



# Концепции языка Prolog

---

- Предложения: факты и правила
- Запросы
- Переменные
- Поиск с возвратом
- Рекурсия

# Введение

---

- Prolog — это язык программирования для **символических, нечисловых** вычислений.
- Он особенно хорошо приспособлен для решения проблем, которые касаются **объектов** и **отношений** между объектами:
  - реализация экспертных систем и оболочек экспертных систем;
  - создание пакетов символьных вычислений;
  - доказательства теорем и интеллектуальные системы и т.д.

# Близость к естественному языку

---



Asm – ассемблер

АЯ – алгоритмические (процедурные) языки

ФП – языки функционального программирования

ЛП – языки логического программирования

ЕЯ – Естественные Языки (русский, английский и др.)

# Концепция языка Prolog

---

- Prolog является **декларативным** языком программирования, который обеспечивает решение задач, выраженных в терминах объектов и отношений между ними
- В Prolog **отсутствуют** операторы присваивания, ветвлений, циклов, безусловных переходов и указателей. Говоря не строго, в Прологе **отсутствуют** какие-либо действия.

# Программирование на языке Пролог

---

- Программа на языке Prolog состоит из следующих компонент :
  - 1) Набор фактов об объектах и отношениях между ними;
  - 2) Множество правил об объектах и отношениях между ними;
  - 3) Единственный вопрос об объектах и отношениях между ними.

# Факты

---

- Факт понимается как запись того **отношения**, значение которого **истинно**.
- **Форма** записи факта :  
*имя\_предиката(аргумент {, аргумент}).*
- В терминологии Prolog любая совокупность фактов (и правил) называется **базой данных**.

# Правила записи фактов

---

- Все имена предикатов и аргументов должны начинаться со **строчной** латинской буквы.
- Перечисление аргументов – через **запятую**.
- Каждый факт должен заканчиваться **точкой**.
- Количество аргументов и вид отношений (направления отношений) определяются **программистом** и не меняются при выполнении программы.

# Примеры фактов

---

- `likes(ivan,programming).` ;Иван увлекается программированием
- `likes(programming,ivan).`  
;Программирование увлекается Иваном



# Правила записи фактов в VisualProlog

---

- Тип отношений описывается в разделе `predicates` :

```
likes(symbol,symbol)
```

- При описании фактов и правил в разделе `clauses` предикаты должны быть сгруппированы :

```
clauses
```

```
likes(ivan,programming).
```

```
likes(ivan,reading).
```

```
likes(mary,reading).
```

# Вопросы

---

- Для того чтобы программа, написанная на языке Prolog, начала работу, к ней нужно обратиться с **вопросом**.
- Для формулировки вопроса в программе на VisualProlog существует раздел **goal**.

**goal**

`likes(ivan,mary).`

- Обращение к Prolog с вопросом инициализирует процедуру **поиска** в базе данных, ранее введенной в систему.

# Формирование ответа на вопрос

---

- 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 в БД, то вопрос согласуется с БД.
- При этом ответом на вопрос будет либо **Yes**, либо **No**.

# Переменные

---

- Под переменной в Prolog 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 Prolog обозначает **объект**, значение которого может быть найдено.

# Переменные

---

- Переменная называется **конкретизированной**, если существует объект, который она обозначает.
- Если не известно, что именно обозначает переменная, то считается, что переменная **не конкретизирована**.

# Использование переменных в вопросах

---

- В вопросах переменные используются для того, чтобы найти **какой-либо** объект.
- Пример :  
`likes(ivan, X)` – ответом будет :  
`X = programming`  
`X = reading`                      2    Solutions  
Yes
- При сопоставлении факта и вопроса, содержащего переменную, переменная **получает** значение

# Использование переменных в вопросах

---

- Если обозначаемый переменной объект не имеет значения в рассматриваемом контексте, то используется **анонимная** переменная.
- В VisualProlog **анонимные** переменные обозначаются символом “\_”.
- Пример :  
`likes(ivan, _)` – ответом будет :  
Yes

# Сложные вопросы и поиск с возвратом

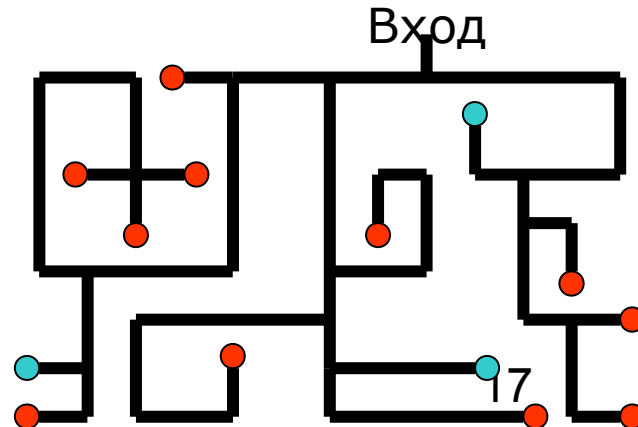
---

- В сложных вопросах, содержащих **конъюнкцию**, должны быть доказаны все подцели.
- Сначала Prolog пытается доказать **первую** подцель.
- Если в базе данных есть факт, соответствующий первой подцели, то Prolog **отмечает** найденное место (приписывает маркер) и пытается согласовать следующие подцели.



