Функциональное и логическое программирование

Лекция 3

Глава 2 Логическое программирование. Основы языка Пролог

Логическое программирование базируется на убеждении, что не человека следует обучать мышлению в терминах операций компьютера, а компьютер должен выполнять инструкции, свойственные человеку. В чистом виде логическое программирование предполагает, что инструкции не задаются, а сведения о задаче формулируются в виде логических аксиом. Такое множество аксиом является альтернативой обычной программе. Подобная программа может выполняться при постановке задачи, формализованной в виде логического утверждения, подлежащего доказательству (*целевого утверждения*).

Идея использования логики исчисления предикатов I порядка в качестве основы языка программирования возникла давно, когда создавались многочисленные системы автоматического доказательства теорем и вопросно-ответные системы. В нашей стране была разработана система ПРИЗ, которая могла доказать любую теорему из школьного учебника геометрии.

PROLOG (programming in logic) - Колмероэ, Марсельский университет.

Группа занималась проблемой автоматического перевода с одного языка на другой.

Основа PROLOG - исчисления предикатов I порядка и метод резолюций.

PROLOG - язык для описания данных и логики их обработки. Программа на Прологе не является таковой в классическом понимании, поскольку не содержит явных управляющих конструкций типа условных операторов, операторов цикла и т.д. Она представляет собой модель фрагмента предметной области, о котором идет речь в задаче.

Реализации языка Пролог:

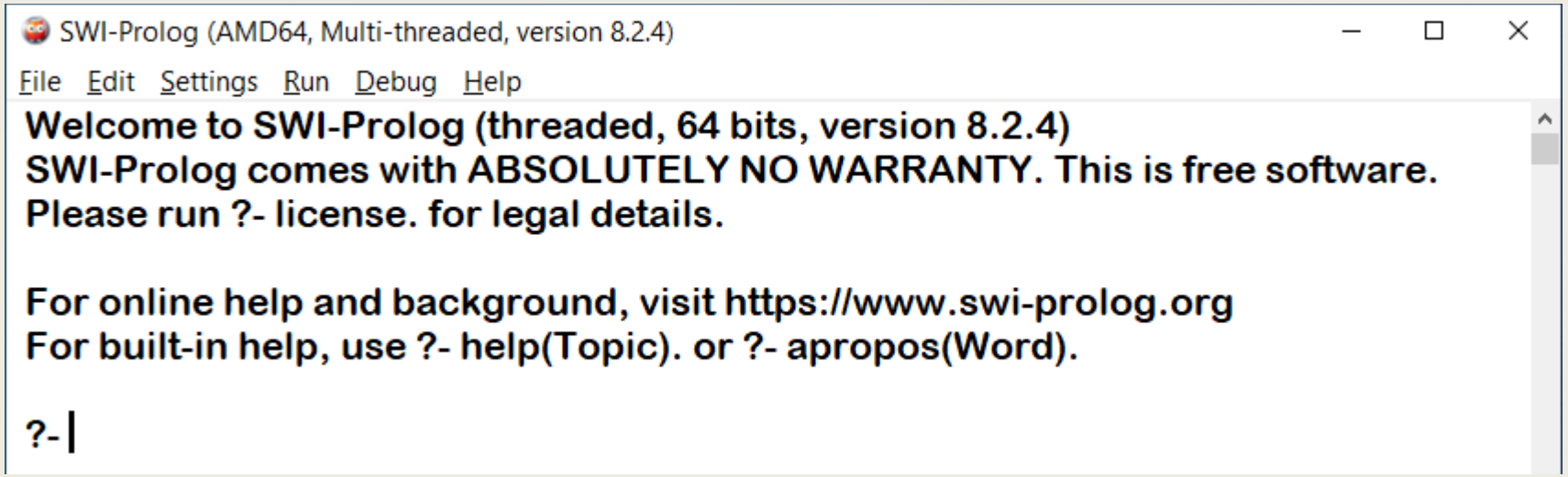
Wisdom Prolog, SWI Prolog, Turbo Prolog, Visual Prolog, Arity Prolog и т.д.

SWI-Prolog (SWI перевод с гол. социально-научная информатика) - Ян Вьелемакер, Амстердамский университет.

