Основные особенности языка Пролог.

Лекция 2.

Специальности : 230105, 010501

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.

Сферы применения Пролога.

- разработка быстрых прототипов прикладных программ;
- управление производственными процессами;
- создание динамических реляционных баз данных;
- перевод с одного языка на другой;
- создание естественно-языковых интерфейсов;
- реализация экспертных систем и оболочек экспертных систем;
- создание пакетов символьных вычислений;
- доказательства теорем и интеллектуальные системы, в которых возможности Пролога по обеспечению дедуктивного вывода применяются для проверки различных теорий.

Отличительные особенности Пролога.

- **Для представления знаний используются фразы Хорна;**
- **Программа описывает логическую модель Предметной Области в виде фактов относительно свойств Предметной Области и отношений между этими свойствами + правила вывода новых свойств и отношений из уже заданных;**
- **Использование терма как единообразной структуры данных;**
- **Отсутствие операторов присваивания, ветвлений, безусловных переходов и указателей.**
- **Для объяснения смысла программы применяются три семантические модели : декларативная, процедурная модели и модель в виде абстрактной машины.**

Процедурная модель.

При *процедурной* трактовке Пролог-программы подчеркивается *последовательность шагов*, которые выполняет интерпретатор *при обработке* запроса. Здесь имеет значение *порядок следования подцелей* в правиле.

Множество фраз, имеющих одно и то же имя и одинаковое количество аргументов, можно рассматривать как процедуру, при этом запрос (или подцель правила) является вызовом процедуры.

Достоинство : позволяет адекватно представлять фразы, в которых важен порядок следования подцелей. Пример : вывод сообщений на экран в определенной последовательности.

Недостаток : невозможность разъяснения смысла фраз, вызывающих побочные эффекты управления. Примеры : остановка выполнения запроса, удаление фразы из программы.

Декларативная модель.

При *декларативной* трактовке Пролог-программы специфицируются *истинностные значения* конкретных случаев отношений.

Для *декларативной* модели фразы Пролога являются формулами *логики предикатов* 1-го порядка, записанными в форме *фраз Хорна*.

Достоинство : эффективность представления знаний ввиду близости к семантике логики предикатов.

Недостаток : невозможность адекватного представления фраз, в которых важен порядок следования подцелей.

Модель в виде абстрактной машины.

С позиций декларативной модели Пролог-программа есть описание логической структуры. При выполнении запроса интерпретатор применяет по отношению к множеству фраз Пролога некоторую стратегию решения задачи. С точки зрения вычислений эта стратегия может быть описана при помощи некоторой абстрактной машины.

В языке Пролог запрос и множество фраз программы имеет вычислительный смысл. Это проявляется в том, что они вызывают определенное поведение интерпретатора Пролога. Модель в виде абстрактной машины описывает смысл запроса и множества фраз через действия этой машины.

Действия такой абстрактной машины можно рассматривать как применение правила резолюции.

Достоинство : модель в виде абстрактной машины наиболее точная из трех рассматриваемых моделей.

Недостаток : большая зависимость от реализации языка.

Факты.

<u>Определение.</u> Факт понимается как запись того отношения, значение которого истинно.

Форма записи факта : имя_предиката(аргумент {, аргумент}).

При этом :

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

2). Перечисление аргументов – через запятую.

3). Каждый факт должен заканчиваться точкой.

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

Факт : who_likes_what (ivan, programming)

не эквивалентен факту : who_likes_what (programming, ivan),

поскольку имеет место изменение направления отношения : “Иван увлекается программированием”≠”Программирование увлекается Иваном”

Запись фактов в Турбо-Прологе.

В Турбо-Прологе тип отношений описывается в разделе predicates :

who_likes_what (symbol, symbol)

При описании фактов и правил в разделе clauses программы на Турбо-Прологе одноименные предикаты должны быть сгруппированы :

clauses

who_likes_what (ivan, programming).

who_likes_what(ivan, reading).

who_likes_what (mary, reading).

