УДК 510.662:004.832.32

А.Б.Николенко

Казахский национальный технический университет им. К.И.Сатпаева, Алматы (E-mail: abnikolenko@gmail.com)

Моделирование итерационных вычислений в системе обобщенного естественного вывода

В статье освещены вопросы формализации итерационных вычислений. Составлена система первого порядка, предназначенная для моделирования содержательных математических рассуждений. Описан процесс построения логических выводов в этой системе, из которых затем извлекаются итерационные программы для решения задач обработки массивов целых чисел. Приведены теоремы о конструктивности выводов и правильности извлекаемых программ.

Ключевые слова: логика предикатов, итерационные вычисления, система естественного вывода, дедуктивный синтез программ.

В [1 и 2; 428, 614] отмечается, что практическое применение логики предикатов обусловлено расширением дедуктивных средств. Это в полной мере касается и прикладных систем условного планирования и синтеза программ. В дедуктивном синтезе программ [2–5], как правило, применяется следующая стандартная схема: исходная задача формулируется в виде формулы логики первого порядка, затем строится ее доказательство в системе логического вывода, после чего из доказательства извлекается программа, решающая исходную задачу. В этой схеме задача представляется в виде формулы

$$\forall \bar{x}(A(\bar{x}) \to \exists \bar{y} B(\bar{x}, \bar{y})),$$

где входные объекты кортежа $\bar{x}$, удовлетворяющие условиям $A(\bar{x})$, связаны с искомыми объектами кортежа $\bar{y}$ условиями $B(\bar{x}, \bar{y})$.

Конструктивное доказательство ищется или методом резолюций, или в некоторой системе натурального типа. Затем из доказательства извлекается заведомо правильная программа (чаще всего) функционального языка программирования (например, LISP или ALGOL-68) [2–5].

Стандартные системы логического вывода не имеют средств моделирования итерационных вычислений. Попытки «встроить» математическую индукцию в используемую систему логического вывода [3–5] малоэффективны и до определенной степени искусственны.

В [6–8] предложен подход к дедуктивному синтезу программ с циклами. В основе формализма лежит система первого порядка GN_1, правила вывода которой (ИС — извлечение следствий и ФП — формирование подцелей), с одной стороны, позволяют моделировать прямые и обратные рассуждения соответственно, а с другой — являются обобщениями правил традиционной системы натурального вывода [9].

В настоящей работе строится система дедуктивного синтеза программ, в рамках которой мы усовершенствуем традиционную схему синтеза. При этом один из основных принципов развиваемого подхода заключается в том, что логический вывод, по которому затем конструируется программа, состоит из двух частей — одна моделирует процесс построения алгоритма (программы), а другая является доказательством обоснования (если это необходимо в явном виде) его (алгоритма) правильности.

Рассмотрен синтез алгоритмов для решения задач поиска и изменения состояния массива и приводятся теоремы о конструктивности выводов и корректности извлекаемых программ.

Для упрощения изложения рассматриваются задачи обработки одномерных массивов целых чисел. Результаты работы могут быть перенесены как на другие структуры данных: многомерные массивы, строки, файлы, так и на конечные математические объекты — множества, последовательности и т.п.

Ввиду ограниченности объема работы, дальнейшее изложение будет до определенной степени схематичным. Это касается полного определения вывода и описания алгоритма извлечения программ из доказательств.

Приведем теперь некоторые обозначения и определения.

Символ ■ используется для обозначения конца определения, замечания, доказательства и т.п.

Термы и формулы используемого языка первого порядка L определяются обычным образом. Символ Ω используется для обозначения пустой формулы. В определение формулы языка L добавляется пункт — «пустая формула есть формула».

Через $\bar{x}=(x_1, x_2,\ldots, x_n)$ обозначаем кортеж n предметных переменных. Аналогично определяются кортежи для других типов символов, а также для термов и т.п. Термы (включая те, которые определяются ниже) будем обозначать буквами $s,t,$ возможно, с индексами и с указанием свободно входящих в них переменных, констант или других термов. Любые формулы будем обозначать прописными курсивными буквами латинского алфавита, при необходимости указывая вхождения свободных переменных и/или термов. Например, $A, B(x,t), P, F(\bar{x}), G(x_1,x_2,...,x_n)$. Причем указанные в терме (формуле) свободные вхождения не обязательно должны в действительности присутствовать в этом терме (формуле).

