УДК 519.712

Г. Ф. ДЮБКО, П. Л. САМОФАЛОВ

ФОРМИРОВАНИЕ ЯЗЫКОВЫХ СРЕДСТВ ДЛЯ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ В ТАБЛИЦАХ РЕШЕНИЙ

Для любого представления знаний и форм вывода в них прикладные аспекты использования знаний в ЭВМ невозможны без программной реализации базы знаний и выводов. Представление знаний таблицами решений также требует автоматического преобразования таблиц в программы, где знания будут представлены структурами памяти, а выводы соответствующими процедурами.

Знания в таблицах решений представляются не только самой формой таблицы, но также условиями и действиями. Условия могут описывать сложные отношения между объектами реального мира, а действия – процедуры преобразования объектов и отношений. Проблема автоматического преобразования таблиц решений в программы требует использования некоторого компактного языка для описания условий и действий в таблицах. С одной стороны, такой язык должен позволять относительно просто и ясно описывать знания, с другой – языковые конструкции должны быть настолько формальными и информативными, чтобы позволить трансляцию языка представления знаний в язык программирования, например C++ или Пролог.

В данной статье предложены языки описания условий и действий. Для языка описания условий построена грамматика, порождающая условия. Эта грамматика в дальнейшем может быть использована для конструирования транслятора, преобразующего таблицы решений в программы.

Рассмотрена методика формирования условий на синтаксическом атрибутном дереве. Предложены языковые конструкции для описания отношений между различными вершинами дерева. Языковые средства выбраны таким образом, чтобы информации в языковых конструкциях было достаточно для трансляции таблиц в программы.

Продемонстрирована методика описания сложных условий в виде функций, аналогичных функциям C++, и в виде функций, реализованных средствами Пролога.

Разработаны языковые средства, позволяющие использовать циклические таблицы решений. Как и в языке программирования, циклы можно использовать с известным числом повторений таблицы решений (тип *for*), по условию (тип *while*). Циклические таблицы решений можно также использовать для реализации хорновских дизъюнктов.

Предложены и проанализированы языковые конструкции, с помощью которых можно формировать действия. В основном в качестве действий должны использоваться функции C++. Параметрами этих функций являются глобальные переменные, определенные в таблицах.

Можно предложить различные схемы использования таблиц решений для представления знаний, логического вывода, нахождения стратегий вывода и т. д. Во всех случаях использования таблиц решений необходимы языковые средства, посредством которых будут выражаться условия и действия в таблицах. Определим формальный язык, достаточный для описания таблицами решений знаний, необходимых для формирования фактов, рабочих баз знаний, стратегий вывода в интеллектуальных системах, где знания и логический вывод представлены исчислением предикатов 1-го порядка.

На рис. 1 представлена схема использования смысла и таблиц решений в экспертной системе.

При разработке языковых средств представления информации в таблицах решений сделано предположение, что все условия формируются над некоторой стандартной структу-

