КОНЦЕПЦИЯ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Терминология логического программирования заимствована из логики. Логика познает принципы человеческого мышления. Логическое программирование – один из подходов к информатике, при котором в качестве языка высокого уровня используется логика предикатов первого порядка.

Суть логического подхода заключается в том, что машине в качестве программы предлагается не алгоритм, а формальное описание предметной области и решаемой проблемы (функции) в виде аксиоматической системы.

КОНЦЕПЦИЯ ЯЗЫКА ПРОЛОГ

Пролог является языком программирования, который обеспечивает решение задач, выраженных в терминах объектов и отношений между ними.

Программирование на языке Пролог состоит из следующих этапов:

1). Объявления некоторых фактов об объектах и отношениях между ними.

2). Определения некоторых правил об объектах и отношениях между ними.

3). Формулировки вопросов об объектах и отношениях между ними.

Программа состоит из предложений, которые могут быть фактами, правилами или вопросами.

ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Синтаксис логики предикатов	
Предложения на естественном языке	**Синтаксис логики предикатов**
Машина красивая	**красивая (Машина)**
Роза красная	**красная (роза)**
Мише нравится машина, если машина красивая	**нравится(миша, Машина) если красивая(Машина)**

БАЗА ЗНАНИЙ

База знаний

Факты

Правила (процедуры)

Факты в Прологе служат для описания конкретных данных и простейших сведений.

Естественный язык ***отношение - факт***

Зина – мама Вовы **мама(зина,вова)**

Миша – папа Вовы **папа(миша,вова)**

Форма записи факта :

<имя факта(предиката)>(аргумент [, аргумент],…).

Каждый факт в Прологе интерпретируется как некоторое истинное утверждение.

1). Все имена предикатов и аргументов должны начинаться со строчной латинской буквы.

2). Перечисление аргументов –через запятую.

3). Каждый факт должен заканчиваться точкой.

4). Аргументы определяются соответствующими типами.

5). Количество аргументов и вид отношений (направления отношений) определяются программистом и не меняются при выполнении программы.

ПРАВИЛА

Под правилами в Прологе понимаются наиболее общие утверждения об объектах и отношениях между ними.

Правила используются для описания процедур принятия решений и обработки данных.

Пролог-правило имеет вид фразовой формы :

заключение:-усл1, усл2, … ,услN.

Заключение(голова правила)=<имя правила>(аргумент [, аргумент],…).

Усл=вызов <факта>|<заключения>

Языки, подобные Прологу, считаются языками типа “если-то”: заключение истинно, если истинными являются условия, перечисленные в правой части. Правила позволяют вывести один факт из других фактов и|или заключений.

Правило - это заключение, для которого известно, что оно истинно, если одно или несколько других найденных заключений или фактов являются истинными.

СТАНДАРТНЫЕ ТИПЫ АРГУМЕНТОВ

Тип данных	Ключевое слово	Диапазон значений	Примеры использования
Символы	**char**	Все возможные символы	**‘a’, 'b', '#', 'B', '\13','%'**
Целые числа	**integer**	От -32768 до 32767	-63, 23, 2349, 32763
Действительные числа	**real**	От E-38 до E+37	42769, -8324, 360, 093, -1.25E23, 5.15E-9
Строки	**string**	Последовательность символов (не более 255)	**today**, "123", ***"пример строки"***
Символьная константа	**symbol**	1. Последовательность букв, цифр и подчеркиваний, начинающаяся с маленькой буквы (латинской или русской) или большой русской буквы. 2. Последовательность любых символов, заключенная в кавычки. Используется тогда, когда имя должно начинаться с большой латинской буквы или содержать пробелы.	***"телефонный номер"***, **alfa_beta_gamma**, **"Alfa_beta_gamma"**
Файлы	**file**	Допустимое имя файла. При операциях с файлами связывается с конкретными файлами или устройствами.	**mail.txt**, **BIRDS.DBA**

ВОПРОСЫ

Вопросы в Прологе служат для записи простых или сложносоставных **запросов** к базам знаний. Ответами на запросы к базам знаний могут быть логические значения **Yes (Да,** истина) или **No (Нет,** ложь) или список конкретных данных, отвечающих запросу.

С помощью запросов можно "спрашивать" базу данных о том, какие утверждения являются истинными. **Запрос называется *целью (goal)***.