Как обычно, через $Q\bar{x}$ обозначаем кванторную приставку $Q_1x_1...Q_kx_k$, где каждое Q_i есть символ $\forall$ либо $\exists$. Формулу с кванторной приставкой $Q(Q_1x_1...Q_kx_k)$ будем называть Q**-формулой** ($Q_1...Q_k$ **-формулой**).

Формулу вида $Q\bar{x}(A_1 \& A_2 \& ... \& A_k)$ называем **конъюнкцией**, формулу вида $Q\bar{x}(A_1 \vee A_2 \vee ... \vee A_k)$ — **дизъюнкцией**, формулу вида $Q\bar{x}(A_1 \to A_2)$ — **импликацией**. Во всех случаях формально не исключается случай $k=1,$ равно как и отсутствие кванторной приставки.

Конъюнкция $Q\bar{x}(A_1 \& A_2 \& ... \& A_k)$ называется **простой конъюнкцией**, если каждая A_i является атомной формулой или ее отрицанием. При $k=1$ будем говорить о **простой формуле**. Дизъюнкция $Q\bar{x}(A_1 \vee A_2 \vee ... \vee A_k)$ называется **простой дизъюнкцией**, если каждая A_i является простой конъюнкцией.

Запись $A=B$ означает графическое совпадение формул.

Определение 1. Формула **содержит** формулу B (B **содержится** в A) в следующих случаях:

1) если A — атомная формула или ее отрицание, то $A=B$;
2) Ω содержится в любой формуле, т.е. если $A=\Omega,$ то B — произвольная;
3) если $A=A_1 \& A_2 \& ... \& A_k, B=B_1 \& B_2 \& ... \& B_m,$ тогда каждая B_j совпадает с некоторой A_i (не исключается случай $k,m=1$);
4) если $A=A_1 \vee A_2 \vee ... \vee A_k,$ то каждая A_i содержит B;
   если $B = B_1 \vee B_2 \vee ... \vee B_m,$ то A содержит хотя бы одну B_1;
5) если $B=\exists x B_1(x),$ то либо найдется такой терм t из A, что $B_1(t)$ содержится в A, либо $A=\exists x B_1(x)$;
6) если $B=\exists x B_1(x),$ то $A=\exists y A_1(y),$ и B_1 содержится в A_1. ■

Определение 2. Формула F языка L называется **канонической**, если:

1) $F=\forall\bar{x}(A(\bar{x})\to\exists\bar{y}B(\bar{x},\bar{y}))$ и найдутся такие термы $\bar{t}(\bar{x})$ из $A(\bar{x}),$ что $B(\bar{x},\bar{t}(\bar{x}))$ содержится в $A(\bar{x});$ переменные кортежа $\bar{y}$ могут отсутствовать;

2) $F=F_1 \& ... \& F_k,$ где каждая F_i является канонической импликацией. ■

Нетрудно показать, что каноническая импликация — это тождественно истинная формула, которая содержит в посылке то, что находится в следствии. Приведение доказываемой формулы к каноническому виду будет критерием окончания доказательства.

Дадим теперь определение правил вывода системы $GN_i,$ которые будем называть правилами обобщенного натурального вывода. Основой этих правил являются преобразования формул [10]. Для наших целей достаточно ограничиться применением правил вывода к $\forall$ - формулам. Символ Ω позволяет любую формулу $Q\ \bar{x}\ A$ представить формально в виде импликации $Q\bar{x}(\Omega\to A),$ что обеспечивает применение правил вывода к любым формулам (при этом считается, что переменные кортежа $\bar{x}$ входят в Ω).

Система GN_i содержит два правила.

1) **Извлечение следствий (ИС).**

$$\frac{\forall\overline{x}(A(\overline{x})\to B(\overline{x}))\quad \forall\overline{u}(C(\overline{u})\to\exists\overline{y}D(\overline{u},\overline{v}))}{\forall\overline{x}\forall\overline{z}(A(\overline{x})\,\&\,D(\overline{x},\overline{z})\to B(\overline{x}))},$$

при этом должны выполняться:

а) формула над чертой слева — доказываемая формула;

б) формула справа может быть либо аксиомой, либо может входить конъюнктивно в $A(\overline{x})$; в последнем случае применение правила называется **внутренним**;

в) C содержится в A при соответствующем переименовании переменных;

г) переменные кортежа $\overline{z}$ не входят в кортеж $\overline{x}$.

