_2=b_1k^{n-3}+b_2k^{n-4}+...+b_{n-3}k^1+b_{n-2}k^0,$$

а в остатке – b_{n-1}, то есть предпоследний разряд k-ичного кода числа N. При третьем делении в остатке получаем b_{n-2} и т. д. При n-ном делении в остатке получаем b_1, то есть первый разряд k-ичного кода, а в целой части частного – нуль.

В итоге приходим к следующему правилу перевода десятичного кода в k-ичный. Для перевода десятичного кода в k-ичный код следует разделить его на k, полученную целую часть частного снова разделить на k и т. д. Деление продолжать до тех пор, пока в целой части частного получится нуль. Считывая остатки этих делений в обратном порядке, получаем k-ичный код числа. В качестве примера переведем число 169 в двоичный код. Выполняем последовательные деления заданного числа на 2, в скобках записываем остатки делений: 169:2=84 (1), 84:2=42 (0), 42:2=21 (0), 21:2=10 (1), 10:2=5 (0), 5:2=2 (1), 2:2=1 (0), 1:2=0 (1). Считывая остатки, получаем двоичный код числа: $169_{10}=10101001_2$.

Введем число $L(n, k)$ всех различных n-разрядных k-ичных кодов. Оно выражается формулой $L(n, k)=k^n$. Для примера подсчитаем число всех трехразрядных восьмиричных кодов: $L(3, 8)=8^3=512$. Наборы значений аргументов $(x_1, x_2,..., x_n)$ n-местных k-ичных предикатов удобно интерпретировать как n-разрядные k-ичные числовые коды. При этом буквы a_1, a_2,..., a_k алфавита A интерпретируем как k-ичные цифры соответственно 0, 1,..., k-1. В таблице значений предиката наборы значений аргументов будем располагать в порядке возрастания представляемых ими чисел. Именно в таком порядке расположены, к примеру, двухразрядные троичные коды в табл. 1. В этом случае буквы а, м, п интерпретируем соответственно как троичные цифры 0, 1, 2. Введем число $M(n, k)$ всех различных наборов значений аргументов n-местного k-ичного предиката. Очевидно, что $M(n, k)=L(n, k)=k^n$. Число столбцов в таблице значений предиката, в которых располагаются наборы значений аргументов, определяется величиной $M(n, k)$. Например, число столбцов табл. 1 равно $M(2, 3)=3^2=9$. Каждому набору значений конечного предиката присвоим свой номер, в качестве которого примем число, соответствующее этому набору при его интерпретации в виде числового кода.

Пронумеруем все k-ичные n-местные предикаты. С этой целью будем интерпретировать последовательность значений предиката, расположенную в нижней строке его таблицы, как kn-разрядный двоичный код. При этом символ 0 интерпретируем как цифру нуль, а символ 1 – как цифру один. Число, соответствующее этому коду, примем в качестве номера конечного предиката. К примеру, предикату, представленному табл. 1, присваиваем номер 001110011, что соответствует числу 115 в десятичной записи. Пусть $N(n, k)$ — число всех различных n-местных k-ичных предикатов. Очевидно, оно совпадает с числом всех k^n-разрядных двоичных кодов, поэтому $N(n, k)=L(k^n, 2)=2^{(k^n)}$. В виде примера определим число всех различных двуместных троичных предикатов: $N(2,3)=2^{(3^2)}=2^9=512$.

4. Применения конечных предикатов

Каждому конечному алфавитному оператору можно поставить в соответствие некоторый свой конечный предикат. Сделать это можно следующим образом. Пусть

$$F(x_1x_2...x_m)=y_1y_2...y_m$$

– произвольно выбранный конечный алфавитный оператор, преобразующий входные слова $x_1x_2...x_m$ длины m в выходные слова $y_1y_2...y_m$ той же длины, составленные из букв алфавита A. Построим

$2m$-местный конечный предикат $f(x_1, x_2,..., x_m, y_1, y_2,..., y_m)$ над алфавитом A, руководствуясь следующим правилом:

$$f(x_1, x_2,..., x_m, y_1, y_2,..., y_m) = \begin{cases} 1, \text{ если } F(x_1x_2...x_m) = y_1y_2...y_m, \\ 0, \text{ если } F(x_1x_2...x_m) \neq y_1y_2...y_m. \end{cases} \quad (1)$$

Запишем уравнение:

$$f(x_1, x_2,..., x_m, y_1, y_2,..., y_m)=1. \quad (2)$$

Это уравнение связывает переменные $x_1, x_2,..., x_m, y_1, y_2,..., y_m$ некоторым отношением [5]. Подставляя в (2) буквы $x_1, x_2,..., x_m$ входного слова $x_1x_2...x_m$ алфавитного оператора F, получим в результате решения этого уравнения буквы $y_1, y_2,..., y_m$ выходного слова $y_1y_2...y_m$. Таким образом, предикат f, построенный указанным способом, содержит в себе всю информацию об интересующем нас алфавитном операторе F. С его помощью можно определить выходное слово алфавитного оператора, представляемого этим предикатом, для любого входного слова.

