1 Inverse binary relation, connection with the inverse mapping

Определение

Пусть $r \subseteq A \times B$ - бинарное отношение. Тогда **обратное отношение** к r - это отношение $r^{-1} \subseteq B \times A$, определённое как:

$$r^{-1} = \{(b, a) | (a, b) \in r\}$$

Замечание

В отличие от обратных отображений, обратные отношения всегда существуют для любого бинарного отношения.

Предложение

Если $f : A \to B$ - отображение и существует обратное к f отображение g, то $g = f^{-1}$

Доказательство

По предложению о единственности обратного отображения, достаточно проверить, что f^{-1} - обратное отображение, т.е. что $f \circ f^{-1} = id_A$ и $f^{-1} \circ f = id_B$. Проверим первое утверждение. Пусть $a \in A$, $b = f(a)$, т.е. $(a, b) \in f$. Тогда по определению, $(b, a) \in f^{-1}$, следовательно, $(a, a) \in f^{-1} \circ f$. Это означает, что $id_A \subseteq f^{-1} \circ f$. С другой стороны, если $(a_1, a_2) \in f^{-1} \circ f$, то существует такой $b \in B$, что $(a_1, b) \in f$ и $(b, a_2) \in f^{-1}$, т.е. $(a_2, b) \in f$. Так как f инъективно, $a_1 = a_2$, поэтому $f^{-1} \circ f \subseteq id_A$. Следовательно, равенство $f \circ f^{-1} = id_A$ доказано. Второе равенство доказывается аналогично.

2 Term rewriting in λ-calculus: call-by-value and call-by-name strategies

Две основные стратегии редукции:

- **вызов по значению**: в любом терме вида $((\lambda x.t)s)$ сначала s сводится к s', и только после этого к нему применяется β-редукция и результат сводится к $t[x = s']$.

  Пример: $(\lambda pq.pqp)(\lambda ab.a)(\lambda ab.b)$

  1. α эквивалентная формула: $(\lambda pq.pqp)(\lambda ab.a)(\lambda cd.d)$.
  2. Редукция: используя редекс $(\lambda pq.pqp)(\lambda ab.a)$: $\lambda q(\lambda ab.a)q(\lambda ab.a)(\lambda cd.d)$.
  3. Подстановка $\theta_1 = [q = (\lambda cd.d)]$: $(\lambda ab.a)(\lambda cd.d)(\lambda ab.a)$.
  4. Подстановка $\theta_2 = [a = (\lambda cd.d)]$: $(\lambda b.(\lambda cd.d))(\lambda ab.a)$.
  5. Подстановка $\theta_3 = [b = (\lambda ab.b)]$: $(\lambda cd.d)(\lambda ab.a)$.

- **вызов по имени**: к любому терму вида $((\lambda x.t)s)$ сначала применяется β-редукция, а затем результат сводится к $t[x = s]$.

  Пример: $(\lambda pq.pqp)(\lambda ab.a)(\lambda ab.b)$

  1. α эквивалентная формула: $(\lambda pq.pqp)(\lambda ab.a)(\lambda cd.d)$.
  2. Подстановка $\theta_1 = [c = (\lambda ab.a)]$: $(\lambda pq.pqp)(\lambda d.d)$.
  3. Подстановка $\theta_2 = [d = (\lambda pq.pqp)]$: $(\lambda pq.pqp)$.

3 Conditions of program correctness: partial and total. Floyd method

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.

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**.

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:

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

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:

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

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.

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;
}
```

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.

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.