Отсечение и его использование.

Лекция 4 (Часть 2).

Примеры совместного использования отсечения и рекурсии при описании теоретико-множественных операций.

Специальности : 230105, 010501

Множества и операции над ними.

Множества представляются в виде списков.

Отличие :

– Множество не содержит повторяющихся элементов.

– Представляется неупорядоченной последовательностью.

Примеры.

Списку [1,2,3,4,3,2] соответствует множество [1,2,3,4].

Множества [1,2,3,4], [4,3,2,1], [1,3,4,2], [1,2,4,3] эквивалентны.

Основные операции над множествами :

- **Принадлежность элемента множеству ;**
- **Включение ;**
- **Пересечение множеств ;**
- **Объединение множеств ;**
- **Вычитание множеств.**

Преобразование списка в множество.

```
/* Определение
принадлежности списку */
member(H,[H|T]) :- !.
member(Elem,[H|T]) :-
            member(Elem,T).
```

```
/* Преобразование списка в
   множество (вариант 1) */
list_set( [ ], [ ] ).
list_set( [ H | T ], Res) :-
        member ( H, T ), !,
        list_set ( T, Res ).
list_set( [ H |T ], [H | Res]) :-
        list_set(T, Res).
```

```
/* Преобразование списка
   в множество (вариант 2) */

list_set1 ( [ ], [ ] ).
list_set1 ( [ H |T ], Res):-
        member (H, T), !,
        delete(H, T, Res1),
        list_set1 ( Res1, Res2 ),
        append ( [ H ], Res2, Res).
list_set1 ( [ H | T ], Res) : -
        list_set1 ( T, Res1),
        append ( [ H ], Res1, Res).
```

Примеры.

Согласование ЦУ list_set ([1, 2, 3, 2, 3, 4, 5, 4, 5, 2, 1], Res) дает Res = [3, 4, 5, 2, 1].

Согласование ЦУ list_set1 ([1, 2, 3, 2, 3, 4, 5, 4, 5, 2, 1], Res) дает Res = [1, 2, 3, 4, 5].

Принадлежность множеству.

Операция принадлежности элемента множеству реализуется аналогично принадлежности списку, отличие состоит в отсутствии необходимости отсечения.

Для сравнения :

```
/* Определение принадлежности
   списку */

member(H, [ H | _ ] ) : - !.
member( Elem, [ _ | T ] ) : -
        member ( Elem, T ).
```

```
/* Определение
   принадлежности множеству */

memb_set (H, [ H | _ ] ) .
memb_set ( Elem, [ _ | T ] ) : -
        memb_set ( Elem, T ).
```

X принадлежит некоторому множеству Y, если X является одним из элементов Y.

Включение.

Множество Y включает в себя множество X, если каждый элемент множества X является также элементом множества Y. Причем множество Y может содержать некоторые элементы, которых нет в X.

```
/* Принадлежность элемента множеству */
   memb_set(H, [ H | _ ] ).
   memb_set(Elem, [ _ | T ] ) :- memb_set (Elem,T).
/* Определение, является ли одно множество
   подмножеством другого */
   sub_set ( [ ], _ ).
   sub_set ( [ H | T ], Set2 ) :- memb_set ( H, Set2 ),
                  sub_set ( T, Set2 ).
```

Пример. **Согласование ЦУ *sub_set ([3, 4, 5], [1, 2, 3, 4, 5, 6])* дает *Yes*.**

Таким образом, sub_set (X,Y) **завершается успешно, если X является подмножеством Y, то есть Y включает X. Условие завершения рекурсии опирается на математическое соглашение о том, что пустое множество является подмножеством любого множества.**

Пересечение множеств.

Пересечением множеств X и Y является множество, содержащее те элементы, которые одновременно принадлежат X и Y.

```
/* Пересечение */
    cross_sets ( [ ], _, [ ] ).
    cross_sets ( [ H | T ], Set2, [ H | Res ] ) :-
                member ( H, Set2 ), !,
                cross_sets ( T, Set2, Res ).
    cross_sets ( [ H | T ], Set2, Res ) :-
                cross_sets (T, Set2, Res).
```

Пример. **Согласование ЦУ *cross_sets ([1,2,3,4], [5,6,3,4], Res)* дает *Res=[3,4]*.**

***Таким образом*, ЦУ *cross_sets(X,Y,Z)* доказуемо, если пересечением множеств X и Y является множество Z.**

Объединение множеств.

Объединением множеств X и Y является множество, содержащее элементы, принадлежащие одновременно X и Y.

```
/* Объединение множеств */

unit_sets( [ ], Set2, Set2).
unit_sets( [H | T], Set2, [ H | Res ] ) :-
                not ( member ( H, Set2 )), !,
                unit_sets ( T, Set2, Res ).
unit_sets( [_ | T ], Set2, Res ) :-
                unit_sets ( T, Set2, Res ).
```

Пример. **Согласование ЦУ *unit_sets ([1,2,3,4], [7,8,5,6,3,4], Res)* дает *Res=[1,2,7,8,5,6,3,4]*.**

***Таким образом*, ЦУ *unit_sets (X, Y, Z)* доказуемо, если пересечением множеств *X* и *Y* является множество *Z*.**

Вычитание множеств.

Разностью множеств X и Y называется множество, содержащее все элементы множества X, не вошедшие в Y.

```
/* Вычитание множеств */
    differ_sets ( [ ], _, [ ] ).
    differ_sets ( [ H | T ], Set2, Res ) :-
          member ( H, Set2 ), !,
          differ_sets ( T, Set2, Res ).
    differ_sets ( [ H | T ], Set2, [ H | Res ] ) :-
          differ_sets ( T, Set2, Res ).
```

Пример 1. **Согласование ЦУ *differ_sets ([1,2,3,4], [3,4], Res)* дает *Res*=[1,2].**

Пример 2. **Согласование ЦУ *differ_sets ([1,2,3,4], [], Res)* дает *Res*=[1,2,3,4].**

***Таким образом*, ЦУ *differ_sets (X, Y, Z)* доказуемо, если *Z* есть разность множеств *X* и *Y*.**

Дополнение до пересечения.

Дополнением до пересечения для множеств X и Y является множество, содержащее элементы множеств X и Y, не вошедшие в их пересечение.

```
/* Симметрическая разность */
s_diff_sets ( [ ], Set2, Set2 ).
s_diff_sets ( [ H1 | T1 ], Set2, [H1 | Res ] ) :-
            not (member ( H1, Set2 )),
            s_diff_sets ( T1, Set2, Res ).
s_diff_sets ( [ H1 | T1 ], Set2, Res ) :-
            member ( H1, Set2 ),
            delete ( H1, Set2, Res1 ),
            s_diff_sets ( T1, Res1, Res ).
```

Пример.

Согласование ЦУ s_diff_sets ([1,2,3,4], [3,4,5,6], Res) дает *Res=[1,2,5,6]*.

Таким образом, ЦУ *s_diff_sets (X, Y, Z)* доказуемо, если *Z* есть дополнение до пересечения множеств *X* и *Y*.

Литература.

Клоксин У., Меллиш К. Программирование на языке Пролог : Пер. с англ. - М.: Мир, 1987. С. 174-176