

# Математическая логика и логическое программирование

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

## Блок 31

Хорновские логические программы:  
синтаксис,  
декларативная семантика,  
правильные ответы

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

# Несколько слов о парадигмах программирования

Две основные парадигмы программирования:

## 1. Императивная

- ▶ Программа — это набор команд (инструкций)
- ▶ Семантика программы задаётся как способ пошагового (последовательного) выполнения команд
- ▶ На каждом шаге выполнения текущей командой преобразуются текущие значения данных и определяется то, какая команда должна быть выполнена следующей

## 2. Декларативная

- ▶ Программа — это набор свойств, задающих (определяющих) правильный результат выполнения
- ▶ В **основной семантике** программы не используются понятия выполнения, промежуточных значений данных, текущей команды и т.п., определяется только общий вид желаемого результата
- ▶ В декларативно разработанной программе, запускающейся на «обычном» компьютере, в связи с пошаговой природой компьютера и вычислений используется **вспомогательная семантика**: императивная, согласующаяся с основной декларативной

# Несколько слов о парадигмах программирования

## Пример

**Задача:** вскипятить воду в электрическом чайнике

**Решение** в императивной парадигме (*упрощённое*):

- ▶ Если в чайнике нет воды, то налить её
- ▶ Нажать на кнопку кипячения
- ▶ Дождаться, когда кнопка кипячения отожмётся

**Решение** в декларативной парадигме (*местами упрощённое, местами усложнённое для более наглядной аналогии*):

- ▶ Желаемый результат — это чайник, в котором горячая вода
- ▶ Вода появляется в чайнике, если её налить
- ▶ Вода в чайнике бывает горячей, когда отжимается кнопка кипячения
- ▶ Чтобы кнопка кипячения отжалась, нужно перед этим её нажать

# Несколько слов о парадигмах программирования

Императивная парадигма программирования — самая популярная и известная, так как эта парадигма

- ▶ используется на «низком уровне» в современных вычислительных устройствах, и при этом
- ▶ достаточно понятна и естественна для программиста и
- ▶ традиционно изучается программистами как первая (и иногда единственная)

Современные языки программирования, как правило, не принадлежат только к одной парадигме, то есть **мультипарадигменны**, но зачастую можно определить *преобладающую* парадигму языка

Примеры языков, в которых преобладает императивная парадигма: C, C++ , Pascal, Java, Python (*хотя местами можно и поспорить о том, что в нём преобладает*), Perl, PHP, машинные коды и ассемблерные языки, ..., ..., ...

Основная математическая вычислительная модель этой парадигмы: машина Тьюринга

# Несколько слов о парадигмах программирования

Две самых известных декларативных парадигмы:

1. Функциональная
2. Логическая

В **функциональной парадигме**:

- ▶ Программа — это функция (в математическом понимании), записанная при помощи базового набора функций и операций композиции функций
- ▶ В **основной семантике** не говорится, **как** вычислить функцию программы, говорится только **что** это за функция

Несколько известных языков программирования с преобладающей функциональной парадигмой: Lisp/Scheme, Erlang, Scala (см. JVM), Haskell, ML (см. OCaml), Python (*хотя он по большей части императивный, но и в функциональном смысле тоже применяется*), ...

Основная математическая вычислительная модель этой парадигмы:  
**λ-исчисление**

# Несколько слов о парадигмах программирования

В **логической парадигме**:

- ▶ Программа — это набор логических формул, представляющих собой совокупность знаний о понятиях задачи / описание необходимых и достаточных свойств результата / определение результата
- ▶ Правильный результат — это **следствие** программы как набора формул
- ▶ В **основной семантике** не говорится, **как** вычислять (извлекать) логические следствия, говорится только **что** является основанием для извлечения следствий

Несколько известных языков программирования с преобладающей логической парадигмой: **Prolog**, Datalog, Planner, Mercury, Gödel, ...

