**ТЕОРІЯ КОМПІЛЯТОРІВ**

Лекція 10

# АКСІОМАТИЧНА СЕМАНТИКА МОВ ПРОГРАМУВАННЯ

**2020**

**Повний текст лекції буде розміщений на сайті baklaniv.at.ua**

Даний підхід ґрунтується на застосуванні апарату числення предикатів і теорії доказів.

Семантику кожної конструкції мови визначають як певний набір аксіом або правил виведення, який використовується для виведення результатів виконання цієї конструкції.

Щоб зрозуміти сенс всієї програми (тобто розібратися, що і як вона робить), ці аксіоми і правила виводу слід застосовувати так само, як при доказі звичайних математичних теорем.

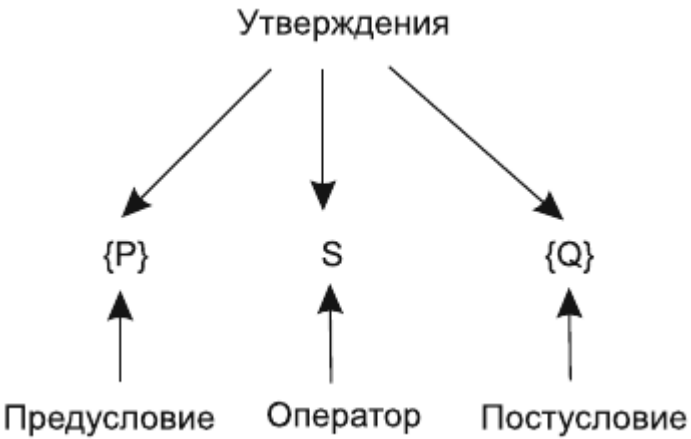
Аксіоми правила виведення використовують для оцінки значень змінних після виконання кожного оператора програми. У підсумку, коли програма виконана, формується доказ того, що обчислені результати задовольняють заданим обмеженням на їх значення щодо вхідних значень.

Словом, доводиться, що вихідні дані є коректними результатами обчислення функцій, обробних відповідні вхідні дані. Прикладом описаного підходу вважається метод *аксіоматичної семантики*, створений Тоні Гоаром (Хоаром).

**Тоні Гоар (народився 11 січня 1934)** - серед його досягнень — розробка логіки Гоара, наукової основи для конструювання коректних програм, яка використовується для визначення та розробки мов програмування.

Отже, *метод аксіоматичній семантики* служить для доведення правильності програми. Доказ правильності демонструє, що програма виконує обчислення, описувані її специфікацією. При доказі кожен оператор оточується логічними виразами (умовами), які задають обмеження на програмні змінні. Вони використовуються для визначення сенсу оператора.

Програма представляється у вигляді хоаровських трійок:



Для отримання трійки до кожного оператору потрібно додати зліва передумову, а праворуч - постумову.

*Умови* (твердження, висловлювання) записуються в фігурних дужках.

*Передумова* описує обмеження на програмні змінні перед виконанням оператора і позначається як {P}.

*Постумова (або післяумова)* задає нові обмеження на ці змінні після виконання оператора і позначається як {Q}.

*Прийнято*: передумова обчислюється на основі постумови операторів.

Серед безлічі передумов особливу роль відіграє *слабка передумова*.

*Слабка передумова* - це найменше обмежуюча передумова, яка гарантує правильність пов'язаної постумови.

Розглянемо приклад хоаровской трійки:

