Математическая логика

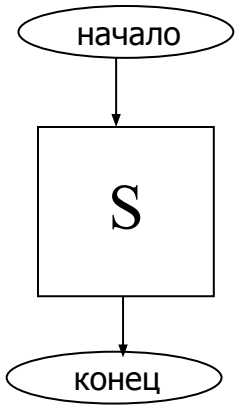
Карпов Юрий Глебович

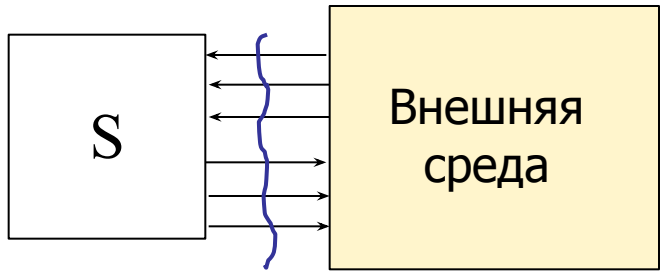
karpovkarpov@dcn.infos.ru

Верификация реагирующих программ
Лекция 7а

Необходимость верификации реагирующих программ

Трансформационные и реагирующие программы





- В 70-х годах прошлого века стала ясной необходимость разделения систем обработки информации (компьютерных программ и аппаратных систем) на два класса:
  - Системы обработки данных (трансформационные системы *transformation systems*).
    - Это обычные системы обработки: стартуют, вводят данные, обрабатывают их (трансформируют), выдают результат и останавливаются
  - Системы, взаимодействующие с окружением (реагирующие системы, *reactive systems*).
    - Эти системы всегда активны, не останавливаются, работают с СОБЫТИЯМИ, реагируя на них выдачей управляющих действий

Трансформационные и реагирующие программы (Amir Pnueli, 1977)

Тип системы	***Трансформационные программные системы***	***Reactive systems - Реагирующие программные системы***
Примеры	Пакетная обработка данных, вычисление значения функции, распознавание образов, ...	Протоколы, ОС, системы логического управления, программы встроенных систем (мобильные телефоны), ...
Цель	Преобразование данных, получение выходных значений как функции исходных данных	Обеспечение правил взаимодействия с окружением
Вычисления	Всегда конечны с получением результата в конце вычислений	Всегда активны; никогда не завершаются
Семантика	***Функция***: изменение состояния: состояние выхода программы является функцией от входного состояния (входных данных)	***Поведение***: последовательность событий и реакций системы на внешние события
Спецификация (требования к системе)	Например, в виде предикатов: пред- и постусловий: {Pre} S {Post}	Например, в виде формул темпоральной логики: S |= **G**(p ⇒**F**q)

Промышленные роботы

- THE NEW YORK TIMES. Промышленные роботы собирают Jeep Grand Cherokee на сборочном заводе Chrysler в Детройте, 2013 г.

Как гарантировать правильность работы систем дискретного управления?

Проблема корректности реагирующих систем

Проблема

- **Компьютеры – повсюду** *Ubiquitous computing* (*вездесущие вычисления*)
- Раньше компьютер был компьютером, а телефон – телефоном, и любой мог отличить одно от другого. Сейчас и компьютер – не только компьютер, и телефон – не только телефон (*A.Tanenbaum*)
- **Эра параллельных реагирующих систем**
- Параллельные системы распространяются все более широко.
- Эти системы - многоядерные чипы, встроенные бортовые системы логического управления, аппаратные комплексы – все они реагирующие системы
- **Параллельные системы полны ошибок**
- Параллельные системы непостижимы для человеческого мозга: они очень часто неправильны, содержат ошибки

