1 Корректность программы

1.1 Проблема корректности программы

Предположим, что нам дана некоторая программа π (написанная на некотором языке программирования).

Проблема

Корректна ли π, т.е. соответствует ли она нашим *ожиданиям*?

Такая постановка задачи носит неформальный характер, и обычно наши ожидания также неформальны и приводятся, например, в виде набора технических требований к программе на естественном языке.

Решение

Чтобы проверить корректность π относительно множества технических спецификаций S обычно используется набор **тестов**, т.е. набор предопределенных начальных состояний программы (представленный, например, в виде набора входных параметров) вместе с набором соответствующих ответов, которые ожидаются от π.

НО: можем ли мы быть *уверены*, что набор тестов является исчерпывающим, т.е. программа *не содержит ошибок*?....

1.2 Математическая природа программ

Существуют ли другие способы переформулировать проблему корректности?

Информационная природа π

Существует фундаментальное наблюдение, что программа π, в действительности, скорее является *информационным* объектом, нежели физическим.

Существует очень важное следствие. этого наблюдения: любая программа π подчиняется законам *математики и логики*, так же, как физические объекты подчиняются физическим законам. Следовательно, мы можем переформулировать природу программы:

Математическая природа π

Любая программа π может рассматриваться как **математический** объект, к которому могут быть применены все математические методы.

1.3 Формальная корректность

Итак, мы можем переформулировать проблему корректности программы, используя понятия математической логики. Но сначала разделим программы на два класса:

- **завершающиеся** (такие как компилятор, конвертер любых данных и т. д.)
- **не завершающиеся** (такие как ОС, IDE и т. д.)

Далее будем говорить только о *завершающихся* программах.

Проблема: формальная корректность

Дана программа π, и некоторое множество входных данных, соответствующее формуле ϕ (**предусловие**), будут ли выходные данные соответствовать формуле ψ (**постусловие**)?

Отметим, что здесь мы *формализовали* технические требования к программе, используя *формулы* логики предикатов. В сокращённых обозначениях проблема корректности записывается как:

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

и называется **тройка Хоара** или **условие частичной корректности**.

1.4 Пример условия частичной корректности: целочисленный квадратный корень

Представим, что нам дана программа π_{sqr}, вычисляющая целую часть квадратного корня натурального числа. Пусть n - целочисленный входной параметр этой программы,, а m - возвращаемое значение. Тогда мы можем *формализовать* требования к этой программе путем явного представления соответствующих предусловий и постусловий.

- $\phi = n \geq 0$ - предусловие для π_{sqr}

- $\psi = (m^2 \leq n) \wedge ((m+1)^2 > n)$ - постусловие для π_{sqr}

Отметим, что мы можем указать другое постусловие с тем же значением. Например, мы могли бы использовать формулу с кванторами:

$$\psi' = (m^2 \leq n) \wedge \forall x((x^2 \leq n) \rightarrow (x \leq m))$$

1.5 Выполнение условий корректности

Теперь мы можем ввести понятие *выполнения* или *истинности* тройки Хоара (частичной корректности).

Определение

Дана тройка Хоара $\{\phi\}\pi\{\psi\}$ (или условие частичной корректности), будем говорить, что оно **истинно** или **выполнено**, тогда и только тогда, когда для любых входных данных, соответствующих ϕ, программа π либо не завершается, либо, в случае её завершения, возвращаемое значение соответствует ψ.

Отметим, что это определение допускает, что π *не завершается* на входных данных, соответствующих предусловиям. Если мы хотим установить более строгое ограничение на корректность, и потребовать, чтобы для любых данных, соответствующих предусловиям, программа π завершалась, то эти ограничения называются **условиями полной корректности** и обозначаются как:

$$[\phi]\pi[\psi]$$

2 Неформальная верификация

2.1 Метод Флойда - определение частичной корректности

Итак, дана тройку Хоара $t = \{\phi\}\pi\{\psi\}$, как мы можем проверить, является ли она истинной? Процесс определения истинности тройки Хоара называется **верификацией** программы. Существуют различные способы верификации. Например, мы можем использовать *неформальный* метод *верификации* t.

Метод Флойда (определение частичной корректности)

1. построить блок-схему π
2. определить множество **контрольных точек** внутри блок-схемы. Вход и выход π должны входить в это множество. Внутри любого цикла блок-схемы должна быть хотя бы одна контрольная точка..
3. определить **инвариант** (некоторую формулу) для каждой контрольной точке. Инвариант для входа - предусловие, а для выхода - постусловие.
4. для любой **пары контрольных точек**, связанных в блок-схеме и не имеющих контрольных точек между ними, **доказать**, что если инвариант первой контрольной точки выполняется, то инвариант последующей точки также будет выполняться.

