
# СЕМАНТИКА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ (СЯП)

**Варшавский Павел Романович**

e-mail: VarshavskyPR@mpei.ru
varp@appmat.ru

# ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

**Целью освоения дисциплины** является:

- формирование способности к проведению рассуждений о программах;
- изучение конструкций, определяющих семантику программ;
- обучение способам анализа программ, включая верификацию программ, распознавание свойств программ (завершаемость, эквивалентность, зацикливание);
- изучение операционной семантики программ;
- изучение денотационной семантики программ.

**Задачи дисциплины**:

- освоение основ теории семантики программ;
- изучение операционной семантики программ;
- изучение денотационной семантики программ;
- приобретение навыков описания денотационной семантики программ, решения задач анализа программ (верификация, распознавание завершенности программы, эквивалентность программ).

# РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА


1. **Кораблин Ю.П. Семантические методы анализа программ : учебное пособие по курсу "Семантика языков программирования" по направлению 01.03.02 "Прикладная математика и информатика" / Ю.П. Кораблин, Нац. исслед. ун-т "МЭИ" (НИУ"МЭИ") . –М.: Изд-во МЭИ, 2019. – 68 с.**
2. Кораблин Ю.П. Семантика языков программирования: Учебное пособие. – М.: Изд-во МЭИ, 1992. – 100 с.
3. Кораблин Ю.П. Семантика языков распределенного программирования: Учебное пособие. – М.: Изд-во МЭИ, 1996. – 102 с.
4. В. Н. Вагин, Е. Ю. Головина, А. А. Загорянская, М. В. Фомина- "Достоверный и правдоподобный вывод в интеллектуальных системах", (2-е изд., испр. и доп.), Издательство: "Физматлит", Москва, 2008 - (712 с.)

# ВВЕДЕНИЕ

В теории языков программирования **семантика** – это область, связанная с математическим изучением смысла программ.

Смысл программы заключается в формальном описании процесса вычисления по программе с помощью математической модели (вычислительного автомата) как «абстрактной машины».

Формальная семантика помогает при разработке интерпретаторов и компиляторов для языков программирования.

Знание семантики способствует лучшему пониманию процесса вычисления по программе. Поскольку формальная семантика оперирует математическими объектами, становится возможным математический анализ программ, включающий доказательство корректности и получение асимптотических оценок вычислительной сложности программы.

# ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О СЕМАНТИКЕ ПРОГРАММ

Имеется различные подходы к определению семантики программ:

**Операционный подход** полезен для реализации языков. В данном случае исполнение программы описывается непосредственно как процесс вычисления на абстрактном автомате, а не путем трансляции.

**Денотационный подход** предполагает, что фразы программы интерпретируются как денотаты, т.е. математические объекты, определенным образом связанные друг с другом. Денотаты описываются с помощью математической нотации, которая в свою очередь формализуется как денотационный метаязык. Денотационные семантические описания могут представлять трансляции с языка программирования в денотационный метаязык. Такие описания используются как основа для построения интерпретаторов и компиляторов.

# ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О СЕМАНТИКЕ ПРОГРАММ

**Аксиоматический подход** полезен для разработки программ. При аксиоматическом подходе с программой связываются логические аксиомы. Используя процесс логического вывода, доказывают различные свойства процесса вычисления по программе такие, как завершенность процесса вычисления, корректность отношения «вход–выход» и другие свойства.

**Алгебраический подход** полезен на теоретическом уровне. Он обеспечивает общий фундамент для теории языков программирования, позволяя на единой математической модели рассматривать такие различные концепции как синтаксис, семантика и типы данных.

# ФОРМАЛЬНАЯ СЕМАНТИКА

Необходимость формальной семантики была обнаружена в середине прошлого века. В середине 60-х годов прошлого века венская лаборатория IBM разработала первый технический язык описания семантики, получивший название **VDL (Vienna Definition Language)**.

VDL позволяет задавать операционную семантику, где язык программирования описывается посредством гипотетического выполнения на некоторой абстрактной машине. Такое описание может быть точным и полным, но оно довольно сложное, что затрудняет его понимание.

Сложность описания языка в VDL отражает не столько сложность реальной реализации, как сложность абстрактной машины, VDL был использован в ряде проектов и, в частности, для описания семантики языков *Алгол-60* и *PL-1*. Эти попытки доказали возможность формального описания семантики языков программирования.

# ОПЕРАЦИОННЫЙ ПОДХОД

При задании операционной семантики языка программирования предварительно определяется «абстрактная машина», которая характеризуется множеством состояний и набором простейших команд.

Машина может быть определена спецификацией того, как изменяется ее состояние каждой командой из заданного набора команд. Затем семантика конкретного языка программирования определяется в терминах этой машины, т.е. семантическое описание языка программирования определяет трансляцию конструкций языка в коды этой машины.

Гипотетическое выполнение этого кода на абстрактной машине, находящейся вначале в исходном состоянии, позволяет получить точный результат.

