УДК 519.95

ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ В ЛОГИЧЕСКОМ ПРОГРАММИРОВАНИИ

О.П. Бондарь

Розглянуто взаємозв'язок деяких елементів математичної логіки з логічним програмуванням.

We consider the connection between elements of mathematical logic and elements of logical programming.

Как известно, логическое программирование широко используется в:

а) запросно-ответных системах, исследованиях по искусственному интеллекту, синтезе программ, доказательстве теорем, формальных вычислениях (символьное дифференцирование и интегрирование, решение уравнений);

б) различных областях автоматизированного проектирования:

- архитектурных сооружений;
- химических соединений;
- различных схем;

в) обработке текстов естественного языка;

г) управлении базами данных в случаях, когда база данных содержит не простые факты, либо когда требуется сделать не предусмотренные создателем изменения и запросы.

Очевидно (см,, напр., [1], [2]), что для решения этих задач в основу логического программирования положен аппарат математической логики. Мы рассмотрим некоторые его элементы (см., напр., [3],[4]) в логических программах, написанных на одном из наиболее распространенных языков логического программирования Prolog (программирование в терминах логики).

Так же, как элементы математической логики, логическая программа может содержать факты, правила и вопросы.

Классические демонстрационные программы «Смертен ли Сократ?»

```
predicates                          /*описания Функторов программ*/
  chelovek (symbol) smerten (symbol)
clauses                              /* факты и правила */
  chelovek ("Socrat").               /* факт: Сократ – человек */
  smerten (X): - chelovek (X).       /* правило: каждый человек
                                     смертен */

Вопросы:
  smerten ("Socrat") .              /* Смертен ли Сократ ?*/
Ответ:Yes
smerten (X).                       /* перечислить всех описанных
                                        в программе смертных */
```

Ответ: X =”Socrat”

и «Ошибаются ли греки?»

predicates
chelovek (symbol) oshibaetsa (symbol) grek (symbol)
clauses
chelovek (“Socrat”).
chelovek (“Tjuring”).
grek (“Socrat”).
*oshibaetsa (X): - chelovek (X). /*человеку свойственно ошибаться*/*

достаточно наглядно демонстрируют это.

Унификация и правило резолюции в логическом программировании фактически означают замену переменных, а также использование необходимых и достаточных условий математической логики. Например, известные с детства логические загадки

«Сын моего отца, но не я»

predicates
son (symbol)
clauses
son (me).
son (brother).
Вопросы: 1) son(X). Ответ: X=me.
X= brother.
2)son(X),<>”me”. Ответ:X=brother.

и «Ребенок моего отца, но мне не брат»

child (symbol)
clauses
child (me).
child (brother).
child (sister).
Вопросы: 1) child(X), X<>brother.
Ответ: X=me. X=sister.
2) child (X), X<>brother, X<>me.
Ответ: X=sister

показывают процесс совмещения предложений (clauses (клауз) – фактов и правил), то есть так наз. унификацию. Схематически унификацию и правила резолюции можно представить так:

если A: - B_1, …, B_n – правило,

то в вопросе R_1, … A, … R_m условие A унифицируется с заголовком правила и по принципу резолюции вместо этого вопроса ищется ответ на вопрос: R_1, …, B_1, …, B_n, … R_m. С точки зрения математической логики это означает, что условия B_1, …, B_n являются необходимыми и достаточными для A.

Симметричные и транзитивные отношения в логическом программировании так же, как в математической логике, описываются отдельными правилами, только с заменой заголовков, обусловленной механизмом логического поиска.

Рассмотрим следующий пример. Предположим, для лечения некоторой болезни требуется вводить больному компоненты лекарств $К_1$, $К_2$, $К_3$. Причем известно что компоненты $К_1$ и $К_2$ должны быть введены вместе, а компоненты $К_1$ и $К_3$ с перерывом. Следующая программа позволит ответить на вопрос: «Можно ли задаваемые компоненты вводить вместе или с перерывом?»

```
predicates
vm (symbol, symbol) pe (symbol, symbol)
vm (symbol, symbol) vmes (symbol, symbol)
per (symbol, symbol) perer (symbol, symbol)
clauses
vm (k1,k2).            /* компоненты k1, k2 нужно вводить вместе */
pe (k3,k1).            /*компоненты k3 и k1 нужно вводить с перерывом*/
vme (X,Y): - vm (X,Y); vm (Y,X).   /* компоненту X нужно вводить
                                      вместе с компонентой Y, либо
                                      компоненту Y вместе с
                                      компонентой X */
vmes (X,Y): - vme (X,Z), vme (Z,Y);  /* компоненту X нужно вводить
                                      вместе с компонентой Y,если
                                      существует компонента Z,
                                      которую нужно вводить вместе с
                                      компонентами X и Y, либо с
                                      перерывом от каждой из них*/
per (X,Z), per (Z,Y).
per (X,Y): - pe (X,Y); pe (Y,X).   /* правило симметрии отношения с
                                      "перерывом"
 perer (X,Y): - per (X,Z), vme (Z,Y).  /* правило транзитивности
                                      отношений "вместе " и "с перерывом"*/
```

Вопросы:

1) perer (k1,k3). Ответ: yes

2) vmes (k1,X). Ответ:X=k2

Как необходимые и достаточные условия в математической логике позволяют упорядочивать множества, так и соответствующие факты и правила (достаточно прозрачные) логического программирования позволяют делать это же.

