

# Математическая логика и логическое программирование

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

## Блок 53

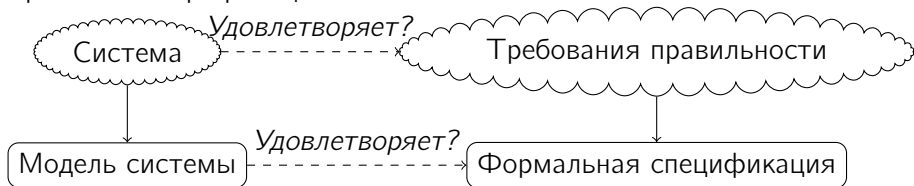
Модельные императивные программы  
Постановка задачи верификации программ

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

ВМК МГУ, 2023/2024, осенний семестр

# Вступление

Формальная верификация:



Обсудим то, как можно использовать **логику предикатов** для формальной верификации **императивных программ**

- ▶ Как может быть устроена математическая модель программы?
- ▶ Как можно записывать требования к программе с использованием логики предикатов?
- ▶ Как проверить, удовлетворяет ли модель программы записанным требованиям?

# Императивные программы: синтаксис

Далее считаются заданными **сигнатура**  $\sigma$  логики предикатов и множество **предметных переменных**  $\text{Var}$

Синтаксис императивных программ зададим следующей БНФ:

$\pi$	$::=$	$stmt \mid stmt \pi$	
$stmt$	$::=$	$\emptyset \mid$	(пустая команда)
		$x := t; \mid$	(присваивание)
		<b>if</b> $C$ <b>then</b> $\pi$ <b>else</b> $\pi$ <b>fi</b> $\mid$	(ветвление)
		<b>while</b> $C$ <b>do</b> $\pi$ <b>od</b>	(цикл)

Здесь:

- ▶  $\pi$  — программа
- ▶  $stmt$  — команда программы (или, по-другому, инструкция)
- ▶  $x \in \text{Var}$
- ▶  $t$  — выражение: произвольный терм, такой что  $\text{Var}_t \subseteq \text{Var}$
- ▶  $C$  — условие: произвольная формула без кванторов, такая что  $\text{Var}_C \subseteq \text{Var}$

# Императивные программы: синтаксис

В примерах будут использоваться

- ▶ сигнатура, содержащая общеизвестные арифметические символы, включая
  - ▶ константы  $0, 1$ ,
  - ▶ функциональные символы  $+^{(2)}, -^{(2)}, \cdot^{(2)}$ ,
  - ▶ предикатные символы  $=^{(2)}, >^{(2)}, \geq^{(2)}$
- ▶ арифметическая интерпретация  $Ar_X$  логики предикатов над множеством чисел  $X$ , в которой эти символы оцениваются «естественно» как соответствующие числа, операции и отношения, в том числе  $0, 1, +, -, \cdot, =, >, \geq$  — соответственно как
  - ▶ числа  $0$  и  $1$ ,
  - ▶ операции сложения, вычитания и умножения чисел и
  - ▶ отношения равенства чисел и их строгого и нестрогого неравенства в большую сторону

# Императивные программы: синтаксис

**Пример:** реализация алгоритма Эвклида  
вычисления наибольшего общего делителя чисел в переменных  $x$ ,  $y$

```
while  $\neg(x = y)$  do  
  if  $x > y$  then  
     $x := x - y;$   
  else  
     $y := y - x;$   
  fi  
od
```

## Императивные программы: операционная семантика

Значение программы — это **вычисляемая ей функция** преобразования входных данных в выходные данные

Для задания этой функции определим следующие понятия:

- ▶ **Состояние данных**: совокупность значений переменных, преобразуемая при выполнении программы
- ▶ **Состояние управления**: описание того, как текущее состояние данных будет изменяться программой в дальнейшем выполнении
- ▶ **Состояние вычисления**: состояние данных + состояние управления, то есть описание значений данных сейчас и в оставшейся части выполнения программы