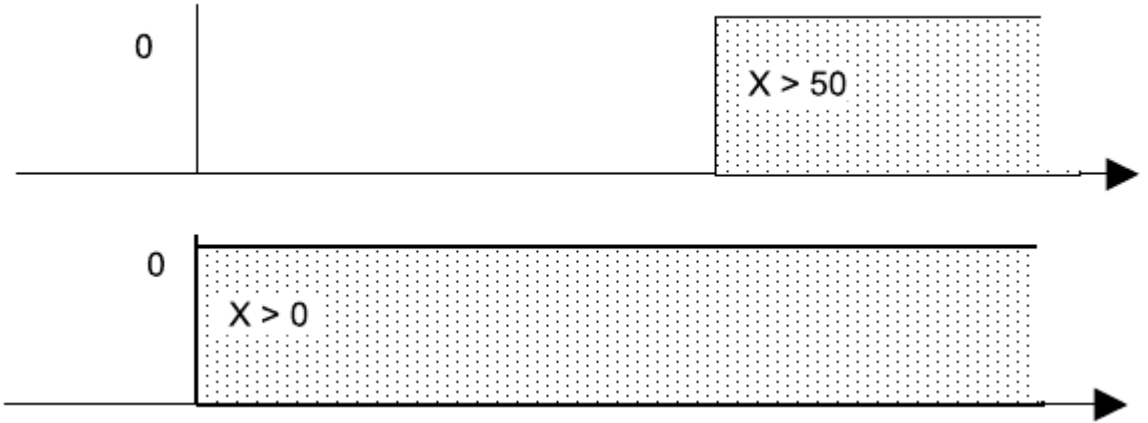
```
{x > 10}   result := 5 * x + 1   {result > 1}
{x > 50}
{x > 1000}
```

У прикладі записано, що при правильній роботі оператора **result: = 5 * x + 1** на результат накладається обмеження **result> 1**.

Тут перераховані три можливих передумови, що накладають різні обмеження на значення «вхідної» змінної x, але слабка передумова має наступний вигляд:

**{X> 0} - слабка передумова**

Слабка передумова все ще забезпечує коректну роботу оператора, при якому дотримується постумови, але при цьому надає найбільшу «свободу» змінної з передумови. Ослаблення передумови можна проілюструвати таким чином.




Ослаблення передумови

Тут показана числова вісь, на якій штрихуванням позначені області дозволених значень для змінної x. Бачимо, що слабішій передумові відповідає максимальна область допустимих значень.

У аксіоматичної семантиці прийнята наступна схема доказу програми:

1. Слабкі передумови оператора обчислюються на базі його постумови.

2. Виконується зворотний прохід по програмі - доказ починається з останнього оператора, завершується першим оператором.

3. Перша передумова задає умови, при яких програма формує останню постумову (бажані результати).

Для деяких операторів обчислення слабшої передумови виконується просто і може бути визначено аксіомою. У більшості випадків слабку передумову потрібно обчислювати за правилом виведення.

*Аксіома* - це логічне твердження, що приймається без доведення в силу безпосередньої переконливості.

*Правило виводу* - це метод отримання істинного твердження на основі інших тверджень.

Аксіоматична семантика вимагає, щоб *для кожного оператора мови програмування була визначена або аксіома, або правило виведення*.

**Аксіома присвоювання**

Ця аксіома «обслуговує», тобто дозволяє «довести» оператор присвоювання. Нехай **x: = E** - оператор присвоювання з постумовою Q. Тоді його передумова, що визначається за аксіомою

$$P = Q_{x \to E}$$

означає, що P обчислюється як Q, в якому всі входження змінної x заміщуються на E, тобто на вираз із правої частини оператора.

Аксіома записується у вигляді такої трійки:

$$\{P = Q_{x \to E}\} \qquad x := E \qquad \{Q\}$$

Повторимо, що аксіоматична семантика розробляється для доведення правильності програм. З цієї точки зору оператор присвоювання з передумовою і постумовою може розглядатися як теорема. Якщо аксіома присвоювання при застосуванні до постусловія і оператора виробляє «правильну» передумову, то теорема доведена.

Рассмотрим примеры применения аксиомы присваивания, то есть примеры до-казательства теорем по поводу операторов присваивания.

Пусть исходные данные для доказательства теоремы имеют вид:

**a := b/2 + 1 {a < 50}**

Тогда для заданного оператора **a := b/2 + 1** слабейшее предусловие вычисляется подстановкой в постусловие **{a < 50}** вместо **a** выражения **b/2 + 1**:

**{b/2 + 1 < 50} ⇒ {b < 98}**

Как использовать этот результат? Достаточно проверить соблюдение предусловия и постусловия в ходе реальных вычислений.