2.2 Пример: корректность программы для вычисления целочисленного квадратного корня

Рассмотрим следующую программу для вычисления целочисленного квадратного корня. Напомним, что $n : int$ является входным параметром, а $x : int$ - выходным. Также нам потребуется локальная переменная $y : int$.

```
sqr(n : int) : int {
  x : int; y : int;
  x := 0; y := 0;
  while (y <= n) {
      y := y + x + x + 1;
      x := x + 1
  }
  x: = x - 1;
  return x;
}
```

2.3 Блок-схема с инвариантами

Применим к программе *sqr* метод Флойда. Здесь $x \div y = x - y$ в случае, если $y \leq x$ и 0 во всех остальных случаях.

2.4 Доказательство частичной корректности

Теперь проанализируем все пары контрольных точек..

- (1) $\Rightarrow$ (2). На входе: $n \geq 0$, на выходе: $(y = x^2) \wedge (x \div 1 \leq n)$ - очевидно, что оба условия выполняются.

- (2) $\Rightarrow$ (2). Обозначим значения x и y на входе как a и b, а значения x и y на выходе как a' и b' соответственно. Тогда

  $$(b = a^2) \wedge ((a \div 1)^2 \leq n)$$

  В точке выхода (2) будет выполняться:

  1. $b \leq n$ (по условию)
  2. $a' = a + 1$
  3. $b' = b + a + a + 1 = a^2 + 2a + 1 = (a + 1)^2 = (a')^2$

  следовательно, $b' = (a')^2$ и $(a' \div 1)^2 = a^2 = b \leq n$ - инвариант в точке выхода выполняется.

- (2) $\Rightarrow$ (3). Обозначим значения x и y на входе как a и b, а значения x и y на выходе как a' и b' соответственно. Тогда

  $$(b = a^2) \wedge ((a \div 1)^2 \leq n)$$

  В точке выхода (3) будет выполняться:

  1. $a' = a - 1 \Rightarrow a = a' + 1$
  2. $a^2 = b > n$, т.е. $(a' + 1)^2 > n$ (по условию)
  3. $(a')^2 = (a - 1)^2 = (a \div 1)^2 \leq n$

  Следовательно, $(a')^2 \leq n$ и $(a' + 1)^2 > n$ - инвариант в точке выхода выполняется.

3 Формальная верификация

3.1 Формальная верификация частичной корректности

Использование неформальной верификации - трудоемкий и не очень надежный процесс, потому что имеет место человеческий фактор. Можно

ли как-либо автоматизировать этот процесс и использовать компьютеры для поиска доказательства корректности? Ответ "да и такой способ называется **формальной верификацией** программ.

Аксиоматическая семантика

Дан язык программирования l, его **аксиоматическая семантика** - это множество аксиом и правил вывода, действующих на формулы и тройки Хоара и адекватных семантике l.

Аксиомы и правила для основных операторов

- $\{(\phi)^x_e\}x := e\{\phi\}$ - аксиома присваивания
- $\frac{\{\phi\}\sigma\{\chi\} \quad \{\chi\}\tau\{\psi\}}{\{\phi\}\sigma;\tau\{\psi\}}$ - правило композиции
- $\frac{\{\phi\wedge\chi\}\sigma\{\psi\} \quad \{\phi\wedge\neg\chi\}\tau\{\psi\}}{\{\phi\}if\ \chi\ then\ \sigma\ else\ \tau\{\psi\}}$ - условное правило
- $\frac{\{\phi\wedge\chi\}\sigma\{\phi\}}{\{\phi\}while\ \chi\ do\ \sigma\ \{\phi\wedge\neg\chi\}}$ - итеративное правило, здесь ϕ - **инвариант** цикла.

Правило следствия

Существует специальное правило, операндами которого могут быть не только тройки Хоара, но и обычные формулы:

$$\frac{\phi \to \phi' \quad \{\phi'\}\sigma\{\psi'\} \quad \psi' \to \psi}{\{\phi\}\sigma\{\psi\}}$$

Это правило помогает извлекать формальные математические вопросы, не зависящие от каких-либо объектов языка программирования, из программы и спецификации. Этот процесс известен как **генерация условий корректности**.

3.2 Пример формальной верификации

Рассмотрим простую программу *swap*, меняющую местами значения двух целочисленных переменных x и y, используя только арифметические операции:

```
swap {
 x := x + y;
 y := x - y;
 x := x - y;
}
```

Тогда спецификация будет выглядеть следующим образом:

$$\{(x = a) \wedge (y = b)\}swap\{(x = b) \wedge (y = a)\}$$

Используя аксиоматическую семантику, мы можем построить дерево вывода для этой тройки Хоара:

$$(x = a \wedge y = b) \rightarrow ((x + y) - ((x + y) - y) = b \wedge (x + y) - y = a)$$

Таким образом, вопрос о корректности *swap* сводится к вопросу о тождественной истинности некоторой формулы логики первого порядка.