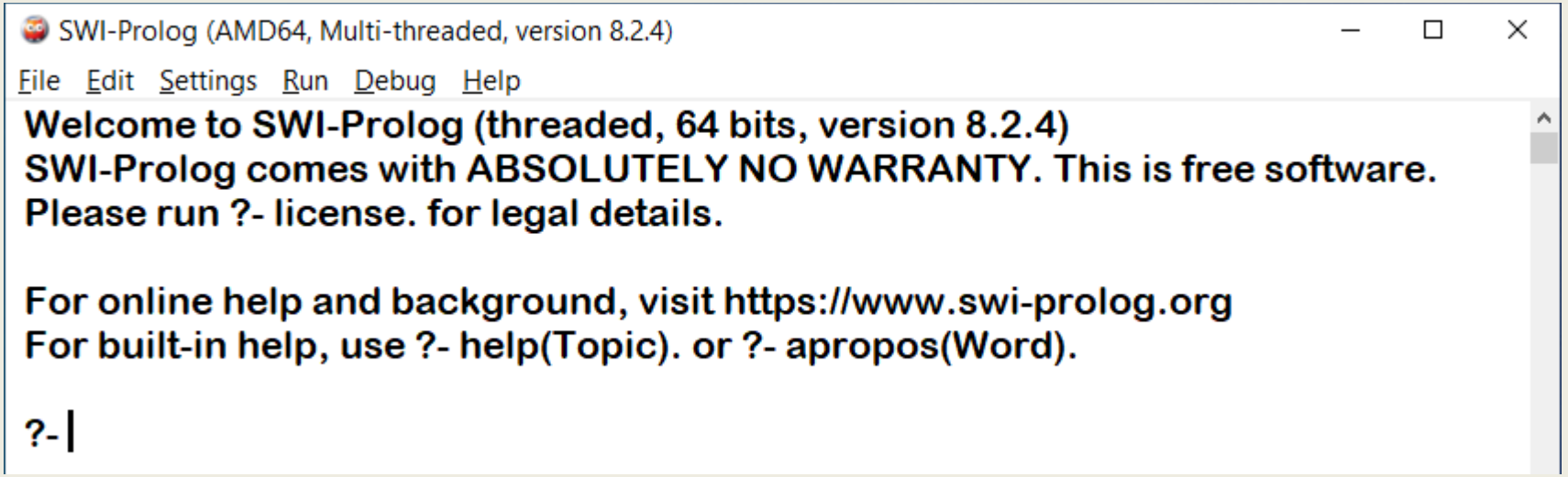
SWI-Prolog позволяет разрабатывать приложения любой направленности, включая Web-приложения и параллельные вычисления, но основным направлением использования является разработка экспертных систем, программ обработки естественного языка, обучающих программ, интеллектуальных игр и т.п. Это интерпретатор. Файлы, содержащие программы, написанные на языке SWI Prolog, имеют расширение pl.

Ссылка для скачивания.

https://www.swi-prolog.org/download/stable



Загрузка существующего файла интерпретатору File-Consult. После загрузки можно задавать вопросы или утверждение для доказательства после знака вопроса.

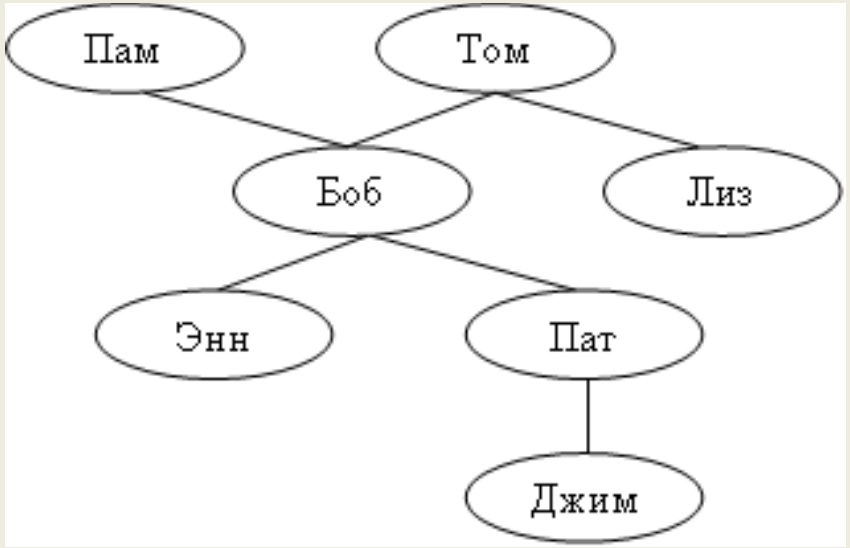


Если создается новый файл, то можно открыть окно редактора File-New (откроется новое окно).

Загрузить файл интерпретатору Compile-Compile buffer или выполнить пункт меню File-Consult, если файл не открыт в окне редактора.

2.1 Факты и правила

Пример 1: Написать программу, описывающую следующее дерево семейных отношений:



Вопросы к программе:

1. Боб является родителем Пат?

