Логические основы Пролога. Рекурсия на Прологе

Лекция №2

План

1. Логика предикатов
2. Унификация
3. Метод резолюции

Почему логика предикатов, а не логика высказываний?

- ЛВ:
  - Все люди смертны.
  - Сократ - человек.
  - Следовательно, Сократ смертен.
- ЛП:
  - Быть_человеком(Сократ).
  - Быть_смертным(человек).

Логика предикатов

- Формальная система
- Алфавит логики предикатов
- Терм
- Атомарная формула
- Формулы логики предикатов
- Литерал
- Дизъюнкт
- Конъюнктивная, пренексная, сколемовская нормальная форма

Терминология какой дисциплины здесь используется?

Логика предикатов

- В формальной системе определены:
  - Алфавит:
    - Переменные (например, x, y, z)
    - Константы (например, a, b, c)
    - Функциональные символы (обычно f, g)
    - Предикатные символы (обычно p, r)
    - Пропозициональные константы (true, false)
    - Логические связки (конъюнкция, дизъюнкция, отрицание, импликация, эквивалентность)
    - Кванторы (существования и всеобщности)
    - Вспомогательные символы: ( ) , .
  - Формулы (словарь)
  - Аксиомы
  - Правила вывода

Что из перечисленного является конечным, а что может быть бесконечным?

Основные понятия

- Терм:
  - Всякая переменная – терм
  - Всякая константа – терм
  - Если $t_1,...,t_n$ - термы, а f - n-местный функциональный символ, то $f(t_1, ..., t_n)$ - терм
  - Других термов нет
- Все объекты в Прологе – термы

Основные понятия

- Атомарный предикат (атомная формула) – результат применения предиката к термам, задают отношения между сущностями
  - отец(x,y)
- Все объекты в Прологе – термы
- Литерал – атомарная формула или отрицание атомарной формулы
- Аргументы формулы упорядочены по смыслу
  - любит(x,y)

Унификация

- Унификация – отождествление формулы путем замены свободных переменных на термы
- Унификация – процесс поиска такой подстановки термов из одного выражения в переменные второго выражения, что оба эти выражения становятся эквивалентными
  - птица(воробей)
  - птица(сорока)
  - птица(X)

Какую подстановку выбрать?

Унификация

- A=p(f(x),z) и B=p(y,a).
- Первый вариант подстановки: (y/f(x),z/a)
- Второй вариант подстановки: (y/f(a),x/a,z/a)
- Существуют ли другие варианты подстановок?
- Какой вариант более общий?

Теорема унификации

- Теорема унификации: для любого унифицируемого конечного множества простых выражений S алгоритм унификации закончит свою работу и выдаст наиболее общий унификатор для S

Правило резолюции

- Если для двух дизъюнктов существует атомная формула, которая в один дизъюнкт входит положительно, а в другой отрицательно, то, вычеркнув соответственно из одного дизъюнкта положительное вхождение атомной формулы, а из другого — отрицательное, и объединив эти дизъюнкты, мы получим дизъюнкт, называемый резольвентой.

Правило резолюции

- Исходные дизъюнкты – револьвируемые
- Вычеркнутые формулы – контрарные литералы
- Конечный дизъюнкт – резольвента

Метод резолюции

- Метод резолюции – доказательство от противного: добавляем к множеству аксиом отрицание гипотезы и выводим противоречие). Если это удается, то исходная формула была выводима
- Процесс резолюции обязательно завершится, если исходное множество дизъюнктов невыполнимо

Пример резолюции

- «Кто умеет читать, тот грамотный»: $(\forall x)(Ч(x) \rightarrow Г(x))$
- «Дельфины неграмотны»: $(\forall x)(Д(x) \rightarrow \neg Г(x))$
- «Некоторые дельфины обладают интеллектом»: $(\exists x)(Д(x) \rightarrow И(x))$
- Предположим, что из этих посылок мы хотим доказать утверждение:
- «Некоторые из тех, кто обладают интеллектом, не умеют читать»: $(\exists x)(И(x) \wedge \neg Ч(x))$

Пример резолюции

- Множество предложений, соответствующих утверждениям 1-3, будет таким:

1) $\neg Ч(x) \vee Г(x)$

2) $\neg Д(y) \vee \neg Г(y)$

3а) Д(А)

3б) И(А)

Отрицание теоремы, которую требуется доказать, преобразованное к форме предложения, имеет вид:

4) $\neg И(z) \vee Ч(z)$

Пример резолюции

- Множество предложений, соответствующих утверждениям 1-3, будет таким:

1) $\neg Ч(x) \vee Г(x)$

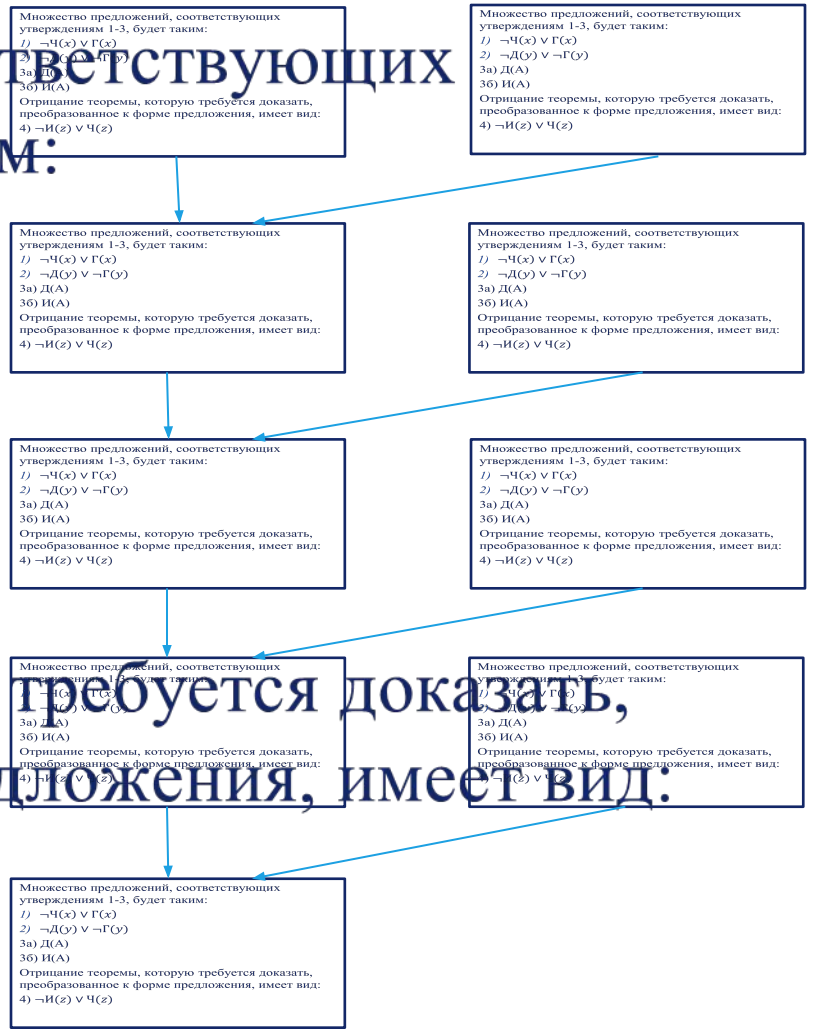
2) $\neg Д(y) \vee \neg Г(y)$

3а) Д(А)

3б) И(А)

Отрицание теоремы, которую требуется доказать, преобразованное к форме предложения, имеет вид:

4) $\neg И(z) \vee Ч(z)$



Логическая программа

- Логическая программа – конечное непустой множество хорновских дизъюнктов (фактов и правил)
- В Прологе реализована линейная резолюция (правило резолюции применяется к самому левому литералу цели и первому унифицируемому к ней дизъюнкту)
- При каждом выборе унифицируемого с целью дизъюнкта запоминается точка возврата

Стратегии резолюции

- Стратегии полного перебора (поиск в ширину)
- Стратегия опорного множества
- Стратегия предпочтения одночленам
- Стратегия, линейная по входу
- Комбинирование стратегий

Предложения

1) Факты
2) Правила
3) Вопросы

Общий вид:

$A :- B_1, \ldots, B_n.$

Факты и правила

Пример факта:

мама(«Наташа», «Даша»).

константа, переменная, составной объект

Пример правил:

бабушка(X,Y) :- мама(X,Z), мама(Z,Y).

бабушка(X,Y) :- мама(X,Z), папа(Z,Y).

процедура

Переменные

- Неявно связаны квантором всеобщности
- Не поддерживается механизм деструктивного присваивания
- Идентификатор указывает не на адрес ячейки памяти, а на объект
- Свободные (неконкретизированные) и связанные (конкретизированные)
- Область определения – одно предложение
- Все анонимные переменные – отдельные объекты

Вопросы. Вычисление цели

```
мама("Наташа","Даша").
мама("Даша","Маша").
goal
    %мама("Наташа","Даша").
    %мама("Наташа","Маша").
    %мама(X, "Даша").
    %мама("Наташа",X).
    %мама(X,Y).
    %мама(X,_).
    %мама(_,_).
```

Возможные результаты работы программы:

1) Цель достигнута (Yes): либо значения переменных, либо No solutions
2) Цель не достигнута (No): либо отношение не выполняется, либо нет достаточной информации

Вычисление цели

```
мама("Наташа","Даша").
мама("Даша","Маша").
бабушка(X,Y) :- мама(X,Z),
        мама(Z,Y).
goal
    бабушка("Наташа",X).
```