2) **Формирование подцели (ФП).**

$$\frac{\forall\overline{x}(A(\overline{x})\to B(\overline{x}))\quad \forall\overline{u}(C(\overline{u})\to D(\overline{u}))\to\forall\overline{v}(F(\overline{v})\to\exists\overline{w}G(\overline{v},\overline{w}))}{\forall\overline{x}\forall\overline{z}(A(\overline{x})\,\&\,G(\overline{x},\overline{z})\to B(\overline{x}))\quad \forall\overline{x}\forall\overline{u}(A(\overline{x})\,\&\,C(\overline{u})\to D(\overline{u}))},$$

при этом должны выполняться:

а) доказываемая формула — над чертой слева;

б) справа над чертой — аксиома, формула, входящая конъюнктивно в $A(\overline{x})$, или гипотеза; в последнем случае справа вверху должен стоять вывод формулы $\forall\overline{v}(F(\overline{v})\to\exists\overline{w}G(\overline{v},\overline{w}))$ из формулы $\forall\overline{u}(C(\overline{u})\to D(\overline{u}))$;

в) при соответствующем переименовании переменных F содержится в A, а G содержится в B;

г) формула под чертой справа — **подцель**, которую нужно доказывать вместо G;

д) если формула справа над чертой содержится в посылке доказываемой формулы, то подцель — **внутренняя**;

е) формулы C и F могут отсутствовать;

ж) слева под чертой — доказываемая формула с учетом того, что подцель уже доказана.

Замечание 1. На самом деле формула, стоящая слева в правой формуле над чертой, может быть конъюнкцией импликаций. В этом случае подцелей — несколько. ■

Определение 3. Выводом (**доказательством**) называется дерево, растущее вниз, в корне которого (самая левая вверху) стоит доказываемая формула; все листья дерева — канонические формулы. ■

Замечание 2. Нам будет достаточно приведенных формулировок правил ИС и ФП. В действительности правила вывода определяются для произвольных кванторных приставок и расширяются на случай вхождения термов. Данная система вывода разрабатывалась для моделирования математических доказательств, причем такие элементы содержательных доказательств, как «разбор случаев» и «доказательство от противного» можно интерпретировать в терминах приведенных правил ИС и ФП; вместе с тем для целей синтеза программ мы должны конструктивно ограничивать применение указанных правил. Автором доказана полнота и непротиворечивость системы GN_i. ■

Объекты рассматриваемой предметной области — целые числа и массивы целых чисел. Массив рассматривается как совокупность независимых переменных. Фрагмент языка первого порядка L для формализации предметной области определяется так, что большинство используемых функций и предикатов имеют общеупотребительное значение. Для того, чтобы показать, каких типов функции и предикаты используются в данной предметной области, и рассмотреть примеры, демонстрирующие рассматриваемый подход, приведем фрагмент сигнатуры языка L.

1. Функциональные константы (функции):

а) **первичные функции**:

+, -, *, / (/ — деление нацело);

- x [i, s] — значение i-ого элемента массива x в состоянии s;

б) **функции**:

– *ср.арифм.* (x, y) — среднее арифметическое;

– *min* (j) — наименьшее значение переменной (левая граница отрезка);

– *max* (j) — наибольшее значение переменной (правая граница отрезка);

– *кол-во (x)* — количество элементов множества x;

– $\sum(x)$ — сумма всех элементов из x.

2. Предикатные константы (предикаты):

а) **первичные предикаты:**

– >, <, =, ≤, ≥ — порядок и равенство на целых числах;

б) **типы**:

– *x* (1: *n*) — массив целых чисел размерности от 1 до *n*;

– *цел. (x)* — *x* целое;

в) **свойства:**

– *чет.* (*x*) — *x* — четное число;

г) **предикаты:**

– $x \in y$ — принадлежность;

– *макс* (*x*) — максимальный элемент множества *x*;

– *миним.* (*x*) — минимальный элемент множества *x*;

– *сост. (x, s)* — *s* есть состояние объекта *x*;

– $i = \overline{1,n}$ — *i* изменяется от 1 до *n*.