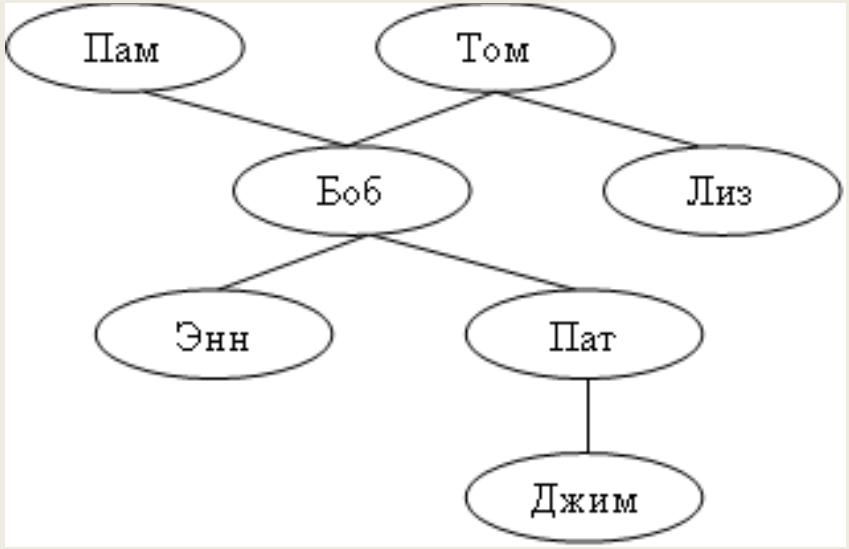
?- родитель(боб,пат).
true.

2. Пат – ребенок Лиз?

?- родитель(лиз,пат).
false.

3. Кто родители Лиз?

?- родитель(X,лиз).
X = том.



Вопросы к программе:

4. Кто дети Энн?

?- родитель(энн,Ч).
false.

5. Кто дети Боба?

?- родитель(боб,Ч).
Ч = энн ;
Ч = пат.

6. Есть ли дети у Пам?

?- родитель(пам,_).
true.

Имена людей в программе начинаются со строчных букв не случайно, в Прологе константы должны начинаться со строчных букв, переменные должны начинаться с заглавных букв или подчерка.

В конце каждого утверждения в программе и каждого вопроса к программе ставится точка.

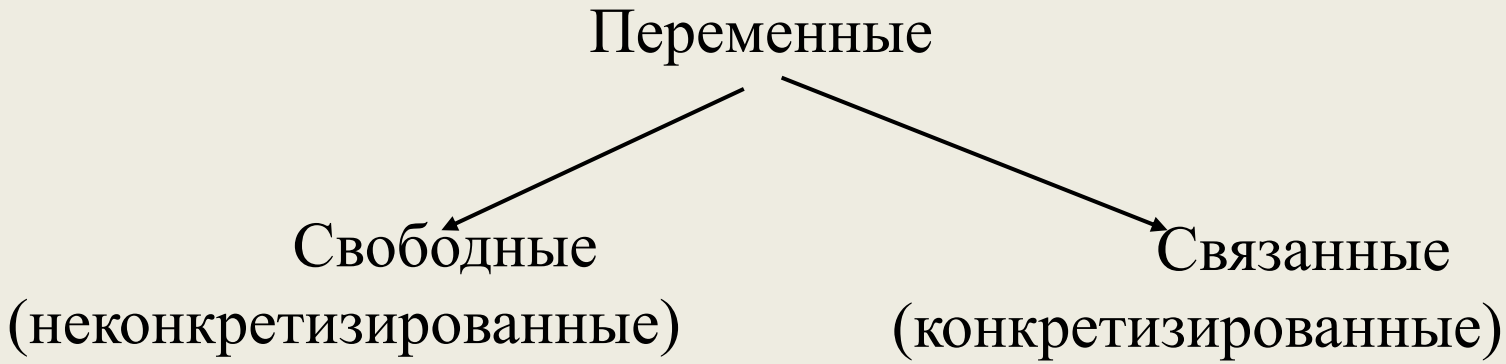
Если существует несколько значений переменных а ответе, то после очередного найденного решения Пролог ждет дальнейших указаний: либо продолжить поиск решений (тогда нажимаем ;), либо прекратить (тогда нажимаем . или Enter).

Вопросы могут быть простые и сложные (в качестве связки «и» при составлении сложного вопроса используется запятая).
Варианты ответов Пролог-системы на заданные вопросы:

- true
- false
- перечисление возможных значений переменных в ответе, при которых утверждение истинно. Если решение не единственно, то Пролог ожидает дальнейших указаний по продолжению поиска решений (продолжить или остановиться).