В терминологии Пролога любая совокупность фактов (и правил) называется базой данных.

Вопросы.

Запись вопросов в Турбо-Прологе сходна с записью фактов и отличается местоположением. Для формулировки вопросов в программе на Турбо-Прологе существует раздел *goal*.

Обращение к Прологу с вопросом инициализирует процедуру поиска в базе данных, ранее введенной в систему. Пролог просматривает БД в поисках предиката, сопоставимого с вопросом.

Предикаты считаются совпадающими, если они совпадают посимвольно и их соответствующие аргументы попарно совпадают. Если предикат вопроса совпадает с предикатом одного из фактов/правил в БД, то вопрос согласуется с БД.

При этом ответом на вопрос будет либо Yes, либо No.

Использование переменных в вопросах.

Определение. Под переменной в Прологе понимается любое имя, начинающееся с прописной латинской буквы. Примеры : *Who, What, Ivan*.

В отличие от процедурных языков, где имя переменной связывается с областью памяти, переменная в Прологе обозначает объект, значение которого может быть найдено.

Определение. Переменная называется конкретизированной, если существует объект, который она обозначает.

Определение. Если не известно, что именно обозначает переменная, то считается, что переменная не конкретизирована.

В вопросах переменные используются для того, чтобы найти какой-либо объект. Пример : who_likes_what (ivan, X) – ответом будет :

X=programming

X=reading 2 Solutions

Yes

Если обозначаемый переменной объект не имеет значения в рассматриваемом контексте, то используется анонимная переменная. В Турбо-Прологе анонимные переменные обозначаются символом “_”.

Сложные вопросы.

В сложных вопросах, содержащих конъюнкцию, и в правилах Пролог при согласовании каждому целевому утверждению приписывает его собственный маркер. Если в базе данных есть факт, соответствующий целевому высказыванию, то Пролог отмечает найденное место и пытается согласовать оставшиеся целевые высказывания. Для каждой подцели просмотр БД начинается с начала.

Пример для приведенной на *слайде 9* БД. Выясним, имеют ли Иван и Мария общие интересы.

goal

who_likes_what (ivan, X), who_likes_what (mary, X).

Вначале согласуем who_likes_what (ivan, X). Переменная X конкретизируется : X= programming. Теперь пытаемся согласовать who_likes_what (mary,programming). Данный факт отсутствует в БД. Поэтому Пролог автоматически расконкретизирует переменную X.

Затем Пролог пытается согласовать утверждение :

who_likes_what (ivan, reading), who_likes_what (mary, reading).

Оба факта, входящие в выражение, присутствуют в БД. Других альтернатив для *who_likes_what(ivan,X)* нет и в качестве результата будет выдано : X=reading, 1 solution, Yes.

Показанный процесс получил название поиска с возвратом.

Правила.

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

Пролог-правило имеет вид фразовой формы :

заключение:-усл1, усл2, … ,услN.

Данное выражение считается основным в Прологе. Языки, подобные Прологу, считаются языками типа “если-то” : заключение истинно, если истинными являются все условия, перечисленные в правой части.

Пример правила для приведенной на слайде 9 БД :

```
common_interests (X,Y ) : -
        who_likes_what (X,Z), who_likes_what (Y,Z), X<>Y.
```

Операторы

Иногда удобно записывать некоторые функторы как операторы.. В Прологе операторы не вызывают выполнения каких-либо арифметических операций.

Так, терм 3+4 - другой способ записи структуры +(3,4)

При построении арифметических выражений необходимо знание ***трех основных свойств*** каждого оператора :

- **Позицию;**
- **Приоритет;**
- **Ассоциативность**.

Позиция оператора

Операторы **+,-,*,/** записываются между своими аргументами и называются ***инфиксными***. : ***x oper y***.

