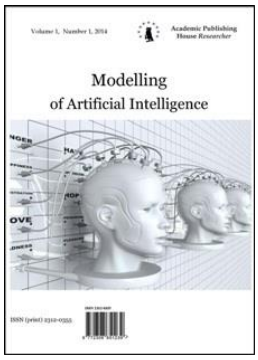
Published in the Russian Federation
Modeling of Artificial Intelligence
Has been issued since 2014.
ISSN: 2312-0355
E-ISSN: 2413-7200
Vol. 10, Is. 2, pp. 104-116, 2016

DOI: 10.13187/mai.2016.10.104
www.ejournal11.com



UDC 004

Intelligent Planning and Simulation of Iterative Calculations

Aleksei B. Nikolenko [a, *]

[a] Almaty branch of St. Petersburg Humanitarian University of labor unions, Republic of Kazakhstan

Abstract

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 some classes of problems of integer numbers array processing. Theorems on deductions constructability and correctness of the derived programs are shown.

Keywords: intelligent planning, formalization of tasks, iterative data structures, program synthesis, systems of logical inference.

1. Введение

Под системами интеллектного управления понимаются системы автоматического управления на базе тех или иных средств искусственного интеллекта (Васильев, 2008). Интеллектное планирование, рассматриваемое как частный случай интеллектного управления, тесно связано с автоматическим доказательством теорем и дедуктивным синтезом программ (Математическая логика в программировании, 1991; Manna, Waldinger, 1995). Следует отметить, что упор на применение чисто синтаксических методов привел к отсутствию существенного прогресса в этой области как в теории, так и на практике. В частности, это касается и языков логического программирования (Мартьянов, Николенко, 1988; Математическая логика в программировании, 1991; Непейвода, 2000). В значительной степени такое относительное «топтание на месте» вызвано тем, что основной используемый здесь теоретический метод – метод резолюций, разработан более пятидесяти лет назад (Robinson, 1965).

Поэтому, несомненно, продолжает оставаться актуальной задача модификации существующих логических инструментов (равно как и построения новых методов) автоматического планирования в соответствии с логикой развития области обработки и представления знаний.

Задачи представления знаний в самых различных предметных областях в последние годы решаются многими исследователями. При этом нередко молчаливо обходятся проблемы, связанные с формализацией содержательных разделов математики и программирования. Почти сорокалетний опыт автора позволяет говорить о том, что при решении задач автоматического доказательства теорем и дедуктивного синтеза программ

* Corresponding author
E-mail addresses: abnikolenko@gmail.com (A.B. Nikolenko)

определенное значение имеют изучение и формализация тех математических и программистских приемов, которые используются на практике и которые влияют как на структуру логических выводов, так и на тактики их построения. Одним из таких направлений является формализация итерационных вычислений в рамках дедуктивного синтеза программ.

Известно, что в дедуктивном синтезе программ (Непейвода, 1979; Николенко, 2009; Manna, Waldinger, 1995), как правило, применяется следующая стандартная схема: исходная задача формулируется в виде формулы логики первого порядка, затем строится ее конструктивное доказательство в системе логического вывода, после чего из доказательства извлекается программа, решающая исходную задачу. В этой схеме задача представляется в виде формулы

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

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 рекурсивная) функционального языка программирования.

В то же время моделирование итерационных вычислений в системах автоматизации планирования и дедуктивного синтеза программ сводится, как правило, к попыткам «встроить» математическую индукцию в используемую систему логического вывода. Отметим в этой связи следующие обстоятельства.

Во-первых, упомянутые выше методы поиска вывода являются чисто синтаксическими и не позволяют учесть все детали выбора структур данных для программной реализации исходных объектов из (*). Так, например, при автоматизации построения программ для решения задач линейной алгебры возникает вопрос: когда в логическом выводе переходить от объекта «матрица» к объекту «двумерный массив»?

Во-вторых, преобразование формулы

$$\forall x(A(x) \to \forall y(y \in x \to B(x, y)))$$

в формулу

$$\forall x \forall y(A(x) \,\&\, y \in x \to B(x, y))$$

при ближайшем рассмотрении оказывается итерационным действием - y пробегает все элементы x.

Далее, даже для простых формул вида (*) при реализации схемы синтеза оказывается, что кванторы всеобщности, вообще говоря, не перестановочны. И, наконец, существует проблема появления в выводе переменных, которых нет в программе, а с другой стороны – в программе могут понадобиться переменные, которых нет в выводе.

Все вышесказанное, а также итерационная природа основных структур данных требуют, на наш взгляд, построения специализированных систем вывода, которые должны обладать рядом свойств, относящихся к изучаемой предметной области. Выявление таких свойств возможно при анализе классов задач, связанных с обработкой итерационных структур данных (массивов, строк, файлов и т.п.) и соответствующих этим задачам формул, логических выводов и программ.

