Применение логического программирования для решения задач обработки естественного языка на ЭВМ.

Лекция 8.

Специальности : 230105, 010501

Проблема общения на естественном языке.

Под Естественным Языком (ЕЯ) в лингвистике понимается любой язык общения между людьми. Под естественностью некоторого языка понимается наличие синонимии и омонимии слов и словосочетаний, а также свободный порядок слов в предложении.

Целью исследований в области обработки естественного языка является построение программных средств, обеспечивающих восприятие и генерацию ЕЯ-текстов. При этом текст рассматривается как дискретное представление ЕЯ-высказывания, подлежащее восприятию (представление в том виде, в котором текстовый материал вводится с клавиатуры или сканирующего устройства). Отдельную область исследований составляет звуковое распознавание и синтез звучащей речи как перевод речевого сигнала в текстовое представление и наоборот.

Виды анализа ЕЯ-информации.

Анализ ключевых слов - метод анализа ЕЯ-высказываний на предмет наличия ключевых слов, которые становятся значениями объектов предикатов. При этом компьютер одинаково реагирует на различные варианты входного текста, наличие грамматической правильности предложений не является обязательным, роль играет лишь наличие ключевых слов. Применение - построение ЕЯ-интерфейсов к БД.

Грамматический анализ : контекстно-свободный и контекстно-зависимый. Контекстно-Свободный (КС) анализ - ЕЯ-фразы классифицируются в зависимости от их внутренней структуры вне зависимости от контекста в соответствии с грамматическими правилами, задающими порядок следования допустимых языком символов (слов). Здесь следует отметить синтаксический анализ предложений.

Прагматический анализ - наиболее сложный, связан с изучением смысла предложения в связи с внеязыковой действительностью.

Идентификация ключевых слов.

При разработке ЕЯ-интерфейса к БД с ключевым словом связывается либо характер действия программы (что хочет пользователь), либо значение некоторого атрибута искомого объекта. Например, при поиске по БД предприятий таким атрибутом может быть торговая марка (Стародусприбор, Новтрак, Квант, Планета), а ключевыми словами, определяющими действия - “Найти”,”Выдать”,”Вывести”,”Выход”. Процедура анализа входного текста с целью выделения ключей состоит из трех шагов :

Ввод команды как строки символов.

Преобразование введенной строки в список слов.

Идентификация ключевых слов.

Создание списка ключевых слов.

Для преобразования строки в список слов требуется написание правила, которое преобразует строку в список. Для этой цели служит встроенный предикат fronttoken. Однако данный предикат предназначен для работы с предложениями на английском языке и не работает со строками, записанными на русском. Здесь программист должен написать свою процедуру разделения русского предложения на слова. Пример такой процедуры приводится в программе NL_DBASE.PRO (ЕЯ-интерфейс к БД предприятий Новгородской области).

Идентификация ключевых слов может быть реализована двумя способами. Первый состоит в задании всех ключевых слов в утверждениях БД. Тогда внутренние унификационные процедуры Турбо-Пролога смогут сопоставить элементы предложения со значениями, взятыми из БД. Второй способ основан на частоте применения определенных грамматических конструкций : ключевые слова определяются позицией, которую они занимают в предложении. В примере для БД предприятий в качестве ключевых берутся первое и последнее слово высказывания.

Использование программирования второго порядка для решения задачи поиска ключевых слов в контексте.

Нахождение ключевых слов в контексте связано с поиском всех вхождений множества ключевых слов в определенный текст и выделением контекстов, в которых они встречаются. Рассмотрим следующую задачу.

Дан список названий публикаций по заданной теме, представленной названием рубрики (например, “logic programming”) и набором ключевых слов (например : "algorithmic“, "debugging“, "logic“, "problem“, "program“, "programming“, "prolog“, "solving"). Требуется сформировать список всех вхождений в данные заголовки множества ключевых слов вместе с их контекстами. Контекст описывается как вращение заголовка, конец которого отмечен вертикальной чертой. Предполагается, что каждый заголовок представляется списком слов.

Рассмотрим формирование интересующего нас списка с помощью квантификации по всем возможным заголовкам. Для этой цели воспользуемся комбинацией поиска всех решений для заданной цели с помощью встроенного предиката findall и возможностями Visual Prolog’a по работе с предикатами в качестве аргументов (предикатными доменами и предикатными значениями).