Простой вопрос:

goal **<имя факта|правила>(<арг>[,<арг> ..])**

Сложный вопрос:

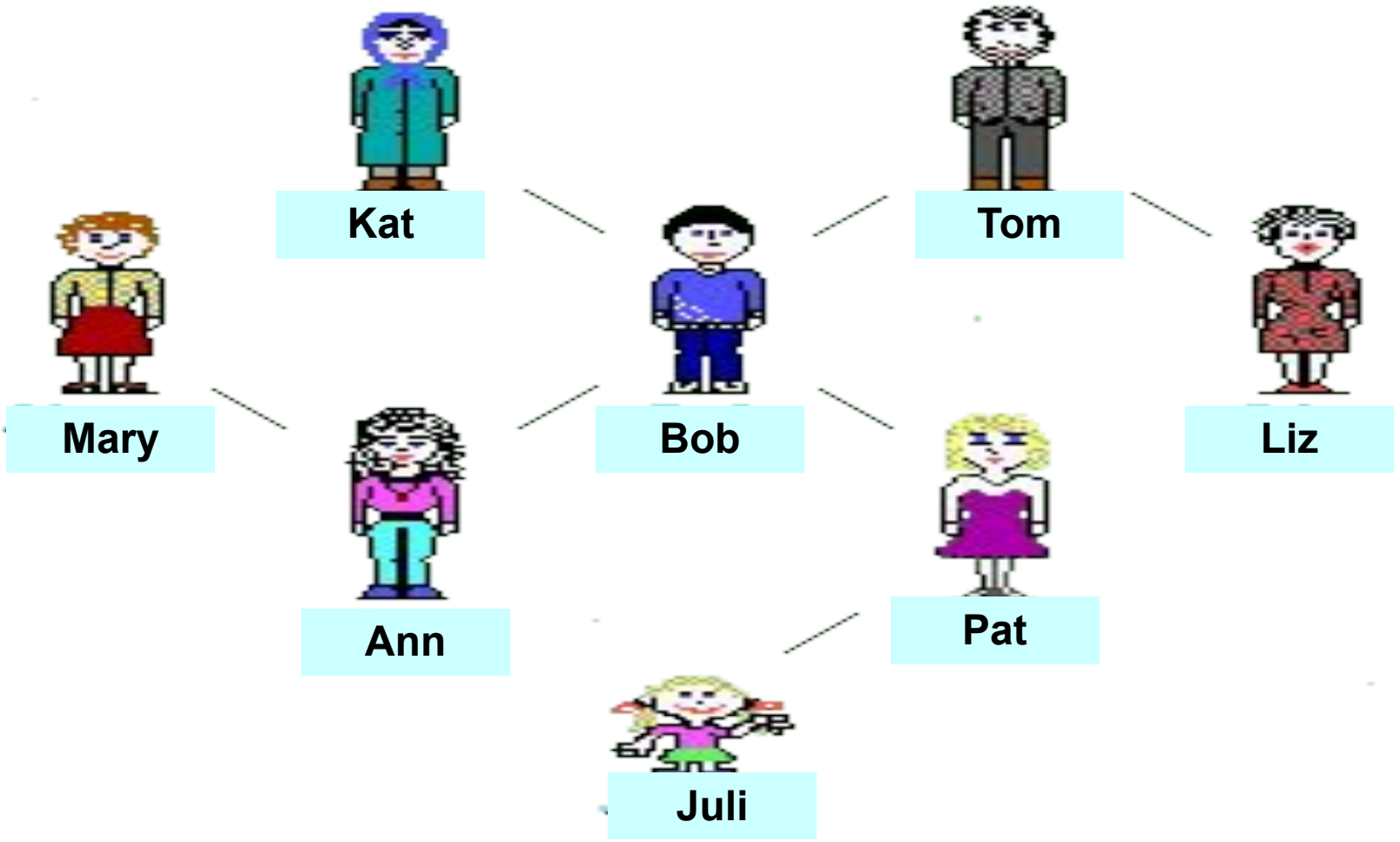
goal **<вопрос>[<,|;><вопрос> ..]**

Пролог включает механизм вывода, который основан на сопоставлении образцов. С помощью подбора ответов на запросы он извлекает хранящуюся в виде фактов и|правил (известную) информацию.