Программа на Прологе состоит из фактов и правил.

Факт – безусловное истинное утверждение
<имя предиката>(O_1,O_2,…,O_n).
O_i – конкретный объект (константа) или абстрактный объект (переменная).
В конце факта ставится точка!



Свободная переменная – переменная, которая еще не получила значения. Она не равняется ни нулю, ни пробелу; у нее вообще нет никакого значения. Такие переменные еще называют *неконкретизированными*.

Переменная, которая получила какое-то значение и оказалась связанной с определенным объектом, называется *связанной*. Если переменная была конкретизирована каким-то значением и ей сопоставлен некоторый объект, то эта переменная уже не может быть изменена в текущем предложении (типа X=1, X=2 или X=1, X=X+2).

Переменная обозначает объект, а не область памяти!

Анонимная переменная начинается с символа подчеркивания и предписывает интерпретатору проигнорировать значение этой переменной.

Если в предложении несколько анонимных переменных, то все они отличаются друг от друга, несмотря на то, что записаны с использованием одного и того же символа.

Правило – утверждение, которое истинно при выполнении некоторых условий, оно позволяет описывать новые отношения. Правило имеет вид:

<голова правила > :– <тело правила>.

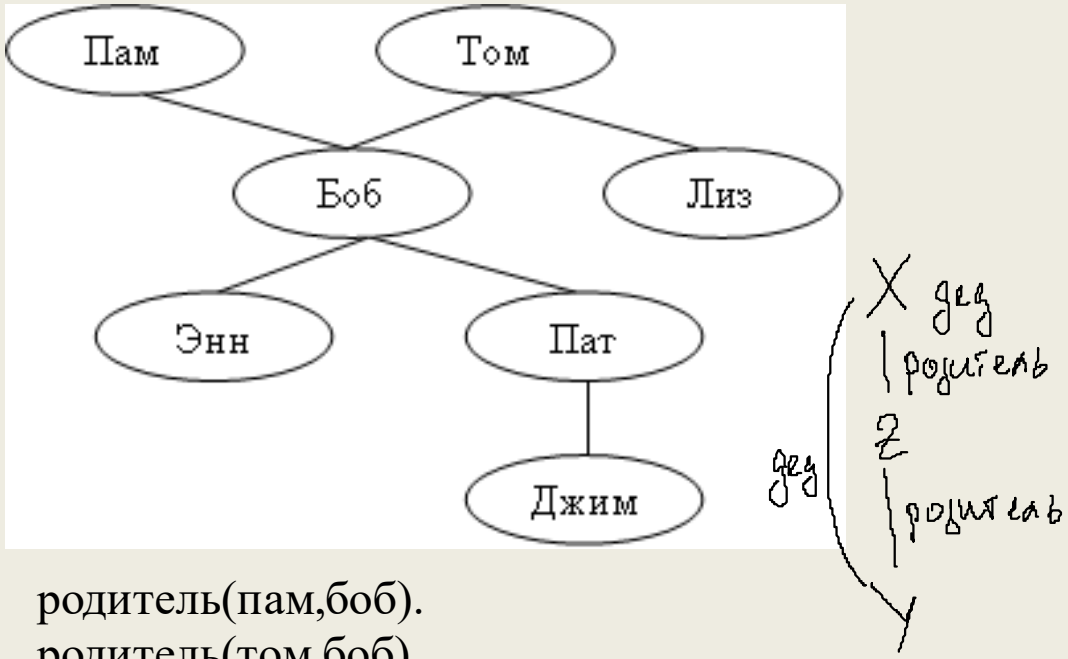
Головой правила является предикат, истинность которого следует установить.

Тело правила состоит из одного или нескольких предикатов, связанных логическими связками: конъюнкция (обозначается запятой), дизъюнкция (обозначается точкой с запятой) и отрицание (означается not или \+). Так же как в логических выражениях, порядок выполнения логических операций можно менять расстановкой скобок.

Можно в теле правила использовать разветвление вида:

(<условие>-><действие 1>;<действие 2>)

Пример 2: Добавим одноместное отношение мужчина (факты). Опишем новое двуместное отношение дед в виде правила.



родитель(пам,боб).
родитель(том,боб).
родитель(том,лиз).
родитель(боб,энн).
родитель(боб,пат).
родитель(пат,джим).
мужчина(том).
мужчина(боб).
мужчина(джим).
дед(X,Y):-мужчина(X),родитель(X,Z),родитель(Z,Y).