<u>Примечание.</u> Ввиду того, что Visual Prolog (как и Turbo Prolog) является типизированным языком, в полной мере работа с предикатами высших порядков в нем реализована быть не может.

Формирование списка вхождений ключевых слов в предложения заданного текста.

```
domains

   slist = string*
   list_of_slist = slist*
   pred_dom1 = nondeterm (slist, list_of_slist) - (o,i)
   pred_dom2 = nondeterm (slist, slist) - (i,o)

predicates

   member:pred_dom1
   keyword (string)
   nondeterm append (slist, slist, slist)
   rotate_and_filter:pred_dom2
   nondeterm set_of (slist, slist, list_of_slist, pred_dom1, pred_dom2)
   kwic (list_of_slist, list_of_slist)
   titles (string, list_of_slist)
   test_kwic (string, list_of_slist)

clauses

% Словарь ключевых слов

   keyword("algorithmic").
   keyword("debugging").
   keyword("logic").
   keyword("problem").
   keyword("program").
   keyword("programming").
   keyword("prolog").
   keyword("solving").
```

Формирование списка вхождений ключевых слов в предложения заданного текста (продолжение).

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

   append ([], L, L).
   append ([H|T], L, [H|T1]) : - append (T, L, T1).

% Недетерминированное описание вращения списка слов

   rotate_and_filter(Xs, Ys) : -
      append (As, [Key|Bs], Xs), keyword (Key), append ([Key, "|"|Bs], As, Ys).

   set_of(Ys,Xs,Titles,Pred1,Pred2):-Pred1(Xs,Titles),Pred2(Xs,Ys).

   kwic(Titles, KWTitles) : - findall(Ys, set_of(Ys, Xs, Titles, member, rotate_and_filter), KWTitles).

% Список названий книг по заданной теме. Тема соответствует первому аргументу предиката titles
% Названия книг представляются списками слов.

   titles("logic programming",[["logic","for","programming","solving"],
                               ["logic","programming"],
                               ["algorithmic","program","debugging"],
                               ["programming","in","prolog"]]).

   test_kwic (Books, Kwic) : -
      titles (Books, Titles), kwic (Titles, Kwic).

goal

   test_kwic (Books, Kwic).
```

Пример формирования списка вхождений ключевых слов для заданного списка заголовков.

Исходный список заголовков публикаций :

```
[["logic","for","programming","solving"],
  ["logic","programming"],
  ["algorithmic","program","debugging"],
  ["programming","in","prolog"]]
```

Заданный набор ключевых слов :

```
keyword("algorithmic").
keyword("debugging").
keyword("logic").
keyword("problem").
keyword("program").
keyword("programming").
keyword("prolog").
keyword("solving").
```

Список вхождений с учетом контекстов :

```
[["logic","|","for","programming","solving"],["programming","|","solving","logic","for"],["solving","|","logic","for","programming"],["logic","|","programming"],["programming","|","logic"],["algorithmic","|","program","debugging"],["program","|","debugging","algorithmic"],["debugging","|","algorithmic","program"],["programming","|","in","prolog"],["prolog","|","programming","in"]]
```

Грамаматический разбор предложений .

Ряд задач обработки ЕЯ требует анализа лексико-синтаксической структуры введенного предложения. К числу таких задач относятся машинный перевод, установление семантической эквивалентности высказываний, управление роботами. Анализ заключается в выделении подлежащего, сказуемого, дополнений, определений, обстоятельств и порядка следования членов предложения. Если выясняется, что структура предложения корректна, значения составляющих предложение слов могут послужить программе сигналом предпринять определенные действия.

Анализ предложения включает два этапа : лексический и синтаксический. На этапе лексического анализа определяется принадлежность слов предложения занесенному в память словарю. На этапе синтаксического анализа строится модель предложения в соответствии с описанной посредством КС-грамматики синтаксической структуры фраз рассматриваемого ЕЯ. Пример использования КС-грамматики для анализа простых распространенных предложений русского языка с возможным наличием однородных подлежащих и определений приводится в программе CFG_ANAL.PRO.

Реализация КС-грамматики для синтаксического анализа фраз русского языка.