Пролог пытается проверить истинность гипотезы (другими словами - ответить на вопрос), запрашивая для этого информацию, о которой уже известно, что она истинна.

Прологовское знание о мире - это ограниченный набор фактов и|или правил, заданных в программе.

ПРИМЕР. РОДСТВЕННЫЕ ОТНОШЕНИЯ



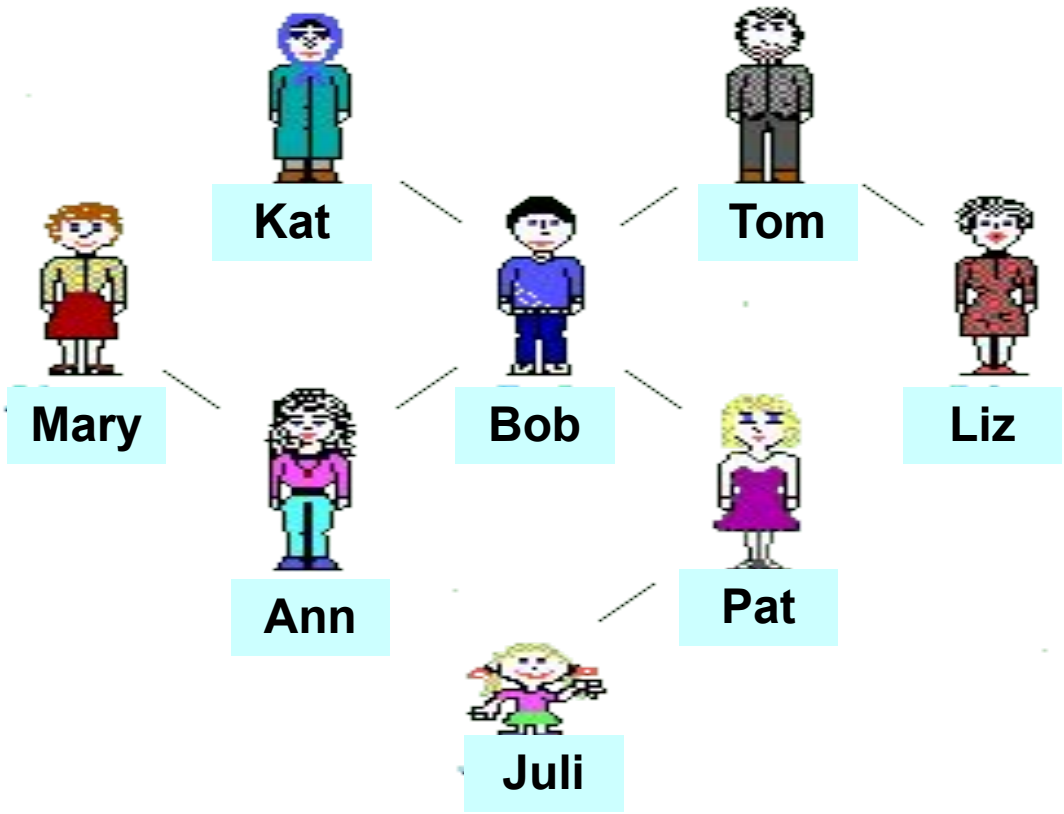
ЗНАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Отношение -родитель(parent) между объектами **Кто** , **Чей**

parent (<Кто>, <Чей>).

Факты:

parent (kat, bob).
parent (tom, bob).
parent (tom, liz).
parent (bob, ann).
parent (bob, pat).
parent (mary, ann).
parent (pat, juli).