## Императивные программы: операционная семантика

**Состояние данных** над переменными  $\text{Var}$  в интерпретации с предметной областью  $D$  — это отображение  $\sigma : \text{Var} \rightarrow D$

**Обозначение:**  $[x_1/\sigma(x_1), \dots, x_n/\sigma(x_n)]$ , если  $\text{Var} = \{x_1, \dots, x_n\}$

**Состояние управления** — это произвольная программа

**Состояние вычисления** — это пара  $\langle \pi \mid \sigma \rangle$ ,

где  $\pi$  — состояние управления и  $\sigma$  — состояние данных

$\Sigma$  — множество всех состояний данных

$\tilde{\Sigma}$  — множество всех состояний вычисления

$\sigma\{x \leftarrow d\}$  — состояние данных, получающееся из состояния данных  $\sigma$  в результате **присваивания** переменной  $x$  значения  $d$ :

$$\sigma\{x \leftarrow d\}(x) = d$$

$$\sigma\{x \leftarrow d\}(y) = \sigma(y), \text{ если } y \neq x$$

## Императивные программы: операционная семантика

Шаг выполнения программы в интерпретации  $\mathcal{I}$  описывается двуместным **отношением переходов**  $\xrightarrow{\mathcal{I}}$  на множестве  $\tilde{\Sigma}$ , состоящим из всех пар следующего вида:

- ▶  $\langle x := t; \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid \sigma \{x \leftarrow t\sigma\} \rangle$
- ▶  $\langle \text{if } C \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_1 \mid \sigma \rangle$ , если  $\mathcal{I} \models C\sigma$
- ▶  $\langle \text{if } C \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_2 \mid \sigma \rangle$ , если  $\mathcal{I} \not\models C\sigma$
- ▶  $\langle \text{while } C \text{ do } \pi \text{ od} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid \sigma \rangle$ , если  $\mathcal{I} \not\models C\sigma$
- ▶  $\langle \text{while } C \text{ do } \pi \text{ od} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi \text{ while } C \text{ do } \pi \text{ od} \mid \sigma \rangle$ , если  $\mathcal{I} \models C\sigma$
- ▶  $\langle \pi_1 \pi_2 \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi'_1 \pi_2 \mid \sigma' \rangle$ , если  $\langle \pi_1 \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi'_1 \mid \sigma' \rangle$
- ▶  $\langle \emptyset \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi \mid \sigma \rangle$



## Императивные программы: операционная семантика

**Трасса** программы  $\pi$  из состояния данных  $\sigma$  в интерпретации  $\mathcal{I}$  — это последовательность состояний вычисления вида

$$\langle \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_1 \mid \sigma_1 \rangle \xrightarrow{\mathcal{I}} \langle \pi_2 \mid \sigma_2 \rangle \xrightarrow{\mathcal{I}} \dots$$

**Вычислениями** программы называются бесконечные трассы и трассы, оканчивающиеся состоянием управления  $\emptyset$

Последнее состояние данных конечной трассы называется **результатом** этой трассы

Запись  $\langle \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}}^* \tilde{\sigma}$  будет означать, что существует конечная трасса программы  $\pi$  из состояния данных  $\sigma$  в интерпретации  $\mathcal{I}$ , оканчивающаяся состоянием вычисления  $\tilde{\sigma}$

Программой  $\pi$  в интерпретации  $\mathcal{I}$  вычисляется частичная функция  $\mathcal{I}[\pi] : \Sigma \rightarrow \Sigma$  следующего вида:

$$\mathcal{I}[\pi](\sigma) = \sigma' \quad \Leftrightarrow \quad \langle \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}}^* \langle \emptyset \mid \sigma' \rangle$$

## Императивные программы: операционная семантика (пример)

$$\text{Var} = \{x, y\}$$

$\pi = \mathbf{while} \neg(x = y) \mathbf{do if} x > y \mathbf{then} x := x - y; \mathbf{else} y := y - x; \mathbf{fi od}$