```
/* Применение КС-грамматики для синтаксического
   анализа фраз русского языка :
   Простое распространенное предложение с однородными членами. */

domains

    str_list = string*
    char_list = char*
    sentence = sentence (subject_phrase, predicate_phrase)
    subject_phrase = subject_phrase (def_list, subject);
                homog_subject_phrase (subject_list)
    predicate_phrase = predicate_phrase (predicate, addition)
    subject = noun (string);
                pronoun(string)
    particle = particle (string)
    verb = verb (string)
    predicate = verb (string);
                  predicate (particle, verb)
    def_list = definition*
    subject_list = subject*
    definition = adjective (string);
                  participle (string)
    addition = noun (string);
                 pronoun (string)

database

    d_noun (string)
    d_pronoun (string)
    d_verb (string)
    d_adjective (string)
    d_participle (string)
    d_particle (string)
```

Реализация КС-грамматики для синтаксического анализа фраз русского языка (раздел clauses, описание основных правил).

```
/* Построение модели предложения */

   sentex (Tklist, sentence (Subject_phrase, Predicate_phrase)) : -
          subj_phrase (Tklist,Tklist1, Subject_phrase),
          pred_phrase (Tklist1, Predicate_phrase),!.
   sentex(_,_) : -
          write("Синтаксис предложения некорректен. "), nl,
          write("Нажмите Enter для продолжения..."),
          readchar(_),fail.

/* Построение модели группы подлежащего */

   subj_phrase(Tklist,Tklist1,subject_phrase (Def_list, noun (Subject))) : -
          def_list_make (Tklist, Def_list, [Subject | Tklist1]),
          d_noun (Subject).

   subj_phrase(Tklist,Tklist1,subject_phrase(Def_list,pronoun(Subject))) : -
          def_list_make (Tklist, Def_list, [Subject | Tklist1]),
          d_pronoun (Subject).

   subj_phrase (Tklist, Tklist1, homog_subject_phrase (Subject_list)) : -
          subj_list_make (Tklist, Subject_list, Tklist1).

/* Построение списка однородных определений */

   def_list_make ([Word | Tklist] , [ ], [Word | Tklist]) : -
          not (d_adjective (Word)), not (d_participle (Word)).

   def_list_make([Word | Tklist],[adjective(Word) | Res_list], Tklist1) : -
          d_adjective(Word),!,
          def_list_make(Tklist,Res_list,Tklist1).

   def_list_make([Word | Tklist], [participle (Word) | Res_list], Tklist1) : -
          d_participle (Word),!,
          def_list_make(Tklist, Res_list, Tklist1).
```

Реализация КС-грамматики для синтаксического анализа фраз русского языка (продолжение).

```
/* Построение списка однородных подлежащих */

   subj_list_make([Word | Tklist],[noun(Word) | Res_list], Tklist1) : -
          d_noun(Word),!,
          subj_list_make(Tklist,Res_list,Tklist1).
   subj_list_make([Word | Tklist], [pronoun(Word) | Res_list],Tklist1):-
          d_pronoun(Word),!,
          subj_list_make(Tklist,Res_list,Tklist1).
   subj_list_make([Word|Tklist],[],[Word|Tklist]):-
          not (d_noun (Word)),not ( d_pronoun(Word)).

/* Построение модели группы сказуемого */

   pred_phrase([Verb|[Addition]],predicate_phrase(verb(Verb),noun(Addition))):-
          d_verb (Verb), d_noun (Addition).

pred_phrase([Verb|[Addition]],predicate_phrase(verb(Verb),pronoun(Addition))):-
          d_verb (Verb), d_pronoun (Addition).

   pred_phrase([Particle|[Verb|[Addition]]],
             predicate_phrase(predicate(particle(Particle),verb(Verb)),noun(Addition))) : -
          d_verb (Verb), d_particle (Particle),
          d_noun (Addition).