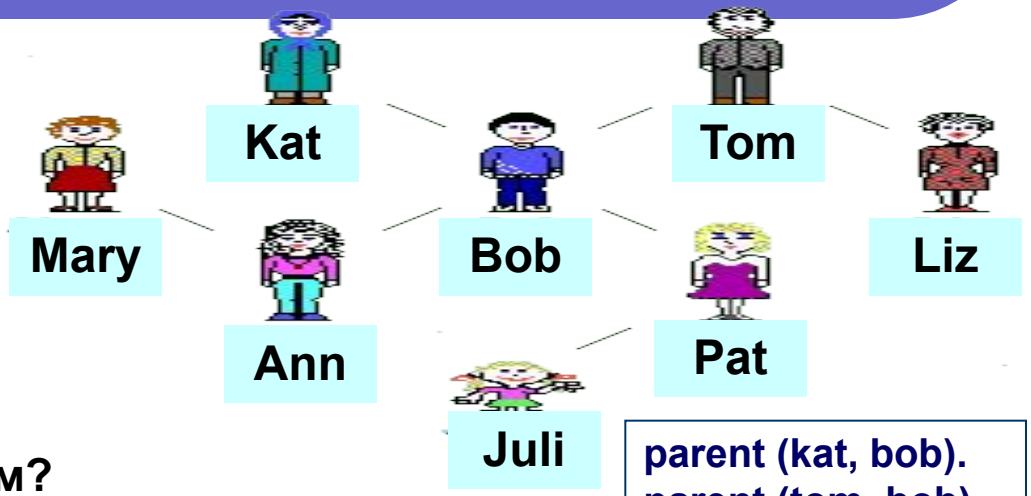


ВОПРОСЫ К БАЗЕ ЗНАНИЙ

goal parent (bob, pat).
yes
goal parent (bob,mary).
no

Переменные в запросах

кто родитель liz?
goal parent (X, liz).
X= tom
Кто является чьим родителем?
(Или найти такие X и Y, что X является родителем Y).
goal parent (X, Y).
X= kat, Y= bob
X= tom, Y= bob
и т.д.
Кто является родителем родителя juli?
goal parent (Y, juli), parent (X, Y). X=bob, Y=pat
Кто внуки тома?
goal parent (tom, Y), parent (Y, X). Y=bob, X=ann
Y=bob, X=pat



parent (kat, bob).
parent (tom, bob).
parent (tom, liz).
parent (bob, ann).
parent (bob, pat).
parent (mary, ann).
parent (pat, juli).

ПРАВИЛА И ФАКТЫ

Отношение child(ребенок) обратное к **parent(родитель)**

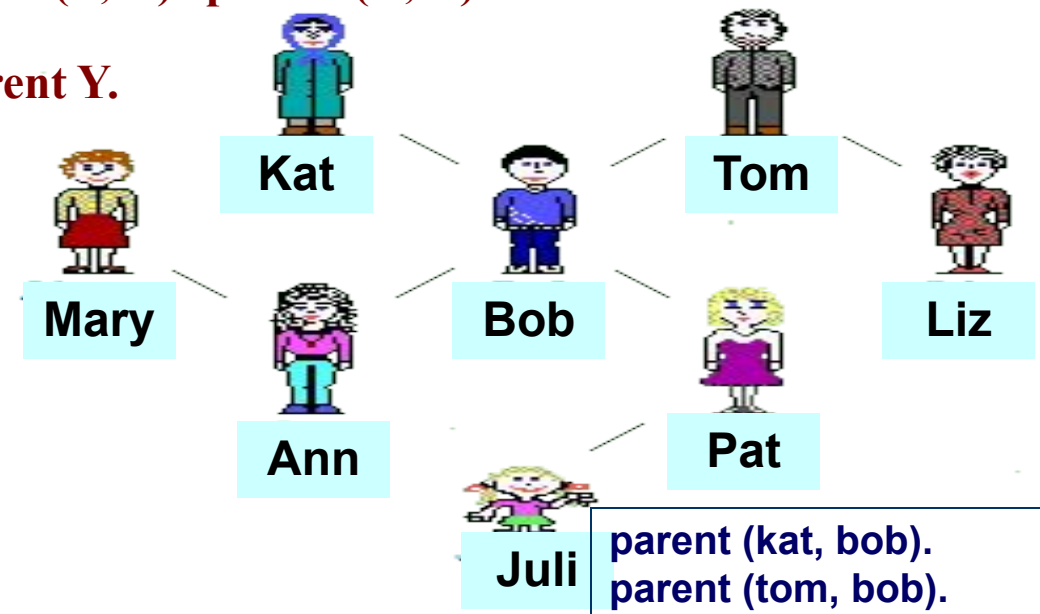
Утверждение- **правило child(Y, X):-parent (X, Y).**

Для всех Y и X

Y -child X, если X -parent Y.

goal child(liz, tom)

```
male(tom).
male(bob).
female(liz).
female(kat).
female(pat).
female(ann).
female(juli).
```




```
parent (kat, bob).
parent (tom, bob).
parent (tom, liz).
parent (bob, ann).
parent (bob, pat).
parent (mary, ann).
parent (pat, juli).
child(Y, X):-parent (X, Y).
```

Отношение mother(<Кто>,. <Чья>).