Аксиомы предметной области и другие элементы сигнатуры будут вводиться по мере надобности.

Для извлечения программ из доказательств используем целевой язык программирования ***LP***, который содержит:

– ***integer***, ***array*** — описания типов для целых чисел и массивов целых чисел;

– ***skip*** — пустой оператор;

– ***exit*** — оператор выхода из программы;

– **:=** — оператор присваивания;

– ***if then do od else do od*** — условный оператор (с операторными скобками);

– ***for to do od*** — оператор цикла;

– ***while do od*** — оператор цикла;

– ***input***, ***output*** — операторы ввода/вывода;

– ***begin end*** — скобки для программы, подпрограммы или блока;

– ***print*** — оператор печати.

В сигнатуру языка ***LP*** входят:

– **+, -, *,** / — арифметические операции (последняя — деление нацело);

– >, <, =, ≤, ≥ — отношения равенства и порядка на числах.

Первичным функциям языка ***L*** поставим в соответствие операции языка ***LP***, первичным предикатам — отношения из ***LP***, а типам — описания ***integer*** и ***array***. Термы, составленные из первичных функций, называем **вычислимыми (относительно языка *LP*)**, а формулы, составленные из первичных предикатов и типов и содержащие только вычислимые термы, — **разрешимыми (относительно языка *LP*)**. Более точные и общие определения приводятся в [11].

Для определения понятия итерационного вывода рассмотрим следующую задачу:

Задача поиска: найти все элементы массива, удовлетворяющие заданному свойству.

Рассмотрим в качестве примера следующую задачу: «в массиве целых чисел найти все элементы больше нуля».

Сформулируем эту задачу по-другому:

Дано: *n* — произвольное целое число;

x — произвольный массив целых чисел размерности *n*. (*)

Найти: ***все*** *i* — такие, что *x* [*i*]>0.

Программа для решения этой задачи на языке высокого уровня выглядит следующим образом:

```
begin integer n; array x (1: n); input (n, x);
        for i = 1 to n do                                   (1)
                if x [i] > 0 then do output (i) od; end.
```

Формула для спецификации (*):

$$\forall n \forall x (x(1:n) \to \exists i (x[i] > 0) \vee \neg \exists i (x[i] > 0)). \qquad (2)$$

Замечание 3. Понятно, что порядок кванторов всеобщности имеет значение. Переставить универсальные кванторы из (2) нельзя. Кроме того, квантору существования в данном случае мы придаем смысл «существуют все *i* такие, что…», а не «найти хотя бы одно *i* такое, что…». Это связано с

тем, что мы хотим построить алгоритм нахождения всех элементов, удовлетворяющих заданному свойству. То есть в данном случае используется тот факт, что существование (на конечных объектах) является обобщением дизъюнкции. ■

Замечание 4. Мы видим, что нам приходится доказывать формулы вида

$$\forall\overline{x}(A\to B\vee\neg B).$$

С точки зрения классической логики данная формула общезначимая. Однако для синтеза программы необходимо конструктивное доказательство, т.е. явное построение требуемых объектов. Поэтому такие формулы надо доказывать. Кроме того, в классической логике для доказательства импликации с произвольной дизъюнкцией в следствии достаточно получить доказательство хотя бы одного из ее членов. В нашем случае мы должны строить части программы для проверки каждого члена дизъюнкции. И если для простых задач проверка отрицания может быть тривиальной (просто выход из программы), то в более сложных задачах проверка негативного члена дизъюнкции в программе соответствует таким действиям, как возврат, чистка и сдвиг. ■

Для построения вывода формулы (2) нам понадобятся следующие аксиомы.

А1. $\forall n\forall x(x(1:n)\to\exists s(\textit{исх\_сост.}(x,s))$ — для любого массива существует исходное состояние, в котором элементы массива имеют произвольное значение.

А2. $\forall n\forall x\forall s(x(1:n)\ \&\ \textit{сост.}(x,s)\to\forall i(\textit{цел.}(x[i,s]))$ — определение массива целых чисел.

