# 1 Семантика программы

## 1.1 Семантика программы

Предположим, что нам дана некоторая программа $\pi$ (написанная на некотором языке программирования). Каково значение $\pi$? Как выразить его математически? Для этого введем понятие **семантики** программы. Далее рассмотрим простой императивный язык программирования с целью иллюстрации понятий, связанных с семантикой программы.

## 1.2 Простой язык программирования

### Простой язык программирования (SPL)

Определим некоторый очень простой язык программирования $SPL$.

- в $SPL$ существует только один тип - все переменные имеют целочисленные значения.
- мы можем выполнять основные арифметические операции: сложение, умножение, вычитание и деление над выражениями. Также у нас есть числовые константы.
- мы можем сравнивать арифметические выражения и комбинировать эти сравнения с логическими операторами: $(x < 2*y) \wedge (x \neq 1)$
- элементарной операцией в программе является присваивание: $x := e$, где $x$ - переменная, а $e$ - некоторое арифметическое выражение вида $(y * 2 + 1) * z$
- в общем случае программа состоит из операций и операторов:
  - $\pi; \rho$ - последовательная композиция $\pi$ и $\rho$
  - $if(cond)then\{\pi\}else\{\rho\}$ - Условный оператор
  - $while(cond)do\{\pi\}$ - цикл while

## 1.3 Семантика типов данных

### Математическая семантика типов данных

В общем случае, для данной системы типов, её **математическая семантика** состоит из:

- для каждого типа $\tau$ множество всех значений $D_\tau$ - **область допустимых значений** типа $\tau$
- для каждого арифметического оператора $\star$, действующего на типы $\tau_1, \ldots, \tau_n$ и возвращающего значение типа $\tau_0$, **интерпретация** этого оператора, а именно отображение $\star : D_{\tau_1} \times \ldots \times D_{\tau_n} \to D_{\tau_0}$

#### Семантика типов в $SPL$

В $SPL$ существует только один тип: $int$, следовательно, его область определения - это просто множество целых чисел $Z$. Все арифметические операторы: $+, \cdot, -, /$ интерпретируются как стандартные операции над целыми числами, аналогично для равенства и отношений $<, \leq, >, \geq$. Объединяя множество и все интерпретации, получим алгебраическую структуру $\mathbb{Z} = (Z, \{+, \cdot, -, /, <\})$ в качестве семантики для типов данных в $SPL$.

## 1.4 Состояние программы

Дана $SPL$ программа $\pi$, обозначим все переменные, входящие в $\pi$ как $V(\pi)$.

#### Состояние программы

Дана $SPL$ программа $\pi$, её **состояние** - это отображение:

$$s : V(\pi) \to D_{int} = Z$$

Если существует больше одного типа, переменные становятся типизированными, и это отображение должно учитывать их типизацию. Множество всех состояний программы $\pi$ обозначается как $SP(\pi)$.

Состояние программы $\pi$ описывает значения всех её переменных. Если нам дано состояние $s$ и некоторое выражение $e$, то **значение** $e$ **в состоянии** $s$ однозначно определено и обозначается как $e[s]$. Для любого состояния $s$, переменной $x$ и значения $a$, определим состояние $s_a^x$ следующим образом:

$$s_a^x(y) = \begin{cases} s(y) & , y \neq x \\ a & , y = x \end{cases}$$

## 1.5 Семантика операций

### Семантика операций

Дана программа $\pi$, её **семантика** $< \pi >$ - это отображение (частичное!)

$$< \pi >: SP(\pi) \to SP(\pi)$$

Значение: начиная с состояния $s_0$, программа $\pi$ либо не завершится, и в этом случае $< \pi > (s_0)$ не определена, либо завершится с состоянием $s_1$, и в этом случае $< \pi > (s_0) = s_1$.

Неформально, семантика программы исчерпывающе описывает ее поведение и, таким образом, выражает её значение.

## 1.6 Истинность формулы в состоянии

Как было отмечено выше, математическая семантика типов данных в $SPL$ в частности и в любом другом языке программирования в целом - это просто *фиксированная* (многокомпонентная) структура, а любое состояние - это просто означивание переменных. Следовательно, может возникнуть вопрос, истинна ли некоторая формула в этой фиксированной структуры при данном означивании переменных.