Важно отметить, что точно таким же способом можно задать с помощью конечных предикатов не только всюду определенные, но также и любые частичные конечные алфавитные операторы. Если для входного слова $x_1x_2...x_m$ алфавитный оператор F не ставит в соответствие никакого выходного слова, это значит, что уравнение (2) для заданного набора значений аргументов $(x_1, x_2,..., x_m)$ не имеет ни одного решения относительно набора переменных $(y_1, y_2,..., y_m)$, то есть что решения этого уравнения не существует. Важно, что с помощью одного конечного предиката можно задать целое семейство конечных алфавитных операторов. Это достигается введением в предикат дополнительной переменной z, значениями которой служат номера задаваемых алфавитных операторов. Пусть, к примеру, нам нужно представить в виде конечного предиката семейство, состоящее из трех алфавитных операторов:

$$F_1(x_1x_2...x_m)=y_1y_2...y_m,$$
$$F_2(x_1x_2...x_m)=y_1y_2...y_m,$$
$$F_3(x_1x_2...x_m)=y_1y_2...y_m.$$

Строим $2m+1$-местный предикат $f(x_1, x_2,..., x_m, y_1, y_2,..., y_m, z)$, полагая, что при $z=1$ он соответствует алфавитному оператору F_1, при $z=2$ – оператору F_2, при $z=3$ – оператору F_3.

С помощью конечных предикатов можно представить не только интересующие нас конечные алфавитные операторы с однозначными значениями, но и так называемые многозначные операторы. *Многозначным конечным алфавитным оператором* называется такое соответствие F, которое связывает каждое входное слово из A^m с некоторым семейством входных слов из того же множества. Если входному слову $x_1x_2...x_m$ алфавитный оператор F ставит в соответствие несколько входных слов, значит, уравнение (2) для заданного набора значений аргументов $(x_1, x_2,..., x_m)$ имеет несколько решений относительно набора переменных $(y_1, y_2,..., y_m)$.

Многозначный оператор можно рассматривать как обобщение понятия однозначного конечного алфавитного оператора. *Однозначный конечный алфавитный оператор* – это такой многозначный конечный оператор, который каждому входному слову ставит в соответствие некоторое множество выходных слов, состоящее не более чем из одного слова. Если каждому входному слову ставится в соответствие множество, состоящее не менее чем из одного слова, то такой конечный алфавитный оператор называется *всюду определенным*, если же некоторым из входных слов ставится в соответствие пустое множество слов, то – *частичным*.

Поначалу, быть может, несколько обескураживает то, что конечные предикаты дают нам больше, чем мы от них ожидали. В самом деле, введя конечные предикаты, мы хотели получить средство представления только для однозначных конечных алфавитных операторов, фактически же получили в виде «бесплатного приложения» возможность представления и многозначных операторов. Хорошо это или плохо? Если бы оказалось, что многозначные алфавитные операторы бесполезны для теории интеллекта, то это, конечно, было бы плохо, так как в теории интеллекта появились бы неинтерпретируемые математические структуры. К счастью, однако, обнаруживается, что многозначные алфавитные операторы (а это покажет дальнейшее изложение) весьма удобны для математического описания высказываний или суждений интеллекта – важных объектов теории интеллекта. Однозначным же алфавитным операторам отводится роль средства математического описания деятельности интеллекта.

Читатель, вероятно, заметил, что при переходе от конечных алфавитных операторов к соответствующим им конечным предикатам мы одновременно совершили переход от переменных X и Y, значениями которых были слова, к переменным $x_1, x_2,..., x_m, y_1, y_2,..., y_m$, пробегающим буквенные значения. Переход к буквенным переменным очень важен, он дает нам в руки удобный математический язык для описания функций интеллекта. Важно понять, что буквенные переменные было бы невозможно использовать, если б мы в свое время не перешли к конечным алфавитным операторам, а продолжали базироваться на понятии алфавитного оператора, заданного на бесконечном множестве слов произвольной длины.

При попытке представления алфавитных операторов с помощью предикатов с буквенными

переменными нам пришлось бы столкнуться с необходимостью введения бесконечного числа буквенных переменных. Если б мы, уже после введения конечного множества слов, сохранили в данном множестве слова различной длины, то это также помешало бы прийти к буквенным переменным, так как потребовалось бы ввести предикаты от непостижимого переменного числа переменных. Конечные предикаты с буквенными переменными – это та награда, которую мы получаем в обмен на отказ от традиционного понятия алфавитного оператора, базирующегося на абстракции потенциальной бесконечности.

