УДК 681.3.06

ПЕРЕВОД ФОРМУЛ ИСЧИСЛЕНИЯ ПРЕДИКАТОВ ПЕРВОГО ПОРЯДКА В ПРЕФИКСНЫЙ ВИД

Г.П. Неговорова

Программа преобразует правильно построенные формулы исчисления предикатов первого порядка в префиксный вид, являющийся разновидностью так называемой польской записи. Префиксная форма в значительной степени облегчает исследование и преобразование логических формул. Если формула представлена в префиксном виде, то легко построить дерево формулы, помещая в его вершины логические связки.

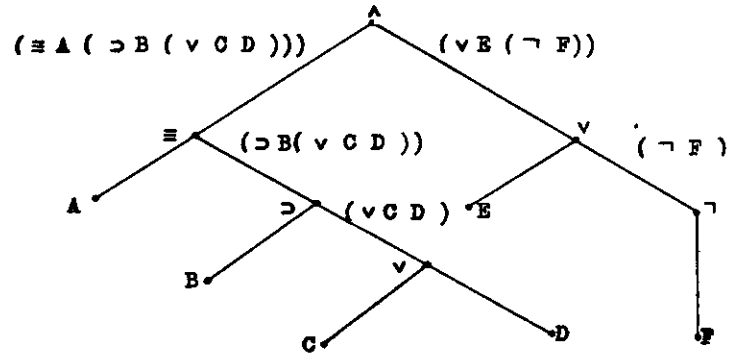
Пусть дана логическая формула

$$(A \equiv (B \supset C \vee D)) \wedge (E \vee \neg F),$$

префиксный вид которой

$$(\wedge (\equiv A (\supset B (\vee C D))) (\vee E (\neg F))).$$

Тогда получаем следующее дерево формулы:



На этом дереве легко просматривается структура формулы. В случае бинарных операций – (∧ , ∨ , ≡ , ⊃) из каждой вершины выходят два ребра, в случае унарной – (¬) – только одно.

I. Определение входной и выходной формул. Опишем правила задания правильно построенных формул (ППФI) исчисления предикатов первого порядка, являющихся входными для описываемого алгоритма, и префиксного вида (ППФ2), в который переводится исходная запись.

Упорядочим логические связи по старшинству: ¬(отрицание) – I, ∧ (конъюнкция) – 2, ∨ (дизъюнкция) – 3, ≡ (тождество) – 4, ⊃ (импликация) – 5.

Область действия логической связки в формуле распространяется до следующей логической связки с большим или равным ей старшинством либо до первой открывающейся скобки. Самая узкая область действия, согласно введенному старшинству логических связок, у отрицания, самая широкая – у импликации.

Воспользуемся записью в бэкусовой нормальной форме [3], являющейся вариантом теоретико-множественного задания классов через подклассы или составляющие их элементы.

Расширим это описание введением связки :=: – синтаксической эквивалентности. А :=: В читается как А эквивалентно В, то есть любая формула на описываемом языке, включающая конструкции "А", считается тождественной той же формуле, в которой все вхождения конструкции "А" заменены на конструкции "В".

Тогда входная формула (ППФI) определяется следующим образом: ⟨ ППФI ⟩ ::= ⟨ имя предиката ⟩ | (⟨ имя предиката ⟩ ⟨ последовательность термов ⟩) | (⟨ ППФI ⟩) | (¬ ␣ ⟨ ППФI ⟩) | (⟨ ППФI ⟩ ␣ ∧ ␣ ⟨ ППФI ⟩) | (⟨ ППФI ⟩ ␣ ∨ ␣ ⟨ ППФI ⟩) | (⟨ ППФI ⟩ ␣ ≡ ␣ ⟨ ППФI ⟩) | (⟨ ППФI ⟩ ␣ ⊃ ␣ ⟨ ППФI ⟩) | ((⟨ кванторный комплекс ⟩) ⟨ ППФI ⟩).