Операторы “+” и “-” могут быть записанными перед своими аргументами. Пример : изменение знака : ***-x+y***. В данной ситуации оператор “-” является ***префиксным***.

Оператор, записываемый после своего аргумента, называется ***постфиксным***. Пример : математическая запись факториала : ***n!***

Приоритет оператора

Приоритет оператора определяет, какая операция выполняется первой. В Прологе каждый оператор связан со своим ***классом приоритета***. Класс приоритета есть целое число, величина которого зависит от конкретной версии Пролога. Однако в любой версии оператор с большим приоритетом имеет класс приоритета, более близкий к 1.

Ассоциативность операторов

От свойства *ассоциативности* операторов зависит порядок выполнения операторов с одинаковым приоритетом. Пример : Y=8/2/2. В подобных случаях необходимо знать, является ли данный оператор *левоассоциативным* или *правоассоциативным*.

***Левоассоциативный* оператор должен иметь слева операции одинакового или более высокого приоритета, а справа - операции низшего приоритета. В Прологе арифметические операции (+,-,*,/) являются левоассоциативными. Это означает, что выражения, подобные Y=8/2/2 интерпретируются как Y=(8/2)/2.**

Равенство и установление соответствия

При попытке согласования с базой данных целевого утверждения X=Y Пролог пытается установить соответствие между X и Y. Целевое утверждение доказуемо, если соответствие имеет место. При согласовании с базой данных цели вида X=Y, где X,Y - любые термы, в которых могут содержаться неконкретизированные переменные, действуют следующие правила.

- **Если X - неконкретизированная переменная, а переменная Y конкретизирована, то X и Y равны, а X становится конкретизированной.**
- **Целые числа и атомы всегда равны сами себе.**
- **Две структуры равны, если они имеют один и тот же функтор и одинаковое число аргументов, причем все соответствующие аргументы равны.**

Действия над числами

Для сравнения чисел Пролог имеет ряд встроенных предикатов. Каждый из них может быть использован в качестве инфиксного оператора.

X=Y X и Y представляют одно и то же число

X<>Y X и Y представляют разные числа

X<Y X меньше Y

X>Y X больше Y

X>=Y X больше или равно Y

X<=Y X меньше или равно Y

Вычисления в Прологе

Арифметические операции могут использоваться для вычислений. Набор допустимых операций зависит от используемой реализации Пролога. Все реализации Пролога обеспечивают выполнение следующих операций :

X+Y	**сумма X и Y**
X-Y	**разность X и Y**
X*Y	**произведение X и Y**
X/Y	**частное от деления X и Y**
X MOD Y	**остаток от деления X на Y**

Турбо-Пролог также допускает целочисленное деление : X DIV Y

Согласование целевых утверждений

Для доказательства *целевых утверждений* Пролог использует известные *утверждения*. Если подцель представляет собой факт, то согласование заканчивается, как только факт будет найден в базе. Если подцель есть правило, то задача сводится к конъюнкции предикатов-подцелей. При согласовании Пролог руководствуется следующими *правилами* :

- **Константа равна только самой себе;**
- **Переменная может быть конкретизирована любым объектом Пролога;**
- **Структуры совпадают, если совпадают их функторы , положение и количество аргументов.**

Для каждой подцели Пролог генерирует свой маркер. Если целевое утверждение не доказано, возбуждается *процесс возврата*.

Успешное доказательство конъюнкции целевых утверждений

Как в теле правила, так и в вопросе Пролог пытается согласовать входящие в конъюнкцию целевые утверждения в том порядке, в каком они написаны (слева направо). Доказательство согласованности целевого утверждения с базой данных включает в себя поиск соответствующих (сопоставимых) утверждений, пометку этого места базы данных и затем доказательство возникших подцелей. В ходе доказательства согласованности целевых утверждений с базой данных происходит конкретизация переменных.

Согласование целевого утверждения при наличии правил.