Вопросы к программе:

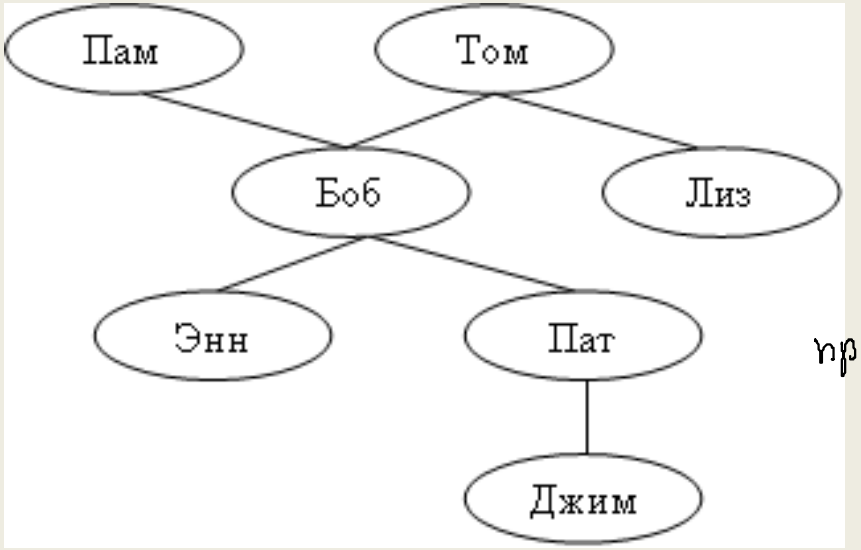
1. Кто дед Джима?

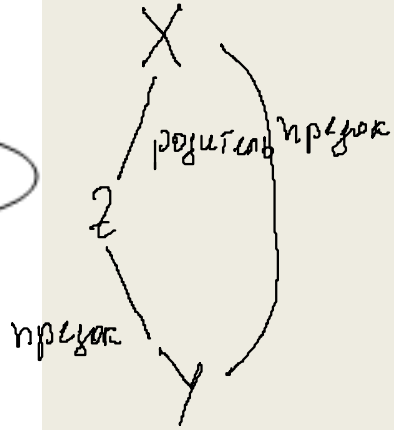
?- дед(Ч,джим).
Ч = боб ;
false.

2. Кто внуки Тома?

?- дед(том,Ч).
Ч = энн ;
Ч = пат ;
false.

Пример 3: Опишем новое двуместное отношение предок.





Вопрос к программе:
Кто предок Джима?

?- предок(X,джим).
X = пат ;
X = пам ;
X = том ;
X = боб ;
false.

родитель(пам,боб).
родитель(том,боб).
родитель(том,лиз).
родитель(боб,энн).
родитель(боб,пат).
родитель(пат,джим).
мужчина(том).
мужчина(боб).
мужчина(джим).
дед(X,Y):-мужчина(X),родитель(X,Z),родитель(Z,Y).
предок(X,Y):-родитель(X,Y).
предок(X,Y):-родитель(X,Z),предок(Z,Y).

2.2 Поиск решений Пролог-системой

Вопрос к системе – это последовательность, состоящая из одной или нескольких целей.

Для ответа на поставленный вопрос Пролог-система должна *достичь всех целей*, т.е. показать, что утверждения вопроса истинны в предположении, что все отношения программы истинны.

Если в вопросе имеются переменные, то система должна найти конкретные объекты, которые, будучи подставлены вместо переменных, обеспечат достижение цели. Если система не в состоянии вывести цель из имеющихся фактов и правил, то ответ должен быть отрицательный.

Факты и правила в программе соответствуют аксиомам, вопрос – теореме.

При поиске ответа на поставленный вопрос находится факт или правило для содержащегося в вопросе предиката и выполняется операция сопоставления (*унификации*) объектов предиката.

Операция унификации объектов успешна:

- сопоставляются две одинаковые константы;
- сопоставляется свободная переменная с константой (при этом свободная переменная становится означенной);
- сопоставляется связанная переменная с константой, равной значению переменной;
- сопоставляется свободная переменная с другой свободной переменной (переменные не получают значений, но становятся сцепленными, т.е. когда одна из них получит значение, то и вторая получит это же значение).

После успешного сопоставления все переменные получают значения и становятся связанными, а предикат считается успешно выполненным (если сопоставление выполнялось с фактом) или заменяется на тело правила (если сопоставление выполнялось с головой правила). Связанные переменные освобождаются, если цель достигнута или сопоставление неуспешно.