Примеры последствий программных ошибок

- INTEL: Микропроцессоры содержат ~5 М переходов. В 1994 выпущен чип с ошибкой. Замена *миллионов* дефектных процессоров ⇒ потери ~$500 М
- 4.06.96 ракета *Ариан 5* (аналог Протона) взорвалась через 39 с. - ошибка переполнения при преобразовании 64-битового вещественного числа в 16-битовое целое. Ущерб > $600 млн.
- 1985-87 гг: Therac-25 прибор лучевой терапии. Пациенты получили передозировку, шестеро умерли, несколько стали инвалидами
- Война в Ираке, 23.03.2003. Ошибка в программе ⇒ система Patriot определила свой бомбардировщик Tornado как приближающуюся ракету. До 24% потерь в живой силе в первой Иракской войне – *"Friendly Fire"*
- Boeing 757, 1995 г (рейс из Майами в Кали, Колумбия). Ошибка в одном символе в Flight Management System привела к катастрофе. Погибли 159 чел
- Связь с советской АМС "Фобос-1" прервалась 2.09.88 г. из-за ошибочной команды, посланной с Земли. АМС потеряна, где-то летает сама по себе. То же с "Фобос-2"

СМИ о программных ошибках – каждый день

- Апрель, 2010. Авария в Мексиканском заливе: возможна ли программная ошибка? *Don Shafer, Phillip Laplante. The BP Oil Spill: Could Software be a Culprit? IEEE IT Pro September/October 2010, IEEE, 2010. ...* Не могла ли одной из причин бедствия стать ошибка в программном обеспечении?
- 05.07.2010. Apple: ошибка связи iPhone 4 — программная. Компания Apple признала существование в iPhone 4 проблем, касающихся качества связи.
- 06.12.2010. "Спутники ГЛОНАСС и ракету "Протон-М" утопили программисты". Ракета-носитель «Протон-М» со спутниками «Глонасс-М» отклонились от заданного курса из-за ошибок в математическом обеспечении (основная причина - неправильно написанная формула в документации на заправку кислородом разгонного блока)
- 08.12.2010 — Японский зонд «Акацуки» не смог выйти на орбиту Венеры

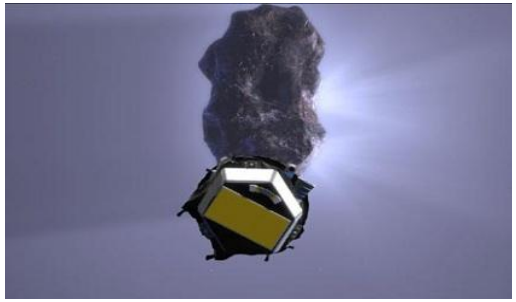
  Космический исследовательский аппарат «Акацуки», запущенный Японией для исследования Венеры, не смог выйти на орбиту планеты, сообщает РИА «Новости» со ссылкой на специалистов Японского аэрокосмического агентства JAXA
- 12.12.2014 - Воздушное пространство Великобритании закрыто из-за компьютерного сбоя. Об этом сообщает Би-би-си

Примеры недавних ошибок

- 2010. Toyota отозвала с рынков >8 млн проданных автомобилей из-за неисправности педали газа и дефекта ПО тормозной системы. Это обойдется ей в $2 млрд. Глюк автомобилей Toyota стал причиной не менее 5 смертей.
- Этот инцидент подрывает доверие к торговой марке Тойоты на многие годы. Акио Тойда, президент компании Тойота, 24.02.2010 выступил в Конгрессе США и принес извинения за проблемы, связанные с качеством автомобилей
- По мнению экспертов, проблемы Toyota вызваны переходом автомобильной индустрии на электронные системы даже в критически важных функциональных узлах
  - Эксперты отмечают, что основная проблема именно в компьютерной системе управления, в которой не было предусмотрено экстренное торможение: педаль тормоза просто не срабатывала, когда сенсор акселератора командовал автомобилю разгоняться.
  - Эта история поднимает новые проблемы перед конструкторами и программистами, которые разрабатывают системы управления. Проблемы с электроникой и компьютерные сбои будут встречаться всё чаще

Проблемы с программным управлением в космосе

Октябрь, 2010. Новые спутники GPS IIF, выпущенные корпорацией Boeing, требуют доработки программного обеспечения, отвечающего за слежение за ядерными взрывами на поверхности Земли, заявил генерал-лейтенант Том Шеридан, исполнительный офицер космических программ ВВС США.