При отсутствии скобок выделение аргументов логических связок осуществляется в соответствии со старшинством логических связок:

(⟨ ППФI ⟩ ⟨ логическая связка старшинства n ⟩ ⟨ ППФI ⟩)
⟨ логическая связка старшинства m ⟩ :=:
⟨ ППФI ⟩ ⟨ логическая связка старшинства n ⟩ ⟨ ППФI ⟩
⟨ логическая связка старшинства m ⟩ , где m ≥ n.

⟨ имя предиката ⟩ ::= ⟨ ЛИСП – атом общего типа ⟩

⟨ ЛИСП – атом общего типа ⟩ ::= ⟨ знак числа ⟩ | ⟨ буквенно-знаковый атом ⟩.

⟨ знак числа ⟩ ::= + | -

⟨ буквенно-знаковый атом ⟩ ::= ⟨ буква ⟩ | ⟨ специальный знак ⟩ | ⟨ буквенно-знаковый атом ⟩ ⟨ буква ⟩ | ⟨ буквенно-знаковый атом ⟩ ⟨знак ⟩ | ⟨ буквенно-знаковый атом ⟩ ⟨ цифра ⟩

Буквенно-знаковый атом должен состоять не более чем из 29 литер.

⟨ литера ⟩ ::= ⟨ буква ⟩ | ⟨ цифра ⟩ | ⟨ знак ⟩ | ⟨ ограничитель ⟩

⟨ буква ⟩ ::= А|Б|В|Г|Д|Е|Ж|З|И|К|Л|М|Н|О|П|Р|С|Т|У|Ф|Х|Ц|Ч|Ш|Щ|Ы|Ь|Э|Ю|Я|D|F|G|I|J|L|N|Q|R|S|U|V|W|Z

⟨ цифра ⟩ ::= 0|I|2|3|4|5|6|7|8|9

⟨ знак ⟩ ::= ⟨ знак числа ⟩ | ⟨ специальный знак ⟩

⟨ специальный знак ⟩ ::= !|,|$_{10}$|↑|×|=|;|[|]|+|'|"|*|/|<|:|¬|≤|≥|[illegible]|⊃|≡|¬|∧|∨|~|–|!|÷

⟨ ограничитель ⟩ ::= (|) | . | [illegible]

⟨ последовательность термов ⟩ ::= ⟨ терм ⟩ | ⟨ терм ⟩ ⟨последовательность термов ⟩

⟨ терм ⟩ := ⟨ ЛИСП – атом общего типа ⟩ | (⟨ имя функции ⟩ ⟨ последовательность термов ⟩)

⟨ имя функции ⟩ ::= ⟨ ЛИСП – атом общего типа ⟩

В качестве ЛИСП – атомов общего типа запрещается использовать следующие специальные знаки и комбинации: ¬, ∧, ∨, ≡, ⊃, А:, Е:

⟨ кванторный комплекс ⟩ ::= ⟨ квантор ⟩ ⟨ последовательность объектных переменных ⟩

⟨ квантор ⟩ ::= ⟨ квантор всеобщности ⟩ | ⟨ квантор существования ⟩

⟨ квантор всеобщности ⟩ ::= А:

⟨ квантор существования ⟩ ::= Е:

⟨ последовательность объектных переменных ⟩ ::= ⟨ ЛИСП – атом общего типа ⟩ | ⟨ ЛИСП – атом общего типа ⟩ ⟨ последовательность объектных переменных ⟩

Правила записи на языке ЛИСП подразумевают следующие эквивалентности, которые справедливы и для закрывающихся скобок:

␣ (:=: ((␣ :=: (␣ (␣ :=: ␣ (:=: (

Один пробел эквивалентен любому числу пробелов.

Тогда (исходя из введенных выше определений) префиксная форма определяется следующим образом:

⟨ ПФ2 ⟩ ::=⟨ имя предиката ⟩|(⟨ имя предиката ⟩ ⟨ после - довательность термов ⟩) | (¬ ␣⟨ ПФ2 ⟩) | (∧ ␣ ⟨ ПФ2 ⟩⟨ ПФ2 ⟩) |(∨ ␣⟨ ПФ2 ⟩ ⟨ ПФ2 ⟩) | (≡ ␣⟨ ПФ2 ⟩ ⟨ ПФ2 ⟩) | (⊃ ␣⟨ ПФ2 ⟩ ⟨ ПФ2 ⟩) | ((⟨ кванторный комплекс ⟩)⟨ ПФ2 ⟩).