Нам представляется, что система вывода, основанная на использовании позитивно-образованных языков (Васильев, 2008) может иметь такие свойства. Причем на первом этапе необходимо описать с точки зрения синтеза программ процесс перевода обычной формулы первого порядка в позитивно-образованную формулу.

Настоящая работа посвящена вопросам моделирования итерационных вычислений в системах первого порядка (Николенко, 2013a; Николенко, 2013b). Вводится понятие итерационного вывода в системе натурального типа (Николенко, 2015), которое позволяет обосновать дедуктивный синтез итерационных программ обработки упорядоченных структур данных (в работе приведены примеры для массивов целых чисел) в системе обобщенного естественного вывода $\boldsymbol{GN_I}$. Выделены классы формул, для которых описан

процесс механического построения итерационных выводов. Приводятся теоремы о конструктивности выводов и правильности извлекаемых программ.

В работе строится система обобщенного естественного вывода $\boldsymbol{GN_1}$, которая содержит два крупноблочных правила вывода, которые позволяют моделировать прямые и обратные рассуждения, являются обобщениями правил традиционной системы натурального вывода и допускают расширение до логики второго порядка. Для упрощения изложения рассматриваются задачи обработки одномерных массивов целых чисел. Результаты работы могут быть перенесены на другие структуры данных: многомерные массивы, строки, файлы.

В заключение приводятся соображения по структуре интерактивной системы синтеза программ и пользовательскому интерфейсу.

2. Обсуждение

Предварительные сведения

Приведем некоторые обозначения и определения.

Термы и формулы используемого языка первого порядка $\boldsymbol{L}$ определяются обычным образом. Символ Ω используется для обозначения пустой формулы. В определение формулы языка $\boldsymbol{L}$ 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, $F(\bar{x})$, $G(x_1, x_2, \ldots, x_n)$. 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 $Q_1x_1\ldots Q_kx_k$, где каждое Q_i есть символ $\forall$ либо $\exists$. Формулу с кванторной приставкой Q ($Q_1x_1\ldots Q_kx_k$,) будем называть *Q-формулой* (*$Q_1\ldots Q_k$-формулой*).

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

Запись $A = B$ означает графическое совпадение формул.

Определение 1. Формула A *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 $A = A_1\&A_2\&\ldots\&A_k$, $B = B_1\&B_2\ldots\&B_m$, тогда каждая B_j совпадает с некоторой A_i (не исключается случай $k, m = 1$);

4) если $A = A_1 \vee A_2 \vee \ldots \vee A_k$, то каждая A_i содержит B;

если $B = B_1 \vee B_2 \ldots \vee B_m$, то A содержит хотя бы одну B_j;

5) если $B = \exists x\, B_1(x)$, то либо найдется такой терм t из A, что $B_1(t)$ содержится в A, либо $A = \forall x B_1(x)$;

6) если $B = \forall x\, B_1(x)$, то $A = \forall y A_1(y)$ и B_1 содержится в A_1.

Определение 2. Формула F языка $\boldsymbol{L}$ 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) $F = F_1 \& \dots \& F_k$, 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 $\boldsymbol{GN_I}$, которые будем называть *правилами обобщенного натурального вывода*. Для наших целей достаточно ограничиться применением правил вывода к $\forall$ -формулам. Символ Ω позволяет любую формулу $Q\,\overline{x}\,A$ представить формально в виде импликации $Q\overline{x}(\Omega \to A)$, что обеспечивает применение правил вывода к любым формулам (при этом считается, что переменные кортежа $\overline{x}$ входят в Ω).

Система $\boldsymbol{GN_I}$ содержит два правила:

1. **Правило прямого вывода (ППВ)**

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

и при этом должно 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. **Правило обратного вывода (ПОВ)**

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

при этом должно 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 вспомогательный вывод формулы $\forall\overline{v}(F(\overline{v}) \to \exists\overline{w}G(\overline{v},\overline{w}))$ из формулы $\forall\overline{u}(C(\overline{u}) \to D(\overline{u})))$; тогда применение ПОВ будет иметь вид

$$[\ \forall\overline{u}(C(\overline{u}) \to D(\overline{u})))\ ]$$

$$\vdots$$

$$\forall\overline{v}(F(\overline{v}) \to \exists\overline{w}G(\overline{v},\overline{w}))$$

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

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, корнем которого (самая левая вверху) считается доказываемая формула; все листья дерева - канонические формулы.

Замечание 2. Нам будет достаточно приведенных формулировок правил вывода. 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; вместе с тем, для целей синтеза программ мы должны конструктивно ограничивать применение указанных правил. Автором доказана полнота и непротиворечивость подобной системы в 1984 году (Николенко, 1984).

Формализация предметной области и итерационные выводы

Одномерный массив рассматривается как совокупность независимых переменных. Фрагмент языка первого порядка для формализации предметной области определяется так, что большинство используемых функций и предикатов имеют общеупотребительное значение.