21.09.2013. Специалисты NASA объявили о прекращении попыток восстановить связь с исследовательским зондом Deep Impact

- Последний успешный сеанс связи с Deep Impact состоялся 8 августа 2013, после чего сообщение с зондом было потеряно
- По предварительным оценкам инженеров NASA, причиной обрыва связи стали ошибки в программном обеспечении, из-за которых зонд потерял ориентацию в пространстве

Примеры ошибок

- Почти в каждой крупной техногенной аварии существуют следы ошибок в системах программного мониторинга и управления
- Авария на Саяно-Шушенской ГЭС 17.08.2009 унесла жизни 75 человек
- Автоматическая система управления технологическими процессами станции не предупредила о надвигающейся аварии по показаниям датчиков

Несанкционированный запуск двигателя Союза – МКС переместилась

- 9 июня 2015 г. на несколько секунд несанкционированно запустились двигатели корабля «Союз», пристыкованного к МКС. На нем космонавты должны будут возвращаться на Землю. В результате произошло изменение положения МКС.
- РИА Новости: "*Причиной нештатного запуска двигателей «Союза» является программная ошибка*"

Ошибки параллельных программ

- Системы программного управления обычно строятся из параллельных взаимодействующих модулей. Ошибки в них часто являются критическими
- Легкость написания синтаксически правильного кода составляет контраст с наиболее сложной составляющей программной системы – с ее поведением, что является причиной наибольшей части сложных ошибок в программах
- Параллельные программы работают правильно “почти всегда”
- Параллельные программы могут годами сохранять ошибки, проявляющиеся после долгой эксплуатации как реакция на возникшую специфическую комбинацию многочисленных факторов, в частности, непредсказуемых скоростей выполнения отдельных процессов в параллельных программах

Параллельные системы сохраняют ошибки

Нетривиальные параллельные и распределенные программные системы непостижимы для человеческого мозга

“Существует обширный печальный опыт того, что параллельные программы упорно сопротивляются серьезным усилиям по выявлению в них ошибок”

S. Owicki, L. Lamport “Proving liveness properties of concurrent programs”, ACM TOPLAS, N4, 1982

Несет ли профессионал ответственность за человеческие жизни?

Программирование является единственной областью инженерной деятельности,
где разработчик не отвечает за качество своей работы

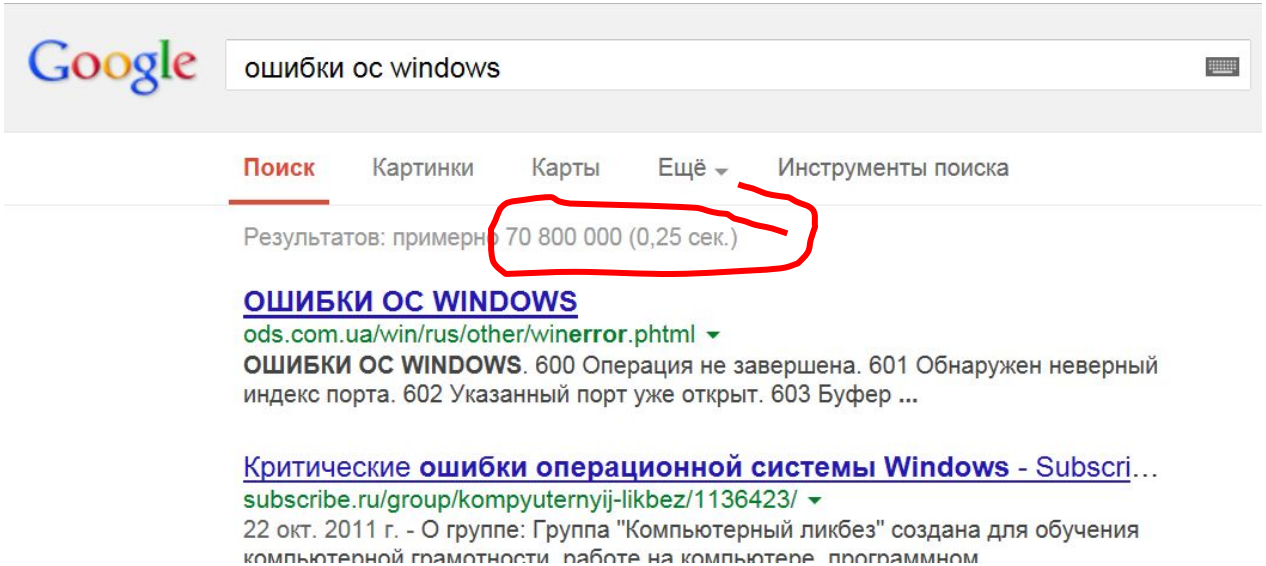
Обычно ПО имеет 10-15 ошибок на 1000 строк кода,
ПО высокого качества – 3 ошибки на 1000 строк кода