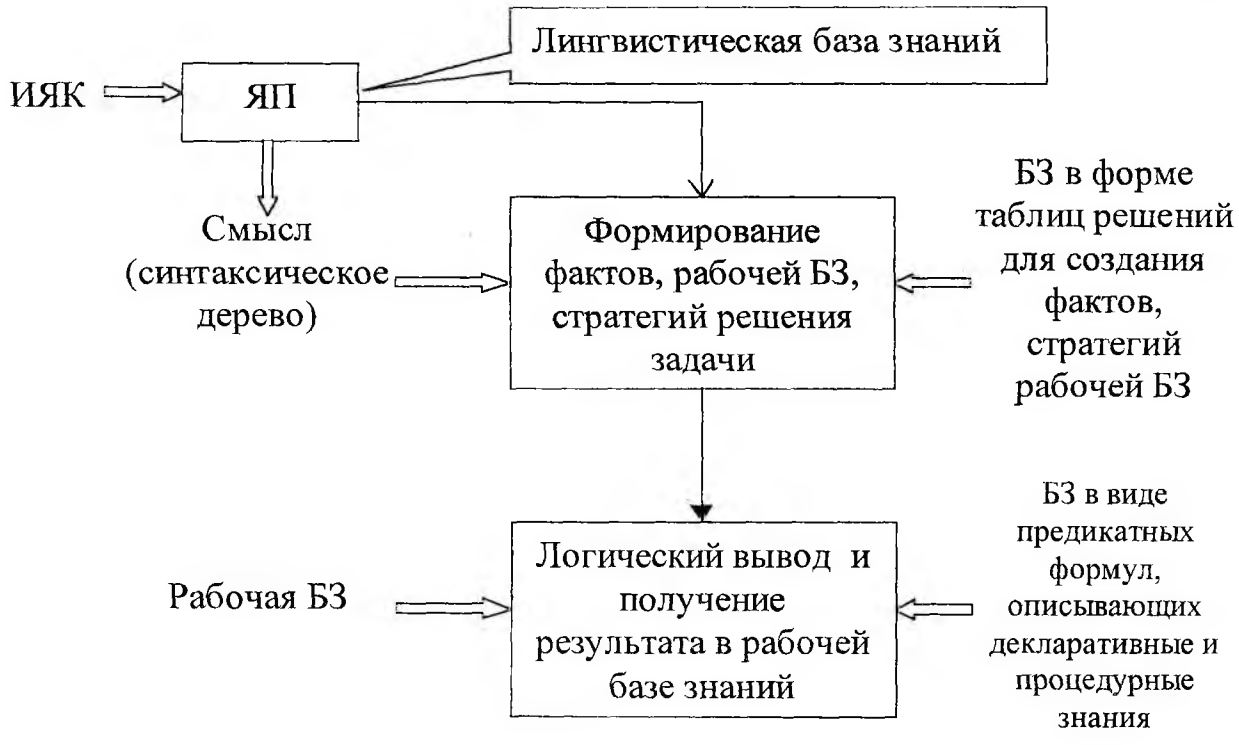

Рис. 1. Использование таблиц решений для генерирования фактов и рабочих баз знаний.

рой STR, задающей в явном виде структурные компоненты (концепты) и отношения между ними. Поскольку смыслом языкового сообщения являются концепты с их свойствами и отношениями между ними, то структуру STR можно считать выражением смысла. В качестве STR можно было бы выбрать множество предикатных формул, семантическую сеть, фреймовое представление или любую другую структуру представления знаний. Так как получение смысла в схеме экспертной системы (рис. 1) осуществляется с помощью языкового процессора, а сам языковой процессор основан на синтаксически-ориентированных принципах, реализованных путем введения контекстно-свободных грамматик (КС-грамматик) специального вида, результат работы ЯП (STR) удобно представлять синтаксическим атрибутным деревом.

Определение

– Помеченное упорядоченное дерево D называется синтаксическим атрибутным деревом в атрибутной *КС*-грамматике $G = (V_T, V_N, P, S)$ если выполнены следующие условия:

– Корень дерева помечен меткой S;

– Если $D_1, \ldots, D_K$ – поддеревья, над которыми доминируют (являются корнями) прямые потомки корня дерева, и корень дерева $D_i (1 \le i \le k)$ помечен символом X_i, то $S \to X_1 \ldots X_k$ – продукция из множества P. D_i должно быть синтаксическим деревом в грамматике $G = (V_T, V_N, P, X_i)$, если X_i – нетерминал; и D_i состоит из единственной вершины, помеченной X_i, если X_i – терминал;

– Если корень дерева имеет единственного потомка, помеченного ε (пустая цепочка), то этот потомок образует дерево, состоящее из единственной вершины, и $A \to \varepsilon$ есть продукция из множества P.

– Все метки синтаксического дерева могут иметь атрибуты (наследуемые или синтезируемые).

Покажем на конкретном примере методику создания таблицы решений (фрагмента БЗ), с помощью которой в ЭС будут формироваться факты, необходимые для логического вывода. Затем на основе анализа этого примера проведем синтез языковых средств, посредством которых будут описываться условия рассматриваемых таблиц решений.

Пусть рассматривается предметная область "эквивалентные преобразования выражений элементарной алгебры". Одним из эквивалентных преобразований является выполнение операции умножения над заданными операндами. В качестве операндов могут быть одночлены, многочлены, алгебраические выражения. Представим фрагмент базы знаний для выполнения умножения следующими предикатами:

$$AB(\alpha_1 * \alpha_2) \wedge MHC(\alpha_1) \wedge ODH(\alpha_2) \supset MHC\left(f_*^{(MH-O)}(\alpha_1, \alpha_2)\right), \quad (1)$$

$$AB(\alpha_1 * \alpha_2) \wedge ODHC(\alpha_1) \wedge ODH(\alpha_2) \supset ODHC\left(f_*^{(ODH-ODH)}(\alpha_1, \alpha_2)\right), \quad (2)$$

$$AB(\alpha_1 * \alpha_2) \wedge ODH(\alpha_1) \wedge ODH(\alpha_2) \supset ODH\left(f_*^{ODH-ODH}(\alpha_1, \alpha_2)\right), \quad (3)$$

$$AB(\alpha_1 * \alpha_2) \wedge MHC(\alpha_1) \wedge MHC(\alpha_2) \supset MHC\left(f_*^{(MH-MH)}(\alpha_1, \alpha_2)\right), \quad (4)$$