Рассмотрим пример представления конечного алфавитного оператора с помощью конечного предиката. Пусть нам дан оператор $FX=Y$, преобразующий двухбуквенные слова $X=x_1x_2$ русского алфавита в однобуквенные слова $Y=y_1$ того же алфавита. Вид преобразования указан в табл. 2. Этому алфавитному оператору ставим в соответствие предикат $t=f(x_1, x_2, y_1)$, описываемый табл. 3. В таблице указаны не все столбцы, а только те из них, у которых в последней строке стоит значение 1. Полагаем, что во всех отсутствующих столбцах предикат принимает значение 0.

Таблица 2

X	до	ре	ми	фа	ля	си
Y	а	о	у	и	е	я

Таблица 3

x_1	д	р	м	ф	л	с
x_2	о	е	и	а	я	и
y_1	а	о	у	и	е	я
t	1	1	1	1	1	1

Поскольку в русском алфавите 33 буквы, то общее число столбцов таблицы должно было бы составлять $33^3 \approx 30$ тысяч. В связи со столь резким увеличением размера таблицы у читателя может сложиться впечатление, что представление операторов в виде конечных предикатов не дает никаких преимуществ и даже усложняет дело, однако это не так. Дело в том, что конечные предикаты, в отличие от конечных алфавитных операторов, которые пришлось бы описывать средствами многозначной логики, допускают весьма удобную аналитическую запись, сходную с формулами алгебры логики [6]. К разработке этой аналитической записи мы сейчас и приступаем.

5. Формулы алгебры конечных предикатов

Для того чтобы получить возможность записывать конечные предикаты в виде формул, мы введем специальную алгебраическую систему, которая называется *алгеброй конечных предикатов*. Точнее, будет введена не одна алгебра, а целое семейство таких алгебр. Каждая алгебра конечных предикатов полностью характеризуется *алфавитом букв* A, состоящим из k символов $a_1, a_2, ..., a_k$, и *алфавитом переменных* B, состоящим из n символов $x_1, x_2,..., x_n$. Средствами алгебры конечных предикатов с алфавитом букв $A=\{a_1, a_2,..., a_k\}$ и алфавитом переменных $B=\{x_1, x_2,..., x_n\}$ может быть записан любой n-местный k-ичный предикат $f(x_1, x_2,..., x_n)$, заданный над алфавитом A. Мы будем считать, что буквы и переменные алфавитов A и B пронумерованы.

Введем понятие *формулы алгебры конечных предикатов* с алфавитом букв $\{a_1, a_2, ..., a_k\}$ и алфавитом переменных $\{x_1, x_2,..., x_n\}$. Формулы будем строить из следующих символов: букв $a_1, a_2,..., a_k$, переменных $x_1, x_2,..., x_n$, знаков дизъюнкции $\vee$ и конъюнкции $\wedge$, открывающей и закрывающей скобок (,), *логических констант* 0 и 1, называемых соответственно *ложью* и *истиной*. Понятие формулы алгебры конечных предикатов определим индуктивно с помощью следующей системы из четырех правил: 1) символы 0 и 1 называем формулами; 2) все выражения вида $a_i(x_j)$, где индекс i находится в пределах от 1 до k, а индекс j – в пределах от 1 до n называем формулами; 3) если выражения A и B являются формулами, то выражение $(A\vee B)$ называем формулой; 4) если выражения A и B – формулы, то выражение $(A\wedge B)$ – тоже формула.

Каждую формулу будем рассматривать как обозначение некоторого предиката $f(x_1, x_2,..., x_n)$, заданного над алфавитом $\{a_1, a_2,..., a_k\}$. Закон соответствия между формулами и обозначаемыми ими предикатами определяем индуктивно следующими правилами: 1) формула 0 обозначает *тождественно ложный предикат*, то есть предикат, равный 0 для всех наборов значений аргументов; 2) формула 1 обозначает *тождественно истинный предикат*, то есть предикат, равный 1 для всех наборов значений аргументов; 3) формула $a_i(x_j)$ обозначает предикат, равный 1 для всех тех наборов значений аргументов, у которых $x_j=a_i$, и равный 0 – для всех остальных наборов. Пусть формула A обозначает предикат f, а формула B – предикат g; тогда: 4) формула $(A\vee B)$ обозначает предикат, равный 0 для всех тех наборов значений аргументов, при которых $f=0$ и $g=0$, и равный 1 – для остальных наборов; 5) формула $(A\wedge B)$ обозначает предикат, равный 1 для всех тех наборов значений аргументов, при которых $f=1$ и $g=1$, и равный 0 – для остальных наборов.

