Дискретная математика

Формальная верификация

Константин Чухарев

Осень 2026

Формальная верификация

> *Тестирование программы может убедительно продемонстрировать наличие ошибок, но безнадёжно несостоятельно для демонстрации их отсутствия.*
>
> ***— Эдсгер Дейкстра***

Что значит «программа корректна»

Программа прошла миллион тестов — и всё равно может упасть на миллион первом. Миллион входов — конечное подмножество бесконечного множества возможных запусков. Свойство вида «программа не упадёт ни на одном входе» — утверждение обо всех элементах бесконечного множества. Один контрпример его опровергает, но ни одно конечное число примеров не подтверждает.

Корректность — отношение между текстом программы и требованием к нему. Прежде чем доказывать, требование нужно записать. Невнятное «работает правильно» превращается в точное утверждение.

> Утверждение о программе — это формула, а проверка программы — доказательство этой формулы.

Тройки Хоара

Определение 1 (Тройка Хоара)

Тройка Хоара

$$\{P\}\ S\ \{Q\}$$

означает: если предусловие P выполнено перед оператором S, то постусловие Q выполнено после его завершения — при условии, что S вообще завершается.

Оговорка о завершении фиксирует *частичную* корректность: тройка не требует, чтобы оператор остановился. Тотальную корректность — завершение плюс постусловие — рассмотрим дальше.

Тройка как контракт

Пример (Корректная тройка)

$$\{x = 5\}\ x := x + 1\ \{x = 6\}$$

До присваивания $x = 5$. После присваивания $x = 6$.

Предусловие кодирует предположения, постусловие — гарантии. Вместе они образуют контракт: если вызывающая сторона удовлетворяет P, процедура обеспечивает Q.

Пример: модуль числа

Пример (Модуль числа)

$$\{x \geq 0\} \quad \text{if } x > 0 \text{ then } y := x \text{ else } y := -x \ \{y = |x|\}$$

При $x > 0$ записывается x. При $x = 0$ записывается 0. Тогда $y = -x = 0 = |x|$. Случай $x < 0$ исключён предусловием. К этой тройке вернёмся на слайде про условный оператор.

Аксиома присваивания

Определение 2 (Аксиома присваивания)

Тройка

$$\{P[x := E]\}\ x := E\ \{P\}$$

верна для любого постусловия P. Здесь $P[x := E]$ — формула P, в которой каждое свободное вхождение x заменено выражением E.

Тонкость подстановки

Прямой и обратный проход

Прямой проход — рассуждение от предусловия к постусловию. После присваивания $x := E$ новое значение x равно значению E до присваивания. Подставляя это значение в известные утверждения, продвигаем их вперёд по программе. Обратный инструмент — слабейшее предусловие: оно восстанавливает предусловие из постусловия.

Пример (Удвоение числа)

Программа `y := 2 * x` удваивает число из x. Тройка Хоара:

$$\{x = a\}\ y := 2 \cdot x\ \{y = 2a\}$$

. Прямой проход: предусловие даёт $x = a$. После присваивания $y = 2 \cdot x$. Из $x = a$ получаем $y = 2 \cdot a = 2a$. Обратный проход: аксиома присваивания из постусловия даёт $(y = 2a)[y := 2 \cdot x]$. Это $2 \cdot x = 2a$. Из предусловия $x = a$ оно следует сразу.

Правило композиции

Определение 3 (Правило композиции)

Из тройки

$$\{P\}\ S_1\ \{R\}$$

и тройки

$$\{R\}\ S_2\ \{Q\}$$

следует тройка

$$\{P\}\ S_1 ; S_2\ \{Q\}$$

. Промежуточное условие R склеивает два оператора в последовательность.