А3. $\forall y(\textit{цел.}(y)\to y>0\vee\neg(y>0))$ — линейность порядка на целых числах.

Вывод в системе GN_i формулы (2) выглядит следующим образом (через A обозначено следствие формулы (4.3)):

$$\frac{\forall n\forall x(x(1:n)\to A(x,i))}{\forall n\forall x\forall j(x(1:n)\ \&\ A(x,j)\to A(x,i))\quad \underset{\Sigma}{\forall i(\Omega\to x[i]>0\vee\neg(x[i]>0))}}. \qquad (3)$$

Здесь формула под чертой справа — подцель, которую нужно доказывать и которая означает, что необходимо проверить разрешимость свойства A; слева под чертой — измененная доказываемая формула с учетом того, что подцель уже доказана; эта формула — каноническая и доказывать ее не надо; фигура $\sum$ является выводом подцели и содержит применения правила ИС с аксиомами А1–А3.

Вывод (3) является обобщением всех натуральных выводов для конкретных размерностей массива. Вывод (3) называем **итерационным выводом** для решения задачи (*). Из вывода (3) **извлекается программа** (1).

Перейдем теперь к формализации задач в GN_i и синтезу программ.

Сформулируем задачу нахождения элементов массива с заданным свойством.

Пусть $A(x[i],A_1,\ldots,A_k,\overline{t}(x,i))$ — простая дизъюнкция, в которой x — переменная для массива; i — номер элемента, который надо найти; $A_1,\ldots,A_k$ — простые бескванторные конъюнкции, входящие в A; $\overline{t}(x,i))=(t1(x,i),\ldots,tl(x,i))$ — все термы, входящие в A.

Формула, соответствующая исходной задаче, имеет вид:

$$F=\forall n\forall x(x(1:n)\to\exists i(A(x[i])\vee\neg\exists i(A(x[i])). \qquad (4)$$

Правило вывода (левая нижняя формула в правиле ФП опущена) для формул вида (4):

$$\frac{F}{F_1\quad\ldots\quad F_k\quad G}, \qquad (5)$$

где

$$F_1=\forall j(j=\overline{1,n})\to A_1(x[j])\vee\neg A_1(x[j]));$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \qquad (6)$$
$$F_k=\forall j(j=\overline{1,n})\to A_k(x[j])\vee\neg A_k(x[j]));$$
$$G=\forall j(j=\overline{1,n})\to\exists u\exists v(u=\min(j)\ \&\ v=\max(j))).$$

Правило (5) означает, что вместо формулы F достаточно доказать $k+1$ подцелей. Первые k формул в (6) относятся к проверке разрешимости членов дизъюнкции, которые по определению являются простыми конъюнкциями, а последняя — устанавливает границы цикла.

Правилу (5) ставится в соответствие следующая схема программы для решения исходной задачи:

begin integer n; ***array*** x (1: n); ***input*** (n); ***input*** (x);
for j=1 ***to*** n ***do***
if A_1 ***then do*** *output* (j); ***exit***; ***od else skip***;
…….
if A_k ***then do*** *output* (j); ***exit***; ***od else skip***;
od; ***print*** («такого элемента не существует»); ***end***. (7)

Пусть каждая простая конъюнкция из A имеет вид $A_w = B_{w1} \& ... \& B_{wr}$. Поэтому далее в рамках доказательства каждой подцели

$$F_w = \forall j(j = \overline{1,n}) \to A_w(x[j]) \vee \neg A_w(x[j])), \ (w \text{ от 1 до } k)$$

формируются r подцелей

$$F_{wh} = \forall j(j = \overline{1,n}) \to B_{wh}(x[j]) \vee \neg B_{wh}(x[j])), \ (h \text{ от 1до } r).$$

Пусть все эти подцели доказаны и из доказательства каждой такой подцели извлечена программа P_{wh} [y_{wh}]. В скобках указан только выходной параметр: $y_{wh} = 1$ в случае B_{wh} (x [j]) и $y_{wh} = 0$ в случае $\neg B_{wh}$ (x [j]). И пусть в результате доказательства подцели G получены значения t (n) = min (j) и s (n) = max (j).