где AB – алгебраическое выражение; MHC – многочлен в скобках; $ОДН$ – одночлен; $ОДНС$ – одночлен в скобках; $f_*^{(MH-O)}$ – процедура умножения многочлена на одночлен с результатом в скобках; $f_*^{(ODH-ODH)}$ – процедура умножения одночлена на одночлен с результатом в скобках; $f_*^{ODH-ODH}$ – процедура умножения одночлена на одночлен.

Положим, что исходная строка, подлежащая преобразованию, есть

$$(x^2 - 2x) * x. \quad (5)$$

Подвергнув строку (5) обработке языковым процессором, настроенным на определенную грамматику, получим синтаксическое дерево (рис. 2).

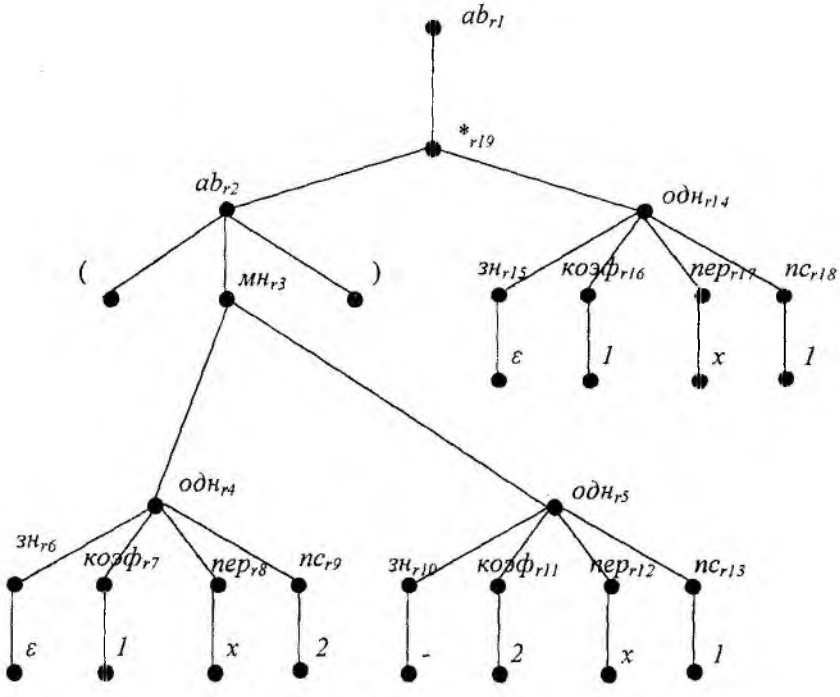

Рис. 2. Синтаксическое дерево с атрибутными символами – метками (r_1, r_2, …, r_{19} - атрибуты).

Таблица решений из соответствующей *БЗ* (см. рис. 1) должна сформировать предикаты *АВ, МНС, ОДН* с их конкретными значениями аргументов. Условия этой таблицы тестируют узлы *STR*. Таким образом, для написания таблицы решений, выполняющей требуемые действия, по меткам узлов *STR* создают условия так, как например, в таблице *ТР1*, обслуживающей формулы (1- 2) (табл. 1).

Таблица 1

	ТР1	R_1	R_2	R_3	E
1	*STR*	1	1	1	
2	*Корень 'ab'*	1	1	1	
3	*Вершина ξ1={потомок [корень]='*'}.*	1	1	1	
4	*Потомок [Вершина ξ1]=(('ав','одн'∧ 'одн''ав') ∧ Потомок['ав']='(','мн',)'*	1	-	-	
5	*Потомок [Вершина ξ1]='одн','одн'*	-	1	-	
6	*Потомок [Вершина ξ1]='мн','мн'*	-	-	1	
		$Д_1$	$Д_2$	$Д_3$	$Д_4$

Действия ТР1 выполняют следующие процедуры:

Д1 – формирование *АВ, МНС, ОДН* с их аргументами;

Д2 – формирование *АВ, ОДН, ОДН* с их аргументами;

Д3 – формирование *АВ, МНС, МНС* с их аргументами;

Д4 – обработка ошибки.