Пример: композиция в действии

Пример (Композиция в действии)

Докажем тройку

$$\{x = 3\}\ x := x + 1 ; x := x \cdot 2\ \{x = 8\}$$

. Идём от постусловия назад. Для второго присваивания промежуточное условие: $R = (x = 8)[x := x \cdot 2]$. То есть $x \cdot 2 = 8$. Значит, $x = 4$. Для первого присваивания аксиома даёт $(x = 4)[x := x + 1]$. То есть $x + 1 = 4$. Значит, $x = 3$. Промежуточное условие $R \equiv x = 4$ склеивает операторы, и тройка доказана.

Правило следствия

Определение 4 (Правило следствия)

Из тройки $\{P'\}\ S\ \{Q'\}$, из посылки $P \to P'$ и посылки $Q' \to Q$ следует тройка $\{P\}\ S\ \{Q\}$. Предусловие можно усилить, постусловие — ослабить.

Слабейшее предусловие

Определение 5 (Слабейшее предусловие)

Слабейшее предусловие $\mathrm{wp}(S, Q)$ — самая широкая формула P. Для неё верна тройка

$$\{P\}\ S\ \{Q\}$$

. Для присваивания: $\mathrm{wp}(x := E, Q) = Q[x := E]$. Для последовательности: $\mathrm{wp}(S_1 ; S_2, Q) = \mathrm{wp}(S_1, \mathrm{wp}(S_2, Q))$. Для условного оператора: $\mathrm{wp}(\text{if } B \text{ then } S_1 \text{ else } S_2, Q) = (B \wedge \mathrm{wp}(S_1, Q)) \vee (\neg B \wedge \mathrm{wp}(S_2, Q))$.

Обратный проход, вычисляющий wp, — стандартный способ генерации условий верификации.

Правило условного оператора

Определение 6 (Правило условного оператора)

Из тройки

$$\{P \wedge B\}\ S_1\ \{Q\}$$

и тройки

$$\{P \wedge \neg B\}\ S_2\ \{Q\}$$

следует тройка

$$\{P\}\ \ \text{if } B \text{ then } S_1 \text{ else } S_2\ \{Q\}$$

.

Пример: возвращаемся к модулю

Пример (Возвращаемся к модулю)

Ветка «если»: $x \geq 0 \wedge x > 0 \Rightarrow x = |x|$. После присваивания $y := x$. Тогда $y = |x|$. Ветка «иначе»: $x \geq 0 \wedge \neg(x > 0) \Rightarrow x = 0$. После присваивания $y := -x$. Тогда $y = -0 = 0 = |x|$. По правилу условного оператора тройка с модулем доказана.

Правило цикла

Определение 7 (Правило цикла)

Из тройки

$$\{I \wedge B\}\ S\ \{I\}$$

следует тройка

$$\{I\}\ \ \text{while } B \text{ do } S\ \{I \wedge \neg B\}$$

. Это частичная корректность: завершение цикла не гарантировано.

Цикл — единственный оператор, где доказательство идёт по индукции. Инвариант I — то, что сохраняется каждой итерацией.

Инвариант цикла

Определение 8 (Инвариант цикла)

Формула I — инвариант цикла while B do S, если:

1. Тело цикла сохраняет I: верна тройка

$$\{I \wedge B\}\ S\ \{I\}$$

.

2. $I \wedge \neg B$ влечёт постусловие.

Инвариант должен выполняться перед первой итерацией. Это база индукции.

Как угадать инвариант

Линейный поиск

Пример (Инвариант цикла: линейный поиск)

Линейный поиск ищет x в массиве $A[0..n-1]$. Если элемент не найден, возвращается n.

```
i := 0
while i < n and A[i] != x do
  i := i + 1
```

Инвариант: $I \equiv 0 \leq i \leq n \wedge \forall j \in \{0, ..., i-1\}\ A[j] \neq x$. Все позиции до i проверены и не содержат x.

База: перед циклом $i = 0$. Диапазон $\{0, ..., -1\}$ пуст, и $\forall j \in \emptyset$ истинно вакуумно.

