

# Математическая логика и логическое программирование

mk.cs.msu.ru → Лекционные курсы  
→ Математическая логика и логическое программирование (3-й поток)

## Блок 52

Формальная верификация

Лектор:

**Подымов Владислав Васильевич**

E-mail:

**valdus@yandex.ru**

ВМК МГУ, 2023/2024, осенний семестр

# Вступление

В математической логике есть ещё много такого, что было бы полезно и интересно обсудить, но не получится, так как времени до конца осталось не так уж и много

Поэтому остановимся подробно только на одном разделе: как можно при помощи логических методов решить какую-нибудь очень нетривиальную и очень полезную прикладную задачу, *казалось бы* не связанную с логикой

Обсудим в деталях одну такую задачу из области программирования

# Ошибки в программах

«Любая нетривиальная программа  
содержит хотя бы одну ошибку»

*(автор неизвестен)*

**Правильная** программа<sup>1</sup> не содержит ошибок и решает задачу, которую требовалось решить при помощи этой программы

Эти утверждения выглядят разумно, но порождают ряд нетривиальных вопросов, например:

- ▶ Что такое «ошибка», насколько плохо её наличие, и нужно ли её вообще искать?
- ▶ Можно ли как-нибудь убедиться, что ошибок в программе нет и она действительно решает поставленную задачу?

---

<sup>1</sup> Из утверждения выше следует, что такой программы не существует среди нетривиальных — но можно и пофантазировать

# Ошибки в программах

Подход к проверке правильности работы программы, про который знает каждый программист — это **тестирование**:

- ▶ придумывается показательный набор входных данных программы (тестовое покрытие)
- ▶ программа выполняется на тестовом покрытии
- ▶ результаты выполнения программы сравниваются с ответами к задаче, которые считаются правильными

Основной недостаток такого подхода:

**ошибки всё равно остаются** в программе, хотя их и становится меньше

Кроме того, в современном мире широко применяются вычислительные системы, для которых даже после «качественного» тестирования

**не гарантируется отсутствие критичных ошибок**:

аппаратные и программно-аппаратные, интерактивные и распределённые системы, сложные программные комплексы, ...

# Ошибки в программах

В некоторых случаях протестировать код — это приемлемо («более-менее работает, а дальше пусть пользователь разбирается»), но далеко не всегда: даже от небольших ошибок серьёзно зависит **прибыль компаний**<sup>1</sup>, **успешность проектов мирового масштаба**<sup>2</sup>, **быт людей**<sup>3</sup> и даже **их жизни**<sup>4</sup>

---

1 1994. Процессор Intel, ошибка в реализации деления чисел с плавающей точкой ⇒ замена дефектных процессоров, сотни миллионов \$ убытка

2 1962–сейчас. Ракеты и спутники, взорвавшиеся и исчезнувшие из-за программных ошибок: Фобос (пропущена кавычка), Ariane 5 (ошибка в округлении чисел), Mars Global Surveyor (перепутаны английская и метрическая системы мер), Hitomi (ошибка в программе стабилизации вращения), ...

3 2003. Полное отключение электричества в нескольких областях США и Канады ⇐ ошибка в реализации взаимодействия программ оповещения об электрической нагрузке (race condition)

4 1980-е гг. Пять смертей при лечении рака аппаратом Therac-25 ⇐ ошибочное увеличение мощности радиационного облучения при очень редком сочетании времён выполнения параллельных подпрограмм (race condition)

# Формальная верификация

Есть и другой подход к проверке правильности вычислительных систем:

- ▶ формулируется набор требований к выполнению системы, означающих «система функционирует правильно»
- ▶ **строго** доказывается или опровергается утверждение о том, что математическая модель, достаточно точно описывающая систему, удовлетворяют этим требованиям

Требования такого вида, записанные на математическом языке, называются **формальной спецификацией** системы, и соответствующий язык — **языком спецификаций**

Проверка соблюдения требований с помощью математических методов называется **формальной верификацией** системы

Если в качестве языка спецификаций выбрать какой-либо **логический язык**, то для проверки правильности программы можно будет использовать **логические методы**