Предикат $(A\vee B)$ называется *дизъюнкцией* или *логической суммой*, а предикат $(A\wedge B)$ – *конъюнкцией* или *логическим произведением предикатов* A и B. Функция, которая ставит в соответствие любым предикатам A и B предикат $(A\vee B)$, называется *операцией дизъюнкции* или *логического сложения предикатов*. Аналогично функцию, сопоставляющую произвольным предикатам A и B предикат $(A\wedge B)$, назовем *операцией конъюнкции* или *логиче-*

ского умножения предикатов. Операции дизъюнкции и конъюнкции будем называть *элементарными операциями алгебры конечных предикатов*. Пусть x, y – логические переменные. Функция $x \vee y$ со значениями $0 \vee 0=0$, $0 \vee 1=1$, $1 \vee 0=1$, $1 \vee 1=1$ называется *функцией дизъюнкции*, а функция $x \wedge y$ со значениями $0 \wedge 0=0$, $0 \wedge 1=0$, $1 \wedge 0=0$, $1 \wedge 1=1$ – *функцией конъюнкции*. В ряде случаев формулы алгебры конечных предикатов удобно рассматривать как значения соответствующих предикатов, то есть как логические константы. При такой интерпретации записи вида $(A \vee B)$ и $(A \wedge B)$ можно рассматривать как функции дизъюнкции и конъюнкции, а входящие в их состав выражения A и B – как логические переменные.

Каждую формулу вида $a_i(x_j)$ удобно рассматривать как одноместный предикат, зависящий только от переменной x_j и определяемый следующим образом:

$$a_i(x_j)=\begin{cases}1, \text{ если } x_j = a_i,\\ 0, \text{ если } x_j \neq a_i.\end{cases} \qquad (3)$$

Предикат $a_i(x_j)$ как бы "узнаёт" одну единственную букву a_i среди возможных букв алфавита A, поэтому назовем его *предикатом узнавания буквы* a_i или просто *узнаванием буквы* a_i, зависящим от переменной x_j. Всего имеется kn различных узнаваний букв. Действуя на различные узнавания букв операциями дизъюнкции и конъюнкции, многократно и в различном порядке, мы получаем различные предикаты. Узнавания букв будем считать *элементарными предикатами алгебры конечных предикатов*. Совокупность элементарных операций и элементарных предикатов образует *базис алгебры конечных предикатов*.

Рассмотрим пример формулы алгебры конечных предикатов. Пусть $k=3$, $n=4$, a_1=а, a_2=м, a_3=п, $x_1=x$, $x_2=y$, $x_3=z$, $x_4=t$. Выражение

$$((\text{а}(y) \wedge \text{а}(t)) \wedge ((\text{м}(x) \wedge \text{м}(z)) \vee (\text{п}(x) \wedge \text{п}(z)))) \qquad (\text{а})$$

есть формула алгебры конечных предикатов. Действительно, согласно второму правилу, выражения а(y), а(t), м(x), м(z), п(x), п(z) – формулы. С помощью четвертого правила из имеющихся уже формул строим следующие формулы: (а(y)$\wedge$а(t)), (м(x)$\wedge$м(z)), (п(x)$\wedge$п(z)). Из двух последних формул по третьему правилу образуем формулу ((м(x)$\wedge$м(z))$\vee$(п(x)$\wedge$п(z))). Наконец, применяя четвертое правило к формулам (а(y)$\wedge$а(t)), ((м(x)$\wedge$м(z))$\vee$(п(x)$\wedge$п(x))), получаем формулу (а).

Недостатком введенных формул является то, что они выглядят неоправданно громоздкими. Это хорошо видно на приведенном примере. Этот недостаток мы компенсируем тем, что будем пользоваться сокращенной записью формул. При переходе от полной записи формулы к сокращенной будем опускать внешние скобки, то есть формулы $(A \vee B)$ и $(A \wedge B)$ мы будем писать в виде $A \vee B$ и $A \wedge B$. Далее, узнавания букв $a_i(x_j)$ будем записывать в более краткой и удобной форме $x_j^{a_i}$, называя букву a_i в этой записи *показателем узнавания буквы*. Наконец, знак конъюнкции в сокращенных записях формул будем заменять точкой или же вовсе опускать $A \wedge B = A \cdot B = AB$.

Кроме того, в записях формул будем опускать все те скобки, наличие которых не обязательно для правильного понимания смысла формулы. Если скобки, регулирующие порядок выполнения действий в сокращенной записи формулы не указаны, то, по принимаемому нами соглашению о старшинстве операций, вначале будем выполнять операции конъюнкции и лишь затем – операции дизъюнкции, например запись $A \vee B \wedge C$ следует понимать как формулу $A \vee (B \wedge C)$. Этим соглашением операция конъюнкции принимается *старшей* по отношению к операции дизъюнкции. При отсутствии скобок в формуле условимся первой из однотипных операций выполнять ту, знак которой стоит в формуле левее, например, $A \vee B \vee C = (A \vee B) \vee C$, $A \wedge B \wedge C = (A \wedge B) \wedge C$. Принятые нами правила сокращения записи формул таковы, что при необходимости всегда можно от сокращенной записи формулы перейти к самой формуле. Применяя, к примеру, только что рассмотренные способы сокращенной записи, формулу (а) можно представить в более компактном и удобном для чтения виде:

$$y^{\text{а}}t^{\text{а}}(x^{\text{м}}z^{\text{м}} \vee x^{\text{п}}z^{\text{п}}). \qquad (\text{б})$$

