

Математическая логика

mk.cs.msu.ru → Лекционные курсы → Математическая логика (318, 319/2, 241, 242)

Блок 47

Формальная верификация

Лектор:

Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

ВМК МГУ, 2025, февраль–май

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 детально одну такую задачу из области программирования

Ошибки в программах

«Любая нетривиальная программа
содержит хотя бы одну ошибку»

(автор неизвестен)

Правильная программа¹ 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?

¹ 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 ошибок¹ серьёзно зависит прибыль компаний², успешность проектов мирового масштаба³, быт людей⁴ и даже их жизни⁵

1 См., например, https://en.wikipedia.org/wiki/List_of_software_bugs

2 1994. Процессор Intel, ошибка в реализации деления чисел с плавающей точкой
⇒ замена дефектных процессоров, сотни миллионов \$ убытка

3 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), ...

4 2003. Полное отключение электричества в нескольких областях США и Канады
⇐ ошибка в реализации взаимодействия программ оповещения об электрической нагрузке (race condition)

5 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:

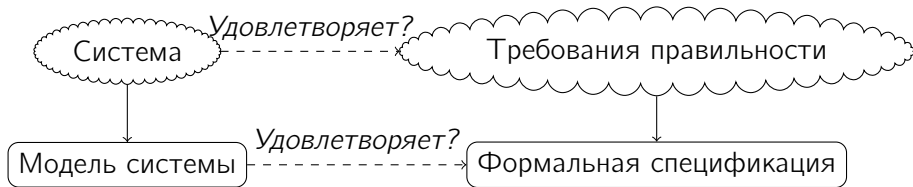
- ▶ формулируется набор требований к выполнению системы, означающих «система функционирует правильно»
- ▶ **строго** доказывается или опровергается утверждение о том, что математическая модель, достаточно точно описывающая систему, удовлетворяют этим требованиям

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**

Формальная верификация



Обсудим то, как можно использовать **логику предикатов** для формальной верификации **императивных программ**

- ▶ Как может быть устроена математическая модель программы?
- ▶ Как можно записывать требования к программе с использованием логики предикатов?
- ▶ Как проверить, удовлетворяет ли модель программы записанным требованиям?