Следовательно, создавая таблицу решений, подобную ТР1, необходимо указать *ту* структуру (имя), узлы которой должны обрабатываться. Наличие имени этой структуры будет одним из условий создаваемой таблицы решений. Остальные условия создаются исходя из предикатной БЗ (формулы (1-5)), концептов и отношений предметной области.

Предикатная БЗ задает также действия, т. е. процедуры формирования фактов или процедуры вывода.

Структурой данных, над которой формируются условия, является дерево STR. Следовательно, операнды операций условий должны быть элементами дерева. В качестве элементов дерева рассматриваются корень, вершина, прямые потомки некоторой вершины, просто потомки, метки, атрибуты меток. При формировании условий между элементами дерева используют операции сравнений, включения, логические операции, операции определения потомков.

Поскольку условия в таблице решений независимы между собой, операнды условий являются локальными для каждого условия. Анализируя таблицу ТР1, нетрудно заметить глобальные операнды, каковыми являются "корень", "вершина 1". При этом все операнды берутся из структуры, задаваемой в первом условии.

Здесь, ниже, приведено описание конструкций языка формирования условий. Семантика конструкции описывается естественным языком, синтаксис - продукциями КС-грамматики.

Сначала формулируем продукции для условий.

$$<\text{Условие}> \rightarrow < \text{имя структуры}> \qquad (6)$$

Семантика продукции (6) заключается в следующем: все предикаты условия проверяются на структуре с указанным именем; имя структуры является глобальной переменной таблицы решений.

<Условие> → < метка > ∈ < фиксированная вершина > |
{< метка >: вершина < целое >} ∈ < фиксированная вершина >. (7)

Семантика продукции (7): вершина, помеченная меткой, включена в поддерево с корнем "фиксированная вершина"; вершина, помеченная меткой, фиксируется и включена в поддерево с корнем "фиксированная вершина".

<Условие> → <фиксация вершины 1> = <последовательность меток>|
<фиксация вершины> = <последовательность меток>. (8)

Семантика продукции (8) такова: под "фиксацией вершины 1" понимается множество потомков, некоторой зафиксированной вершины; это множество потомков одного и того же уровня помечено, с упорядочиванием потомков слева – направо, "последовательностью меток".

<Условие> → < фиксированная вершина > = < метка>. (9)

Семантика продукции (9): некоторая определенная вершина помечена указанной меткой.

<Условие> → <потомок> = (<последовательность фиксаций>)|
<потомок> = (<последовательность меток>). (10)

Семантика продукции (10) заключается в том, что упорядоченная последовательность вершин, являющихся потомками определенной вершины, помечена указанными метками в порядке их перечисления; вершины, помеченные метками, могут быть зафиксированы.

<Условие> → < обращение к функции>. (11)

Семантика продукции (11) такова: условием может быть процедура, параметрами которой являются вершины обрабатываемой структуры.

<Условие> → <значение атрибута> = < значение атрибута>. (12)

Семантика продукции (12): сравниваются значения атрибутов, т. е. строки для двух вершин.

<Условие> → < логическое выражение>. (13)

Семантика продукции (13) заключается в следующем: условие комбинируется из условий, определенных в (6 - 12), с использованием логических связок и кванторов логики предикатов первого порядка.

Далее аналогичным образом составляются продукции для определения нетерминальных символов из продукции (6 - 13).

Языковые средства для формирования циклических таблиц решений

Таблицу решений можно интерпретировать как модель представления знаний [1]. Формализовав представление условий и действий, таблицы решений можно считать средством проектирования базы знаний. Разработка транслятора, превращающего таблицу решений в программу, делает из таблицы решений средство реализации.

Таким образом, таблицы решений, являясь средством проектирования и реализации баз знаний, могут рассматриваться как язык программирования. Любой язык программирования является эффективным средством представления процессов, если он имеет средства описания циклов. Циклы характеризуются средствами управления, которые обеспечивают повторение некоторой последовательности инструкций, изменяющей значения структур данных, представленных в цикле.

Для таблиц решений цикл состоит в повторении данной таблицы решений столько раз, сколько позволяет управляющий элемент. В качестве управляющего элемента цикла можно использовать любое условие таблицы решений, поэтому разрабатывать для управления циклом специальный элемент не целесообразно. Повторение таблицы решений можно осуществлять с помощью стандартного действия "повторить". Действие "повторить" может иметь параметры, изменяющие значения некоторых глобальных переменных таблицы. Другая альтернатива параметрам может состоять в том, что изменение значений глобальных переменных производятся с помощью специальных действий, специфицированных в таблице решений. В этом случае важен порядок действий. Действие "повторить" должно выполняться последним.