Линейный поиск: шаг и выход

Пример (Шаг индукции и выход)

Шаг: пусть инвариант и условие цикла истинны. Условие цикла: $i < n \land A[i] \neq x$. Тело увеличивает счётчик на единицу. Условие цикла даёт $i + 1 \leq n$. По инварианту все позиции до i не содержат x. Условие цикла даёт $A[i] \neq x$. Значит, все позиции до $i + 1$ не содержат x, и инвариант сохраняется.

Выход: отрицание условия цикла: $\neg(i < n \land A[i] \neq x) \equiv i \geq n \lor A[i] = x$. С учётом инварианта $i \leq n$. Из $i \geq n$ получаем $i = n$: элемент не найден. Случай $A[i] = x$: элемент найден.

Верификация через индукцию

Доказательство сохранения инварианта телом цикла — шаг индукции. База индукции — инвариант, установленный перед первой итерацией.

Инструменты верификации работают так:

1. Программист записывает предусловия, постусловия и инварианты циклов.
2. Инструмент генерирует *условия верификации* — логические формулы, истинность которых гарантирует корректность программы.
3. SMT-солверы проверяют эти формулы, решая, выполнимо ли отрицание каждой из них.
4. Ответ `unsat` означает: отрицание невыполнимо, условие верификации истинно.

> Человек придумывает инварианты, машина проверяет следствия.

Условия верификации

Пример (От программы к условию верификации)

Программа обменивает значения двух переменных:

```
t := x
x := y
y := t
```

Докажем тройку

$$\{x = a \wedge y = b\}\ t := x \,;\, x := y \,;\, y := t\ \{x = b \wedge y = a\}$$

. Вычисляем слабейшее предусловие обратным проходом. $\mathrm{wp}(y := t, x = b \wedge y = a) = x = b \wedge t = a$. $\mathrm{wp}(x := y, x = b \wedge t = a) = y = b \wedge t = a$. $\mathrm{wp}(t := x, y = b \wedge t = a) = y = b \wedge x = a$. Слабейшее предусловие — в точности предусловие тройки, поэтому программа корректна.

От программы к SMT-запросу

Каждое присваивание — равенство между свежей константой и старым значением. Условие верификации говорит: если выполнены предусловие и все равенства переходов, выполнено и постусловие. Отрицание условия передаём солверу на языке SMT-LIB:

```
(assert (not (=> (and (= x0 a) (= y0 b)
                      (= t0 x0) (= x1 y0) (= y1 t0))
                 (and (= x1 b) (= y1 a)))))
(check-sat)
```

Солвер отвечает `unsat`: отрицание невыполнимо, тройка верна. Если бы солвер ответил `sat`, он предъявил бы модель — контрпример, на котором тройка нарушается.

Контрпримеры и тестирование

Пример (Неверная тройка)

Тройка

$$\{x = 5\}\ x := x + 1\ \{x = 7\}$$

ложна. Из предусловия $x = 5$. После присваивания получаем $x = 6$. Отрицание её условия верификации выполнимо: солвер находит модель, на которой постусловие не выполнено.

Падающий тест — контрпример к утверждению «программа корректна на этом входе». Тестирование может продемонстрировать наличие ошибок, но никогда их отсутствие. Один контрпример опровергает $\forall$, но никакое количество примеров его не доказывает.

Тестирование против доказательства

- Тестирование — поиск контрпримера.
- Доказательство — демонстрация того, что контрпримеров не существует.

И то, и другое необходимо: тестирование находит лёгкие ошибки, доказательство — глубокие.

Терминируемость и тотальная корректность

Частичная корректность не требует завершения. Тройка

$$\{P\}\ S\ \{Q\}$$

утверждает: если S завершается, то Q верно. Тотальная корректность требует и завершения, и постусловия.

Определение 9 (Тотальная тройка)

Тотальная тройка $[P]\ S\ [Q]$ означает: если P верно перед выполнением, то S завершается и после него верно Q.