Для определения понятия итерационного вывода рассмотрим задачу *поиска в массиве элементов с заданным свойством*:

«**в массиве целых чисел найти все элементы, большие нуля**» (**)

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.
```

Пусть теперь k – конкретное число (произвольная константа). Следующая программа не содержит цикл и является решением задачи (**), когда массив имеет размерность k:

```
begin array x(1:k); input(x);
        if x[1]>0 then do output(1) ) od;
    if x[2]>0 then do output(2) ) od;                        (2)
        ..............................
        if x[k]>0 then do output(k) ) od;  end.
```

Сформулируем задачу (**) в виде следующей формулы первого порядка:

$$\forall n \forall x(x(1:n) \rightarrow \exists i(x[i] > 0) \vee \neg \exists i(x[i] > 0)) \qquad (3)$$

Для построения натурального вывода формулы (3) нам понадобятся следующие аксиомы и определения.

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

А2. $\forall x \forall s(исх\_сост(x,s) \rightarrow x = \{x[1,s], x[2,s],..., x[k,s]\})$ - состояние массива определяется значениями его элементов, k – произвольная константа.

А3. $\forall n \forall x \forall s \forall i(x(1:n) \,\&\, сост(x,s) \,\&\, i \in <1,n> \rightarrow цел(x[i,s])$ - определение массива целых чисел ($i \in <1,n>$ - предикат «i изменяется от 1 до n»).

Для случая, когда размерность массива является конкретной константой k, имеем k соответствующих аксиом.

А3'. $\forall x \forall s(x = \{x[1,s], x[2,s],..., x[k,s]\} \rightarrow цел(x[1,s]))$,

$\forall x \forall s(x = \{x[1,s], x[2,s],..., x[k,s]\} \rightarrow цел(x[2,s]))$,

.....

$\forall x \forall s(x = \{x[1,s], x[2,s],..., x[k,s]\} \rightarrow цел(x[k,s]))$ - определение массива целых чисел.

А4. $\forall y(цел(y) \rightarrow y > 0 \vee \neg(y > 0))$ - линейность порядка на целых числах.

A5. $\forall x \forall s(\mathit{исх\_состм}(x,s) \to \mathit{состм}(x,s))$ - исходное состояние массива является состоянием.

Построим натуральный вывод (Prawitz, 1965) для решения задачи (**) для случая n=2. В данном случае аксиомы A2 и A3' имеют вид:

$$\forall x \forall s(\mathit{исх\_состм}(x,s) \to x = \{x[1,s], x[2,s]\}),$$
$$\forall x \forall s(x = \{x[1,s], x[2,s]\} \to \mathit{цел}(x[1,s])),$$
$$\forall x \forall s(x = \{x[1,s], x[2,s]\} \to \mathit{цел}(x[2,s])).$$

Введем обозначения для формул, встречающихся в выводе (рисунок 1):

$$\alpha_1 = x[1,s] > 0 \,\&\, x[2,s] > 0,\ \alpha_2 = x[1,s] > 0 \,\&\, \neg(x[2,s] > 0),$$
$$\alpha_3 = \neg(x[1,s] > 0) \,\&\, x[2,s] > 0,\ \alpha_4 = \neg(x[1,s] > 0) \,\&\, \neg(x[2,s] > 0).$$

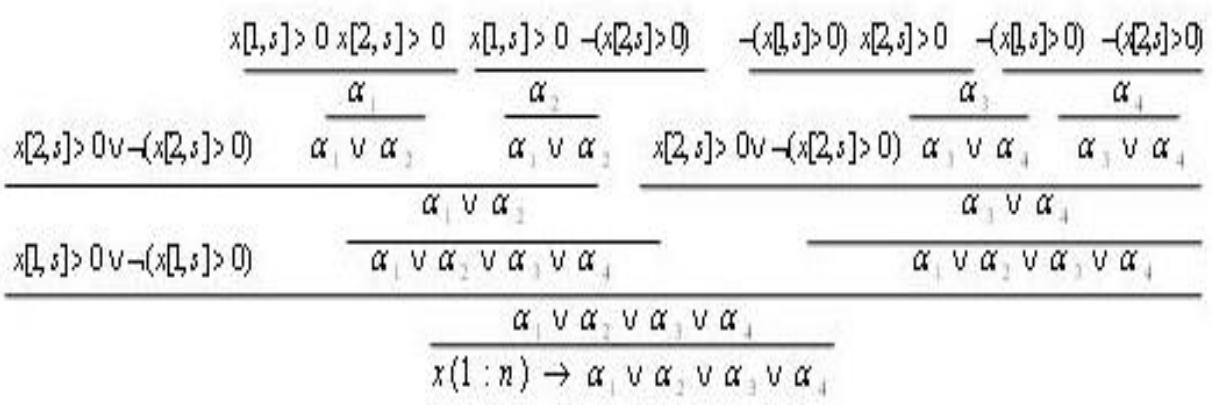


Рис. 1. *Вывод формулы (3) для случая k =2*

В доказанной формуле $\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4$ первые три члена дизъюнкции соответствуют случаю, когда искомый элемент существует, а последний - отрицанию из (3).

Из-за недостатка места вывод формулы $x[1,s] > 0 \vee \neg(x[1,s] > 0)$ на рисунке 1 приводится отдельно (рисунок 2). Правила снятия квантора всеобщности опущены ввиду очевидности их применения. Вывод формулы $x[2,s] > 0 \vee \neg(x[2,s] > 0)$ строится аналогично.

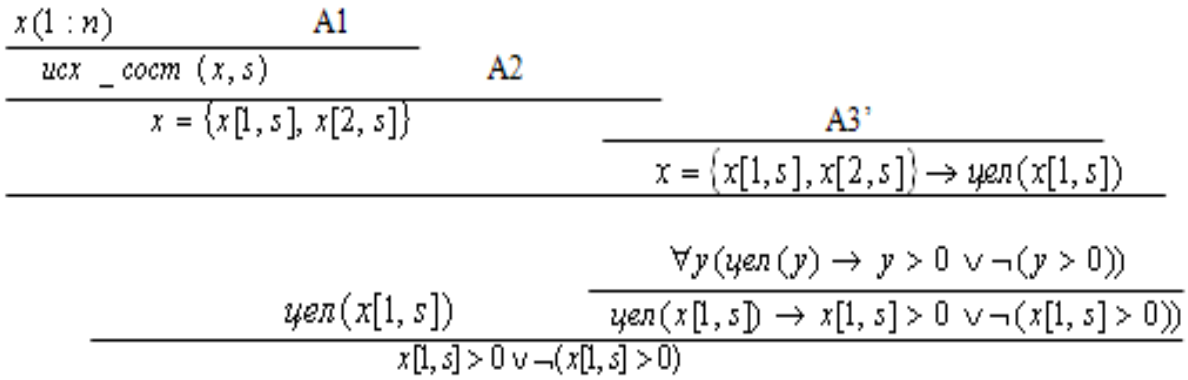


Рис. 2. *Вывод формулы* $x[1,s] > 0 \vee \neg(x[1,s] > 0)$

Построенному выводу поставим в соответствие программу:

```
begin  array x(1:2); input(x);
       if x[1]>0 then do output(1) ) od;                    (4)
       if x[2]>0 then do output(2) ) od;   end.
```

Аналогичные выводы можно построить для каждого произвольного k. Каждому такому выводу будет соответствовать программа вида (2). Программа с циклом (1) является обобщением всех программ вида (2).

Вывод в системе $\boldsymbol{GN_I}$ формулы (3) выглядит следующим образом (через A обозначено следствие формулы (3)):

$$\frac{\forall n \forall x(x(1:n) \rightarrow A(x,i))}{\begin{array}{ll}\alpha & \forall i(i \in <1,n> \rightarrow x[i,s] > 0 \vee \neg(x[i,s] > 0))\end{array}} \qquad (5)$$
$$\sum$$

Здесь:

а) формула под чертой справа – подцель, которую нужно доказывать;

б) $\alpha = \forall n \forall x \forall j(x(1:n) \& A(x,j) \rightarrow A(x,i))$ – измененная доказываемая формула с учетом того, что подцель уже доказана; α – тождественно истинная и доказывать ее не надо;

в) фигура $\sum$ является выводом подцели и содержит несколько применений правила ППВ с аксиомами А1, А5, А3, А4.

Вывод (5) является обобщением всех натуральных выводов для конкретных размерностей массива. Вывод (5) называем *итерационным выводом* для решения задачи (**). Из (5) *извлекается программа* (1). Таким образом, совокупность натуральных выводов для k = 1,2,... является теоретическим обоснованием для итерационного вывода. Основное преимущество выводов в системе $\boldsymbol{GN_I}$ перед натуральными выводами – крупноблочность применяемых правил. Кроме того, вывод (5) в точности соответствует содержательному алгоритму: «последовательно от 1 до n проверить каждый элемент на указанное свойство».

Задача поиска элемента с заданным свойством

Сформулируем задачу нахождения элементов массива с заданным свойством в более общем виде.

Пусть $A(x[i], A_1, \ldots, A_k, \bar{t}(x,i))$ - простая дизъюнкция, в которой:

x – переменная для массива; $A_1,\ldots, A_k$ – простые бескванторные конъюнкции, входящие в A; $\bar{t}(x,i)) = (t_1(x,i),\ldots, t_l(x,i))$ – все термы, входящие в A.

Формула, соответствующая исходной задаче, имеет вид:

$$F = \forall n \forall x(x(1:n) \rightarrow \exists i A(x[i]) \vee \neg \exists i A(x[i])) \qquad (6)$$

Правило вывода (левая нижняя формула в правиле ПОВ опущена) для таких формул:

$$\frac{F}{F_1 \quad \ldots \quad F_k \quad G}, \qquad (7)$$

где

$$\begin{aligned} F_1 &= \forall j(j = \overline{1,n} \rightarrow A_1(x,j) \vee \neg A_1(x,j)), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ F_k &= \forall j(j = \overline{1,n} \rightarrow A_k(x,j) \vee \neg A_k(x,j)), \\ G &= \forall j(j = \overline{1,n} \rightarrow \exists u \exists v(u = \min(j) \& v = \max(j))). \end{aligned} \qquad (8)$$

Правило (7) означает, что вместо формулы F достаточно доказать k+1 подцелей. Первые k формул в (8) относятся к проверке разрешимости членов дизъюнкции, которые по определению являются простыми конъюнкциями, а последняя устанавливает границы цикла. Правилу (7) 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 A1 then do output(j); exit; od else skip;
                    ..................................................
                    if Ak then do output(j);exit; od else skip;
                od; print(«такого элемента не существует»); end.
```
(9)

