Математические методы верификации схем и программ

mk.cs.msu.ru → Лекционные курсы
→ Математические методы верификации схем и программ

Блок 1

Что такое и зачем нужна формальная верификация

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2022/2023, осенний семестр

Что такое формальная верификация

Популярный подход к устранению ошибок в вычислительной системе (программе, микросхеме, распределённой системе, ...), известный каждому разработчику, — это тестирование: система выполняется в разных обстоятельствах (*на тестовых сценариях*), результаты выполнения сравниваются с ожидаемыми, и при расхождении ищутся и устраняются ошибки

В ряде случаев (например, при разработке схем, распределённых систем и больших программных и программно-аппаратных комплексов) получить версию разрабатываемой системы для устранения в ней ошибок — это трудоёмко, дорого или просто невозможно, и тогда применяется имитационное моделирование: составление и выполнение *модели* разрабатываемой системы

В большинстве случаев нельзя исследовать систему или её модель во **всевозможных** обстоятельствах, поэтому после тестирования и моделирования **ошибки всё равно остаются**

Что такое формальная верификация

> The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness.[1]
>
> Dijkstra. The humble programmer.
> Turing award speech. 1972.

Хотя с этим утверждением можно до некоторой степени поспорить, но уже в 1970-х годах обозначилась необходимость строго обосновывать отсутствие ошибок проектирования и программирования в вычислительных системах

Формальная верификация — это подход к проверке правильности вычислительных систем, устроенный в целом так:

- Придумывается формальная спецификация: набор требований, означающих правильность функционирования системы и записанных на формальном языке
- Строго математически обосновывается (или опровергается) утверждение о том, что система удовлетворяет спецификации

1 Существенно повысить уверенность в правильности программы можно только предоставив убедительное обоснование правильности

Какие бывают формальные спецификации

Формальной спецификацией может быть, например:

- «эталонная» правильная система

Тогда формальная верификация представляет собой проверку эквивалентности заданной системы и эталонной

Пример

Пусть имеется программа π_1, заведомо правильно (*хотя, быть может, и неэффективно*) сортирующая массив целых чисел

Кто-то написал другую (*эффективную, но совершенно непонятную*) программу π_2 и утверждает, что она тоже сортирует массив целых чисел

Чтобы убедиться, что программа π_2 действительно работает правильно, достаточно убедиться, что она эквивалентна программе π_1

Какие бывают формальные спецификации

Формальной спецификацией может быть, например:

- «эталонная» правильная система

Другой пример

Цифровая микросхема при разработке, как правило, проходит через много уровней абстракции, например (*если чуть упростить реальность*):

Системный уровень ($\sim$ SystemC): высокоуровневое алгоритмическое описание

$\downarrow$

Поведенческий уровень ($\sim$ Verilog):
способ пересылки логических значений смежду схемными регистрами

$\downarrow$

Логический уровень ($\sim$ AIG): соединение триггеров и логических вентилей

$\downarrow$

Транзисторный уровень: принципиальная схема соединения МОП-транзисторов

$\downarrow$

Топологический уровень: размещение КМОП-транзисторов на кристалле

Схема на предыдущем уровне служит эталоном для следующего, и описание схемы на следующем уровне должно быть эквивалентно её описанию на предыдущем уровне

Какие бывают формальные спецификации

Формальной спецификацией может быть, например:

- набор логических формул, выражающих свойство правильности системы на достаточно простом языке

Для проверки правильности системы не обязательно иметь готовую эталонную систему, достаточно

- выбрать подходящий формальный язык спецификаций,
- представить фразу «система работает правильно» в виде формулы и
- применить известный метод проверки соответствия системы формуле

Спецификации в таком подходе обычно основываются на

- логике предикатов (*полагаю, что все тут знают, что это такое, но всё равно позже коротко напомню*) и
- темпоральных логиках (*позволяющих описывать взаимосвязи между событиями, происходящими в системе в разные моменты времени*)

Этому подходу и этим языкам будет посвящена подавляющая часть курса

Что такое формальная верификация

Формальная верификация естественным образом дополняет тестирование и имитационное моделирование,

- предлагая «доказательство без выполнения» вместо «выполнения без доказательства» и
- покрывая те случаи, в которых тестирование и моделирование не могут должным образом повысить уверенность в том, что система работает «достаточно правильно»