2. О п и с а н и е п р о г р а м м ы. Программа написана на языке ЛИСП (ВЦ АН СССР) для ЭВМ БЭСМ-6 [1],[2].

В программе описаны три функции класса SEXPR , внесенные под один оператор DEFINE : PREF , REV , SUBLIS и две функции класса SFEXPR : INPEX и CONST.

Функция INPEX с помощью функции SUBLIS снабжает каждую логическую связку индексом, согласно введенному старшинству логических связок. С полученной после работы SUBLIS формулой начинает работать функция PREF - основная функция программы.

Функция PREF осуществляет последовательное выделение элементов формулы с целью обнаружения логических связок и установления по приписанному им индексу их области действия.

При последовательном выделении элементов списка выявляются логические связки и сравниваются их индексы - числа. Если последующий индекс меньше предыдущего (другими словами, соответствующая последующему индексу область действия логической связки уже, чем у предыдущей), то функция PREF выделяет следующие элементы формулы, пока не обнаружит логическую связку с более широкой или равной областью действия. Тогда формируется список, первым элементом которого является последняя из выделенных логических связок (F), затем обработанная к этому моменту часть формулы слева от этой связки - (L), а к оставшейся правой части формулы - (R) функция PREF обращается рекурсивно.

Функция PREF просматривает всю формулу до конца, поскольку выделение элементов происходит последовательно и при выделении очередного элемента формулы производится проверка на конец списка. Функция перестает работать только тогда, когда исчерпает все элементы формулы. При выделении очередного элемента формулы им может оказаться элемент, сам являющийся списком. Тогда к нему снова рекурсивно обращается функция PREF . В результате работы функции PREF получается формула в префиксном виде.

Рассмотрим простейший пример.

Пусть имеется исходная формула (A ␣⊃ ␣ B ␣ ∨ ␣ C), которая является аргументом функции INPEX. После работы функции SUBLIS логическим связкам сопоставляются индексы, с которыми они образуют точечные пары: (A (⊃ · 5)B (∨·3) C).

После этого работает функция PREF , которая и приводит формулу к префиксному виду: (⊃ A (∨ B C)).

Блок-схема функции PREF приводится в приложении. Она записана неформально в терминах ЛИСП-функций.

В своей работе функция PREF использует вводимую функцию REV. Функция REV переставляет элементы списка в обратном порядке, то есть накапливает перевернутый список. Причем если элемент списка сам является списком, то внутри него функция REV не работает.

Вспомогательной функцией является также вводимая функция SUBLIS [1]. Значением её первого аргумента должна быть таблица соответствия a, состоящая из пар вида (u_i · v_i), где u_i - атом. Если u_i входит в выражение x , являющееся значением второго аргумента, то повсюду в этом выражении оно заменяется выражением v_i . В данной программе функция SUBLIS , выделяя в формуле логические связки, образует точечные пары, снабжая каждую логическую связку индексом, согласно введенному старшинству связок: (¬ · 1), (∧ · 2), (∨ · 3), (≡ · 4), (⊃ · 5). Для изменения старшинства связок достаточно лишь задать другие выражения v_i , например, (¬ · 5),(≡ · 4),(⊃ · 3),(∨ · 2),(∧ · 1).

В программе вводятся также константы A:, E:, ¬ с помощью функции CONST .

Автор глубоко признателен В.С.Лозовскому за ценные указания по написанию статьи.