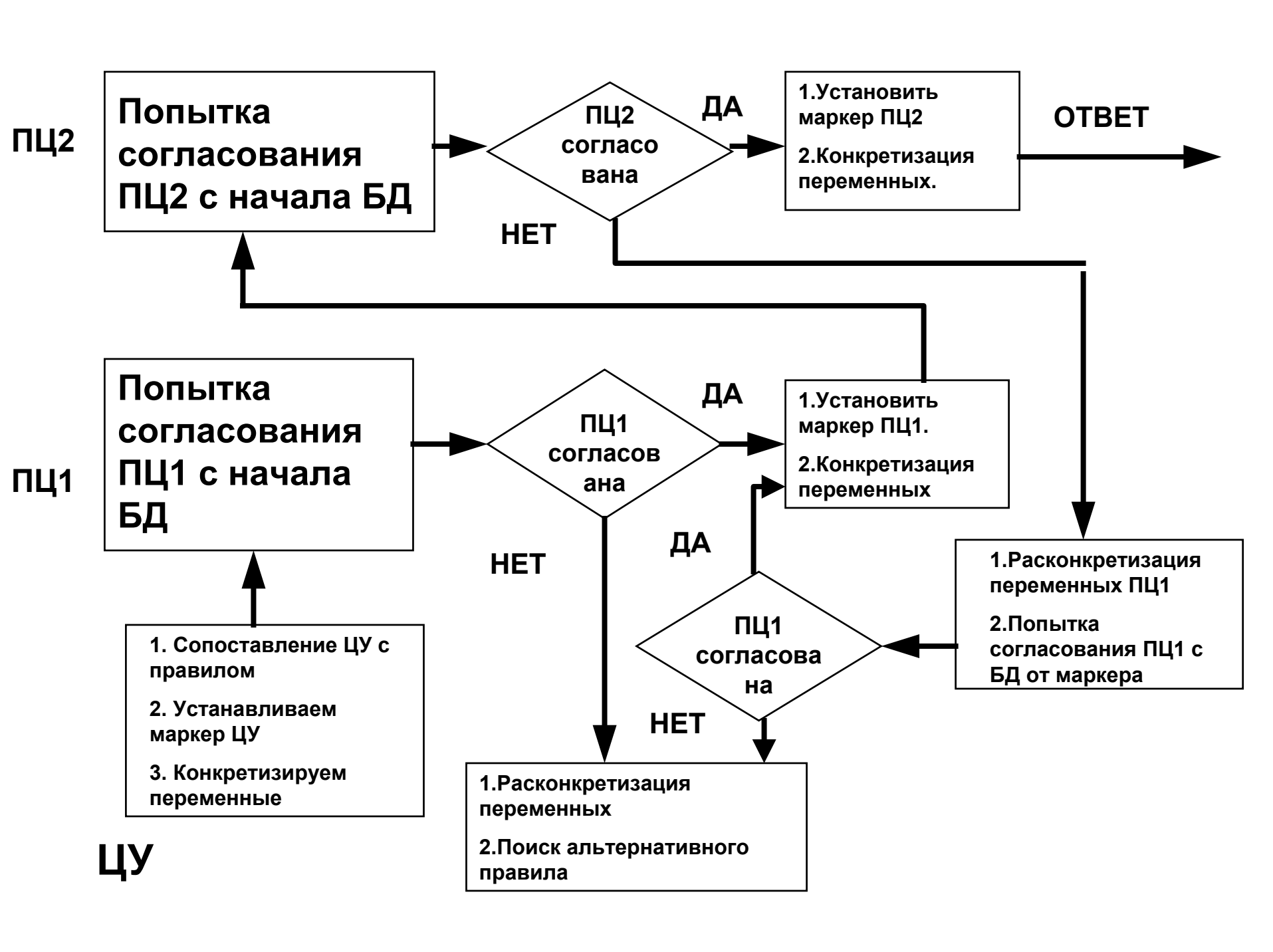
Рассмотрим последовательность действий по согласованию целевого утверждения *common_interests (ivan, mary)* при наличии показанного правила в БД. Факт *common_interests(ivan, mary)* в БД отсутствует, но есть приведенное выше правило. Пролог отмечает это место в БД. При этом переменная *X* конкретизируется значением *ivan*, *Y* – значением *mary*.