Два основных популярных подхода к формальной верификации вычислительных систем — это

- дедуктивный анализ и
- проверка моделей (проверка на модели; верификация моделей программ; model checking)

Дедуктивный анализ

Дедуктивный подход к формальной верификации обычно устроен так

Для системы задаётся формальная семантика: описание поведения в строгих математических терминах

Формулируется **теорема о корректности системы**: теорема в обычном математическом понимании, утверждающая правильность системы относительно формальной семантики и формальной спецификации

Теорема о корректности **доказывается** каким-либо общепризнанным способом, в том числе

- вручную, как обычно доказывается всё в математических статьях и книгах, или
- с использованием средств автоматизированного доказательства теорем (пруверов)

Подробный рассказ о методах формальной верификации в курсе начнётся с краткого введения в дедуктивную верификацию последовательных программ

Проверка моделей (model checking)

Общая схема метода проверки моделей:

1. Спецификация системы представляется в виде формулы φ некоторого логического языка
2. Для системы строится модель M, представляющая собой
   - описание того, какие возможности пошагового выполнения имеет система, и вместе с этим
   - интерпретацию формул выбранного языка спецификаций
3. Проверяется выполнимость формулы φ на модели M:

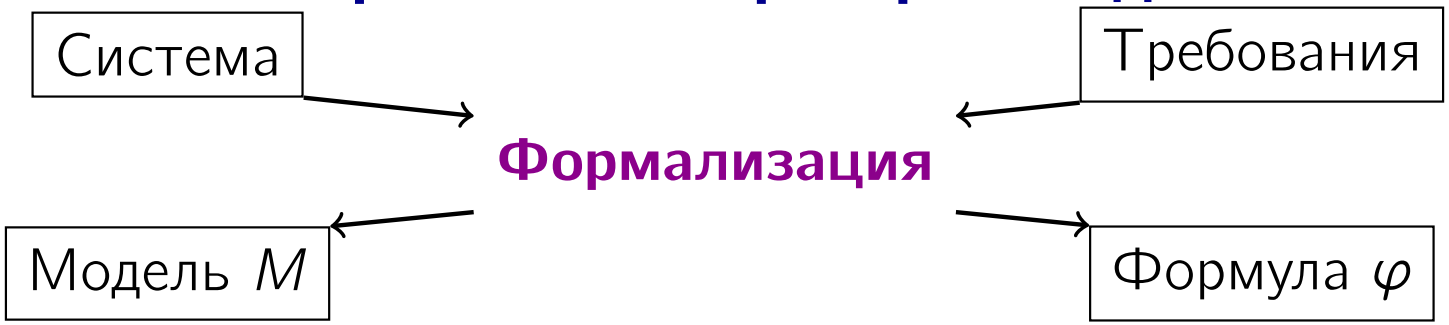
$$M \models \varphi?$$

Этому подходу посвящена бóльшая часть курса

Проверка моделей (model checking)

Проверка моделей ...	**Дедуктивный анализ ...**
обычно алгоритмически разрешима	обычно алгоритмически неразрешим
обычно применяется для моделей и спецификаций простого вида — например, формула несложного пропозиционального языка и конечный автомат	ограничивает устройство моделей и спецификаций в основном только возможностями ручного доказательства пользователем
используется в основном для случаев, когда проверить выполнимость формулы на модели можно автоматически	хотя до некоторой степени и автоматизирован, но в основном выполняется вручную и требует соответствующих навыков в области построения доказательств
ввиду автоматического решения предпочтительна там, где может быть применена	может применяться для случаев, с которыми не справляется проверка моделей

Основные этапы применения проверки моделей



Чтобы применить метод проверки моделей, для начала требуется построить

- модель по реальной исследуемой системе и
- формулу по (неформальной) спецификации этой системы

Моделирование системы иногда оказывается простым и даже автоматическим, а иногда требует большого труда и продвинутых знаний

(например, знаний о методах абстракции моделей для уменьшения их размера — это будет обсуждаться в лекциях)

Составление неформальной и формальной спецификации — тоже непростое дело: во многом ручное, и иногда непросто понять, правильна ли спецификация; если ошибочна, то верификация бессмысленна