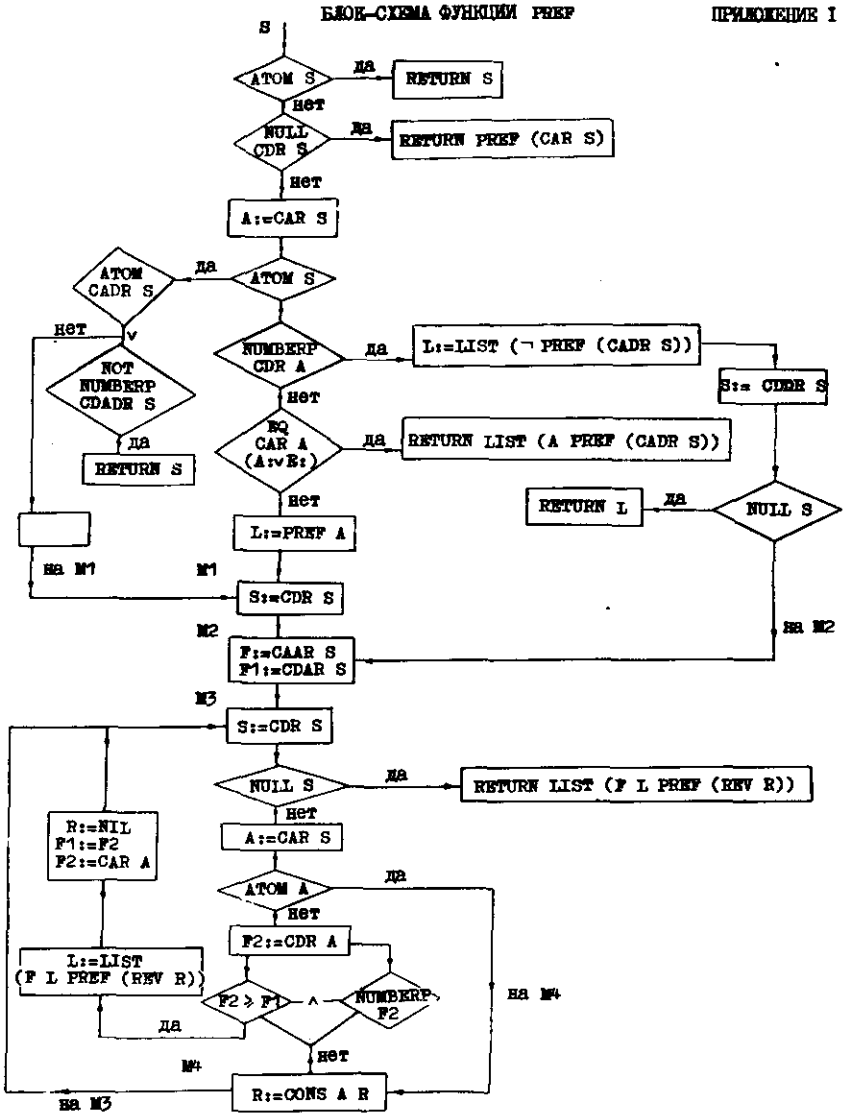
Л и т е р а т у р а

1. ЛАВРОВ С.С., СИЛАГАДЗЕ Г.С. Входной язык и интерпретатор системы программирования на базе языка ЛИСП для машины БЭСМ-6 М.,ВЦ АН СССР,1969.

2. СИЛАГАДЗЕ Г.С. Система программирования с языка ЛИСП для машины БЭСМ-6. Диссертация на соискание ученой степени кандидата физ.-мат.наук. М., ВЦ АН СССР, 1971.

3. Алгоритмический язык АЛГОЛ-60. Пересмотренное сообщение. Под редакцией А.П.Ершова, С.С.Лаврова, М.Р.Шура-Бура. М.,Изд-во "Мир", 1965.