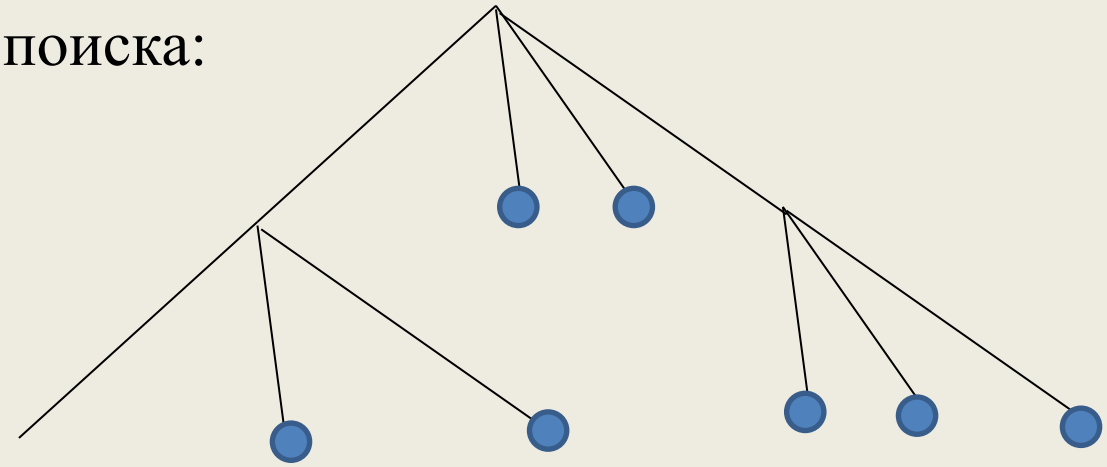
Процесс унификации похож на использование оператора =.

A=B может интерпретироваться как присваивание слева направо, справа налево, как сравнение.

Для достижения цели используется механизм отката.

При вычислении цели выполняется сопоставление с фактами и головами правил. Сопоставления выполняются слева направо.

Дерево поиска:



Выделены точки отката, которые Пролог запоминает для поиска альтернативных путей решения.

Если цель была неуспешна, то происходит откат к ближайшему указателю отката.

Если цель достигнута, но использовались не все указатели отката, то будет продолжен поиск решений.

Трассировка

Включение трассировки: trace.

Отказ от трассировки: notrace.

Слова, появляющиеся в окне трассировки:

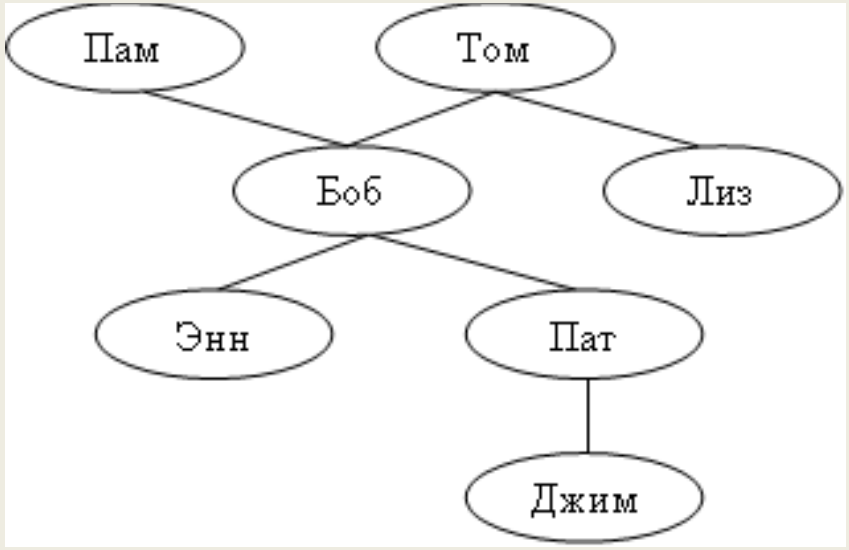
Call Указывается текущая цель

Exit Указывается цель, которая успешна.

Redo Возврат в отмеченную точку возврата для поиска альтернативного решения.

Fail Указанная цель, которая не была достигнута.

В круглых скобках указывается глубина в дереве поиска.



родитель(пам,боб).
родитель(том,боб).
родитель(том,лиз).
родитель(боб,энн).
родитель(боб,пат).
родитель(пат,джим).
мужчина(том).
мужчина(боб).
мужчина(джим).
дед(X,Y):-мужчина(X),родитель(X,Z),родитель(Z,Y).
предок(X,Y):-родитель(X,Y).
предок(X,Y):-родитель(X,Z),предок(Z,Y).

Пример:
Трассировка вопроса «Является ли Том предком Энн?»

?- trace.
true.