# Сложные вопросы и поиск с возвратом

- После согласования (или не согласования) следующих подцелей, Prolog снова **возвращается** к первой подцели и **снова** начинает ее согласование с приписанного маркера.
- Описанный процесс получил название **поиска с возвратом** (backtracking).



# Пример сложного вопроса и поиска с возвратом

---

- Выясним, имеют ли Иван и Мария общие интересы.

goal

`likes(ivan,X),likes(mary,X).`

- Вначале **согласуем** `likes(ivan,X)`.  
Переменная `X` **конкретизируется** : `X=programming`.
- Теперь пытаемся **согласовать** `likes(mary,programming)`. Данный факт отсутствует в БД. Поэтому Prolog автоматически **расконкретизирует** переменную `X`.

# Пример сложного вопроса и поиска с возвратом

---

- Затем Prolog пытается **согласовать** утверждение  
`likes(ivan,reading),likes(mary,reading).`
- Оба факта, входящие в выражение, **присутствуют** в БД.
- Других **альтернатив** для `likes(ivan,X)` **нет** и в качестве результата будет выдано :  
`X=reading 1 solution Yes.`

# Правила

---

- Под правилами в Prolog понимаются наиболее общие **утверждения** об объектах и **отношениях** между ними.
- Правила задают **новые** отношения через уже **существующие**.
- Пролог-правило имеет вид фразовой формы:  
*заклучение:-усл1, усл2, ... ,услN.*
- Данное выражение считается основным в Prolog.

# Правила

---

- Прологоподобные языки считаются языками типа “если-то” : заключение истинно, если истинными являются все условия, перечисленные в правой части.
- Пример правила :

% у X и Y имеются общие интересы

```
common_likes(X,Y):-  
    likes(X,Z),likes(Y,Z),X<>Y.
```

# Согласование целевого утверждения при наличии правил

---

- Выясним, имеют ли Иван и Мария общие интересы.

`goal`

`common_likes(ivan,mary).`

- Вначале Prolog пытается найти в БД факт `common_likes(ivan,mary).`
- Данный факт отсутствует в БД, но есть приведенное выше правило. Пролог отмечает это место в БД. При этом переменная `X` конкретизируется значением `ivan`, `Y` – значением `mary`.

# Согласование целевого утверждения при наличии правил

---

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

# Согласование целевого утверждения при наличии правил

---

- Затем Prolog пытается найти соответствие для следующего предиката в правиле, для чего ищет в базе данных факт `likes(mary,programming)` и, не находя его, расконкретизирует переменную `Z`.
- Теперь Prolog продолжает поиск соответствия для предиката `likes(ivan,Z)` начиная от первого сверху утверждения (где находится маркер) и находит факт `likes(ivan,reading)`, `Z` присваивается значение `reading`.



# Согласование целевого утверждения при наличии правил

---

- Утверждение `likes(mary,reading)` успешно согласуется с БД.
- Последняя цель в конъюнкции также успешно достигается и тем самым доказывается, что факт `likes(ivan,mary)` является истинным, Prolog отвечает Yes.

# Рекурсия как основной метод программирования на Prolog

---

- В общем случае рекурсивное правило имеет следующий вид :

*recursive\_rule(<фактические параметры через запятую>):-  
    <предикаты и правила>,  
    recursive\_rule(<фактич. параметры рекурсивного вызова>).*

- Для передачи значений между правилами используется стек.
- Всякий раз при рекурсивном вызове новые копии используемых значений помещаются в стек.

# Правила построения рекурсивных процедур

---

- Должен быть задан изменяющийся аргумент рекурсивного вызова. При этом положение рекурсивного вызова в теле правила может быть любым.
- Должен быть задан факт, обеспечивающий завершение рекурсии. Такой факт должен помещаться перед правилом, а не после него во избежание левосторонней рекурсии.

# Левосторонняя рекурсия

---

- Возникает в случае, когда правило порождает подцель, эквивалентную исходной цели, которая явилась причиной использования этого правила.
- В процедуре с левой рекурсией рекурсивная подцель стоит слева от других подцелей.

○ Пример:

`dog(X) :- dog(Y), parent(Y,X).`

`dog(reks).`

## Левосторонняя рекурсия

---

- При попытке согласовать целевое утверждение `dog(X)` Prolog вначале пытается использовать правило и рекурсивно порождает подцель `dog(Y)`.
- Попытка найти соответствие этой цели вновь приводит к выбору первого правила и так далее.
- Причина зацикливания заключается в отсутствии возможности использования механизма возврата.
- Для того, чтобы начался возврат, Prolog должен потерпеть неудачу, чего не происходит в приведенном примере.