Вначале Пролог ищет соответствие для предиката *who_likes_what(ivan,Z)*. Факт, с которым происходит сопоставление, есть *who_likes_what(ivan, programming)*, и тем самым первая цель достигнута. Пролог отмечает маркером соответствующее место в БД (первое утверждение сверху) и записывает, что *Z* присвоено значение *programming*.

Затем Пролог пытается найти соответствие для следующего предиката в правиле, для чего ищет в базе данных факт *who_likes_what (mary, programming)* и, не находя его, расконкретизирует переменную *Z*. Теперь Пролог продолжает поиск соответствия для для предиката *who_likes_what (ivan, Z)* начиная от первого сверху утверждения (где находится маркер) и находит факт *who_likes_what (ivan, reading)*, *Z* присваивается значение *reading*. Утверждение *who_likes_what (mary, reading)* успешно согласуется с БД. Последняя цель в конъюнкции также успешно достигается и тем самым доказывается, что факт *common_interests (ivan, mary)* является истинным, Пролог отвечает *Yes*.

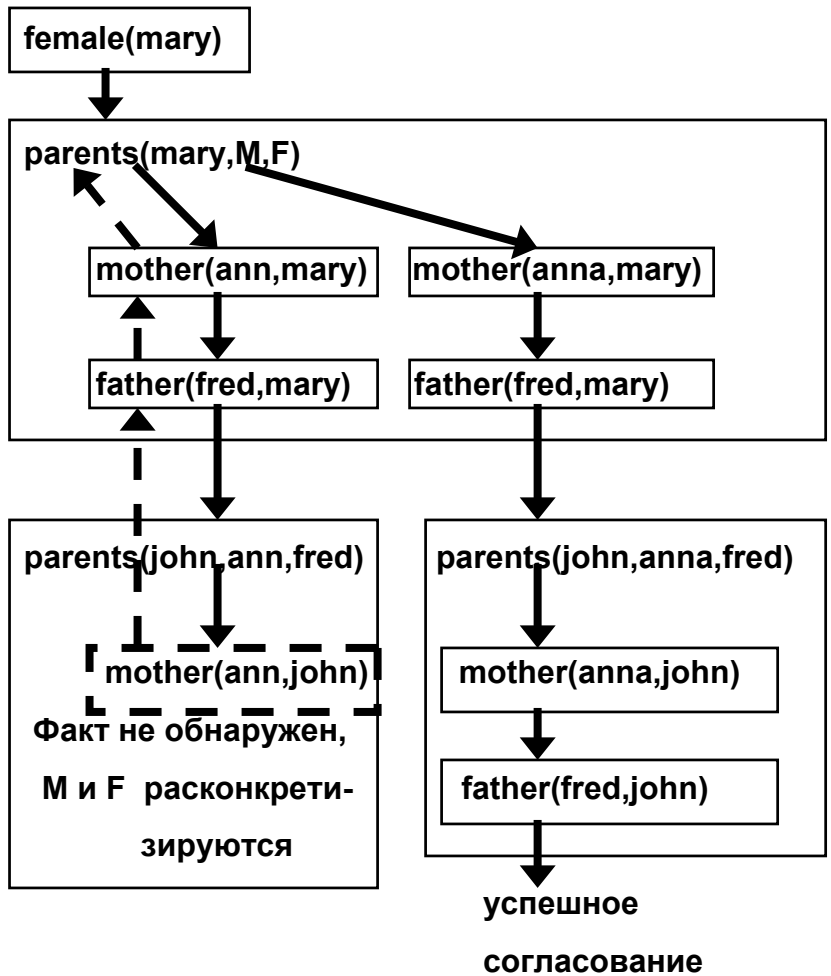
Рассмотрение целевых утверждений при использовании механизма возврата