От формулы всегда можно перейти к таблице обозначаемого ею предиката. Для этого нужно по формуле вычислить значения предиката для всевозможных наборов значений аргументов. Вычислим, к примеру, значения предиката $f(x, y, z, t)$, представленного формулой (б) для наборов значений аргументов (м, а, м, а) и (м, а, п, а):

$$f(\text{м, а, м, а}) = \text{а}^{\text{а}}\text{а}^{\text{а}}(\text{м}^{\text{м}}\text{м}^{\text{м}} \vee \text{м}^{\text{п}}\text{м}^{\text{п}}) =$$
$$= 1 \cdot 1 \cdot (1 \cdot 1 \vee 0 \cdot 0) = 1 \cdot (1 \vee 0) = 1 \cdot 1 = 1,$$
$$f(\text{м, а, п, а}) = \text{а}^{\text{а}}\text{а}^{\text{а}}(\text{м}^{\text{м}}\text{п}^{\text{м}} \vee \text{м}^{\text{п}}\text{п}^{\text{п}}) =$$
$$= 1 \cdot 1 \cdot (1 \cdot 0 \vee 0 \cdot 1) = 1 \cdot (0 \vee 0) = 1 \cdot 0 = 0.$$

Аналогичные вычисления для всевозможных четырехбуквенных наборов, составленных из букв а, м, п (их всего $3^4=81$), показывают, что предикат f «узнаёт» два слова *мама* и *папа*: он ставит им в соответствие значение 1, то есть истину. Остальным словам предикат f ставит в соответствие значение 0, то есть ложь.

Возникает вопрос, *полна* ли введенная нами алгебра конечных предикатов, то есть для каждого ли конечного предиката найдется обозначающая его формула? Оказывается, что алгебра конечных предикатов полна. Действительно, тождественно ложный предикат может быть записан в виде формулы 0.

Предикат, принимающий значение 1 только на одном наборе значений аргументов (σ_1, σ_2,..., σ_n), можно представить формулой $x_1^{\sigma_1}x_2^{\sigma_2}...x_n^{\sigma_n}$. Предикат, принимающий значение 1 на произвольных наборах (σ_{11}, σ_{12},..., σ_{1n}), (σ_{21}, σ_{22},..., σ_{2n}),..., (σ_{m1}, σ_{m2},..., σ_{mn}), может быть представлен формулой:

$$x_1^{\sigma_{11}}x_2^{\sigma_{12}}...x_n^{\sigma_{1n}} \vee x_1^{\sigma_{21}}x_2^{\sigma_{22}}...x_n^{\sigma_{2n}} \vee ... \vee x_1^{\sigma_{m1}}x_2^{\sigma_{m2}}...x_n^{\sigma_{mn}}.$$

Таким образом, любой конечный предикат может быть записан на языке алгебры конечных предикатов.

Алгебра конечных предикатов не только полна, но даже в некотором смысле *избыточна*. Так, мы видим, что введенная в ней первым правилом формула 1 нам не понадобилась для записи произвольных предикатов. Обозначаемый формулой 1 тождественно истинный предикат может быть выражен с помощью других средств алгебры конечных предикатов, например, в форме дизъюнкции всевозможных конъюнкций вида $x_1^{\sigma_1}x_2^{\sigma_2}...x_n^{\sigma_n}$. В случае, когда $k \geq 2$, то есть когда алфавит A содержит по крайней мере две буквы a_1 и a_2, можно обойтись и без формулы 0. Обозначаемый этой формулой тождественно ложный предикат может быть записан, например, в виде формулы $x_1^{a_1}x_1^{a_2}$.

Исключим из этого определения формулы алгебры конечных предикатов первое правило, вводящее формулы 0 и 1; кроме того, предположим, что $k \geq 2$, и рассмотрим вопрос, можно ли в этих условиях еще более упростить определение формулы при сохранении полноты алгебры. Оказывается, что при принятых предположениях алгебра конечных предикатов *несократима* в том смысле, что любые дальнейшие сокращения в оставшихся правилах образования ее формул ведут к неполноте данной алгебры. Действительно, исключим из числа формул одно из выражений: $a_i(x_j)$, вводимое вторым правилом. Тогда мы не сможем образовать формулу для обозначения предиката, обращающегося в 1 на всех тех наборах значений аргументов, у которых $x_j = a_i$, и в 0 – на всех остальных наборах. Если же мы исключим третье правило, тогда станет невозможным представление в виде формулы тождественно истинного предиката. Исключая четвертое правило, мы не сможем представить в виде формулы тождественно ложный предикат.