Вычисление  $\pi$  из  $[x/2, y/4]$  в  $Ar_{\mathbb{Z}}$ , где  $\mathbb{Z}$  — множество всех целых чисел:

$$\langle \pi \mid [x/2, y/4] \rangle$$

$$Ar_{\mathbb{Z}} \downarrow \quad \text{т.к. } Ar_{\mathbb{Z}} \models \neg(x = y)[x/2, y/4]$$

$$\langle \mathbf{if} x > y \mathbf{then} x := x - y; \mathbf{else} y := y - x; \mathbf{fi} \pi \mid [x/2, y/4] \rangle$$

$$Ar_{\mathbb{Z}} \downarrow \quad \text{т.к. } Ar_{\mathbb{Z}} \not\models (x > y)[x/2, y/4]$$

$$\langle y := y - x; \pi \mid [x/2, y/4] \rangle$$

$$Ar_{\mathbb{Z}} \downarrow \quad \text{т.к. } [x/2, y/4]\{y \leftarrow (y - x)[x/2, y/4]\} = [x/2, y/2]$$

$$\langle \emptyset \pi \mid [x/2, y/2] \rangle$$

$$Ar_{\mathbb{Z}} \downarrow$$

$$\langle \pi \mid [x/2, y/2] \rangle$$

$$Ar_{\mathbb{Z}} \downarrow \quad \text{т.к. } Ar_{\mathbb{Z}} \not\models \neg(x = y)[x/2, y/2]$$

$$\langle \emptyset \mid [x/2, y/2] \rangle$$

Результат этого вычисления:  $[x/2, y/2]$

Следовательно,  $Ar_{\mathbb{Z}}[\pi]([x/2, y/4]) = [x/2, y/2]$

# Задача верификации программ

Требования правильности выполнения программы могут быть записаны как два отношения на состояниях данных:

- ▶ **предусловие**  $\varphi$ ,  
задающее общий вид **допустимых** входных данных
- ▶ **постусловие**  $\psi$ ,  
описывающее устройство **правильных** выходных данных

Принято рассматривать два вида правильности выполнения программы относительно заданных предусловия и постусловия:

- ▶ **частичная корректность**: результат любого конечного вычисления программы на допустимых входных данных правилен
- ▶ **полная корректность**: любое вычисление программы на допустимых входных данных конечно, и результат этого вычисления правилен

Остановимся подробнее на частичной корректности программ

# Задача верификации программ

Тройка Хоара (по-другому — триплет Хоара) — это запись вида  $\{\varphi\}\pi\{\psi\}$ , где

- ▶  $\varphi$  — формула логики предикатов, называемая **предусловием**
- ▶  $\pi$  — программа
- ▶  $\psi$  — формула логики предикатов, называемая **постусловием**

Триплет  $\{\varphi\}\pi\{\psi\}$  **истинен в интерпретации**  $\mathcal{I}$  ( $\mathcal{I} \models \{\varphi\}\pi\{\psi\}$ ),

если для любых состояний данных  $\sigma$ ,  $\sigma'$  верно следующее:

если  $\mathcal{I} \models \varphi\sigma$  и значение  $\sigma' = \mathcal{I}[\pi](\sigma)$  определено, то  $\mathcal{I} \models \psi\sigma'$

Программа  $\pi$  **частично корректна** в интерпретации  $\mathcal{I}$  относительно предусловия  $\varphi$  и постусловия  $\psi$ , если  $\mathcal{I} \models \{\varphi\}\pi\{\psi\}$

**Задача верификации императивных программ:**

для заданных программы  $\pi$ , предусловия  $\varphi$ , постусловия  $\psi$  и интерпретации  $\mathcal{I}$  проверить справедливость соотношения  $\mathcal{I} \models \{\varphi\}\pi\{\psi\}$