   pred_phrase([Particle|[Verb|[Addition]]],
            predicate_phrase(predicate(particle(Particle),verb(Verb)),pronoun(Addition))) : -
          d_verb (Verb), d_particle (Particle),
          d_pronoun (Addition).
```

Модель языка как преобразователя “Смысл⇔Текст”.

В рамках теоретического подхода “Смысл⇔Текст” Естественный Язык (ЕЯ) рассматривается как особого рода преобразователь, который выполняет переработку заданных смыслов в соответствующие им тексты и заданных текстов в соответствующие им смыслы. Причем предполагается многозначное отображение множества текстов на множество смыслов.

Под текстом в рассматриваемом подходе понимается особый конструкт, являющийся дискретным представлением звучащей (устной) речи как последовательности непрерывных акустических сигналов.

Смысл, как и текст, представляет собой особый конструкт, но еще более сложный и далекий от наблюдения. В подходе “Смысл⇔Текст” под смыслом понимается информация, подлежащая передаче и восприятию. Причем относительно понятия смысла вводится понятие равнозначности текстов как эквивалентности (с точки зрения носителя данного ЕЯ в пределах некоторой оговоренной точности) означаемых. Тогда смысл представляется как инвариант синонимических преобразований одних равнозначных текстов в другие.

Под самой моделью языка понимается некоторый гипотетический преобразователь, который существует в сознании носителя ЕЯ и устанавливает соответствие между смыслами и текстами данного ЕЯ.

Основные аспекты модели “Смысл⇔Текст”.

Руководствуясь идеями Н.Хомского по формальным грамматикам как исчислениям, в модели “Смысл⇔Текст” следует различать :

–“Лингвистическая часть”, в которой собраны сведения о конкретном ЕЯ (то есть соответствия “смыслы⇔тексты”) и которая представляет собой исчисление, то есть набор предписаний (что можно/нельзя и что нужно делать).

–“Алгоритмическая часть”, в которой содержится описание механизма (или процедур) использования сведений о ЕЯ, представляемых “лингвистической частью”.

Модель “Смысл⇔Текст” с формальной точки зрения есть не порождающее, а преобразующее устройство – не генератор текстов, а транслятор смыслов в тексты и обратно.

Следует отметить, что в формальной грамматике (по Н.Хомскому) *алгоритм использования грамматики* для порождения или распознавания фраз *должен задаваться отдельно от нее* !

Уровни представления высказываний и строение модели “Смысл⇔Текст”.

Понятие смысла, которое используется в подходе “Смысл⇔Текст”, предполагает наличие более чем одного текста, которые несут заданный смысл (явление синонимии). При этом возможна также и омонимия – наличие более чем одного смысла, которые соответствуют заданному тексту. В силу наличия в ЕЯ указанных явлений прямой одношаговый переход от смыслов непосредственно к текстам (и наоборот) оказывается практически крайне затруднительным и запутанным. С целью решения указанной проблемы в модели “Смысл⇔Текст” вводится ряд уровней промежуточных представлений ЕЯ-высказываний при переходе от смысла к тексту :