Вариант цикла

Определение 10 (Вариант цикла)

Функция f со значениями во вполне упорядоченном множестве (обычно в натуральных числах) — *вариант цикла*, если она строго убывает на каждой итерации. Вполне упорядоченное множество не содержит бесконечно убывающих цепочек. Рано или поздно вариант достигает минимума, и условие выхода срабатывает.

Правило цикла с вариантом

Определение 11 (Правило цикла с вариантом)

Из тройки

$$\{I \wedge B \wedge f = z\}\ S\ \{I \wedge f < z\}$$

следует $[I]$ while B do S $[I \wedge \neg B]$. Здесь f — вариант цикла. А z — свежая переменная, фиксирующая значение f до итерации.

Пример (Вариант цикла)

Программа `while x > 0 do x := x - 1` завершается. Вариант $f = x$. На каждой итерации x уменьшается на 1. Бесконечная убывающая цепочка $x > x - 1 > x - 2 > \ldots$ невозможна.

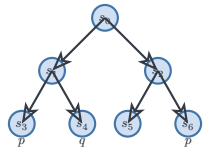
Model checking против дедуктивной верификации

Тройки Хоара — дедуктивная верификация: доказательство по правилам вывода. Model checking — другой ответ: автоматический перебор состояний конечной модели.

Определение 12 (Model checking)

Строится конечная модель системы — граф состояний и переходов. Свойство записывается в темпоральной логике, например в CTL. AG p — «во всех состояниях любого пути выполнено p». EF p — «существует путь, где когда-нибудь выполнено p». Алгоритм обходит состояния модели и проверяет свойство точно. При нарушении выдаётся контрпример — путь, на котором свойство не выполнено.

Сравнение подходов



Сравнение подходов:

- Model checking точен на конечных моделях и полностью автоматичен.
- Дедуктивная верификация работает с бесконечными состояниями, но требует инвариантов от человека.
- Model checking отвечает «да» или «нет» с контрпримером, дедуктивная — доказательством.

Инструменты верификации

Труд верификации делится между человеком и машиной. Инварианты придумывает человек, условия верификации проверяет SMT-солвер.

- Dafny — верификатор аннотированных программ на собственном языке.
- Why3 — портал, передающий условия верификации разным солверам.
- Frama-C — анализ кода на языке C.
- Z3 — SMT-солвер, проверяющий условия верификации на языке SMT-LIB.

Разметка программы утверждениями и проверка следствий машиной — рабочий процесс современной верификации. Падающий тест — контрпример к утверждению о корректности. Тестирование и доказательство работают вместе: тесты находят лёгкие ошибки, доказательство закрывает глубокие.

Что мы поняли?

- Тройка Хоара — контракт: предусловие, оператор, постусловие.
- Аксиома присваивания читается «назад»: из постусловия восстанавливает предусловие.
- Слабейшее предусловие — точный обратный инструмент для генерации условий верификации.
- Инвариант превращает шаг цикла в шаг индукции: база, шаг, выход.
- Условия верификации проверяют SMT-солверы: отрицание условия должно быть невыполнимо.
- Частичная корректность — постусловие после завершения, тотальная — ещё и завершение.
- Вариант цикла доказывает завершение: функция, убывающая на каждой итерации.
- Model checking перебирает состояния конечной модели, дедуктивная верификация доказывает по правилам.

Вопросы для самопроверки

- Что такое тройка Хоара, и почему она задаёт частичную корректность?
- Как аксиома присваивания читается «назад»?
- Что такое слабейшее предусловие, и как оно вычисляется для присваивания и последовательности?
- Какие условия проверяются у инварианта цикла?
- Почему инвариант линейного поиска содержит границу $0 \le i \le n$?
- Что отвечает SMT-солвер корректной программе, и почему?
- Чем тотальная корректность отличается от частичной?
- Чем model checking отличается от дедуктивной верификации?