Еще один пример. Для заданной пары

```
y := 5 * x - 10 {y > 95}
```

слабейшее предусловие равно:

$$\{5 * x - 10 > 95\} \Rightarrow \{x > 17\}$$

## Правило консеквенции (упрощения)

Рассмотрим хоаровскую тройку

**{a > 7} a := a – 2 {a > 0}.**

В этом случае утверждение, вырабатываемое по аксиоме присваивания, не равно предусловию. И все же очевидно, что из **{a > 7}** следует **{a > 2}**. Для использования этого факта в доказательстве нам потребуется правило вывода «*правило консеквенции*», иначе называемое *правилом упрощения*.

Общая форма правила вывода имеет вид

$$\frac{S1,\ S2,\ldots,\ SN}{S}$$

где утверждается, что если исходные высказывания $S1$, $S2$, …, $SN$ истинны, то может быть выведена истина для заключения $S$.

Правило *консеквенции* записывается в форме

$$\frac{\{P\}S\{Q\},\ P' \Rightarrow P, Q \Rightarrow Q'}{\{P'\}S\{Q'\}},$$

где символ ⇒ обозначает «следует», а $S$ — любой оператор программы.

Правило означает: если логическое высказывание $\{P\}\ S\ \{Q\}$ — истина, а из утверждения $P'$ следует утверждение $P$ и из утверждения $Q$ следует утверждение $Q'$, то может быть выведено истинное логическое высказывание $\{P'\}\ S\ \{Q'\}$.

Это правило говорит, что *постусловие* может быть всегда *ослаблено*, а *предусловие* может быть всегда *усилено*.

Такое ослабление/усиление очень полезно в доказательстве программ. Например, это позволяет завершить доказательство начального логического высказывания.

Примем $P = \{a > 2\}$, $P' = \{a > 7\}$, $Q = Q' = \{a > 0\}$, тогда

$$\{a > 2\}\ a := a - 2\ \{a > 0\}, \{a > 7\} \Rightarrow \{a > 2\}, \{a > 0\} \Rightarrow \{a > 0\}.$$

$$\{a > 7\}\ a := a - 2\ \{a > 0\}$$

Первое исходное утверждение **{a > 2} a := a – 2 {a > 0}** доказывается по аксиоме присваивания. Второе и третье утверждения очевидны. Следовательно, по правилу консеквенции, такое упрощение является истинным.

## Правило вывода для последовательности операторов

Слабейшее предусловие для последовательности операторов не может описываться аксиомой, так как предусловие зависит от конкретных разновидностей операторов последовательности. В этом случае предусловие может быть описано только с помощью правила вывода.

Пусть ***S1*** и ***S2*** — два смежных оператора, имеющих следующие пред- и постусловия: ***{P1} S1 {P2}***, ***{P2} S2 {P3}***. Отметим, что постусловие первого оператора совпадает с предусловием второго. Логически это вполне объяснимо: первый оператор формирует данные для успешного старта второго оператора.

Тогда правило вывода для последовательности из двух операторов принимает вид:

$$\frac{\{P1\}\ S1\ \{P2\},\{P2\}\ S2\ \{P3\}}{\{P1\}\ S1;S2\ \{P3\}}.$$

Для данного примера $\{P1\}\ S1;\ S2\ \{P3\}$ правило вывода предписывает аксиоматическую проверку

последовательности операторов $S1$; $S2$: они должны иметь общее утверждение, иначе ограничение на переменные, — это $\{P2\}$. Мало того, предусловие первого оператора должно стать общим предусловием последовательности, а постусловие второго — общим постусловием.

Правило вывода для последовательности имеет очень большое практическое значение. Именно оно обеспечивает цепочку автоматического доказательства операторов программы. Для запуска цепочки достаточно знать лишь требуемый результат программы, то есть постусловие последнего оператора.

При доказательстве последнего оператора будет вычислено его слабейшее предусловие. Это даст возможность перейти к доказательству предпоследнего оператора. Ведь правило вывода для последовательности гласит: предусловие последнего оператора равно постусловию предпоследнего. Далее шаги перехода от оператора к оператору повторяются.

Процесс завершается доказательством первого оператора программы, то есть вычислением его слабейшего предусловия, которое определяет ограничения на исходные данные программы.

**Наступна лекція №11 буде присвячена темі “Опис семантики контекстно-вільних мов методом математичної індукції”.**