–Семантический – запись смысла без расчленения на фразы;

–Синтаксический и морфологический уровни – с подразделением на поверхностный и глубинный подуровень;

–фонологический – здесь для фразы задается последовательность символов-фонем и просодем (минимальных значащих единиц, задающих интонацию фразы);

–Фонетический – здесь для фразы задается последовательность символов, представляющих звуки и просодии (ударения, паузы и т.п.).

n-арные деревья.

Задачи обработки информации на естественном языке, требующие формального описания преобразований синтаксических структур, имеют дело с помеченными деревьями арностью в общем случае больше 2. Развивая понятие дерева как многодоменной структуры, введем в рассмотрение n-арное дерево как дерево с отсутствием ограничений* на число дочерних поддеревьев узла (дочерние поддеревья задаются списком).

***Примечание. Для представления структуры фраз естественных языков используются не произвольные деревья, а деревья более специальных типов, задаваемых ограничениями на характер ветвления и на размещение пометок на ветвях. В частности, вводится следующее ограничение : для каждой пометки a_i указывается такое число n_i, что из любого узла исходит не более n_i ветвей с пометкой a_i (таким образом, из каждого узла исходит не более $n_1+n_2+ \ldots +n_k$ ветвей, где k - общее число разных пометок).**

Глубинные синтаксические структуры как пример n-арных деревьев.

Согласно данному И.А.Мельчуком теоретическому описанию, информация узла дерева глубинного синтаксиса включает в себя :

- **lex_in - лексическая часть,**
- **gram_in - грамматическая часть,**
- **arrow_label - пометка входящей ветви,**
- **composition_label - композиционная метка узла.**

Описание лексической части узла :

- **c0 - ключевое слово лексической замены,**
- **lex_fun - список лексических функций.**

Описание грамматической части узла :

- **part_of_speech - символьный атом, обозначающий часть речи,**
- **list_semant_categ - список семантически обусловленных словоизменительных категорий.**

Для описания информации узла введем домен dss_node, имя которого будет именем составного объекта, образованного при помощи функтора node и включающего в качестве имен объектов атрибуты информационного наполнения узла.

Описание дерева глубинного синтаксиса в Пролог-программе.

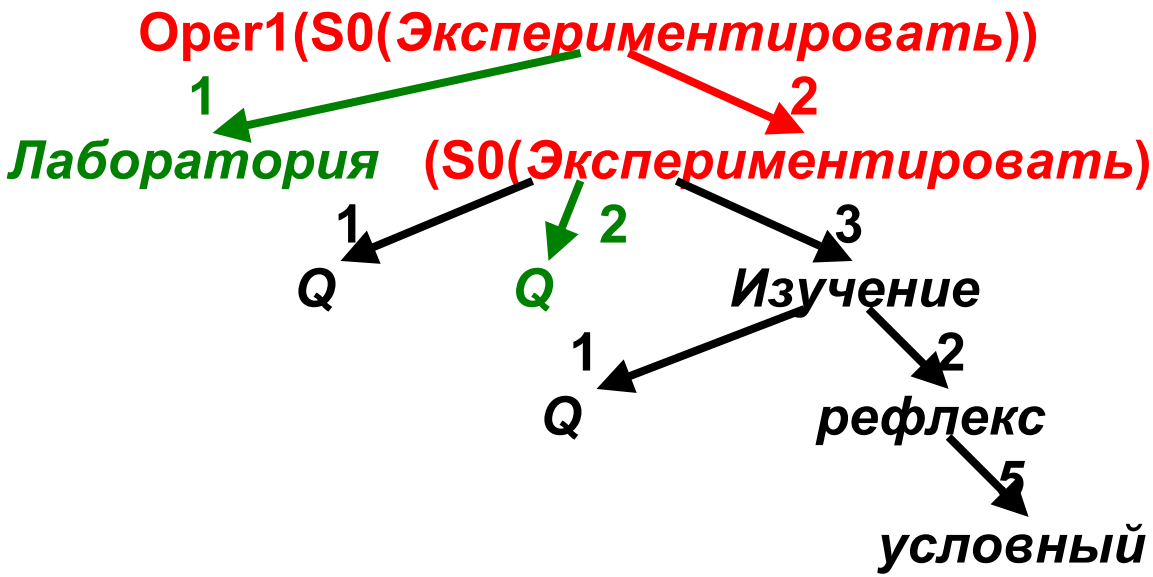
Само дерево может быть задано непосредственно в программе с помощью предиката dss.