Поскольку условия *TP* формируются на глобальной структуре *STR*, при повторении цикла чаще всего будет изменяться значение этой структуры. Начальное значение *STR* должно инициироваться языковым процессором. Это значение представляет собой знания о том, какие сущности представлены на данном этапе решения задачи и в каких отношениях между собой эти сущности находятся. Цикл может описывать формальные рассуждения (вывод), осуществляющие этап решения задачи интеллектуальной системой. В результате выполнения цикла последовательные значения структуры *STR* можно интерпретировать как результаты выполнения шагов вывода, т. е. каждый шаг вывода превращает предыдущее знание в последующее на основе метазнаний.

Рассмотрим формирование циклической таблицы решений на примере приведения подобных в многочлене. Например, многочлен $2x^2y - x^2 + x^2y + 2x^2 - 4x^2y$ с использованием знаний о структуре многочлена и самой операции приведения подобных должен быть приведен к многочлену $-x^2y + x^2$.

Логически знания о приведении подобных в многочлене могут быть описаны следующим множеством формул:

$$Мн(\alpha): - not\ МНП(\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4). \tag{14}$$

$$Мн(\alpha):- МНП(\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4),\ \beta = f_{пр.под.}(\alpha_1, \alpha_2, \alpha_3, \alpha_4), МН(\beta).0 \tag{15}$$

$$\begin{gathered} МНП(\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4):- f_{разд.}(\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4), ОДН(\alpha_2), ОДН(\alpha_3), \\ ПОДОБН(\alpha_2, \alpha_3), (ОДН(\alpha_1); МН(\alpha_1);\varepsilon), \\ (ОДН(\alpha_4); МН(\alpha_4);\varepsilon). \end{gathered} \tag{16}$$

$$\begin{gathered} ПОДОБН(\alpha_2, \alpha_3):- Д_1 = f(атрибут(\alpha_1)),\ Д_2 = f(атрибут(\alpha_2)), \\ arg(3, Д_1, z_1), arg(3, Д_2, z_2), \\ z_1 = z_2. \end{gathered} \tag{17}$$

Нотация в (14-17) близка к прологовской, семантика также. Предикат *arg(x, y, z)* истинен, если *z* есть x-й аргумент в терме y. Однако реализация этих формул в Прологе весьма проблематична. Формулы (14-17) могут служить формальным представлением того, что в действительности нужно выполнить. Например, если формула (14) – истинна, то многочлен не имеет подобных. Если истинна формула (15), то многочлен имеет подобные и получается новый многочлен, который, в соответствии со стратегией вывода в логическом представлении БЗ должен быть подвергнут испытанию на присутствие подобных. Эта формула является рекурсивной и требует цикла для своей организации. Формулы (16-17) определяют, каким образом задается подобность двух подстрок строки, являющейся многочленом.

Взяв за основу знания о приведении подобных, выраженные формулами (14 - 17), нетрудно составить адекватную таблицу решений, которая будет не только описывать знания об операции, но и позволит непосредственную реализацию применения этих знаний (таблица решений ТР2) (табл. 2).

Таблица 2

ТР2	R_1	R_2	E
STR	1	1	
{Мн ∈ STR: вершина 1}	1	1	
Приведение Подобных (вершина 1)	1	0	
$Д_1$	X		
Повторить	X		
Выход из ТР2, так как подобных нет		X	
Выход из ТР2 для определения дальнейших действий			X

Элементы таблицы решений ТР2 определены следующим образом. Функция *ПриведениеПодобных* описана в предыдущем параграфе и вырабатывает условие наличия двух подобных одночленов. Подобные одночлены фиксируются двумя вершинами *ВершинаТек1, ВершинаТек".* Для того, чтобы такая фиксация была возможна, положим, что семантика языковой конструкции *"ВершинаТек<целое>"* требует создания текущей нумерации вершин дерева STR, независимой от основной нумерации. Если *ПриведениеПодобных* есть *true*, выполняются действия в порядке их записи: сначала $Д_1$, затем "*Повторить*". Действие $Д_1$ может быть представлено функцией *ТрансформацияПрПод(STR, ВершинаТек1, ВершинаТек2),* которая перестраивает структуру *STR* в соответствии с правилами приведения подобных. В результате выполнения этой функции структура *STR* будет содержать новое дерево. Действие повторить приведет к выполнению этой же таблицы. Отметим, что все фикцации вершин старого дерева отменяются и производятся новые в соответствии с новым значением *STR.*

Если в структуре *STR* нет подобных или ситуация выходит за рамки правил R_1, R_2, то осуществляется выход из таблицы.