[trace] ?- предок(том,энн).
Call: (10) предок(том, энн) ? creep
Call: (11) родитель(том, энн) ? creep
Fail: (11) родитель(том, энн) ? creep
Redo: (10) предок(том, энн) ? creep
Call: (11) родитель(том, _13336) ? creep
Exit: (11) родитель(том, боб) ? creep
Call: (11) предок(боб, энн) ? creep
Call: (12) родитель(боб, энн) ? creep
Exit: (12) родитель(боб, энн) ? creep
Exit: (11) предок(боб, энн) ? creep
Exit: (10) предок(том, энн) ? creep
true .

2.3 Графический отладчик

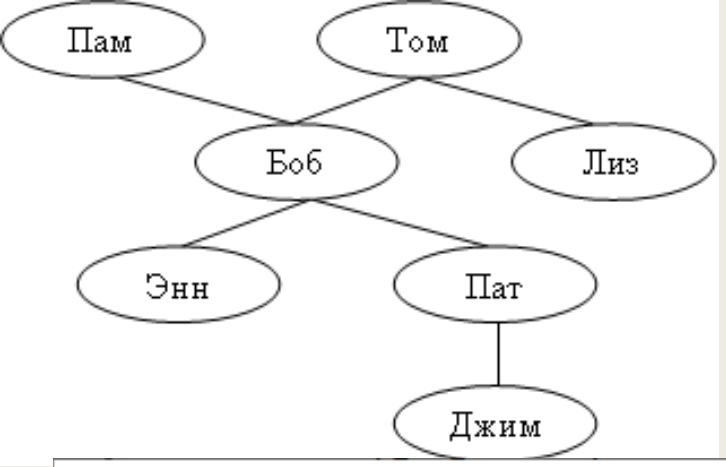
Включение графической трассировки: guitracer или через меню Debug-Graphical debugger.

Выключение графической трассировки: noguitracer.

После включения графической трассировки включаем трассировку:

?-trace, <имя предиката>.

Откроется дополнительное окно графического отладчика. Оно разделено на 3 части: верхнее левое окно показывает текущие значения переменных, верхнее правое окно показывает дерево вызовов, а нижнее – текст программы. Для пошагового выполнения следует нажимать значок стрелки, направленной вправо или пробел.



Пример:

?-trace, дед(том,X).

Tool Edit View Compile Help

Bindings

X = том

Call Stack

11 дед/2

```
родитель(пам,боб).
родитель(том,боб).
родитель(том,лиз).
родитель(боб,энн).
родитель(боб,пат).
родитель(пат,джим).
мужчина(том).
мужчина(боб).
мужчина(джим).
дед(X,Y):-мужчина(X),родитель(X,Z),родитель(Z,Y).
```

Call: дед/2

2.4 Некоторые операции в SWI-Prolog

=	Унификация (присваивание значения несвязанной переменной)
<, =<, >=, >	Арифметические (только для чисел) операции сравнения
=:=	Арифметическое равенство
=\=	Арифметическое неравенство
is	Вычисление арифметического выражения
@<, @=<, @>=, @>	Операции сравнения для констант и переменных любого типа (чисел, строк, списков и т.д.)
==	Равенство констант и переменных любого типа
\==	Неравенство констант и переменных любого типа
A mod B	Остаток от деления A на B
A//B	Целочисленное частное при делении A на B

Примеры вопросов и ответов Пролог-системы:

?- X=6+9.
X = 6+9.

?- X is 6+9.
X = 15.

?- X+7=7+X.
X = 7.

?- 4+7=7+4.
false.

?- 4+7=:=7+4.
true.

2.5 Предикаты ввода-вывода

read(A)	Чтение значения с клавиатуры в переменную A
write(A)	Вывод значения A на экран без перевода строки
writeln(A)	Вывод значения A на экран с переводом курсора в начало следующей строки
nl	Перевод курсора в начало следующей строки
format('<строка~w>',X)	<строка> <значение X>
format('<строка~w~w>', [X,Y])	<строка> <значение X> <значениеY>
format('<строка1~w\n строка2~w>',[X,Y])	<строка1> <значение X> <строка2> <значениеY>

Некоторые полезные сведения при работе с интерпретатором SWI-Prolog

При ожидании ввода выводится приглашение |:

Комментарии заключаются между /* и */

или % комментируется строка правее %.