Таким образом, формулы 0 и 1 не обязательны для алгебры конечных предикатов. Их введение не расширяет выразительные возможности этой алгебры, поскольку и без этих формул алгебра конечных предикатов полна. Символы 0 и 1 включены в число формул лишь потому, что они обеспечивают дополнительные удобства при пользовании алгеброй конечных предикатов. Заметим, что формула 0 для полноты алгебр, у которых $k=1$, необходима. Алгебры этого типа не представляют серьезного интереса для теории интеллекта ввиду их вырожденности. Алфавит A таких алгебр состоит из одной буквы a_1. Предикаты, охватываемые этими алгебрами, принимают значения лишь на одном наборе значений аргументов (a_1, a_1,..., a_1), в котором буква a_1 встречается n раз. Алгеброй охватывается всего два предиката 0 и 1.

6. Тождества алгебры конечных предикатов

Приступим к изучению свойств алгебры конечных предикатов. Сначала рассмотрим ее основные тождества. Назовем *тождественными* формулы, выражающие один и тот же предикат. Факт тождественности формул будем обозначать знаком $\equiv$. Для равенства значений предикатов будем использовать знак $=$. Пусть A, B, C – произвольные формулы алгебры конечных предикатов. Справедливы следующие тождества: *законы коммутативности*

$$A \vee B \equiv B \vee A, \quad (4)$$

$$AB \equiv BA; \quad (5)$$

законы ассоциативности

$$(A \vee B) \vee C \equiv A \vee (B \vee C), \quad (6)$$

$$(AB)C \equiv A(BC); \quad (7)$$

законы дистрибутивности

$$(A \vee B)C \equiv AC \vee BC, \quad (8)$$

$$AB \vee C \equiv (A \vee C)(B \vee C); \quad (9)$$

законы идемпотентности

$$A \vee A \equiv A, \quad (10)$$

$$AA \equiv A; \quad (11)$$

законы элиминации (поглощения)

$$A \vee AB \equiv A, \quad (12)$$

$$A(A \vee B) \equiv A. \quad (13)$$

Названия для приведенных тождеств заимствованы из алгебры логики, где рассматриваются совпадающие с ними по форме законы. Однако в алгебре логики формулы обозначают не конечные предикаты, а булевы функции [7]. Справедливость тождеств легко проверяется перебором всевозможных значений предикатов A, B, C. Например, докажем первый закон коммутативности: $0 \vee 0 \equiv 0 \vee 0$, $0 \vee 1 \equiv 1 \vee 0$, $1 \vee 0 \equiv 0 \vee 1$, $1 \vee 1 \equiv 1 \vee 1$. Справедлив, кроме того, ряд тождеств, в которых участвуют логические константы:

$$A \vee 1 \equiv 1, \quad (14)$$

$$A \cdot 0 \equiv 0, \quad (15)$$

$$A \cdot 1 \equiv A, \quad (16)$$

$$A \vee 0 \equiv A. \quad (17)$$

Пусть x – произвольная буквенная переменная. Имеет место следующее тождество:

$$x^{a_1} \vee x^{a_2} \vee ... \vee x^{a_k} \equiv 1. \quad (18)$$

Действительно, какую бы букву алфавита A мы ни подставили вместо x в левую часть равенства (18), всегда один из дизъюнктивных членов, а вместе с ним и вся дизъюнкция, обращается в 1. Тождество (18) назовем *законом истинности*. Пусть a и b – произвольные, но отличающиеся друг от друга ($a \neq b$) буквы алфавита A. Имеют место следующие тождества:

$$x^a x^b \equiv 0. \qquad (19)$$

Действительно, какую бы букву мы ни подставили вместо x в левую часть равенства (19), всегда хотя бы один из конъюнктивных членов обратится в 0, а вместе с ним обратится в 0 и вся левая часть равенства (19). Тождество (19) назовем *законом ложности*. Закон истинности в некотором смысле можно рассматривать как аналог закона исключенного третьего алгебры логики, а закон ложности – как аналог закона противоречия [1].

С абстрактной точки зрения введенная нами алгебра конечных предикатов есть дистрибутивная решетка с нулем и единицей [8]. А именно – это есть множество всех n-местных k-ичных предикатов с заданными на нем бинарными операциями дизъюнкции и конъюнкции, для которых выполняются аксиомы (4) – (17) дистрибутивной решетки с нулем и единицей. Роль нуля в алгебре конечных предикатов выполняет тождественно ложный предикат 0, роль единицы – тождественно истинный предикат 1. Важно заметить, что в алгебре конечных предикатов выполняются сверх этого еще и тождества (18) и (19). Как будет показано ниже, наличие этих тождеств позволяет ввести в алгебре конечных предикатов третью, унарную операцию – отрицание и рассматривать эту алгебру как булеву алгебру [7]. В силу наличия тождеств (18) и (19) алгебра конечных предикатов оказывается более богатой свойствами, чем булева алгебра, взятая в чистом виде.

Интересно выяснить вопрос: *полна* ли система тождеств, введенных нами в алгебре конечных предикатов. Иными словами, можно ли с помощью этих тождеств доказать тождественность любых двух формул алгебры конечных предикатов, обозначающих один и тот же предикат? Несколько позже будет доказано, что система тождеств (4) – (19) в указанном смысле полна. Эта система тождеств разбивает все множество формул алгебры конечных предикатов на классы эквивалентности [7] таким образом, что каждому из этих классов может быть взаимно однозначно сопоставлен свой конечный предикат.