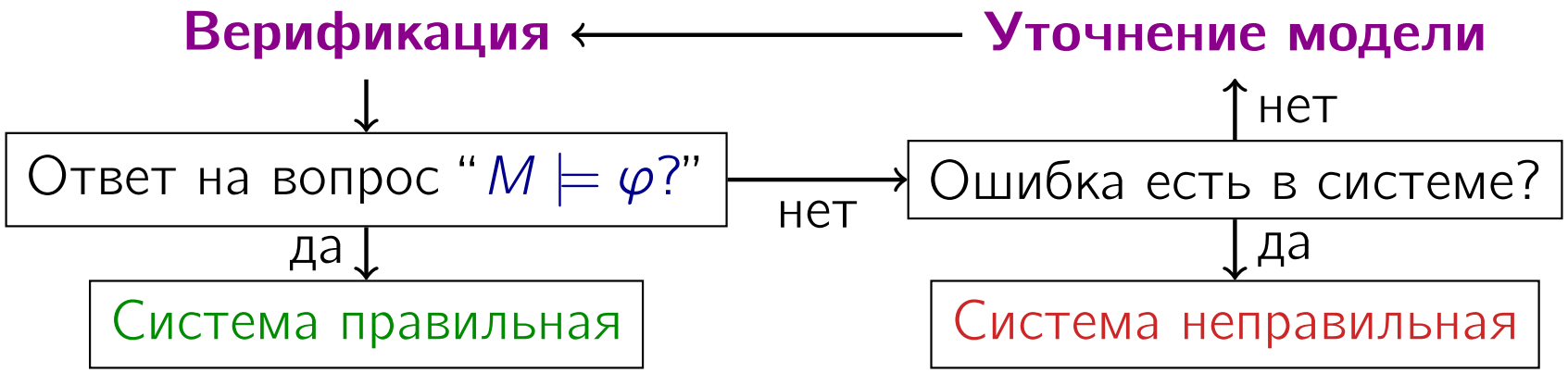
Этому всему будете учиться, выполняя обязательные домашние задания

Основные этапы применения проверки моделей

В идеальном случае верификация модели относительно формулы выполняется автоматически с использованием несложных алгоритмов алгебры и дискретной математики, таких как проверка выполнимости булевых формул, проверка достижимости вершин в графе, проверка достижимости компонент сильной связности в графе, построение схем по булевым формулам, решение систем линейных неравенств, ...

О применении таких алгоритмов в верификации будет идти речь в лекциях

Основные этапы применения проверки моделей



На практике зачастую бывает недостаточно ответить на вопрос "$M \models \varphi$?": если ответ отрицательный, то в зависимости от того, содержится ли найденная ошибка в системе, это может означать как то, что система оказалась неправильной, так и то, что модель следует уточнить

Существуют методы автоматического уточнения модели по найденной ошибке, *и быть может, поговорим и про них ближе к концу курса*

Зачем нужна формальная верификация?

Когда-то мир обходился и без формальной верификации вычислительных систем

Но так жить становится всё труднее

Программно-аппаратные системы повсеместно используются в критических областях (медицина, энергетика, транспорт, военные комплексы, телекоммуникации, банковский сектор, ...)

Если в такой системе возникнет ошибка, то это может привести к серьёзным последствиям и катастрофе, как и отключение системы во избежание ошибки

Помочь избежать катастрофы может строгое обоснование отсутствия ошибок

Зачем нужна формальная верификация?

Когда-то мир обходился и без формальной верификации вычислительных систем

Но так жить становится всё труднее

Даже вне критических областей цена ошибки может быть высока:

- вычислительные системы становятся всё сложнее, объёмнее и дороже, а значит, и вероятность ошибок и трудоёмкость их исправления
- с возрастанием сложности систем усложняются и методы их разработки, и вместе с этим разнообразие ошибок и возможность их «безболезненного» устранения без необходимости возврата к начальным этапам разработки
- в некоторых системах (таких как микросхема на кристалле) исправить ошибку «локально» в готовом изделии вовсе невозможно, требуется полный перевыпуск изделия со всеми сопутствующими затратами

Зачем нужна формальная верификация?