Основная математическая вычислительная модель этой парадигмы: **логический вывод** (логические исчисления)

# ХЛП: синтаксис

Хорновская логическая программа (ХЛП) сигнатуры  $\sigma$  логики предикатов — это конечная последовательность программных утверждений, каждое из которых представляет собой факт или правило

Факт имеет вид « $A$ ;», где  $A$  — атом логики предикатов

Правило имеет вид « $A \leftarrow B_1, \dots, B_k$ ;», где:

- ▶  $k \geq 1$
- ▶  $A$  — **заголовок**: атом логики предикатов
- ▶  $B_1, \dots, B_k$  — **тело**: последовательность атомов логики предикатов, разделённых запятой

Запрос к ХЛП (или, по-другому, **целевое утверждение**, или просто **цель**) имеет вид « $?C_1, \dots, C_k$ », где

- ▶  $k \geq 0$ , и для случая  $k = 0$  запрос принято записывать так:  $\square$
- ▶  $C_1, \dots, C_k$  — **тело**

## ХЛП: синтаксис

Иными словами, ХЛП и запрос к ней задаются следующей БНФ:

<i>ХЛП</i>	::=	<i>утверждение</i> <i>ХЛП</i>
<i>утверждение</i>	::=	<i>факт</i>   <i>правило</i>
<i>факт</i>	::=	<i>атом</i> ;
<i>правило</i>	::=	<i>заголовок</i> ← <i>тело</i> ;
<i>заголовок</i>	::=	<i>атом</i>
<i>тело</i>	::=	<i>атом</i>   <i>атом</i> , <i>тело</i>
<i>запрос</i>	::=	□   ? <i>тело</i>

Для технического единообразия будем считать, что факт — это правило с пустым телом:

$$\langle\langle A \rangle\rangle = \langle\langle A \leftarrow ; \rangle\rangle$$

Каждый атом в теле запроса (цели) принято также называть **подцелью**

Переменные, содержащиеся в запросе (цели), принято называть **целевыми**



# ХЛП: синтаксис

Если захотите транслировать ХЛП и запрос в язык Prolog, то для этого достаточно сделать следующее:

1. Заменить все «;» на «.», «←» на «:-», «?» на «?-» и добавить «.» в конце запроса
  - ▶ В курсе оставим синтаксис ХЛП как есть, чтобы не изменять сложившуюся математику ради одного конкретного языка программирования
2. Начинать все переменные с прописной (большой) буквы, а остальные идентификаторы — со строчной (маленькой)
  - ▶ Будем стараться придерживаться такого написания в курсе, дополнительно различая категории символов шрифтами:  $X$  — переменная,  $\mathbf{a}$  — константа или функциональный символ,  $p$  — предикатный символ

# ХЛП: синтаксис

## Примеры

Правило:  $\text{любит}(\text{паша}, Y) \leftarrow \text{любит}(Y, X), \text{любит}(\text{паша}, X);$

- ▶ Заголовок: «любит(**паша**, Y)»
- ▶ Тело: «любит(Y, X), любит(**паша**, X)»
- ▶ Переменные: Y, X
- ▶ Константы: **паша**
- ▶ Предикатные символы: любит

Факт:  $\text{любит}(\text{паша}, \text{пиво}); \quad \text{любит}(\text{паша}, \text{пиво}) \leftarrow;$

Запрос:  $?умный(X), \text{добрый}(X), \text{красивый}(X), \text{любит}(X, \text{я})$

- ▶ В запросе содержатся 4 подцели: «умный(X)», «добрый(X)», «красивый(X)», «любит(X, я)»
- ▶ X — целевая переменная

# ХЛП: декларативная семантика

Почти каждому элементу  $\xi$  синтаксиса ХЛП можно естественным образом сопоставить формулу логики предикатов  $\Phi_\xi$ :

Элемент	общий вид $\xi$	формула $\Phi_\xi$
Факт	$A$ ;	$\forall \dots A$
Тело	$B_1, \dots, B_k$	$B_1 \& \dots \& B_k$
Правило	$A \leftarrow \beta$	$\forall \dots (\Phi_\beta \rightarrow A)$
Запрос	$?\gamma$	$\Phi_\gamma$

Здесь, как и в блоке 30,  $\forall \dots \varphi(\tilde{x}^n)$  означает  $\forall \tilde{x}^n \varphi(\tilde{x}^n)$

Хорновской логической программе  $\mathcal{P} = (\mathcal{R}_1 \dots \mathcal{R}_m)$  сопоставим систему формул  $S_{\mathcal{P}} = \{\Phi_{\mathcal{R}_1}, \dots, \Phi_{\mathcal{R}_m}\}$

# ХЛП: декларативная семантика

**Содержательно**, декларативная (**основная**) семантика ХЛП  $\mathcal{P}$  и запроса  $\mathcal{Q}$  к ней устроена так:

