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, 2023/2024, осенний семестр

Что такое формальная верификация

Наиболее популярный подход к отладке[1] вычислительной системы,[2]
известный каждому разработчику, — это тестирование:
система выполняется в разных обстоятельствах
(*на наборе тестовых сценариев*),
результаты выполнения сравниваются с ожидаемыми,
и при расхождении ищутся и устраняются причины

Зачастую[3] получить версию системы, подходящую для отладки, —
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**

1 Обнаружению и устранению ошибок
2 Например, программы, микросхемы, распределённой системы и многого другого
3 При разработке микросхем, распределённых систем, больших программных и программно-аппаратных комплексов и не только

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 строго обосновывать (*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 проверить справедливость этого утверждения, достаточно убедиться, что программа π_2 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 (∼ SystemC): высокоуровневое алгоритмическое описание

↓

Поведенческий уровень (∼ Verilog):
способ пересылки логических значений смежду схемными регистрами

↓

Логический уровень (∼ AIG): соединение триггеров и логических вентилей

↓

Транзисторный уровень: принципиальная схема соединения МОП-транзисторов

↓

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» формулой и
- применить известный метод проверки соответствия системы формуле

Спецификации в таком подходе зачастую основываются на

- логике предикатов (*полагаю, что все тут знают, что это такое, но всё равно позже коротко напомню*) и
- темпоральных логиках (*позволяющих описывать взаимосвязи событий, происходящих 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 (другие названия: 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 всё доказывается 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

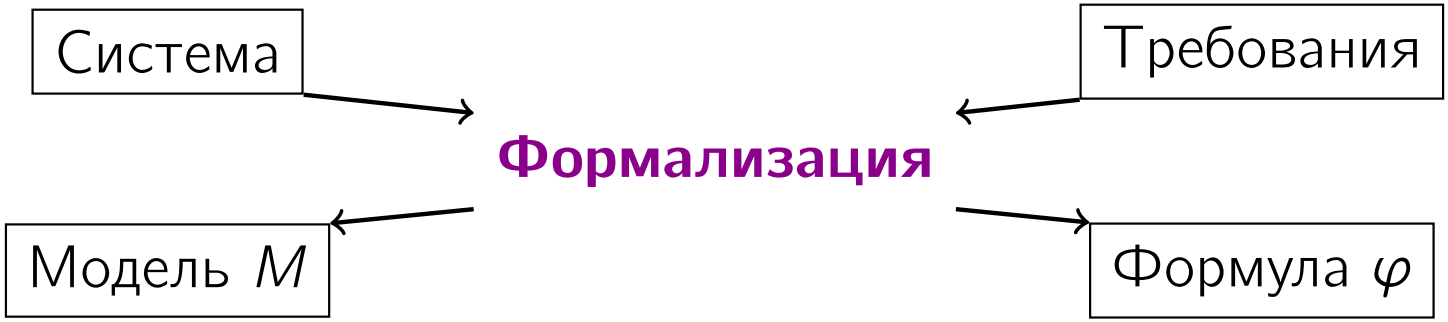
3. Проверяется выполнимость формулы φ на модели M:
$$M \models \varphi?$$

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



В начале применения метода проверки моделей строятся

- модель по реальной исследуемой системе и
- формула 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)

Разработать спецификацию и соответствующую формулу —
непростое дело: во многом ручное, и иногда непросто понять,
правильна ли формула; 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,

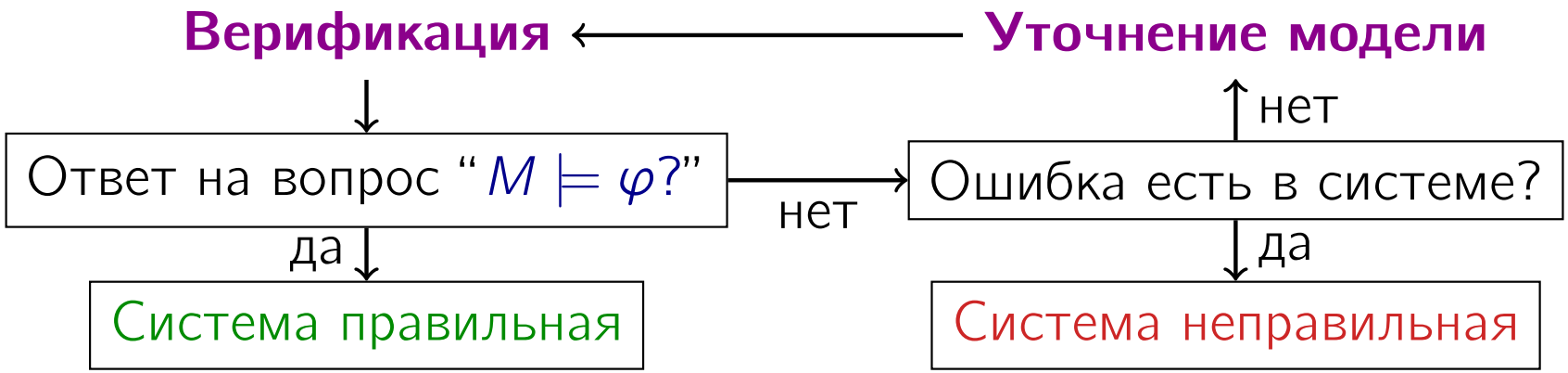
построение схем по булевым формулам,

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 при ошибочном продолжении работы,
так и при экстренном отключении

Помочь избежать такой 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,
  повышается вероятность ошибок и трудоёмкость их исправления
- с возрастанием сложности систем
  усложняются и методы их разработки, и вместе с этим
  разнообразие ошибок и процессы их устранения
  без необходимости возврата к начальным этапам разработки
- в некоторых системах (таких как микросхема на кристалле)
  исправить ошибку «локально» в готовом изделии
  вовсе невозможно, исправление сопряжено
  с полным 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?

Понять, насколько неприятными могут быть последствия даже небольших ошибок, достаточно лишь «слегка» поискать примеры в открытом доступе:[1]

- Отдаляется «светлое будущее человечества»
  - 1962, Маринер 1; 1988, Фобос-1; 1996, Ариан-5; 1999, Mars Climate Orbiter; 2005, зонд Гюйгенс; ...
- Корпорации получают неожиданные огромные убытки
  - 1994, Intel, сотни млн. \$; 2012, Knight Capital Group, 440 млн. \$; 2009, банк UBS, чудом избежали убытка 31 млрд. \$
- Умирают люди
  - 1985–1987, 2000, аппараты радиотерапии Therac-25 и схожий
  - 1995, Боинг-757 врезался в гору
  - 2003, ошибка в системе «свой-чужой» ЗРК MIM-104 Patriot
- Страдает быт людей с широким спектром последствий
  - 2003, массовое отключение электростанций в нескольких штатах США и провинциях Канады от нескольких часов до нескольких дней

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

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
- Нередко средства, методы и результаты формальной верификации
  умалчиваются из коммерческих соображений
  и соображений безопасности

Тем не менее известны и такие примеры

Зачем нужна формальная верификация?

Некоторые 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?

Некоторые 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?

Некоторые 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?

Некоторые 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?

Некоторые 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?

Некоторые 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?

Некоторые 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, ...