Пусть каждая простая конъюнкция из A имеет вид $A_w = B_{w1} \& \ldots \& B_{wr}$. 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 $\mathbf{P}_{wh}[y_{wh}]$. (В квадратных скобках указан только выходной параметр. При этом 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 (9) 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***;
..... (10)
$\mathbf{P}_{w1}[y_{w1}]$;..., $\mathbf{P}_{wr}[y_{wr}]$; ***if*** y_{w1}=1&...& y_{wr}=1 ***then do***
output(j);***exit***; ***od else skip***;
od; ***print***(«такого элемента не существует»); ***end.***

Если некоторые из $B_{wh}(x, j)$ являются отношениями из целевого языка программирования, то программы $\mathbf{P}_{wh}[y_{wh}]$ отсутствуют, а в условии ***if*** программы (10) появятся не y_{wh}=1, а эти самые отношения. В доказательстве подцели G (нахождение границ цикла) должны в явном виде использоваться все термы, входящие в F. Доказательства подцелей с атомными формулами получаются применениями правила ППВ к определениям предикатов.

Справедливы следующие теоремы.

Теорема 1. Пусть $\bar{t}(x, n, i)$ – все термы, входящие в следствие формулы

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

j – параметр цикла программы (10).

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

Теорема 2. Пусть дана формула

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

$A(x[i])$ – простая дизъюнкция. Тогда программа (10) является полностью корректной (Грис, 1983).

Теорема 3. Если конструктивно доказаны подцели (8), то формула

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

- конструктивно истинная.

Расширение формализма

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), $\sum(x)$ вводится соответственно *функция с условием*:

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

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

- $\sum(x, F(x, \bar{p}))$ - сумма всех элементов x, удовлетворяющих условию F с параметрами $\bar{p}$.

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

- *макс*(x, $F(x,\bar{p})$) - максимальный элемент множества x, удовлетворяющий условию F;
- *мин*(x, $F(x,\bar{p})$) - минимальный элемент множества x, удовлетворяющий условию F;

- *сост(x, s, F(x, $\bar{p}$))* - *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 используемой системе вывода. Исходные предикаты и функции языка ***L*** преобразуются в $\Sigma(x, \Omega)$, *сост*(*x*, *s*, Ω), *x*[*i*, *s*, Ω].

Вычислимый (в рамках данного формализма**)** *терм* языка ***LL*** – это либо вычислимый стандартный терм, либо терм, которому сопоставлена программа языка ***LP***. *Разрешимая* (в пределах данного формализма) *формула* языка ***LL*** – это либо разрешимая стандартная формула, либо формула, которой сопоставлена программа целевого языка.

При синтезе программ на изменение состояния массива правила вывода системы $\boldsymbol{GN_1}$ переформулируются для языка ***LL***.

Изменение состояния массива

Пусть $n \geq 1$, а $p_1, p_2, ..., p_k$ - одноместные предикаты такие, что: $p_i \leftrightarrow \neg p_1 \,\&\, ... \,\&\, \neg p_{i-1} \,\&\, \neg p_{i+1} \,\&\, ... \,\&\, \neg p_k$ для каждого *i* от 1 до *k*. Это условие означает, что предикаты $p_1, p_2, ..., p_k$ являются взаимоисключающими.

И пусть формула $\forall a(цел(a) \to p_1(a) \vee ... \vee p_n(a))$ истинна, а $t_1, t_2, ..., t_k$ - произвольные термы.