- ▶ Программа — это база имеющихся заведомо верных знаний
  - ▶ Правило « $A \leftarrow [B_1, \dots, B_k];$ » — это утверждение о том, что для любых значений переменных утверждение  $A$  верно [если для тех же значений верны все утверждения  $B_1, \dots, B_k$ ]
- ▶ Запрос к программе — это входные данные, определяющие вопрос об имеющихся знаниях, на который требуется ответить, записав ответ в целевые переменные (*как запрос к базе данных*)
  - ▶ Запрос « $?C_1, \dots, C_m$ » отвечает вопросу «для каких значений целевых переменных становятся одновременно верными все утверждения  $C_1, \dots, C_m$ ?»
- ▶ **Правильный ответ** на запрос к программе — это значения целевых переменных  $\mathcal{Q}$ , при которых этот запрос как формула **следует** из программы как формулы

# ХЛП: декларативная семантика

**Формально**, центральное понятие декларативной семантики — это **правильный ответ**, и это понятие определяется так

$Var_Q = Var_{\Phi_Q}$  — множество всех переменных запроса  $Q$

**Ответ** на запрос  $Q$  — это подстановка  $\theta$ , такая что  $Dom_\theta \subseteq Var_Q$

**Правильный ответ** на запрос  $Q$  к программе  $\mathcal{P}$  — это ответ  $\theta$  на запрос  $Q$ , для которого выполнено соотношение

$$S_{\mathcal{P}} \models \forall \dots (\Phi_Q \theta)$$

# ХЛП: декларативная семантика

## Пример

Программа  $\mathcal{P}$ :

пернатый( <b>орёл</b> );	есть_живой( <b>орёл</b> );	летает( <b>орёл</b> );
пернатый( <b>чайка</b> );	есть_живой( <b>чайка</b> );	летает( <b>чайка</b> );
пернатый( <b>пингвин</b> );	есть_живой( <b>пингвин</b> );	
пернатый( <b>велоцираптор</b> );		
птица( $X$ ) $\leftarrow$ пернатый( $X$ ), есть_живой( $X$ );		

Запрос  $\mathcal{Q}$ :

?птица( $X$ ), летает( $X$ )

В фактах перечислено всё, что известно про пернатость, современность и способность летать четырёх существ

Единственное правило программы трактуется так:

Если произвольно взятое существо  $X$  пернато и ещё не вымерло, то оно обязательно птица

Запрос к программе трактуется так:

Для какого существа  $X$  верно, что он птица и летает?



# ХЛП: декларативная семантика

## Пример

Программа  $\mathcal{P}$ :

пернатый( <b>орёл</b> );	есть_живой( <b>орёл</b> );	летает( <b>орёл</b> );
пернатый( <b>чайка</b> );	есть_живой( <b>чайка</b> );	летает( <b>чайка</b> );
пернатый( <b>пингвин</b> );	есть_живой( <b>пингвин</b> );	
пернатый( <b>велоцираптор</b> );		
птица( $X$ ) $\leftarrow$ пернатый( $X$ ), есть_живой( $X$ );		

Запрос  $Q$ :

?птица( $X$ ), летает( $X$ )

А на запрос «?птица(**орёл**), летает(**орёл**)» к  $\mathcal{P}$  есть только один правильный ответ:  $\epsilon$

*Потому что согласно  $\mathcal{P}$ , орёл — это действительно летающая птица*

А на запрос «?птица(**пингвин**), летает(**пингвин**)» к  $\mathcal{P}$  нет ни одного правильного ответа

*Потому что из  $\mathcal{P}$  невозможно достоверно заключить, что пингвин летает*



# ХЛП: декларативная семантика

**Другой пример:** снова Даша, Саша, Паша и пиво

Программа  $\mathcal{P}$ :

любит(даша, саша); любит(саша, пиво); любит(паша, пиво);  
любит(паша, X)  $\leftarrow$  любит(паша, Y), любит(X, Y);

Запрос  $\mathcal{Q}$ :

?любит(X, даша)

Единственное правило программы трактуется так:

Для любых произвольно взятых предметов X, Y, если и Паша, и X любят Y, то Паша любит X

Запрос к программе трактуется так:

Какие предметы X обязательно любят Дашу?

Существует ровно один правильный ответ на запрос  $\mathcal{Q}$  к программе  $\mathcal{P}$ :

{X/паша}

То есть {X/паша} — единственный ответ  $\theta$  на  $\mathcal{Q}$ , для которого верно

$S_{\mathcal{P}} \models \forall \dots (\text{любит}(X, \text{даша})\theta)$