

# Математическая логика и логическое программирование

mk.cs.msu.ru → Лекционные курсы  
→ Математическая логика и логическое программирование (3-й поток)

## Блок 43

Логические программы:  
встроенные предикаты и функции

Лектор:  
**Подымов Владислав Васильевич**  
E-mail:  
**valdus@yandex.ru**

# Вступление

При помощи ХЛП можно решать любые разрешимые задачи, но всё же чего-то не хватает

**Например**, если хочется написать программу, складывающую два числа из  $\mathbb{N}_0$ , то это не получится сделать сильно проще, чем:

1. Использовать представление чисел в виде двоичной записи, и для неё использовать списки, перечисляя биты от младшего к старшему:

▶  $13 = (1011)_2 = \mathbf{1.1.0.1.nil}$

▶ Использовать *предикат* сложения  $\text{plus}(X, Y, Z) = \langle Z \text{ есть сумма } X \text{ и } Y \rangle$

▶ Реализовать этот предикат для выбранного представления:

$\text{plus}(\mathbf{nil}, L, L);$

$\text{plus}(L, \mathbf{nil}, L);$

$\text{plus}(\mathbf{0.L}, X.M, X.N) \leftarrow \dots;$

$\text{plus}(\mathbf{1.L}, X.M, Y.N) \leftarrow \dots;$

...

Выходит не очень удобно для использования — а нельзя ли лучше?

# Вступление

Дальше (в этом блоке слайдов и в следующих) будут обсуждаться конструкции, которые содержатся в языке Prolog и существенно повышают удобство практического использования логических программ

Но введение этих конструкций сопряжено с техническими и идеологическими трудностями, из-за которых будем иногда

- ▶ «забывать» про декларативную семантику и обсуждать только операционную
- ▶ «вспоминать» про декларативную семантику и пытаться преодолеть трудности, иногда полноценно, а иногда только отчасти

Чтобы подчеркнуть расхождение между тем, что будет рассказываться про логические программы дальше, и тем, что рассказывалось до сих пор, не будем называть программы с дальнейшими добавлениями и уточнениями «хорновскими», называя их просто «**логические программы**»

# Встроенные предикаты и функции

В языке логических программ на практике содержатся **встроенные предикаты и функции**: предикатные и функциональные символы, имеющие предзаданный смысл, выходящий за рамки SLD-резолюции, и предназначенные для записи в телах правил и запросов

Смысл встроенных предикатов будет обсуждаться **только** в рамках операционной семантики, и он будет задаваться как сочетание **критерия выполнимости** и **унификатора**:

- ▶ Если подцелью выбран не встроенный предикат, то шаг вычисления состоит в обычном построении SLD-резольвенты
- ▶ Иначе:
  - ▶ Если выполнен **критерий выполнимости** встроенного предиката, выбранного в качестве подцели, то
    - ▶ этот предикат удаляется из запроса, и
    - ▶ к оставшейся части запроса применяется **унификатор**
  - ▶ Иначе построенное вычисление считается **тупиковым** (невозможно выполнить следующий шаг)

# Типы данных

В «удобных» языках программирования, как правило, есть хотя бы минимальный набор **типов данных**

**Для примера** подробно обсудим тип **integer** целых чисел:

**0, 1, (-1), 2, (-2), ...**

С точки зрения логики предикатов,

- ▶ **0, 1, (-1), 2, (-2), ...** — это целочисленные **константы**
- ▶ **integer** — это одноместный предикатный символ:  $\text{integer}(x) = \langle\langle x \text{ — целое число} \rangle\rangle$

## Типы данных: integer

`integer` в логических программах — это встроенный одноместный предикат:

- ▶ Критерий выполнимости  $\text{integer}(t)$ :  $t$  — это целочисленная константа
- ▶ Унификатор:  $\varepsilon$

**Например:**

?integer(**3**), p(X)

↓  $\varepsilon$

?p(X)

?integer(**0.nil**)

тупик

?integer(**1 + 2**)

тупик

?integer(X)

тупик

## Типы данных: integer, +, -, · (\*), / (div), % (mod)

Для целых чисел можно естественно ввести

- ▶ операции: +, -, · (она же \*), / (она же **div**), % (она же **mod**), ...
  - ▶ С точки зрения логики предикатов, это функциональные символы подходящей местности в инфиксной записи с естественным смыслом
- ▶ отношения: <, ≤, >, ≥, ...
  - ▶ С точки зрения логики предикатов, это предикатные символы подходящей местности в инфиксной записи с естественным смыслом
  - ▶ Особое место в этом списке занимает отношение равенства, его обсудим отдельно позже

*Небольшая поправка:* знаки операций и отношений, используемые в Prolog, могут отличаться, читайте документацию

## Типы данных: integer, <, ≤, >, ≥

Отношение  $\bowtie \in \{<, \leq, >, \geq\}$  над целыми числами в логических программах — это встроенный двуместный предикат в инфиксной записи:

- ▶ Критерий выполнимости  $t_1 \bowtie t_2$ :  $t_1$  и  $t_2$  — целочисленные константы, входящие в отношение  $\bowtie$
- ▶ Унификатор:  $\varepsilon$

### Например:

$$\begin{array}{c} ?1 < 3, p(X) \\ \downarrow \varepsilon \\ ?p(X) \end{array}$$

$$\begin{array}{c} ?3 < 1 \\ \text{тупик} \end{array}$$

$$\begin{array}{c} ?X < 2 \\ \text{тупик} \end{array}$$

$$\begin{array}{c} ?1 < 1 + 2 \\ \text{тупик} \end{array}$$

Как видно по самому правому и самому левому примерам, выражение  $1 + 2$  расценивается программой не как число  $3$ , а именно как выражение — запись, сама по себе не являющаяся числом