Когда целевое утверждение недоказуемо, осуществляется возврат по “цепочке доказательств” в место выбора утверждения для согласования заново соответствующих целевых утверждений. При этом Пролог пытается найти альтернативное утверждение, соответствующее данной цели. Вначале происходит расконкретизация всех переменных, конкретизированных в ходе доказательства данного целевого утверждения. Затем возобновляется поиск в базе данных, начиная с помеченного маркером места. Если будет найдено другое утверждение, соответствующее целевому, Пролог помещает в это место маркер, поиск в базе данных продолжается, начиная с помеченного места.


ПЦ2
Попытка согласования ПЦ2 с начала БД
ПЦ2 согласована
ДА
1.Установить маркер ПЦ2
2.Конкретизация переменных.
ОТВЕТ
НЕТ
ПЦ1
Попытка согласования ПЦ1 с начала БД
ПЦ1 согласована
ДА
1.Установить маркер ПЦ1.
2.Конкретизация переменных
НЕТ
ДА
1.Расконкретизация переменных ПЦ1
2.Попытка согласования ПЦ1 с БД от маркера
ПЦ1 согласована
НЕТ
1. Сопоставление ЦУ с правилом
2. Устанавливаем маркер ЦУ
3. Конкретизируем переменные
1.Расконкретизация переменных
2.Поиск альтернативного правила
ЦУ

Пример согласования целевого утверждения

```
/*   Семейные связи   */
predicates
        male(symbol).
        female(symbol).
        mother(symbol,symbol).
        father(symbol,symbol).
        parents(symbol,symbol,symbol).
/* goal
        female(mary),parents(mary,M,F),parents(john,M,F).
     */
clauses
        male(john).
        male(fred).
        female(mary).
        female(anna).
        female(ann).
/*факт, противоречащий семантике предметной
области*/
        mother(ann,mary).
/*а здесь - семантически корректная информация*/
        mother(anna,john).
        mother(anna,mary).
        father(fred,mary).
        father(fred,john).
parents(C,M,F):-mother(M,C),father(F,C).
```



Правила установления соответствия

- **Неконкретизированная переменная соответствует любому объекту. Этот объект становится значением переменной.**
- **Целое число или атом соответствует только самим себе.**
- **Между структурами можно установить соответствие при совпадении функторов, числа параметров и соответствии параметров.**

***Примечание.** При выборе утверждения все переменные изначально неконкретизированы.**

Понятие “сцепленных переменных”.

В ряде случаев согласования целевых утверждений некоторая изначально неконкретизированная переменная X может быть автоматически конкретизирована значением другой переменной Y. В таких случаях говорят, что переменные X и Y становятся сцепленными. Пример (контр-пример) – отношение “родная сестра” :

```
sister(X,Y) :-

    woman(X), parent(X, Mother, Father), parent(Y, Mother, Father).
```

Согласование целевого утверждения sister(“Мария”,Y) приведет в одном из возможных случаев успешного доказательства к конкретизации Y значением”Мария”. Для предотвращения порождения нежелательных решений как следствия сцепления переменных в тело правила следует вводить дополнительные утверждения для контроля взаимных соотношений значений переменных в заголовке правила :

```
sister(X,Y) :-

    not(X=Y), woman(X), parent(X, Mother, Father), parent(Y, Mother, Father).
```

Литература.

1. Клоксин У., Меллиш К. Программирование на языке Пролог : Пер. с англ. - М.: Мир, 1987. С. 11-62
2. Маплас Дж. Реляционный язык Пролог и его применение : Пер. с англ. - М.: Наука, 1990. С. 66-87
3. Ин Ц., Соломон Д. Использование Турбо-Пролога : Пер. с англ. - М.: Мир, 1993. С. 32-68, 111-113.