# ПРИМЕР ОПЕРАЦИОННОГО ОПРЕДЕЛЕНИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ

Зафиксируем вначале некоторый простой язык программирования, который назовем языком **L**.

При задании синтаксиса языка **L** будем использовать следующие обозначения:

**Exp** – множество выражений;
**Bexp** – множество булевых выражений;
**Com** – множество команд (программ);
**X** – множество переменных.

Через **E**, **B**, **C**, **x**, возможно с индексами, обозначаются типичные представители соответствующих множеств, т.е. **E** ∈ **Exp**, **B** ∈ **Bexp**, **C** ∈ **Com**, **x** ∈ **X**.

# СИНТАКСИС ЯЗЫКА L

Синтаксис языка **L** задается следующим образом:

$$C :: x:=E \mid C_1; C_2 \mid \underline{\text{if}}\ B\ \underline{\text{then}}\ C_1\ \underline{\text{else}}\ C_2\ \underline{\text{fi}} \mid \underline{\text{while}}\ B\ \underline{\text{do}}\ C\ \underline{\text{od}}$$

При задании синтаксических конструкций языка **L** перевернутое слово (**od**, **fi**) используется для обозначения конца действия соответствующего оператора (**do**, **if**).

Под состоянием абстрактной машины мы будем понимать вектор состояния ($\sigma$), сопоставляющий переменным программы значения из области их значений. Таким образом, вектор состояния может рассматриваться как функция, которая сопоставляет каждой переменной программы значение из некоторой области значений.

# ОПЕРАЦИИ АБСТРАКТНОЙ МАШИНЫ

В качестве операций абстрактной машины определим две операции:

– операция **Out** определяет заключительное состояние конечной последовательности состояний, т.е.

$$\mathbf{Out}\ (\sigma_1 \hat{} \sigma_2 \hat{} \ \ldots \ \hat{} \sigma_n) = \sigma_n\ ,$$

где символ “**^**” используется для обозначения конкатенации последовательности состояний.
Функция **Out** не определена, если последовательность бесконечна.

– операция **Eval** определяет значение выражения, соответствующего текущему состоянию.

# ДОПОЛНИТЕЛЬНЫЕ СОГЛАШЕНИЯ

1) Через $\sigma\{x\}$ будем обозначать значение переменной **x** в состоянии $\sigma$;

2) Если **v** – некоторое значение и **x** – переменная, то через $\sigma[v/x]$ будем обозначать подстановку значения **v** в вектор $\sigma$ вместо **x**, определяемую следующим образом:

$$\sigma[v/x]\{x\} = v,$$
$$\sigma[v/x]\{y\} = \sigma\{y\} \text{ для всех } y \neq x.$$

3) Условное выражение $p \rightarrow e_1, e_2$, где **p** – всюду определенное условие, $e_1, e_2$ – выражения, определяется следующим образом:

$$p \rightarrow e_1, e_2 = \begin{cases} e_1, \text{ если значение } p \text{ истинно,} \\ e_2, \text{ в противном случае.} \end{cases}$$

# ОПЕРАЦИОННОЕ ОПРЕДЕЛЕНИЕ ЯЗЫКА L

Зададим теперь операционное определение языка **L** с помощью функции **O** от двух аргументов – **текста программы** и **вектора состояния**, представляющего текущее состояние вычислений.

Функция **O** имеет следующий вид

$$O: Com \times S \to S^*,$$

где **S** – множество состояний, **S*** – множество последовательностей (возможно, бесконечных) состояний.

# ОПЕРАЦИОННОЕ ОПРЕДЕЛЕНИЕ ЯЗЫКА L

Для задания операционной семантики языка **L** достаточно определить функцию **O** для всех пунктов определения языка **L**:

$O(x:=E)(\sigma) = \sigma[Eval(E)(\sigma)/x]$

$O(C_1; C_2)(\sigma) = O(C_1)(\sigma)\hat{}O(C_2)(Out(O(C_1)(\sigma)))$

$O(\underline{if}\ B\ \underline{then}\ C_1\ \underline{else}\ C_2\ \underline{fi})(\sigma) = Eval\ (B)(\sigma) \rightarrow O(C_1)(\sigma), O(C_2)(\sigma)$

$O(\underline{while}\ B\ \underline{do}\ C\ \underline{od})(\sigma) = Eval(B)(\sigma) \rightarrow O(C)(\sigma)\hat{}O(\underline{while}\ B\ \underline{do}\ C\ \underline{od})\ (Out(O(C)(\sigma))), \sigma$

В этом определении $O(C)(\sigma)$ означает, что функция **O** применяется к аргументам **C** и $\sigma$, являющимися типичными представителями множеств **Com** и **S** соответственно. В дальнейшем будем опускать скобки аргумента $\sigma$ там, где это не будет вызывать недоразумения, т.е. вместо $O(C)(\sigma)$ будем писать $O(C)\sigma$.

Приведенное определение **O** согласуется с обычным операционным пониманием того, как вычислять команды.