Не все из перечисленных тождеств, однако, необходимы для достижения полноты. Поэтому число тождеств в системе можно сократить. Например, справедливость тождеств (14) и (15) может быть доказана на основе остальных тождеств:

$$A \vee 1 \equiv 1 \vee A \equiv 1 \vee A \cdot 1 \equiv 1 \vee 1 \cdot A \equiv 1;\ A \cdot 0 \equiv 0 \cdot A \equiv 0 \cdot (0 \vee A) \equiv 0.$$

Из совокупности остальных тождеств выводится второй закон дистрибутивности (9):

$$AB \vee C \equiv AB \vee C \vee CB \equiv AB \vee C \vee CA \vee CB \equiv$$
$$\equiv AB \vee CC \vee CA \vee CB \equiv BA \vee CA \vee BC \vee CC \equiv$$
$$\equiv (B \vee C) \wedge A \vee (B \vee C)\, C \equiv$$
$$\equiv A(B \vee C) \vee C(B \vee C) \equiv (A \vee C)(B \vee C).$$

Было бы интересно указать какую-нибудь несократимую полную систему тождеств алгебры конечных предикатов, удобную в роли системы аксиом для этой алгебры.

Алгебру конечных предикатов можно определить *абстрактно* как некую алгебраическую систему [8]. Рассмотрим множество M, в котором содержатся, по крайней мере, два элемента 0 и 1 и $k \cdot n$ элементов a_{ij} ($1 \le i \le k$, $1 \le j \le n$). Пусть на множестве M заданы две операции $\circ$ и Δ, удовлетворяющие следующим аксиомам:

$$a \circ b \equiv b \circ a,\ a \Delta b \equiv b \Delta a, \qquad (20)$$

$$(a \circ b) \circ c \equiv a \circ (b \circ c),\ (a \Delta b) \Delta c \equiv a \Delta (b \Delta c), \qquad (21)$$

$$(a \circ b) \Delta c \equiv (a \Delta c) \circ (b \Delta c),\ (a \Delta b) \circ c \equiv (a \circ c) \Delta (b \circ c), \qquad (22)$$

$$a \circ a \equiv a,\ a \Delta a \equiv a, \qquad (23)$$

$$a \circ (a \Delta b) \equiv a,\ a \Delta (a \circ b) \equiv a, \qquad (24)$$

$$a \circ 1 \equiv 1,\ a \Delta 0 \equiv 0,\ a \Delta 1 \equiv a,\ a \circ 0 \equiv a, \qquad (25)$$

$$a_{1j} \circ a_{2j} \circ \ldots \circ a_{kj} \equiv 1,\ (1 \le j \le n), \qquad (26)$$

$$a_{i_1 j} \Delta\, a_{i_2 j} \equiv 0,\ (i_1 \neq i_2,\ 1 \le i_1,\ i_2 \le k,\ 1 \le j \le n). \qquad (27)$$

Тождества (20) – (27) назовем *аксиомами абстрактной алгебры конечных предикатов*. Любая алгебраическая система, удовлетворяющая перечисленным требованиям, изоморфна [7] описанной ранее алгебре конечных предикатов. Чтобы перейти от абстрактного определения алгебры конечных предикатов к прежнему ее определению, следует множество M проинтерпретировать как множество всевозможных k-ичных n-местных предикатов, элементы 0 и 1 – как тождественно ложный и тождественно истинный предикаты, элементы a_{ij} – как предикаты $x_j^{a_i}$, операции $\circ$ и Δ – как дизъюнкцию и конъюнкцию предикатов.

Заметим, что только что приведенное определение абстрактной алгебры конечных предикатов допускает еще одну, двойственную первой, интерпретацию. А именно, множество M, как и прежде, интерпретируем как множество всевозможных k-ичных n-местных предикатов, элементы 0 и 1 – как тождественно истинный и тождественно ложный предикаты, элементы a_{ij} как предикаты вида:

$$P_i(x_j) = \begin{cases} 0, \text{ если } x_j = a_i, \\ 1, \text{ если } x_j \neq a_i. \end{cases} \qquad (28)$$

Операции $\circ$ и Δ интерпретируем как конъюнкцию и дизъюнкцию предикатов. Интересно отме-

тить, что в пределах каждой из этих двух интерпретаций абстрактной алгебры конечных предикатов при $k>2$ двойственности не наблюдается (такая двойственность имеет место в алгебре логики и в алгебре конечных предикатов при $k=2$). Хотя при замене знака $\circ$ на Δ и знака 0 на 1, и наоборот, набор тождеств (20) – (25) остается тем же самым, однако тождества (26) и (27) не переходят друг в друга.