Тогда в результате подстановок в (7) получим следующую программу:

begin integer n; ***array*** x (1: n); ***input*** (n); ***input*** (x);
for j= t (n) ***to*** s (n) ***do***
$\mathbf{P}_{11}[y_{11}]$;…, $\mathbf{P}_{1r}[y_{1r}]$; ***if*** y_{11}=1&…& y_{1r}=1 ***then do output*** (j); ***exit***; ***od else skip***;
……
$\mathbf{P}_{w1}[y_{w1}]$;…, $\mathbf{P}_{wr}[y_{wr}]$; ***if*** y_{w1}=1&…& y_{wr}=1 ***then do*** *output* (j); ***exit***; ***od else skip***;
od; ***print*** («такого элемента не существует»); ***end***. (8)

Замечание 5. Если некоторые из B_{wh} (x, j) являются первичными предикатами, то программы P_{wh} [y_{wh}] отсутствуют, а в условии ***if*** программы (8) появятся не $y_{wh} = 1$, а эти самые предикаты. ■

Замечание 6. Более точно в доказательстве подцели G (нахождение границ цикла) должны в явном виде использоваться все термы, входящие в F. Доказательства подцелей с атомными формулами, как правило, получаются применениями правила ИС к определениям предикатов. ■

Чтобы быть уверенными в обоснованности нашего подхода, мы должны ответить на следующие вопросы.

1. Является ли программа (8) правильной?
2. Обеспечивают ли подцели (6) и программа (8) конструктивное доказательство формулы (4), соответствующей исходной задаче?
3. Как определяются границы цикла в полученной программе?

Справедливы следующие теоремы.

Теорема 1. Пусть $\overline{t}(n, x[i]))$ — все термы, входящие в следствие формулы

$$\forall n \forall x(x(1:n) \to \exists i(A(x[i]) \vee \neg \exists i(A(x[i])),$$

j — параметр цикла программы (8).

Тогда min (j) = max (min ($\overline{t}(n, x[i])$)), max (j) = min (max ($\overline{t}(n, x[i])$)). ■

Теорема 2. Пусть дана формула

$$\forall n \forall x(x(1:n) \to \exists i(A(x[i]) \vee \neg \exists i(A(x[i])),$$

где A (x [i]) — простая дизъюнкция. Тогда программа (8) является полностью корректной. ■

Теорема 3. Если конструктивно доказаны подцели (6), то формула
$\forall n \forall x(x(1:n) \to \exists i(A(x[i]) \vee \neg \exists i(A(x[i]))$ — конструктивно истинная. ■

В доказательствах приведенных теорем используются определения и понятия из [12].

Рассмотрим теперь задачу ***нахождения в массиве первого минимального из всех элементов, удовлетворяющих заданному свойству***. Имеем формулу

$$F = \forall n \forall x(x(1:n) \to \exists i(x[i]) = \min(x, A(x[i])). \quad (9)$$

Приведем определение минимального элемента:

$$G = \forall n \forall x(x(1:n) \to \forall i(x[i] = \min(x) \leftrightarrow \forall j(x[i] \le x[j]))).$$

Пусть вначале A — пустая формула. В этом случае (9) доказывается применением правила ФП, соответствующего итерационному определению массива через отрезки и применением определения минимального элемента. Формализация таких доказательств и извлечение из них программ рассматриваются в [8]. Из доказательства извлекается программа

begin integer n; ***array*** x (1: n); ***input*** (n); ***input*** (x);
y:=x [1]; (10)
for j=2 ***to*** n ***do if*** x [j]<y ***then*** y:=x [j] ***od output*** (y, i); ***end***.

Пусть теперь A (x [i]) — простая дизъюнкция. Тогда к формуле (8) применяется правило, аналогичное правилу (5). Изменится лишь схема программы. Условный оператор в (10) формируется в соответствии со схемой программы (7).

Аналогичным образом строятся доказательства и программы для других предикатов и функций с условиями, которые определяются итерационно. Справедливы также соответствующие теоремы обоснования.

Рассмотрим теперь задачи на изменение состояния массива. Ниже мы лишь опишем расширение рассматриваемого формализма и наметим схемы решения таких задач.

Термы и формулы языка L называем **стандартными**.