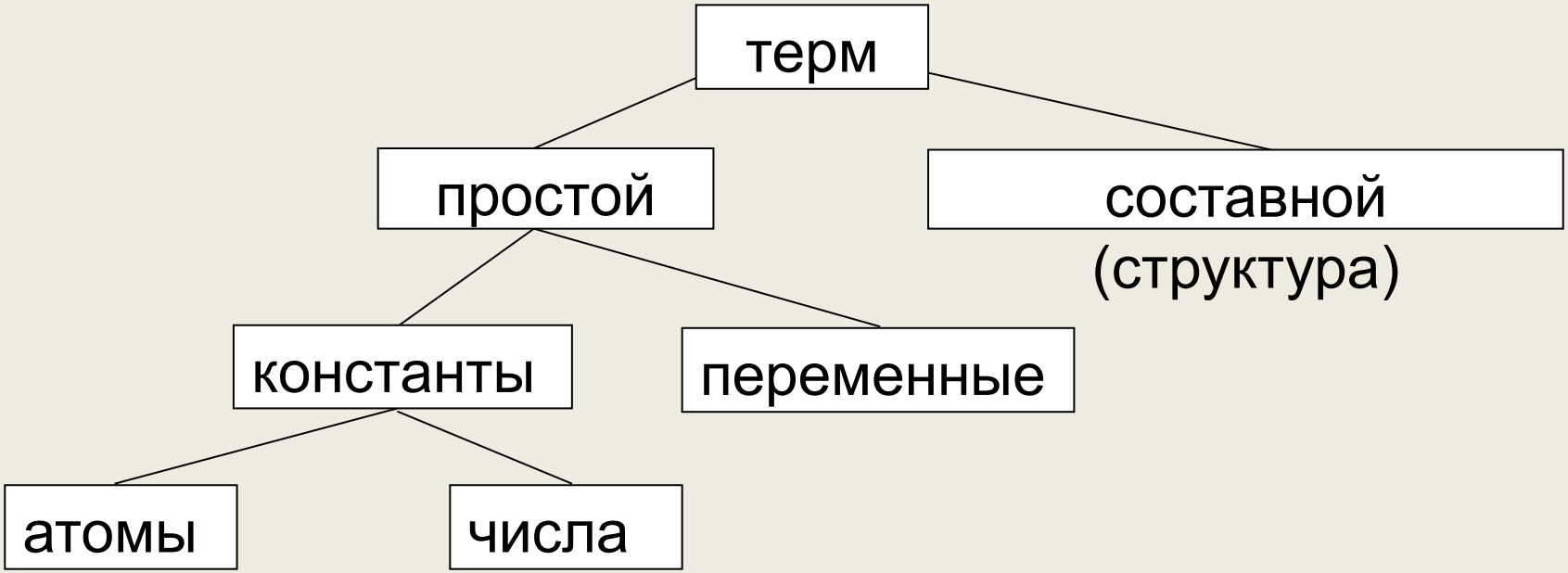
pwd. – текущая папка;

ls. – содержимое текущей папки;

cd(‘<путь к папке, слэши дублируются’>). – сменить текущую рабочую папку.

2.6 Структуры

В соответствии с теорией предикатов первого порядка единственная структура данных в логических программах - *термы*.



Атом - последовательность латинских букв, цифр, начинающаяся со строчной буквы, любая последовательность символов, заключенных в апострофы, или специальный символ.

Составной терм:

$$f(t_1,t_2,...,t_n),$$

где f - имя n-арного функтора, t_i – аргументы-термы.

Единообразие программ и данных также, как в Лиспе.

Примеры составных термов:

дата (число, месяц)

дата (число, Месяц, год)

Каждый функтор и предикат определяется двумя параметрами: именем и арностью (количеством аргументов).

В Пролог-программах допускается использование одного и того же предикатного (или функционального символа) с разным числом аргументов.

Предикаты для проверки типа терма:

var(Term) - свободная переменная

nonvar(Term) - несвободная (конкретизированная) переменная

number(Term) - целое или вещественное число

atom(Term) – атом

atomic(Term) - атом или число

ground(Term) -терм не содержит свободных переменных

2.7 Рекурсия

Рекурсия в Прологе может быть алгоритмическая и по данным. Основное отличие от Лиспа заключается в том, что возвращаемое значение должно находиться в аргументе предиката.

Передача параметров осуществляется по значению.

Пример 1:

Определим одноместный предикат, который заданное количество раз выводит на экран строку ****.

```
vivod(0).
vivod(N):-writeln('****'),N1 is N-1,vivod(N1).
```

```
?- vivod(6).
****
****
****
****
****
****
true .
```

Пример 2:

Определим одноместный предикат, вычисляющий n!.

Ввод n с клавиатуры и вывод результата будет осуществлять предикат goal.

```
goal:-writeln('N-?'),read(N),fact(N,P),format('~w!=~w',[N,P]).
fact(0,1).
fact(N,P):-N1 is N-1,fact(N1,P1),P is N*P1.
```

```
?- goal.
N-?
|: 6.
6!=720
true .
```