Для всех X и Y

X -mother Y, if X- parent Y и X -female.

mother(X, Y):-parent(X, Y), female(X).

ПРАВИЛА И ФАКТЫ

Отношение sister

Для любых X и Y

X sister Y, if
у X и Y есть общий родитель,
и X female

sister (X, Y):- parent(Z,X), parent(Z,Y),
female(X).

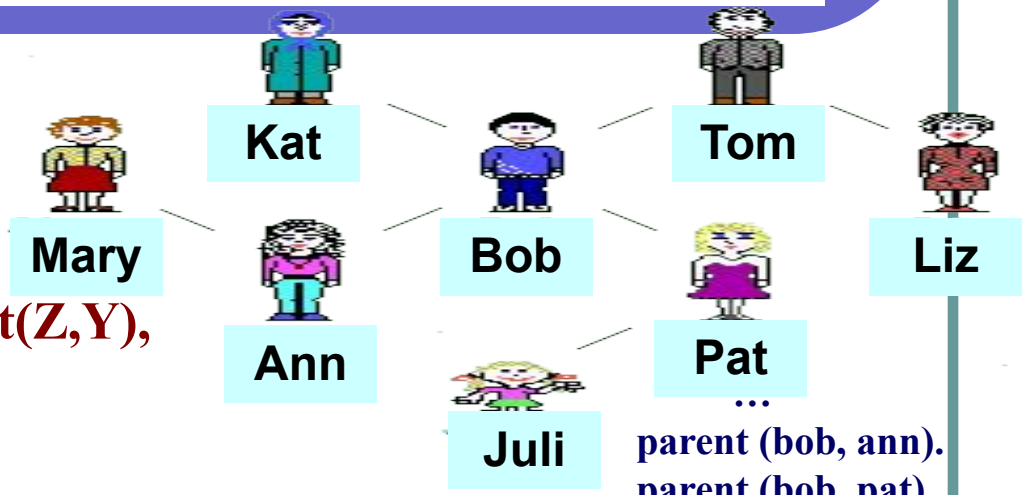
goal sister(pat, pat).

Yes (Почему?)

sister (X, Y):- parent(Z,X), parent(Z,Y), X<>Y, female(X).

Анонимная переменная _

goal sister(_, pat).



…
parent (bob, ann).
parent (bob, pat).
…
female(pat).
female(ann).

Переменные служат для обозначения объектов, значения которых меняются в ходе выполнения программы.

Имена переменных

- могут начинаться с прописной буквы;

- быть символом подчеркивания.

Область действия переменных - одно предложение (правило, запрос).

Одноименные переменные в разных предложениях могут иметь разные значения.

СТРУКТУРА ПРОГРАММЫ

domains
<*имя нового типа*>=<известный тип> …

predicates
<*список описаний используемых предикатов***> …**

clauses
<база знаний (факты|правила)>

Goal
<вопрос>

ПРОГРАММА. РОДСТВЕННЫЕ ОТНОШЕНИЯ

```
domains  name=symbol
predicates
nondeterm male(name) nondeterm female(name)
nondeterm parent(name, name)
nondeterm mother(name, name)
nondeterm sister(name, name)
clauses parent (kat, bob). parent (tom, bob). parent (tom, liz).
        parent (bob, ann).
        parent (bob, pat).
        parent (mary, ann).
        parent (pat, juli).
        male(tom). male(bob). female(juli).
        female(liz). female(kat).
        female(pat).
        female(ann).
sister (X, Y):- parent(Z,X), parent(Z,Y), X<>Y,
                female(X).
mother(X, Y):-parent(X, Y), female(X).

Goal mother (X, Y), write("mother ",X,'\n',Y,'\n').
     %mother (X, Y), write(X),nl,write(Y).
     %mother (X, Y), write(X,'\n',Y),nl,fail.
```

ОПИСАНИЕ ПРЕДИКАТОВ БАЗЫ ЗНАНИЙ

```
predicates
nondeterm male(symbol)
nondeterm female(symbol)
nondeterm parent(symbol, symbol)
nondeterm mother(symbol, symbol)
nondeterm sister(symbol, symbol)
```