Когда-то мир обходился и без формальной верификации вычислительных систем

Но так жить становится всё труднее

При этом в современных вычислительных системах есть классы ошибок, которые практически невозможно обнаружить более «привычными» программисту методами, такими как тестирование

Например, это ошибки в распределённых системах, скрытые в распределённом взаимодействии компонентов

(*А в современном мире большинство систем являются распределёнными*)

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

22.07.1962

Космический аппарат Маринер-1 был уничтожен через 293 секунды после старта

Антенна аппарата потеряла связь с наводящей системой на Земле, и в результате управление взял на себя бортовой компьютер с ошибкой в программе

Из отчёта НАСА, направленного в Конгресс в 1963:

«Пропуск дефиса в редактировании данных вынудил компьютер дать серию ненужных сигналов о коррекции курса, которые сбили корабль с курса и привели к его уничтожению»

Версий о том, что именно произошло, в том числе что это за дефис, есть много, но все они одинаково обидные

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

02.09.1988

На борт советской межпланетной станции «Фобос-1» была передана неверная команда

При обработке исключительной ситуации эта команда была воспринята бортовым компьютером как отключение системы стабилизации и ориентации

В результате станция лишилась управления и затерялась в космосе

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

04.06.1996

Европейская космическая ракета «Ариан-5» взорвалась через 40 секунд после старта

Причина:

- арифметическое переполнение при конвертации 64-битного вещественного числа в 16-битное знаковое целое
- в подсистеме обработки ошибок не была учтена эта ошибка, и надёжность содержащего её модуля не была должным образом проверена
- в итоге на бортовой компьютер ракеты переданы неверные данные о высоте, что и привело к уничтожению

Эта ошибка считается одной из самых дорогостоящих компьютерных ошибок в истории: 360–500 млн. $, отложенные коммерческие запуски ряда ракет, потеря репутации космического агентства

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

23.09.1999

Спутник НАСА Mars Climate Orbiter исчез с орбиты Марса, предположительно, распавшись в атмосфере из-за слишком низкой высоты полёта

Причина: перепутаны метрическая и британская системы мер в одной из подсистем

Ущерб: 327 млн. $

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

15.01.2005

Космический зонд «Гюйгенс» отделился от автоматической межпланетной станции «Кассини», достигшей после 7-летнего полёта орбиты Титана, спутника Сатурна, и начал спуск в атмосферу спутника и передачу изображений поверхности Титана

Половина изображений была утеряна, вероятно, из-за ошибок в бортовой программе зонда

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

1985–1987

Медицинский аппарат радиационной терапии Therac-25 облучал некоторых пациентов дозами, превышающими предписанные в 100 раз

Одна из основных причин: ошибочное увеличение мощности облучения из-за состояния гонки (race condition), возникающего при очень редком сочетании времён выполнения параллельных подпрограмм

Ущерб: по крайней мере трое погибших, множество передозировок

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

2000

В Панаме в больнице Justo Arosemena несколько пациентов получили передозировку облучения при радиотерапии

Причина: ошибочное увеличение мощности из-за состояния гонки (race condition) при введении некоторых данных по параллельным каналам

Ущерб: по крайней мере пятеро погибших, множество передозировок

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

25.02.1991

Зенитный ракетный комплекс MIM-104 Patriot не смог перехватить иракскую тактическую ракету

Причина: ошибка в программном обеспечении системы управления, которая привела к рассинхронизации таймеров разных компьютеров комплекса на 0.3 сек. / 100 часов

Ущерб: погибло 28 военнослужащих армии США

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

23.03.2003

Зенитный ракетный комплекс MIM-104 Patriot неверно идентифицировал британский бомбардировщик Tornado как приближающуюся вражескую ракету

Ущерб: сбит дружественный бомбардировщик, погибли оба пилота

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

20.12.1995

В катастрофе самолёта Боинг-757, рейс Майами-Кали, врезался в гору в Колумбии

Одна из существенных составляющих катастрофы — ошибка в именовании путевых точек в программе управления полётом

Ущерб: 159 погибших, выплата около 300 млн. $ родственникам погибших

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

01.08.2012

Финансовая компания Knight Capital Group во время обновления 45 минут выполняла на своих серверах попеременно старый и новый код

Причина: один необновлённый сервер, использующий устаревшую трактовку одного бита данных

Ущерб: 440 млн. $

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

2009

Японское подразделение швейцарского банка UBS чуть не потратило 31 млрд. $ на покупку акций компании Capcom вместо запланированных 310 тыс. $

Причина: из-за программной ошибки к заказу акций добавилось пять нолей

Ущерб: чудом удалось избежать, но мог быть невероятным

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

1994

Intel выполнил массовую замену дефектных процессоров Intel Pentium за свой счёт

Причина: ошибка в реализации деления чисел с плавающей точкой, из-за которой в некоторых случаях неверно считались знаки «хвоста» мантиссы

Ущерб: сотни млн. $

Зачем нужна формальная верификация?