Нахождение максимума из двух чисел

```
max(X,Y,X) :-
   X>Y. /* если первое число больше второго,
           то первое число - максимум */
max(X,Y,Y) :-
   X<Y. /* если первое число меньше второго,
             то второе число - максимум */
max(X,Y,Y) :-
   X=Y. /* если первое число равно второму,
           возьмем в качестве максимума
           второе число */
```

Нахождение максимума из двух чисел - 2

```
max(X,Y,X):-
   X>Y. /* если первое число больше второго,
           то первое число - максимум */
max(X,Y,Y):-
   X<=Y./* если первое число меньше или равно
              второму, возьмем в качестве
    максимума второе число */
```

Нахождение максимума из двух чисел (отсечение)

```
max2(X,Y,X):-
   X>Y,!./* если первое число больше второго,
                то первое число - максимум */
max2(_,Y,Y).  /* в противном случае
    максимумом будет второе число */
```

Условия

S:-

<условие>,!,P.

S :-

P2.

if <условие> then P else P2

Нахождение максимума из трех чисел

```
max3a(X,Y,Z,X):-
   X>=Y,X>=Z.
   /* если первое число больше или равно второму
      и третьему, то первое число - максимум */
max3a(X,Y,Z,Y):-
   Y>=X,Y>=Z.
   /* если второе число больше или равно первому
      и третьему, то второе число является
      максимумом */
max3a(X,Y,Z,Z):-
   Z>=X,Z>=Y.
   /* если третье число больше или равно первому
      и второму, то максимум - это третье число */
```

Нахождение максимума из трех чисел (отсечение)

```
max3b(X,Y,Z,X):-
   X>Y,X>Z,!.
   /* если первое число больше второго и третьего,
      то первое число - максимум */
max3b(_,Y,Z,Y):-
   Y>=Z,!.
   /* иначе, если второе число больше третьего,
      то второе число является максимумом */
max3b(_,_,Z,Z).
    /* иначе максимум - это третье число */
```

Нахождение максимума из трех чисел (с помощью max2)

```
max3(X,Y,Z,M):-
    max2(X,Y,XY), /* XY - максимум из X и Y */
    max2(XY,Z,M). /* M - максимум из XY и Z */
```

Рекурсия на Прологе

Программа «Родственники»

```
предок(Предок,Потомок):-
        родитель(Предок,Потомок).
        /* предком является родитель */
предок(Предок,Потомок):-
        родитель(Предок,Человек),
        предок(Человек,Потомок).
        /* предком является родитель предка */
```

Правило, реализующее шаг рекурсии

<имя определяемого предиката>:-

[<подцели>],

[<условие выхода из рекурсии>],

[<подцели>],

<имя определяемого предиката>,

[<подцели>].

Программа «Факториал»

1!=1 /* факториал единицы равен единице */

N!=(N-1)!*N /* для того, чтобы вычислить факториал некоторого числа, нужно вычислить факториал числа на единицу меньшего и умножить его на исходное число */

Факториал

```
fact(1,1). /* факториал единицы равен единице */
fact(N,F):-
   N1=N-1,
   fact(N1,F1), /* F1 равен факториалу числа
                   на единицу меньшего исходного
                   числа */
   F=F1*N. /* факториал исходного числа равен
              произведению F1 на само число */
```

Факториал

```
fact(1,1). /* факториал единицы равен единице */
fact(N,F):-
   N>1, /* убедимся, что число больше единицы */
   N1=N-1,
   fact(N1,F1), /* F1 равен факториалу числа,
                   на единицу меньшего исходного
                   числа */
   F=F1*N. /* факториал исходного числа равен
              произведению F1 на само число */
```

Факториал

```
fact(1,1):-!. /* условие останова рекурсии */
fact(N,F):-
   N1=N-1,
   fact(N1,F1), /* F1 равен факториалу числа,
                   на единицу меньшего исходного
                   числа */
   F=F1*N. /* факториал исходного числа равен
              произведению F1 на само число */
```

Факториал
Правосторонняя рекурсия

```
fact2(N,F,N,F):-!. /* останавливаем рекурсию, когда третий
                      аргумент равен первому*/
fact2(N,F,N1,F1):-
   N2=N1+1, /* N2 - следующее натуральное число
               после числа N1 */
   F2=F1*N2, /* F2 - факториал N2 */
   fact2(N,F,N2,F2).
   /* рекурсивный вызов с новым натуральным
      числом N2 и соответствующим ему
      посчитанным факториалом F2 */
```

Факториал

```
factM(N,F):-
    fact2(N,F,1,1). /* вызываем предикат с уже
                       заданными начальными
                       значениями */
```

Цикл с предусловием

w :-

<условие>, p, w.

w :- !.

while <условие> do P

Программа «Родственники» левосторонняя рекурсия

```
предок2(Предок,Потомок):-
        родитель(Предок,Потомок).
        /* предком является родитель */
предок2(Предок,Потомок):-
        предок2(Человек,Потомок),
        /* предком является родитель предка */
        родитель(Предок,Человек).
```