Чтобы **вычислить** это выражение (преобразовать его в число), требуется применить соответствующий предикат



## Вычисляющий предикат (is)

**is** — это встроенный двуместный предикат (записывающийся инфиксно), предназначенный для вычисления значения выражения с записью в переменную:

- ▶ Критерий выполнимости  $t_1$  is  $t_2$ :
  - ▶  $t_1$  — переменная и
  - ▶  $t_2$  — выражение (терм), имеющее значение (без переменных и построенное корректно относительно типов)
- ▶ Унификатор:  $\{t_1/val\}$ , где  $val$  — значение выражения  $t_2$

### Например:

?X is <b>1 + 2</b> , p(X), r(Y)	? <b>3</b> is <b>1 + 2</b>	?X is <b>1 + Y</b>	? <b>1 + 2</b> is <b>3</b>	? <b>1 + 2</b> is X
↓ {X/ <b>3</b> }	тупик	тупик	тупик	тупик
?p( <b>3</b> ), r(Y)				

*Небольшая поправка:*

- ▶ В интерпретаторе языка Prolog с немалой вероятностью во втором слева примере будет не тупик, а шаг вычисления с унификатором  $\epsilon$
- ▶ Выше изложена семантика **is** в «исходном» варианте языка Prolog, а остальное здесь считается «отклонением от стандарта»

## Предикаты равенства и неравенства ( $=$ , $==$ , $\neq$ , $\neq$ )

В Prolog используется два вида равенства термов:

- ▶ **Сильное**: посимвольное совпадение
- ▶ **Слабое**: унифицируемость

$==$  — встроенный двуместный предикат сильного равенства в инфиксной записи:

- ▶ Критерий выполнимости  $t_1 == t_2$ : термы  $t_1$  и  $t_2$  посимвольно совпадают
- ▶ Унификатор:  $\epsilon$

**Например:**

$$?X + 1 == X + 1, p(X)$$

$\downarrow \epsilon$

$$?p(X)$$

$$?X == 1 + 2$$

тупик

$$?3 == 1 + 2$$

тупик

$$?X + 1 == 1 + X$$

тупик

$$?X + 1 == 2 + X$$

тупик

$$?1 + 2 == 2 + 1$$

тупик

## Предикаты равенства и неравенства ( $=$ , $==$ , $\neq$ , $\neq$ )

$=$  — встроенный двуместный предикат слабого равенства в инфиксной записи:

- ▶ Критерий выполнимости  $t_1 = t_2$ :  $\mathcal{V}(t_1, t_2) \neq \emptyset$
- ▶ Унификатор: какой-либо наиболее общий унификатор  $t_1, t_2$

Например:

$$?X + \mathbf{1} = X + \mathbf{1}, p(X)$$

$$\downarrow \varepsilon$$
$$?p(X)$$

$$?X = \mathbf{1} + \mathbf{2}, p(X), r(Y)$$

$$\downarrow \{X/\mathbf{1} + \mathbf{2}\}$$
$$?p(\mathbf{1} + \mathbf{2}), r(Y)$$

$$?\mathbf{3} = \mathbf{1} + \mathbf{2}$$

тупик

$$?X + \mathbf{1} = \mathbf{1} + X, p(X), r(Y)$$

$$\downarrow \{X/\mathbf{1}\}$$
$$?p(\mathbf{1}), r(Y)$$

$$?X + \mathbf{1} = \mathbf{2} + X$$

тупик

$$?\mathbf{1} + \mathbf{2} = \mathbf{2} + \mathbf{1}$$

тупик

## Предикаты равенства и неравенства ( $=$ , $==$ , $\neq$ , $\neq$ )

$\neq$  — встроенный двуместный предикат сильного неравенства в инфиксной записи:

- ▶ Критерий выполнимости  $t_1 \neq t_2$ :  $t_1$  и  $t_2$  не совпадают посимвольно
- ▶ Унификатор:  $\epsilon$

Например:

$$?X + 1 \neq X + 1$$

тупик

$$?X \neq 1 + 2, p(X)$$

$\downarrow \epsilon$   
 $?p(X)$

$$?3 \neq 1 + 2, p(X)$$

$\downarrow \epsilon$   
 $?p(X)$

$$?X + 1 \neq 1 + X, p(X)$$

$\downarrow \epsilon$   
 $?p(X)$

$$?X + 1 \neq 2 + X, p(X)$$

$\downarrow \epsilon$   
 $?p(X)$

$$?1 + 2 \neq 2 + 1, p(X)$$

$\downarrow \epsilon$   
 $?p(X)$

## Предикаты равенства и неравенства ( $=$ , $==$ , $\neq$ , $\neq$ )

$\neq$  — встроенный двуместный предикат слабого неравенства в инфиксной записи:

- ▶ Критерий выполнимости  $t_1 \neq t_2$ :  $\mathcal{U}(t_1, t_2) = \emptyset$
- ▶ Унификатор:  $\varepsilon$

Например:

$$?X + 1 \neq X + 1$$

тупик

$$?X \neq 1 + 2$$

тупик

$$?3 \neq 1 + 2, p(X)$$

$\downarrow \varepsilon$   
 $?p(X)$

$$?X + 1 \neq 1 + X$$

тупик

$$?X + 1 \neq 2 + X, p(X)$$

$\downarrow \varepsilon$   
 $?p(X)$

$$?1 + 2 \neq 2 + 1, p(X)$$

$\downarrow \varepsilon$   
 $?p(X)$