Для формализации задач на изменение состояния массива используется язык ***LL***, который получается из L следующим образом.

1. Вместо каждой из функций x [i, s], *кол-во* (x), $\Sigma(x)$ вводится соответственно **функция с условием**:

– $x[i, s, F(x, \overline{p})]$ — значение i-ого элемента массива x в состоянии s, удовлетворяющего условию F с параметрами $\overline{p}$;

– *кол-во* (x, F (x, $\overline{p}$)) — количество элементов множества x, удовлетворяющих условию F;

– $\sum$ (x, F (x, $\overline{p}$)) — сумма всех элементов x, удовлетворяющих условию F с параметрами $\overline{p}$.

2. Вместо каждого из предикатов макс (x), *миним.* (x), s = *сост.* (y) вводится соответственно **предикат с условием:**

– *макс*. (x, F (x, $\overline{p}$)) — максимальный элемент множества x, удовлетворяющий условию F;

– *миним.* (x, F (x, $\overline{p}$)) — минимальный элемент множества x, удовлетворяющий условию F;

– $сост.(x, s, F(x, \overline{p}))$ - s есть состояние объекта x, удовлетворяющего условию F с параметрами $\overline{p}$.

В пунктах 1 и 2 условие F является стандартной формулой. Атомными формулами языка ***LL*** являются только атомные формулы языка L. Понятно, что определенные таким образом функции и предикаты есть всего лишь сокращения соответствующих формул. Однако роль таких объектов весьма существенна при определении правил вывода и в построении выводов в системе вывода, используемой в [8].

Исходные предикаты и функции языка ***L*** преобразуются в $\Sigma(x,\Omega)$, $сост.(x,s,\Omega)$ x [i, s,Ω].

Вычислимый терм языка ***LL*** **—** это либо вычислимый стандартный терм, либо терм, которому сопоставлена программа языка ***LP***. **Разрешимая формула** языка ***LL*** **—** это либо разрешимая стандартная формула, либо формула, которой сопоставлена программа языка ***LP***.

При синтезе программ на изменение состояния массива правила вывода системы GN_i переформулируются для языка ***LL***.

Рассмотрим задачу: ***в массиве переставить элемент с одним заданным свойством и элемент с другим заданным свойством.***

$$\forall n \forall x(x(1:n) \to \forall s1(исх\_сост.(x,s) \to \exists s2 пер.(s1,s2,A(x,n),B(x,n)))),$$

$пер.(s1,s2,A(x,n),B(x,n)$ — сокращение для формулы $из(s1,s2) \,\&\, сост.(x,s,F))$,

где

$$F(x, n) = \exists i \exists j (A(x[i, s1] \,\&\, B(x[j, s1]) \,\&\, x[i, s2] = x[j, s1] \,\&\, x[j, s2] = x[i, s1]).$$

Вместо формул *A, B* могут стоять элементы массива с индексами; предикат *из* (*s*1, *s*2) означает, что существует переход из одного состояния в другое.

В данном случае речь идет о перестановке первого найденного элемента. Конечно же, необходимо еще добавить условие, что все остальные элементы остаются на месте.

Подобным образом можно представить задачу — ***заменить элемент с заданным свойством на терм с заданным свойством.***

Рассмотрим теперь задачу ***последовательной перестановки всех элементов массива с заданным свойством со всеми элементами, удовлетворяющими другому заданному свойству.***

Доказываемая формула имеет вид

$$\forall n \forall x (x(1:n) \to \forall s1 (\textit{исх\_сост.}(x, s) \to \exists s2 \textit{все}(s1, s2, A(x, n), B(x, n)))) \; \textit{все}(s1, s2, A(x, n), B(x, n)) \; —$$

сокращение для формулы

$$\exists P (P = \textit{посл.сост.}(x) \,\&\, s1 = P[1] \,\&\, s2 = \textit{конечн.}(P) \,\&$$

$$\forall s3 \forall i (s3 = P[i] \to \exists s4 (s4 = P[i+1] \,\&\, \textit{пер.}(s3, s4, A(x, n), B(x, n)))).$$

Здесь вводится предикат *P = посл. сост.* (*x*) — последовательность состояний массива *x*. Предикат *конечн.* (*P*) определяет последний элемент последовательности состояний.