Пример приведения подобных, описанный таблицей ТР2, является типичным для представления знаний, выраженных рекурсивными логическими формулами, где процесс получения результата при манипуляции знаниями сводится к итерации, т. е. к новой обработке полностью вычисленных значений на предыдущем шаге цикла.

Существуют формы представления знаний с помощью рекурсивных логических формул, которые требуют рекурсивного процесса обработки, когда на каждом шаге цикла получаются "недовычисленные" значения структур. Эти "недовычисленные" значения могут быть вычислены окончательно, когда цикл вычислений завершится, т. е. будет получено значение простейшей рекурсивной структуры. Рассмотрим более подробно процесс получения результата с помощью подобной рекурсии на конкретном примере.

Пусть отношение *SumList(Xs, Sum)* описывает отношение "сумма элементов списка *Xs* равна *Sum"* и определяется следующими логическими формулами:

$$SumList([X|Xs], Sum) :- Sum = Sum1 + X, SumList(Xs, Sum1)$$
$$SumList([\,], 0). \qquad (18)$$

Процесс обработки этих формул приводит к последовательности определений $Sum0 = Sum1 + X0$, $Sum1 = Sum2 + X1, \ldots$, $SumN = Xn$, которые не являются операторами присваивания и могут быть вычислены только тогда, когда будут определены все их элементы. Элемент *Xn* считается определенным, поэтому должен быть выполнен цикл последова-

тельного вычисления *Sum1* для получения окончательного результата *Sum0*. Процесс, управляемый формулами (18) может быть представлен таблицей решений ТР3 (табл. 3).

Таблица 3

ТР3	R_1	R_2	E
Xs	1	1	
Sum[i]	1	1	
Xs =[X|Ys]	1	0	
$Д_1$	X		
Sum[i] := X	X		
i:=i+1	X		
Повторить	X		
$Д_2$		X	
$Д_3$			X

Переменные *Xs, Sum*[*i*], *X, Ys, i, Sum* являются глобальными переменными таблицы и их нужно соответствующим образом описывать. *Xs, Sum*[*i*] являются небулевскими переменными, представляющими условие. Они по умолчанию считаются глобальными, а описываемые ими условия приобретают значение true, если эти переменные объявлены и инициализированы вне таблицы решений. Для остальных глобальных переменных необходимо явное указание. Языковая конструкция {<*переменная*>: *глобальная*} годится для указания глобальной переменной, поэтому условия 2, 3 ТР3 нужно представить в виде

Sum[{i: глобальная}],
Xs = [{X: глобальная}|{Ys: глобальная}].

Необходимо добавить соответствующие правила для представления условий, которые позволяют задавать структуры в виде списка и массива, а также разбивать список на голову и хвост.

Опишем теперь действия таблицы ТР3 (табл. 3).

$Д_1$ – преобразование списка Xs в Ys (удаление головы списка *Xs*) и присвоение Xs := Ys.

$Д_2$ – вычисление значения рекурсивной структуры в соответствии с алгоритом

Sum = 0;
For(k=1; kJi; k++)
Sum = Sum + Sum[k];.

$Д_3$ – принятие решения в случае нештатной ситуации.

Правила составления таблицы ТР3.2. являются типичными для процессов обработки знаний, представленных рекурсивными логическими формулами, приводящими к рекурсивному вычислению.

Методы представления знаний таблицами решений, описанные в этой статье, могут быть применены для формирования баз знаний в экспертных системах и являются частью технологии проектирования и реализации экспертных систем [2].

Список литературы: 1. *Дюбко Г.Ф. Самофалов П.Л.* Описание знаний с использованием таблиц решений // Автоматизированные системы управления и приборы автоматики. 1997. Вып. 104. Харьков. С. 81-85. 2. *Дюбко Г. Ф. Самофалов П.Л.* Технология разработки интеллектуальных систем // Информационные системы: Сб науч. тр. 1994. Харьков: НАНУ, ПАНИ, ХВУ. Вып. 2. С. 57-60.

Поступила в редколлегию 30.11.2000