Выводы

Может показаться, что алгебра конечных предикатов есть обобщение алгебры логики. Так, мы видим, что обе алгебры относятся к классу булевых алгебр. Кроме того, в частном случае, когда $k=2$ и $A=\{0,1\}$, тождества (18) и (19) превращаются соответственно в закон исключенного третьего $x\vee\bar{x}=1$ и закон противоречия $x\bar{x}=0$ алгебры логики. Здесь обозначено $x^1=x$, $x^0=\bar{x}$. Все тождества (4) – (19) алгебры конечных предикатов в этом случае по форме совпадают с тождествами алгебры логики. И все же неверно было бы утверждать, что алгебра логики – это частный случай алгебры конечных предикатов.

Дело в том, что смысл, вкладываемый в операцию отрицания $\bar{x}$ в алгебре логики и в предикат узнавания нуля x^0 в алгебре конечных предикатов при $k=2$, $A=\{0, 1\}$, различен. В то время как в алгебре логики операцией отрицания можно действовать на любую булеву функцию, в алгебре конечных предикатов при $k=2$, $A=\{0, 1\}$ положение иное: узнаванием нуля разрешается действовать лишь на буквенные аргументы $x_1, x_2,..., x_n$, на предикаты же действовать узнаванием нуля не разрешается. В алгебре конечных предикатов выражения вида A^0, где A – произвольная формула, не являются формулами. А в алгебре логики выражения типа $\bar{A}$ – это формулы.

Таким образом, алгебра логики не является частным случаем алгебры конечных предикатов. В этом обстоятельстве кроется объяснение того факта, что, как было отмечено выше, алгебра конечных предикатов (без 0 и 1 и при $k\geq 2$) несократима. В алгебре же логики набор ее базисных (элементарных) операций (отрицание, дизъюнкция и конъюнкция) можно сократить без потери данной алгеброй свойства полноты (например, можно исключить операцию дизъюнкции), чего не могло бы случиться, если бы алгебра логики была частным случаем алгебры конечных предикатов. Алгебра конечных предикатов при $k=2$ и $A=\{0, 1\}$, как алгебраическая система, насколько нам известно, в литературе специально не рассматривалась. Однако она неявно присутствует во многих руководствах по алгебре логики. Так, например, в литературе можно встретить термин *формулы с тесными отрицаниями*, обозначающий формулы алгебры логики, у которых отрицания могут стоять только непосредственно над независимыми переменными [8]. К такого рода формулам алгебры логики относятся, в частности, дизъюнктивные и конъюнктивные нормальные формы, а также скобочные формы. Эти формы реализуют в виде двухступенчатых или многоступенчатых комбинационных схем, на входы которых поступают двоичные сигналы вместе с их отрицаниями [1].

Список литературы: 1. *Глушков, В.М.* Введение в кибернетику. [Текст] / В.М. Глушков – К.: Изд. АН УССР, 1964. – 324 с. **2.** Энциклопедия кибернетики [Текст] / Т. 1. – К.: Изд. АН УССР, 1974. – 608 с. **3.** *Глушков, В.М.* Алгебра, языки, программирование [Текст] / В.М. Глушков, Г.Е. Цейтлин, Е.Л. Ющенко – К.: Наук. думка, 1974. – 318 с. **4.** *Новиков, П.С.* Элементы математической логики [Текст] / П.С. Новиков – М.: Наука, 1973. – 400 с. **5.** *Шрейдер, Ю.А.* Равенство, сходство, порядок [Текст] / Ю.А. Шрейдер – М.: Наука, 1971. – 256 с. **6.** Дискретная математика и математические вопросы кибернетики [Текст] / Под ред. С.В. Яблонского и О.Б. Лупанова. – М.: Наука, 1974. – Т. 1. – 311 с. **7.** *Мальцев, А.И.* Алгебраические системы [Текст] / А.И. Мальцев – М.: Наука. – 1970. – 394 с. **8.** *Лавров, И.А.* Задачи по теории множеств, математической логике и теории алгоритмов [Текст] / *И.А. Лавров, Л.Л. Максимова* – М.: Наука, 1975. – 247 с.

Поступила в редколлегию 25.05.2011

УДК 519.7

Про алгебру скінченних предикатів / М.Ф. Бондаренко, Ю.П. Шабанов-Кушнаренко // Біоніка інтелекту: наук.-техн. журнал. – 2011. – № 3 (77). – С. 3-13.

Розробляється математична мова опису структур і функцій систем природного інтелекту. Введені поняття скінченного алфавітного оператора, алгебри скінченних предикатів і формул алгебри скінченних предикатів.

Табл. 3. Бібліогр.: 8 найм.

UDC 519.7

About algebra of final predicates / M.F. Bondarenko, Yu.P. Shabanov-Kushnarenko // Bionics of Intelligense: Sci. Mag. – 2011. – № 3 (77). –P. 3-13.

The mathematical language of structures and functions of the natural intellect systems description is developed. The concepts of finite alphabetical operator, algebras of finite predicates and formulas of finite predicates algebra, are entered.

Tab. 3. Ref.: 8 items.