Требуется построить логический вывод, из которого можно извлечь программу решения следующей задачи: «*В массиве целых чисел* $x(1:n)$ *заменить все элементы, удовлетворяющие свойству* p_1 *- на терм* t_1 *, удовлетворяющие свойству* p_2 *- на терм* t_2 *, ... удовлетворяющие свойству* p_k *- на терм* t_k *».*

Сформулируем задачу в языке логики предикатов.

$$\Phi = \forall n \forall x \forall s_1 (x(1:n) \,\&\, исх\_сост(x, s_1) \to \exists s_2 (сост(x, s_2, P))), \qquad (11)$$

где $P = P(n, x, s_1, s_2, \bar{p}, \bar{t}) \Leftrightarrow$

$$\forall i (i = (1:n) \to \overset{k}{\underset{j=1}{\&}} (p_j(x[i, s_1] \to x[i, s_2] = t_j(\bar{u}))),$$

символ $\Leftrightarrow$ означает «по определению», $\bar{p} =< p_1, p_2, ..., p_k >$, $\bar{t} =< t_1, t_2, ..., t_k >$, $\bar{u}$ - кортеж всех свободных переменных, входящих в $\bar{p}$ и $\bar{t}$. Кроме того, $исх\_сост(x, s_1)$ означает, что s_1 есть начальное состояние массива, а $сост(x, s_2, P)$ означает, что s_2 есть состояние массива *x*, удовлетворяющее условию P .

Если в формуле (11) все p_i разрешимые (относительно языка программирования), а все t_i - вычислимые, то программа, решающая задачу (11) имеет вид:

```
begin integer n; array x(1:n); input(n); input(x);
    for j=1 to n do
        if p1(x[i]) then x[i2] = t1;                    (12)
        .....................................
        if pk(x[i]) then x[i2] = tk; od;
            output(x); end.
```

Приведем теперь схему построения логического вывода формулы Φ, по которому будет строиться программа (12). На первом этапе строится последовательность *SS* состояний

массива *x*. Далее доказывается, что *SS* не является бесконечной и показывается, что последнее состояние этой последовательности и будет искомым, т.е. удовлетворяет условию *P*.

Для данного класса задач последовательность состояний описывается формулой:

$$\forall n \forall x \forall s_0 (x(1:n) \,\&\, сост(x, s_0) \to \exists ss(посл\_сост(x, ss, F))) \qquad (13)$$

где $F = F(n, x, s_0, ss, \bar{p}, \bar{t}) \Leftrightarrow$

$$ss(0) = s_0 \;\&\; \forall i (i = (1:n) \to \underset{j=1}{\overset{k}{\&}} (p_j(x[i, s_1] \to x[i, s_2] = t_j(\bar{u})))$$

В рассматриваемом случае количество состояний в последовательности *SS* изначально определяется числом *n* и совпадает с количеством итераций в программе (12). Поэтому существование последнего элемента доказывать не надо. Это исключительный случай.

Теперь докажем, что последнее состояние в последовательности, определяемой формулой (13), и является искомым. Применим к Φ правило ППВ:

$$\frac{\Phi \qquad \forall n \forall x \forall s (x(1:n) \,\&\, исх\_сост(x, s) \to сост(x, s))}{\Phi_1 = \forall n \forall x \forall s_1 (x(1:n) \,\&\, исх\_сост(x, s_1) \,\&\, сост(x, s_1) \to \exists s_2 (сост(x, s_2, P)))}$$

Формула над чертой справа есть аксиома, означающая, что любое исходное состояние массива является его состоянием. Следующий шаг вывода – применение к формуле Φ_1 правила ППВ для формулы (13):

$$\frac{\Phi_1 \quad (13)}{\Phi_2}$$

где $\Phi_2 = \forall n \forall x \forall s_1 \forall ss (x(1:n) \,\&\, исх\_сост(x, s_1) \,\&\, сост(x, s_1) \,\&$

$\&\, посл\_сост(x, ss, F) \to \exists s_2 (сост(x, s_2, P)))$. (14)

Подцель для доказательства Φ_2 (применение правила ПОВ) имеет вид:

$$\Phi_2 \mapsto (\Omega \to \exists s (кон\_сост(ss, s)) \,\&\, \forall s (кон\_сост(ss, s) \to сост(x, s, P))), \qquad (15)$$

где символ $\mapsto$ означает «для того, чтобы доказать …, достаточно доказать …». Содержательно в (15) сказано, что вместо того, чтобы доказывать формулу слева, докажем, что в построенной последовательности существует конечное состояние (первая подцель), и это конечное состояние удовлетворяет исходному условию P (вторая подцель). В нашем случае необходимо доказать только вторую подцель, т.е.

$$\forall s (кон\_сост(ss, s) \to сост(x, s, P))$$

Для этого применим правило ПОВ к выражению:

$$\forall s (\Omega \to сост(x, s, P)) \mapsto \forall m (m = (1, n-1) \,\&\, P(m, омр(x, m), s_1, ss(m), \bar{p}, \bar{t}) \to$$

$$\to P(m+1, омр(x, m+1), s_1, ss(m+1), \bar{p}, \bar{t})).$$

Далее необходимо *k* раз применить внутренним образом правило ПОВ. Каждая подцель будет строиться по определению соответствующего предиката из $p_1, p_2, \ldots, p_k$.