```
domains
  lex_in=lex_in(c0,lex_fun).
      c0=symbol.
      lex_fun=symbol*.
  gram_in=gram_in(part_of_speech,list_semant_categ).
      part_of_speech=symbol.
      list_semant_categ=symbol*.
  dss_node=node(lex_in,gram_in,arrow_label,composition_label).
      composition_label=symbol.
      arrow_label=integer.
  t_dss_tree=t_dss_tree(dss_node,t_dss_tree_list).
  t_dss_tree_list=t_dss_tree*.
predicates
  dss(t_dss_tree).
  print_forest_dss(t_dss_tree_list).
  print_dss(t_dss_tree).
```

Пример дерева глубинного синтаксиса для простого распространенного предложения русского языка.

Предложение : ***Лаборатория провела эксперименты по изучению условных рефлексов.***

Глубинная синтаксическая структура предложения :



Обозначения :

Поддерево, заменяемое лексическим правилом*

Поддерево, заменяемое синтаксическим правилом*

***Примечание. Здесь имеются в виду лексические и синтаксические правила синонимических преобразований деревьев глубинного синтаксиса теории языка как преобразователя “Смысл⇔Текст”.**

Описание дерева с помощью предиката dss.

```
dss(t_dss_tree(node(lex_in("Экспериментировать",["S0","Oper1"]),
    gram_in("V",["сов_вид","прош_вр","изъявит_накл"]),0,"empty"),
      [t_dss_tree(node(lex_in("Лаборатория",[]),
                              gram_in("S",["ед_ч"]),1,"empty"),[]),
       t_dss_tree(node(lex_in("Экспериментировать",["S0"]),
                              gram_in("S",["мн_ч"]),2,"empty"),
        [t_dss_tree(node(lex_in("Q",[]),
                               gram_in("S",["empty"]),1,"empty"),[]),
         t_dss_tree(node(lex_in("Изучение",[]),
                               gram_in("S",["по","ед_ч"]),3,"empty"),
          [t_dss_tree(node(lex_in("Рефлекс",[]),
                                gram_in("S",["мн_ч"]),2,"empty"),
           [t_dss_tree(node(lex_in("Условный",[]),
                                 gram_in("A",["мн_ч"]),5,"empty"),[])]),
         t_dss_tree(node(lex_in("Q",[]),gram_in("empty",[]),1,"empty"),[])]),
        t_dss_tree(node(lex_in("Q",[]),
                               gram_in("S",["на","empty"]),2,"empty"),[])
       ])
      ])).
```

Организация рекурсивной обработки n-арного дерева.

Для организации рекурсивной обработки каждой из n ветвей по аналогии с двойной рукурсией для бинарного дерева используется рекурсивная обработка леса дочерних поддеревьев каждого узла с помощью вспомогательных правил.

```
/* Пример : печать дерева глубинного синтаксиса */
print_dss(DSS_tree):-
                print_forest_dss([DSS_tree|[]]).
print_forest_dss([t_dss_tree(node(Lex_in,Gram_in,
                                Arrow_label,Composition_label),[])|[]]):-
                write(Lex_in," ",Gram_in," ",Arrow_label," ",Composition_label," ").
print_forest_dss([t_dss_tree(node(Lex_in,Gram_in,
                                Arrow_label,Composition_label),Child_forest)|[]]):-
                write(Lex_in," ",Gram_in," ",
                                Arrow_label," ",Composition_label," "),nl,
                print_forest_dss(Child_forest).
print_forest_dss([t_dss_tree(node(Lex_in,Gram_in,
                    Arrow_label,Composition_label),[])|Cdr_forest_dss]):-
                write(Lex_in," ",Gram_in," ",
                    Arrow_label," ",Composition_label," "),
                print_forest_dss(Cdr_forest_dss).
  print_forest_dss([t_dss_tree(node(Lex_in,Gram_in,
          Arrow_label,Composition_label),Child_forest)|Cdr_forest_dss]):-
                write(Lex_in," ",Gram_in," ",Arrow_label," ",Composition_label," "),
                print_forest_dss(Cdr_forest_dss),nl,
                print_forest_dss(Child_forest).
```

Выводы.

Подход, связанный с анализом ключевых слов, вполне преемлем при создании простейших ЕЯ-интерфейсов к БД. При достаточно большом количестве ключевых слов данный подход позволяет находить в БД достаточно подробную информацию, причем ни размер БД, ни количество ключевых слов не влияет на время работы внутренних унификационных процедур Турбо-Пролога.

Для языков с фиксированным порядком членов предложения (например, английского) КС-грамматики вполне применимы для выполнения анализа подмножеств языка относительно некоторой предметной области.

Литература.

Ин Ц., Соломон Д. Использование Турбо-Пролога : Пер. с англ. - М.: Мир, 1993. С. 93-106, 230-287,452-490

И.А.Мельчук Опыт теории лингвистических моделей “смысл⇔текст” : Семантика, синтаксис / И.А.Мельчук.-[Переизд.]. // Школа “Языки русской культуры”. Москва, 1999. С. 141-176.

Гладкий А.В., Мельчук И.А. Грамматики деревьев. I. Опыт формализации преобразований синтаксических структур естественного языка, сб. “Информационные вопросы семиотики, лингвистики и автоматического перевода”, вып. 1. – М., 1971. – С. 16-41.

Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог : Пер. с англ. - М.: Мир, 1990. С. 54-57.

Доорс Дж. и др. Пролог - язык программирования будущего : Пер. с англ. - М.: Финансы и статистика, 1990. С. 52-57.

Маплас Дж. Реляционный язык Пролог и его применение : Пер. с англ. - М.: Наука, 1990. С. 120-126.

Клоксин У., Меллиш К. Программирование на языке Пролог : Пер. с англ. - М.: Мир, 1987. С. 63-65.