Анализ задач на изменение состояния массива позволяет отметить следующее. Программа строится из определения требуемой последовательности состояний и заданных свойств. Доказательство же необходимо здесь, во-первых, для проверки разрешимости заданных свойств, а во-вторых, для обоснования того, что определяемая последовательность является конечной и последний ее элемент есть требуемое состояние массива.

Для решения практически интересных задач необходимо действовать по следующей схеме.

1. Описать спецификацию задачи (доказываемая формула).
2. Описать спецификацию алгоритма — это определение последовательности состояний вместе с описанием получения следующего состояния из предыдущего.
3. Построить доказательство того, что полученная последовательность приводит к конечному состоянию, определяемому спецификацией задачи.

По указанной схеме автором рассмотрен синтез программ для решения задач сортировки массива.

В заключение отметим, что исследование даже простых задач показывает: для практического применения дедуктивного синтеза программ необходимо применять специализированные системы вывода. При этом не исключается синтез правил вывода с использованием логики второго порядка.

References

1 *Vasilyev S.N.* Formalization of knowledge and management based on positively-created languages // Information technology and computing systems. — Moscow: Nauka, 2008. — № 1. — P. 3–17.

2 *Stuart J.Russel, Peter Norvig.* Artificial Intelligence. A Modern Approach // Prentice Hall. — 2003. — P. 1672.

3 Mathematical logic in programming // Collection of articles, transl. from English., ed. M.V.Zaharyascheva and Yu.I.Yanov // Moscow: Mir, 1991. — P. 408.

4 *Nepeivoda N.N.* Regarding a method for constructing a correct program out of correct subprograms: Programming. — 1979. — № 1. — P. 11–21.

5 *Manna Z., Waldinger R.* Fundamentals of deductive program synthesis // IEEE Transactions on Software Engineering. — 1995. — 18(8). — P. 674–704.

6 *Nikolenko A.B.* On automation of construction of programs using the predicate logic // Bulletin of automation, Engineering and Technical Magazine. — 2009. — № 3. — P. 34–36.

7 *Nikolenko A.B.* Deductive synthesis of iterative programs // Articles of the III International Scientific and Practical Conference «Informatization of Society». — Astana, 2012. — P. 564–568.

8 *Nikolenko A.B.* Implementation of iterative definitions in deductive synthesis of programs // Articles of International Scientific and Practical Conference, St. Petersburg State University. — Almaty, 2010. — P. 50–55.

9 *Prawitz D.* Natural deduction // A Proof-Theoretical Study. Almqvist&Wilksell. — Uppsala, 1965. — P. 14–16.

10 *Martyanov V.I., Nikolenko A.B.* Logical programming language PIFOR // In the book: Instrumentation systems and modeling. — Novosibirsk: Nauka, 1988. — P. 27–32.

11 *Nepeivoda N.N.* Stable truth and computability // Collection of articles: Studies in the theory of algorithms and mathematical logic. — Moscow: Nauka, 1979. — P. 78–89.

12 *Gris D.* Science of programming. — Moscow: Mir, 1983 — P. 234–257.

А.Б.Николенко

Талдап жинақталған нағыз қорытындының жүйесіндегі итерационды есептеулерді модельдеу

Мақалада итерационды есептеулерді нысандандыру мәселелері қарастырылған. Мазмұнды математикалық пайымдаудың модельдеуіне арналған бірінші ретті жүйе құрылған. Бұл жүйедегі логикалық тұжырымдардың құрылу үрдісі, кейіннен олардан шығарылатын, бүтін сандар массивін өңдеу есептерін шешу үшін арналған итерационды бағдарламалар сипатталған. Шығарылған бағдарламалардың тұжырымдарының конструктивтілігі мен дұрыстығы туралы теоремалар келтірілген.

A.B.Nikolenko

Modeling of iterative calculations in the system of General natural deduction

This work is devoted to the formalization of iterative calculations. A first-order system intended to simulate content-rich mathematical reasoning is constructed. The chaining process of logical deductions in this system is described, so that to use these conclusions to derive iterative programs to solve problems of integer numbers array processing. Theorems on deductions constructability and correctness of the derived programs are shown.