Теоремы 1-3 раздела 2 справедливы и для изложенной схемы.

Отметим, что можно усложнить вид формулы (11), вводя в рассмотрение вместо предикатов $p_1, p_2, \ldots, p_k$ простые дизъюнкции. В этом случае теоремы 1, 2 будут также справедливы. С другой стороны, можно отказаться от ограничений на предикаты $p_1, p_2, \ldots, p_k$. В этом случае несколько усложнится процесс построения вывода, но все результаты останутся справедливы.

Рассмотрим теперь задачу «*в массиве переставить элемент с одним заданным свойством и элемент с другим заданным свойством*».

Формула, соответствующая этой задаче, имеет вид:

$$\forall n \forall x(x(1:n) \to \forall s1(ucx\_cocm(x,s) \to \exists s2 nep(s1,s2,A(x,n),B(x,n)))),$$

пер(s1, s2 , A(x, n), B(x, n)) - сокращение для *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 задачу *последовательной перестановки всех элементов массива с заданным свойством со всеми элементами, удовлетворяющими другому заданному свойству***.**

При доказательстве формулы, соответствующей описанию указанной задачи вводятся предикаты:

а) *P = посл. сост*(*x*) - последовательность состояний массива *x*;

б) *конечн*(*P*) - определяет последний элемент последовательности состояний.

3. Заключение

Таким образом, описана система обощенного естественного вывода, предназначенная для моделирования математических рассуждений. Рассмотрен процесс построения логических выводов, из которых извлекаются итерационные программы (в процедурном языке программирования). Предложена формализация задач обработки массивов целых чисел: поиск элемента с заданным свойством и изменение состояния массива. Структура логических выводов при этом такова, что из первой части извлекается программа, решающая исходную задачу, а вторая часть содержит доказательство обоснованности извлекаемой программы. Приведены теоремы обоснования метода о конструктивности выводов и правильности извлекаемых программ.

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.

На основе данного подхода разрабатывается интерактивная система синтеза программ. Для обеспечения дружественного интерфейса на вход системы подаются не формулы логики предикатов, а спецификации задач (Николенко, 2009). Спецификация задачи (на ограниченном русском языке) состоит из двух частей. В первой части описываются входные параметры и условия, которым они удовлетворяют. Эта часть спецификации начинается ключевым словом **Дано**. Вторая часть является «заданием» на построение программы. В этой части ключевым является слово, отражающее действие, которое необходимо выполнить для нахождения решения задачи. За ключевым словом следуют условия, налагаемые на выходные параметры и их связи с входами задачи. Ключевые слова – действия – вводятся посредством определений через формулы и уже определенные действия. Основные действия над массивами: *Найти*, *Найти все*, *Проверить*, *Заменить*, *Переставить*, *Отсортировать* и т.д. Спецификации автоматически переводятся в формулы логики для последующего построения вывода и извлечения программы.

Литература

Васильев, 2008 - *Васильев С.Н.* (2008). Формализация знаний и управление на основе позитивно-образованных языков. // *Информационные технологии и вычислительные системы*, №1. С. 3-17.

Грис, 1983 - *Грис Д.* (1983). Наука программирования. М.: Мир, 416 с.

Мартьянов, Николенко, 1988 - *Мартьянов В.И., Николенко А.Б.* (1988). Язык логического программирования ПИФОР. // *Инструментальные системы и моделирование*. Новосибирск. С. 27-32.

Математическая логика в программировании, 1991 - *Математическая логика в программировании* (1991). Сборник статей. М.: Мир, С. 408.

Непейвода, 1979 - *Непейвода Н.Н.* (1979). Об одном методе построения правильной программы из правильных подпрограмм. // *Программирование*. №1. С. 11-21.

Непейвода, 2000 - Непейвода Н.Н. (2000). Прикладная логика. Новосибирск: НГУ, 491 с.

Николенко, 1984 - *Николенко А.Б.* (1984). Метод инвариантных преобразований и логический вывод. // *Математические заметки*, т.36, вып.1. 1984. с. 3–15.

Николенко, 2009 - *Николенко А.Б.* (2009). Об автоматизации построения программ с использованием логики предикатов. // *Вестник автоматизации, инж.-техн. журнал*, Алматы. №3. с. 34-36.

Николенко, 2013a - *Николенко А.Б.* (2013). Моделирование итерационных вычислений в системе обобщенного естественного вывода. // *Вестник Карагандинского государственного университета. Серия Математика*. Караганда: Изд-во Карагандинского государственного университета. №1. С. 65-73.