Поступила в ред.-изд.отд.
17 апреля 1973 года



Текст программы перевода формулы исчисления предикатов первого порядка в префиксный вид

Функции INPEX CONST DEFINL

```
INPEX 1
        01 00   (SFEXPR INPEX
        02 00    (LAMBDA
        03 03     (S)
        03 00     (PREF
INPEX 2
        04 00      (SUBLIS
        05 00       (QUOTE
        06 00        ((¬.
        10 07          (¬.1))
        07 00         (∧.
        10 07          (∧.2))
        07 00         (∨.
        10 07          (∨.3))
INPEX 3
        07 00         (≡.
        10 07          (≡.4))
        07 00         (⊃.
        10 01          (⊃.5))))S))))
CONST
        01 00   (SFEXPR CONST
        02 00    (LAMBDA
        03 03     (X Y)
        03 00     (PROG NIL    X
        04 00      (CSET
        05 05       (CAR X)
        05 04       (CAR Y))
        04 04      (POP X)
        04 04      (POP Y)
        04 00      (COND
        05 06       ((NULL X)
        06 04        (RETURN T)))
        04 01      (GO X))))
CONST 2
        C1 00   (CONST
        02 02    (A: E: ¬)
        02 01    (A: E: ¬))
DEFINE
        01 00   (DEFINE
```

ФУНКЦИЯ PREF

```
PREF 1
       02 00    (PREF
       03 00     (LAMBDA
       04 04      (S)
       04 00      (PROG
       05 05       (L R A F F1 F2)
PREF 2
       05 00       (COND
       06 07        ((ATOM S)
       07 06         (RETURN S))
       06 00        ((NULL
       10 07          (CDR S))
PREF 3
       07 00         (RETURN
       10 00          (PREF
       11 05           (CAR S)))))
       05 00       (SETQ A
       06 05        (CAR S))
       05 00       (COND
PREF 4
       06 07        ((ATOM A)
       07 00         (COND
       10 00          ((OR
       12 00            (ATOM
       13 12             (CADR S))
       12 00            (NOT
PREF 5
       13 00               (NUMBERP
       14 11                (CDADR S))))
       11 10            (RETURN S))
       10 00           (T
       11 11            (SETQ L A)
       11 06            (GO M1))))
       06 00         ((NUMBERP
 REF 6
       10 07           (CDR A))
       07 00          (SETQ L
       10 00           (LIST ¬
       11 00            (PREF
       12 07             (CADR S))))
       07 00          (SETQ S
PREF 7
       10 07           (CDDR S))
       07 00          (COND
       10 11           ((NULL S)
       11 07            (RETURN L)))
       07 06          (GO M2))
```



```
PREF 8
       06 00        (T
       07 00         (COND
       10 00          ((OR
       12 00            (
       13 12             (CAR A) B:)
       12 00            (EQ
       13 11             (CAR A) A:))
PREF 9
       11 00           (RETURN
       12 00            (LIST A
       13 00             (PREF
       14 10              (CADR S)))))
PREF 10
       10 00            (T
       11 00             (SETQ L
       12 05              (PREF A))))))    M1
       05 00       (SETQ S
       06 05        (CDR S))    M2
PREF 11
       05 00       (SETQ F
       06 05        (CAAR S))
       05 00       (SETQ F1
       06 05        (CDAR S))    M3
       05 00       (SETQ S
PREF 12
       06 05        (CDR S))
       05 00       (COND
       06 07        ((NULL S)
       07 00         (RETURN
       10 00          (LIST F L
       11 00           (PREF
       12 05            (REV R))))))
PREF 13
       05 00       (SETQ A
       06 05        (CAR S))
       05 00       (COND
       06 07        ((ATOM A)
PREF 14
       07 06         (GO M4))
       06 00        ((AND
       10 00          (NUMBERP
PREF 15
       11 00           (SETQ F2
       12 10            (CDR A)))
       10 07          (GREQP F2 F1))
       07 00         (SETQ L
       10 00          (LIST F L
```

```
PREF 16
      11 00          (PREF
      12 07           (REV R))))
      07 07         (SETQ R NIL)
      07 07         (SETQ F1 F2)
      07 00         (SETQ F
PREF 17
      10 07          (CAR A))
      07 05         (GO M3)))     M4
      05 00       (SETQ R
      06 05        (CONS A R))
      05 02       (GO M3))))
```