Например в $SPL$, семантика типов данных - это фиксированная структура $\mathbb{Z}$, следовательно,

$$s \models \phi \Leftrightarrow \mathbb{Z} \models \phi[s]$$

Таким образом, поскольку любая логическая комбинация сравнений выражений в $SPL$ может быть рассмотрена просто как формула, не содержащая кванторов, для любого такого выражения либо $cond$, либо $s \models cond$ или $s \not\models cond$.

## 1.7 Семантика операций в $SPL$

### Семантика операций в $SPL$

Дана программа $\pi$, её семантика $< \pi >$ определяется индукцией по построению $\pi$:

1. $< x := e > (s) = s^x_{e[s]}$

2. $< \pi; \rho > (s) =< \pi > \circ < \rho > (s) =< \rho > (< \pi > (s))$

3. $< if(cond)then\{\pi\}else\{\rho\} > (s) = \begin{cases} < \pi > (s) & , s \models cond \\ < \rho > (s) & , s \not\models cond \end{cases}$

4. $< while(cond)do\{\pi\} > (s) = \underbrace{< \pi > \circ \ldots \circ < \pi >}_{n}(s) = s'$, где $n$ - (минимальный) такой, что $s' \not\models cond$ для всех $k < n \underbrace{< \pi > \circ \ldots \circ < \pi >}_{k}(s) \models cond$

Теперь у нас есть строгое математическое описание языка $SPL$

## 1.8 Выполнимость тройки Хоара в $SPL$

Теперь мы можем строго определить понятие *выполнимости* или *истинности* тройки Хоара (частичной корректности) в $SPL$.

### Определение

Дана тройка Хоара $\{\phi\}\pi\{\psi\}$, где $\pi$ - $SPL$ программа, будем говорить, что она **истинна** или **выполняется**, записывается как

$$\models \{\phi\}\pi\{\psi\}$$

тогда и только тогда, когда для любого состояния $s$, если $s \models \phi$ и $< \phi > (s)$ определено, то $< \phi > (s) \models \psi$

# 2 Корректность и полнота аксиоматической семантики.

## 2.1 Корректность аксиоматической семантики для $SPL$

Как и в классической логике, мы хотим убедиться, что все тройки Хоара, являющиеся выводимыми в аксиоматической семантике для $SPL$, истинны.

### Теорема (корректность аксиоматической семантики $SPL$)

Для любой тройки Хоара $\{\phi\}\pi\{\psi\}$, если существует её дерево вывода, все листья которого, являющиеся формулами, тождественно истинны, то

$$\models \{\phi\}\pi\{\psi\}$$

**Доказательство**

Как всегда, докажем эту теорему индукцией по высоте дерева вывода. Сначала необходимо проверить, что аксиома присваивания всегда истинна, затем проверить все остальные правила вывода. Проверим, что $\models \{(\phi)^x_e\}x := e\{\phi\}$. Рассмотрим некоторое состояние $s$, такое, что $s \models (\phi)^x_e$. Тогда, если заменить каждое вхождение $e$ в $(\phi)^x_e$ значением $e[s]$, истинность формулы сохранится. Таким образом, $s^x_e \models \phi$, потому что в этой формуле все вхождения $x$ заменены значениями $e[s]$. Предположим, что $\models \{\phi\}\pi\{\chi\}$ и $\models \{\chi\}\rho\{\psi\}$. Необходимо проверить, что $\models \{\phi\}\pi;\rho\{\psi\}$. Действительно, возьмём некоторое состояние $s_0 \models \phi$ такое, что $< \pi;\rho > (s)$ определено: если $s_1 =< \pi > s_0$, то $s_1 \models \chi$, следовательно, если $s_2 =< \rho > (s_1)$, то $s_2 \models \psi$. Но $s_2 =< \pi;\rho > (s_0) \models \psi$, ч.т.д. Остальные случаи доказываются аналогично.$\square$

## 2.2 Полнота аксиоматической семантики для $SPL$

**Теорема (полнота аксиоматической семантики $SPL$)**

Для любой тройки Хоара $\{\phi\}\pi\{\psi\}$, если $\models \{\phi\}\pi\{\psi\}$, то существует её дерево вывода, все листья которого, являющиеся формулами, тождественно истинны.

Это доказательство намного сложнее, чем доказательство теоремы о корректности.