# Пример использования рекурсии

---

- Вычисление факториала

`fact ( 0, 1 ).`

`fact ( N, M ) : -`

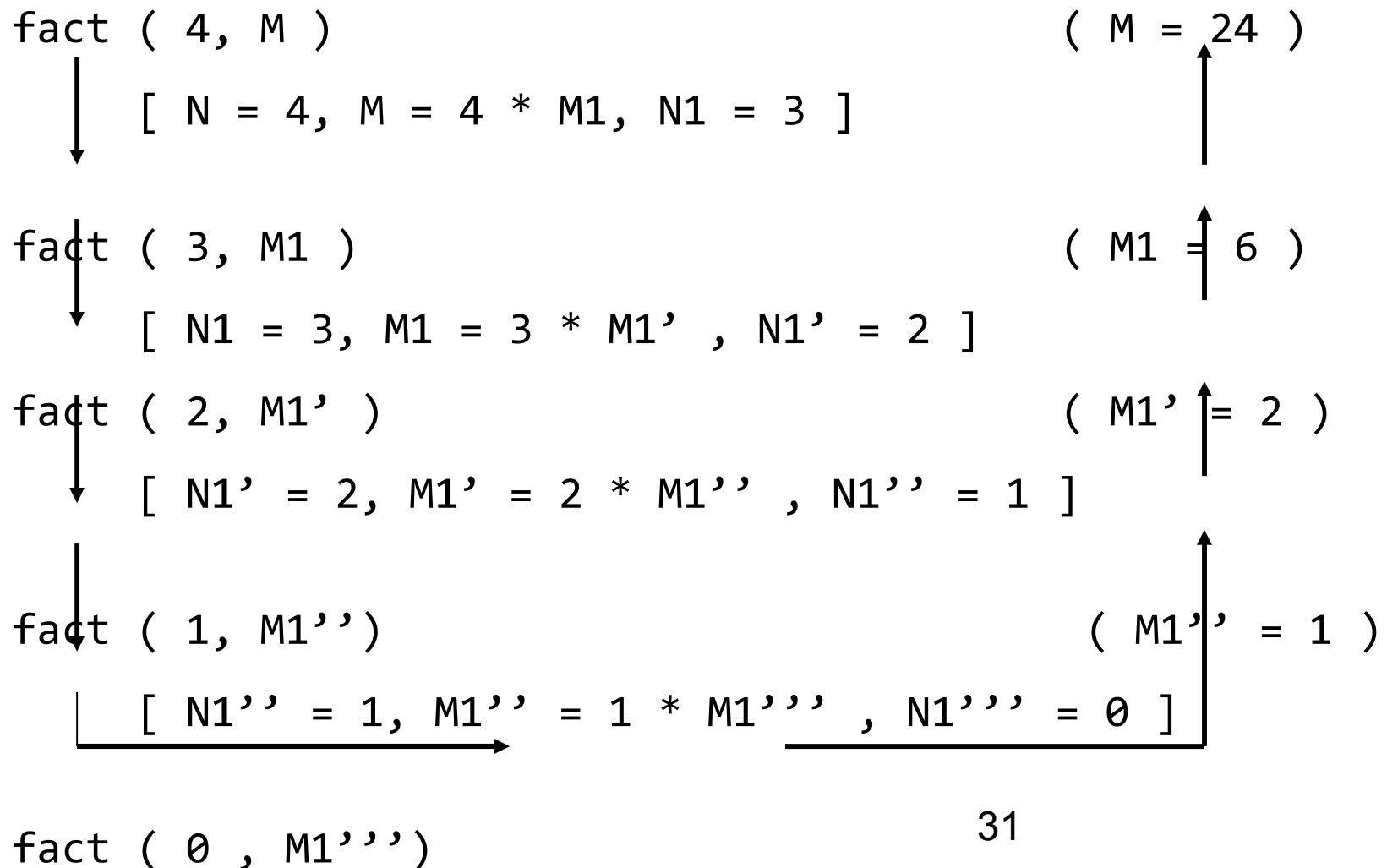
`N1 = N - 1,`

`fact ( N1, M1 ),`

`M = N * M1.`

- Рассмотрим работу правила для вычисления 4!.

# Этапы решения задачи “Вычисление факториала”



# Декларативная трактовка Prolog-программ

---

- Декларативное значение программ позволяет установить, является ли заданная цель истинной, а в случае положительного ответа — при каких значениях переменных она является истинной.
- Никаких шагов в программе не выполняется.



# Декларативная трактовка Prolog-программ

---

- Для доказательства истинности цели достаточно правильно сформулировать все факты и правила. При этом порядок правил и фактов не важен, так же не важен порядок целей в правилах.
- Пролог автоматически, без участия пользователя, доказывает истинность целей. При этом нет необходимости знать, как работает механизм вывода.

# Процедурная трактовка Prolog-программ

---

- При процедурной трактовке Prolog-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 является вызовом процедуры.
- Для получения процедурного значения Prolog-программы необходимо знать, как работает механизм вывода.

# Что читать

---

Братко, Иван.

Б87 Алгоритмы искусственного интеллекта на языке PROLOG, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильямс", 2001. — 640 с.

- Глава 1. Введение в Prolog
- Глава 2. Синтаксис и значение программ Prolog
- Глава 3. Списки, операции, арифметические выражения
- Пункт 3.3. Запись в операторной форме
- Пункт 3.4. Арифметические выражения