Николенко, 2013b - *Николенко А.Б.* (2013). Автоматизация построения итерационных программ // Материалы 6-й Всероссийской мультиконференции по проблемам управления. Ростов-на-Дону: Изд-во Южного федерального университета, т.1. С. 143-146.

Николенко, 2015 - *Николенко А.Б.* (2015). О построении итерационных выводов в системе натурального типа // Отечественная наука в эпоху изменений: постулаты прошлого и теории нового времени. X Международная научно-практическая конференция Ежемесячный научный журнал, ч.5. Екатеринбург. №5. С. 33-37.

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

Prawitz, 1965 - *Prawitz D.* (1965). Natural deduction. A Proof-Theoretical Study. Almqvist&Wilksell, Uppsala, 124 p.

Robinson, 1965 - *Robinson J.A.* (1965). A machine-oriented logic based on the resolution principle. J. A. C. M., 12, pp. 23-41.

References

Vasil'ev, 2008 - *Vasil'ev S.N.* (2008). Formalizatsiya znanii i upravlenie na osnove pozitivno-obrazovannykh yazykov. // Informatsionnye tekhnologii i vychislitel'nye sistemy, №1. S. 3-17.

Gris, 1983 - *Gris D.* (1983). Nauka programmirovaniya. M.: Mir, 416 s.

Mart'yanov, Nikolenko, 1988 - *Mart'yanov V.I., Nikolenko A.B.* (1988). Yazyk logicheskogo programmirovaniya PIFOR. // Instrumental'nye sistemy i modelirovanie. Novosibirsk. S. 27-32.

Matematicheskaya logika v programmirovanii, 1991 - Matematicheskaya logika v programmirovanii (1991). Sbornik statei. M.: Mir, S. 408.

Nepeivoda, 1979 - *Nepeivoda N.N.* (1979). Ob odnom metode postroeniya pravil'noi programmy iz pravil'nykh podprogramm. // Programmirovanie. №1. S. 11-21.

Nepeivoda, 2000 - *Nepeivoda N.N.* (2000). Prikladnaya logika. Novosibirsk: NGU, 491 s.

Nikolenko, 1984 - *Nikolenko A.B.* (1984). Metod invariantnykh preobrazovanii i logicheskii vyvod. // Matematicheskie zametki, t.36, vyp.1. 1984. s. 3–15.

Nikolenko, 2009 - *Nikolenko A.B.* (2009). Ob avtomatizatsii postroeniya programm s ispol'zovaniem logiki predikatov. // Vestnik avtomatizatsii, inzh.-tekhn. zhurnal, Almaty. №3. s. 34-36.

Nikolenko, 2013a - *Nikolenko A.B.* (2013). Modelirovanie iteratsionnykh vychislenii v sisteme obobshchennogo estestvennogo vyvoda. // Vestnik Karagandinskogo gosudarstvennogo

universiteta. Seriya Matematika. Karaganda: Izd-vo Karagandinskogo gosudarstvennogo universiteta. №1. S. 65-73.

Nikolenko, 2013b - *Nikolenko A.B.* (2013). Avtomatizatsiya postroeniya iteratsionnykh programm // Materialy 6-i Vserossiiskoi mul'tikonferentsii po problemam upravleniya. Rostov-na-Donu: Izd-vo Yuzhnogo federal'nogo universiteta, t.1. S. 143-146.

Nikolenko, 2015 - *Nikolenko A.B.* (2015). O postroenii iteratsionnykh vyvodov v sisteme natural'nogo tipa // Otechestvennaya nauka v epokhu izmenenii: postulaty proshlogo i teorii novogo vremeni. X Mezhdunarodnaya nauchno-prakticheskaya konferentsiya Ezhemesyachnyi nauchnyi zhurnal, ch.5. Ekaterinburg. №5. S. 33-37.

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

Prawitz, 1965 - *Prawitz D.* (1965). Natural deduction. A Proof-Theoretical Study. Almqvist&Wilksell, Uppsala, 124 p.

Robinson, 1965 - *Robinson J.A.* (1965). A machine-oriented logic based on the resolution principle. J. A. S. M., 12, pr. 23-41.

УДК 004

Интеллектное планирование и моделирование итерационных вычислений

Алексей Борисович Николенко [a, *]

[a] Алматинский филиал Санкт-Петербургского Гуманитарного университета профсоюзов, Республика Казахстан

Аннотация. Работа посвящена вопросам формализации итерационных вычислений. Строится 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 некоторых классов задач обработки массивов целых чисел. Приводятся теоремы о конструктивности выводов и правильности извлекаемых программ.

Ключевые слова: интеллектное планирование, формализация задач, итерационные структуры данных, синтез программ, системы логического вывода.

* Корреспондирующий автор
Адреса электронной почты: abnikolenko@gmail.com (А.Б. Николенко)