МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«МИРЭА — Российский технологический университет»
РТУ МИРЭА

Институт искусственного интеллекта (ИИИ)
Кафедра промышленной информатики (ПИ)

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7

по дисциплине
«Методы верификации и валидации характеристик программного обеспечения»

Выполнил студент группы ИКМО-05-23 Миронов Д.С.

Принял Петренко А. А.

Москва 2024

12.11.2024

Дедуктивная верификация последовательных программ

На основе изучения материала лекций по дисциплине «Методы верификации и валидации характеристик программного обеспечения» требуется выполнить следующее.

1. Докажите следующее условие корректности, вычислив слабейшее предусловие:

```
{true}
y := 0;
z := a;
y := y + x;
z := z * x;
y := y * z;
z := z / a;
z := z * b;
z := z + c;
y := y + z
{y = a•x2 + b•x + c}
```

2. Докажите следующее условие корректности:

```
{a > 0 & b > 0 & a ≠ b}
x := a;
y := b;
if x > y then
 x := x – y
else
 y := y – x
end
{a > 0 & b > 0 & НОД(x, y) = НОД(a, b)}
```

3. Предложите инварианты циклов для программ P,Q и S, реализующих алгоритм Евклида (см. предыдущее практическое занятие).
4. Докажите частичную корректность реализаций алгоритма Евклида (программы P,Q и S из предыдущего практического занятия).
5. Напишите пред- и постусловие для программ сортировки числовых массивов. Докажите частичную корректность программы, реализующей «метод пузырька».
6. Выполните верификацию следующей программы для заданных пред- и постусловий:

```
{a≥0}
x := a;
n := 1;
y := 0;
while x≠0 do
 y := y + n;
 n := n + 2;
 x := x - 1
end
```

{y = a2}

7. Выполните верификацию следующей программы для заданных пред- и постусловий:

```
{true}
x := a;
n := 0;
while x ≠ 0 do
 x := x & (x - 1);
 n := n + 1
end
{n = count(a)}
```

Здесь & – операция побитового И, а count(x) — функция, возвращающая число единиц в двоичном представлении числа x. Определите формально предметную область (для определенности можете считать, что переменные принимают целочисленные значения из отрезка [0,232-1]).

8. Выполните верификацию следующей программы для заданных пред- и постусловий:

```
{a > 1}
i := a - 1;
x := 1;
while i > 0 do
 if a % i = 0 then
 k := 0;
 j := i - 1;
 while j > 1 do
 if i % j = 0 then
 k := k + 1
 end;
 j := j - 1
 end;
 if k = 0 then
 x := i;
 i := 1
 end
 end;

i := i - 1
end
{x = maxPrimeFactor(a)}
```

Здесь maxPrimeFactor(a) – максимальный простой делитель целого числа a.

1. Вычисление слабейшего предусловия (WP)

Задано условие:

```
{true}
y := 0;
z := a;
y := y + x;
z := z * x;
y := y * z;
z := z / a;
z := z * b;
z := z + c;
y := y + z
{y = a * x^2 + b * x + c}
```

Для доказательства частичной корректности программы необходимо последовательно вычислить слабейшие предусловия (WP), начиная с конца и двигаясь к началу программы.

Шаги вычисления WP:

Постусловие: y=a·x^2+b·x+c.

Шаг 1: Инструкция `y := y + z`

Преобразуем постусловие y=a·x^2+b·x+c, заменяя y на y+z:

(y+z)=a·x^2+b·x+c

Получаем слабейшее предусловие для инструкции `y := y + z`:

y+z=a·x^2+b·x+c

Шаг 2: Инструкция `z := z + c`

Преобразуем предыдущее выражение y+z=a·x^2+b·x+c, заменяя z на z+c:

y+(z+c)=a·x^2+b·x+c

Это упрощается до:

y+z=a·x^2+b·x

Слабейшее предусловие для инструкции `z := z + c`:

y+z=a·x^2+b·x

Шаг 3: Инструкция `z := z * b`

Преобразуем выражение y+z=a·x2+b·x, заменяя z на z·b:

y+(z·b)=a·x^2+b·x

Слабейшее предусловие для инструкции `z := z * b`:

$y+z \cdot b=a \cdot x^2+b \cdot x$

Шаг 4: Инструкция `z := z / a`

Преобразуем выражение $y+z \cdot b=a \cdot x^2+b \cdot x$, заменяя z на z/a:

$y+((z/a) \cdot b)=a \cdot x^2+b \cdot x$

Упрощаем до:

$y+(z \cdot b)/a=a \cdot x^2+b \cdot x$

Слабейшее предусловие для инструкции `z := z / a`:

$y+(z \cdot b)/a=a \cdot x^2+b \cdot x$

Шаг 5: Инструкция `y := y * z`

Преобразуем предыдущее выражение, заменяя y на $y \cdot z$:

$(y \cdot z)+(z \cdot b)/a=a \cdot x^2+b \cdot x$

Шаг 6: Инструкция `z := z * x`

Преобразуем предыдущее выражение, заменяя z на $z \cdot x$:

$(y \cdot z \cdot x)+((z \cdot x) \cdot b)/a=a \cdot x^2+b \cdot x$

Шаг 7: Инструкция `y := y + x`

Преобразуем предыдущее выражение, заменяя y на y+x:

$(y+x) \cdot z \cdot x+((z \cdot x) \cdot b)/a=a \cdot x^2+b \cdot x$

Шаг 8: Инструкция `z := a`

Подставим z=a:

$(y+x) \cdot a \cdot x+((a \cdot x) \cdot b)/a=a \cdot x^2+b \cdot x$

Упростим выражение:

$(y+x) \cdot a \cdot x+b \cdot x=a \cdot x2+b \cdot x$

Следовательно, y=0, так как $0+a \cdot x^2+b \cdot x=a \cdot x^2+b \cdot x$.

Начальное слабейшее предусловие: true.

2. Доказательство корректности для программы:

Код:

```
{a > 0 & b > 0 & a ≠ b}
x := a;
y := b;
if x > y then
  x := x - y
else
  y := y - x
end
{a > 0 & b > 0 & gcd(x, y) = gcd(a, b)}
```

Для доказательства корректности:

1. **Предусловие**: $a>0 \land b>0 \land a \neq b$.
2. **Постусловие**: $a>0 \land b>0 \land gcd(x,y)=gcd(a,b)$.
3. Корректность этой программы можно доказать, применяя эквивалентности алгоритма Евклида для вычисления НОД.

3. Инварианты циклов для программ P, Q и S

Для всех программ можно использовать один и тот же инвариант:

$gcd(x,y) = gcd(a,b)$

4. Доказательство частичной корректности реализаций алгоритма Евклида (программы P, Q и S)

1. Для программы P: доказываем, что на каждой итерации gcd(x,y) остается неизменным, пока $x \neq y$.
2. Для программы Q: доказываем эквивалентность НОД при замене переменных с учетом остатка от деления.
3. Для программы S: проверяем, что при каждой итерации НОД остается неизменным.

5. Пред- и постусловие для программы сортировки числовых массивов

Для алгоритма сортировки методом пузырька:

- **Предусловие**: array — массив чисел длиной n.
- **Постусловие**: Массив отсортирован, то есть $\forall i<j$, $array[i] \leq array[j]$.

6. Верификация программы для пред- и постусловий: $y = a^2$

Код:

```
{a ≥ 0}
x := a;
n := 1;
y := 0;
while x ≠ 0 do
  y := y + n;
  n := n + 2;
  x := x - 1
end
```

```
{y = a^2}
```

Инвариант цикла: y=(a−x)2y = (a - x)^2y=(a−x)2.

7. Верификация программы для пред- и постусловий: n=count(a)

Код:

```
{true}
x := a;
n := 0;
while x ≠ 0 do
  x := x & (x - 1);
  n := n + 1
end
{n = \text{count}(a)}
```

Инвариант цикла: nnn равно количеству единиц в битовой форме aaa, обрезанных на каждом шаге.

8. Верификация программы для нахождения максимального простого делителя

Код:

```
{a > 1}
i := a - 1;
x := 1;
while i > 0 do
  if a % i = 0 then
    k := 0;
    j := i - 1;
    while j > 1 do
      if i % j = 0 then
        k := k + 1
      end;
      j := j - 1
    end;
    if k = 0 then
      x := i;
      i := 1
    end
  end;
  i := i - 1
end
{x = \text{maxPrimeFactor}(a)}
```

Предусловие: a>1, постусловие: x=maxPrimeFactor(a)

Здесь инвариант цикла должен поддерживать условие, что x — наибольший простой делитель числа aaa на каждом шаге выполнения программы.