Функция REV

```
REV 1
      02 00    (REV
      03 00     (LAMBDA
      04 04      (X)
      04 00      (PROG
REV 2
      05 05       (V)    A
      05 00       (COND
      06 07        ((NULL X)
      07 05         (RETURN V)))
REV 3
      05 00       (PUSH
      06 05        (CAR X) V)
      05 05       (POP X)
      05 02       (GO A))))
```

Функция SUBLIS

```
SUBLIS 1
      02 00    (SUBLIS
      03 00     (LAMBDA
      04 04      (A X)
      04 00      (COND
SUBLIS 2
      05 06       ((ATOM X)
      06 00        (PROG
      07 07         (B)
      07 07         (SETQ B A)    L
      07 00         (COND
SUBLIS 3
      10 11          ((NULL B)
      11 10           (PETURN X))
      10 00          ((EQ X
      12 11            (CAAR B))
```



```
      11 00          (RETURN
      12 07           (CDAR B))))
SUBLIS 4
      07 00        (SETQ B
      10 07         (CDR B))
      07 05        (GO L)))
      05 00      (T
      06 00       (CONS
SUBLIS 5
      07 00        (SUBLIS A
      10 07         (CAR X))
      07 00        (SUBLIS A
      10 01         (CDR X))))))))
```

Примеры обращения к функции INPEX

```
      01 00   (INPEX
      02 03    ((A: X E: Y)
      03 00     (F
      04 01      (P X) Y)))

      01 00   (INPEX
      02 03    ((A: X E: Y)
      03 04     ((F X Y)∨
      04 01      (F Y X))))

      01 00   (INPEX
      02 06    ((((A: X E: Y)
      06 07        ((F X Y)∨
      07 05         (F Y X)))∧
      05 06       ((A: X Y)
      06 07         ((F X Y)⊃
      07 04          (F Y Y))))⊃
      04 05      ((E: Z)
      05 01       (F Z Z)))))

      01 00   (INPEX
      02 01    (A ∨ B))
      01 00   (INPEX
      02 01    (¬A))
      01 00   (INPEX
      02 01    (¬ A ⊃ B ∧ C))

      01 00   (INPEX
      02 01    ((¬ A ⊃ B) ∨ C))

      01 00   (INPEX
      02 00    (A ⊃ B ≡ C ⊃
      03 03     (D ∨ ¬ A ⊃ C)≡ ¬ A ∧
      03 01     (¬ B ∨ C)))
```

Примеры формул на входном и выходном языках

```
      EXPRESSION ENTERED ...
(INPEX ((A: X E: Y) (F (P X) Y)))
      R E S U L T ...
((A: X E: Y) (F (P X) Y))

      EXPRESSION ENTERED ...
(INPEX ((A: X E: Y) ((F X Y)∨(F Y X))))
      R E S U L T ...
((A: X E: Y) (∨(F X Y) (F Y X)))

      EXPRESSION ENTERED ...
(INPEX (((((A: X E: Y) ((F X Y)∨(F Y X)))∧((A: X Y)
      ((F X Y)⊃(F Y Y))))⊃((E: Z) (F Z Z)))))
      R E S U L T ...
(⊃(∧((A: X E: Y) (∨(F X Y) (F Y X))) ((A: X Y)
      (⊃(F X Y) (F Y Y)))) ((E: Z) (F Z Z)))

      EXPRESSION ENTERED ...
(INPEX (A∨B))
      R E S U L T ...
(∨ A B)

      EXPRESSION ENTERED ...
(INPEX (¬A))
      R E S U L T ...
(¬A)

      EXPRESSION ENTERED ...
(INPEX (¬A⊃B∧C))
      R E S U L T ...
(⊃(¬A) (∧B C))

      EXPRESSION ENTERED ...
(INPEX ((¬A⊃B)∨C))
      R E S U L T ...
(∨(⊃(¬A) B) C)

      EXPRESSION ENTERED ...
(INPEX (A⊃B≡C⊃(D∨¬A⊃C)≡¬A∧(¬B∨C)))
      R E S U L T ...
(⊃(⊃A(≡B C)) (≡(⊃(∨D(¬A)) C) (∧(¬A) (∨(¬B) C))))
```