По ходу дела заметим, что при обучении любой математической дисциплине студенты должны видеть в каждой конкретной задаче общую схему, под которую подходит еще много других задач, отличающихся

конкретными формулировками. Например, задачи 1 и 2 можно полагать имеющими общую схему.

Задача 1.

Вдоль аллеи, идущей в меридианальном направлении, растут вишня, груша, орех, персиковое дерево, слива и яблоня. Известно, что яблоня располагается севернее груши, но южнее ореха. Груша растет рядом со сливой, а орех рядом с персиковым деревом. Между вишней и персиковым деревом расположено одно дерево, а между яблоней и персиковым деревом – два дерева. Столько же деревьев между грушей и вишней. В каком порядке растут деревья?

Задача 2.

Производственный технологический процесс построен так, что 6 операций (a,b,c,d,e,f) должны выполняться последовательно одна за другой. Известно, что операция f должна выполняться раньше операции b, но позже операции c, операции b и c должны быть рядом, c и d тоже рядом. Между a и d одна операция. Между f и d – две операции. Столько же операций между b и a. В каком порядке должны выполняться операции?

```
domains
  i = integer
predicates
  prav(i,i,i,i,i,i)  p1(i,i)  p2(i,i)  p3(i,i)  m(i)
clauses
  m(1). m(2). m(3). m(4). m(5). m(6).
  prav (X,Y,Z,P,Q,R): - m(X), m(Y), m(Z), m(P), m(Q), m(R), R<Y, Z<R,
                        p1(Y,Q), p1(Z,P),
                        p2(X,P), p3(P,Q), p3(X,Y).
  p1(X,Y): - X=Y+1; Y=X+1.          /* рядом */
  p2(X,Y): - X=Y+2; Y=X+2.          /* через 2 элемента */
  p3(X,Y): - X=Y+3; Y=X+3.          /* через 3 элемента */
```

На вопрос: prav (X,Y,Z,P,Q,R). дается единственно правильный ответ: 352614

Он означает, что операция a должна выполняться третьей, b – пятой, c – второй и т.д., т.е., если указать естественную последовательность операций, то они должны выполняться в таком порядке: e, c, a, f, b, d.

Пусть имеется два упорядоченных множества. Биекцию, т.е. взаимно-однозначное соответствие между ними, устанавливают в математической логике с помощью тех или иных правил – формул, словесных описаний, графиков. Первые два вида правил, вообще говоря, несложно реализуются в логических программах. Например, использование внутренних и внешних целей в Prolog-е позволяет достаточно просто организовать взаимно-однозначное соответствие, например, между множеством имен и чисел (номеров мест, соответствующих именам):

```
%множество имен {a,b,c}
```

```
%множество мест {1,2,3}
domains
  s = symbol  i = integer
predicates
  name(s)  mesto(i)  otv(s,i)
clauses
  name(a). name(b). name(c).
  mesto(1). mesto(2). mesto(3).
  otv(X,Y): - name(X), mesto(Y).
Внешняя цель (вопрос):
otv("a",3), otv("b",2), otv("c",X).
Ответ: X=1, X=2, X=3.
```

А чтобы был выдан единственный правильный ответ, если элементы множеств находятся во взаимно-однозначном соответствии, нужно дополнить вопрос условиями X<>2, X<>3. Вопрос в таком виде неудобен. Более удобно задать внутреннюю цель.

```
goal
  write("место a"), readint(X), write("место b"), readint(Y),
  otv("c",Z), Z<>X, Z<>Y,
  write("место c"), write(Z).
```

Недостаток этой программы: если программа будет иметь несколько ответов, то на экран будет выдан только первый.

Соответствие между двумя множествами предыдущей программы можно организовать, например, и с помощью шестиместного предиката, в котором каждая последующая пара аргументов означает имя и соответствующее ему место.

Подводя итог, можно отметить, что небольшой объем статьи не позволяет рассмотреть более детально и глубоко вынесенную в заголовок тему. Поэтому вышеуказанное можно полагать только введением в данную тематику.

БИБЛИОГРАФИЯ

[1] Адаменко А., Кучуков А. Логическое программирование и Visual Prolog. – С.-Петербург.: 2003.

[2] Братко И. Алгоритмы искусственного интеллекта на языке PROLOG. – М.: 2004.

[3] Мендельсон Э. Введение в математическую логику. – М.: 1986.

[4] Новиков П.С. Элементы математической логики. – М.: 1973.