Насколько серьёзны бывают даже небольшие ошибки

14.08.2003

Полное отключение электричества в нескольких штатах США и и провинциях Канады

В многих областях электричество было восстановлено через несколько часов, в некоторых — через несколько дней

Одна из причин: состояние гонки (race condition), которое на продолжительное время вывело из строя комнату управления энергосетью и не позволило устранить каскадное отключение «в зародыше»

Ущерб: трудно посчитать; около сотни непосредственно погибших, пожары, нарушения в работе всех завязанных на электричество служб, ...

Зачем нужна формальная верификация?

На все эти примеры можно возразить:
«Но где гарантии, что умелое применение формальной верификации помогло бы избежать этих ошибок?»

Двойственные положительные примеры, увы, найти и заметить намного сложнее:

- Если ошибка в программе привела к серьёзным проблемам, то это видно, и об этом говорят
- Если же отсутствие ошибок из-за умелого их поиска привело к нормальной работе программы, то это незаметно, и об этом не говорят

Тем не менее известны и такие примеры

Зачем нужна формальная верификация?

Достижения методов формальной верификации

1988

Использование дедуктивного анализа позволило университету Оксфорда совместно с компанией Inmos разработать микропроцессор т.н. транспьютера и язык программирования этого микропроцессора Occam, существенно ускорив разработку (по сравнению с альтернативными проектами), обнаружив неоднозначность в стандарте IEEE и ошибку в подсхеме FPU у конкурентов

Работа была удостоена награды «Queen's award for technological achievement»

Зачем нужна формальная верификация?

Достижения методов формальной верификации

1988

Компании GEC Alsthom, MATRA Transport и RATP (автономное агентство парижского транспорта) завершили проект по компьютеризации системы управления парижским метро (RER) на языке Modula-2

Использование дедуктивного анализа для обоснования правильности отдельных модулей системы позволило избежать тестирования отдельных модулей и ограничиться только глобальным тестированием

Позже по той же методике была выполнена полная автоматизация одной из линий парижского метро

Зачем нужна формальная верификация?

Достижения методов формальной верификации

1992

National Westminster Bank и компания Platform Seven завершили проект по созданию электронной платёжной системы для smart-cards

Использование дедуктивного анализа позволило обнаружить и исправить уязвимости и получить доказательство соответствия требованиям 200-страничного стандарта безопасности

Зачем нужна формальная верификация?

Достижения методов формальной верификации

1992–настоящее время

Продолжается проект системы автоматического управлени подвижным барьером для защиты Роттердама от наводнений

Метод проверки моделей (в частности, средство Spin) применяется для обнаружения ошибок в спецификациях и коде на языке C

Это средство будет изучаться в курсе

Зачем нужна формальная верификация?

Достижения методов формальной верификации

1993

При помощи методов формальной верификации обнаружены ошибки в протоколе обеспечения когерентности кэшей FutureBus+ , принятого в качестве стандарта IEEE для шины высокопроизводительных компьютеров

Для выявления ошибок использовалась проверка моделей (средство SMV)

В курсе будет рассматриваться потомок этого средства, NuSMV

Зачем нужна формальная верификация?

Достижения методов формальной верификации

2009

Группой австралийский исследователей строго доказана корректность *nix-микроядра L4 (seL4) в предположениях об устройстве и надёжности аппаратуры, на которой запускается это ядро

Для доказательства использовался дедуктивный анализ со средством автоматизированного построения доказательств Isabelle/HOL

«We can predict precisely how the kernel will behave in every possible situation»

Этого средства в курсе не будет, увы

Зачем нужна формальная верификация?

Достижения методов формальной верификации

2003. Тони Хоар инициировал «grand challenge for computing research»: создание верифицирующего компилятора

2006. Создан и пополняется «The Verified Software Repository» (VSR)

Подразделения формальных методов верификации имеются во многих крупных компаниях, занимающихся программами и аппаратурой: Microsoft, Intel, Cisco, IBM, Cadence, Mentor Graphics, ...