Современное ПО содержит миллионы строк кода,
уже сданные программы наполнены ошибками

Тяжелые последствия ошибок ПО - в Интернете:

SOFTWARE HORROR STORIES: http://www.cs.tau.ac.il/~nachumd/horror.html
Collection of Software Bugs: http://www5.in.tum.de/~huckle/bugse.html
Worst software defects in history:
http://cristianpocovnicu.wordpress.com/2011/03/22/worst-software-defects-in-history/
Википедия: http://en.wikipedia.org/wiki/List_of_software_bugs

Миллионы сообщений об ошибках в ОС Windows



- По заявлению Майкрософта, в ОС Windows остаются тысячи ошибок!
- Июнь 2013: Microsoft анонсировала программу премирования пользователей, отыскавших уязвимости в предварительной версии операционной системы Windows 8.1. Тем, кто обнаружит ошибки и найдет способы обойти встроенную в ОС защиту, обещают выплатить до $100 000.

Можно ли поставить ОС Windows на спец компьютер, управляющий атомной электростанцией?

Инженеру необходимо гарантировать правильную работу созданных им артефактов

- Обеспечение качества разработанной технической системы является важнейшей составляющей каждой инженерной специальности
- Конструкторы самолетов изучают аэродинамику, строители – сопротивление материалов, машиностроители изучают теорию механизмов и машин
- Каждая инженерная специальность имеет свою область прикладной математики, на которой основаны теории, позволяющие гарантировать качество инженерных разработок в этих областях
- В области разработки ПО и встроенных систем логического управления также необходимые свои разделы прикладной математики – это формальные методы, на которых основывается верификация

Эти разделы и методы специфические: мы должны гарантировать правильность поведения созданных нами динамических дискретных систем. Нужно определить, что такое поведение дискретных систем управления, определить язык спецификации формальных требований к поведению, алгоритмы проверки того, что эти требования выполняются

Грустная история

- До последнего времени методы верификации могли быть применены для доказательства корректности только “toy systems” а не промышленного ПО, но даже для простых программ требовались огромные усилия

- С 60 годов 20 века много лет большое число исследователей работало над проблемой доказательства правильности программ, однако, эта проблема еще недавно не имела удовлетворительного решения
- Еще 20 лет назад использование формальных методов и верификации было весьма далеким от практики:
  - нотация сложна и неясна, ошибки встречались и в доказательствах
  - методы анализа были не масштабируемы (not scalable)
  - автоматические инструменты неадекватны и трудны в использовании; фактически, верификация проводилась интерактивно
  - были доказаны только тривиальные примеры
  - требовалась высокая квалификация для спецификации и анализа

Model checking: прорыв в области верификации

В последнее время – качественный прорыв в одном из направлений исследований в области верификации ПО и дискретных систем, основанный на изящных формальных методах - model checking

Model checking (проверка модели):

методы и алгоритмы верификации аппаратуры и программ разработаны от теоретических изысканий до *индустриальной технологии*

(*Индустриальные (industrial) – используемые на практике*)

Model checking

- Определение:

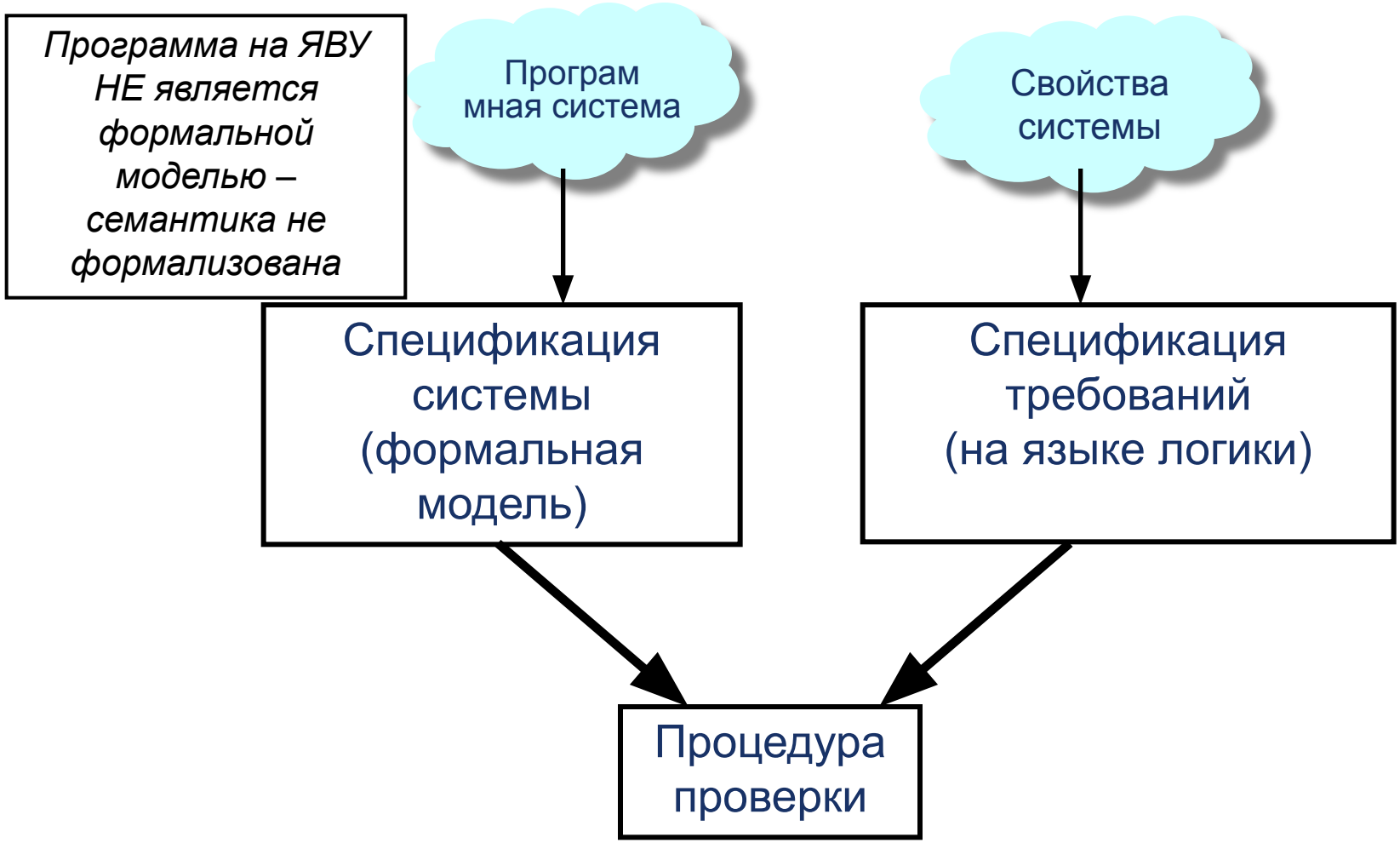
Model checking – это процедура, которая устанавливает, выполняется ли на данной структуре M логическая формула Ф, или, короче, удовлетворяет ли M формуле Ф:

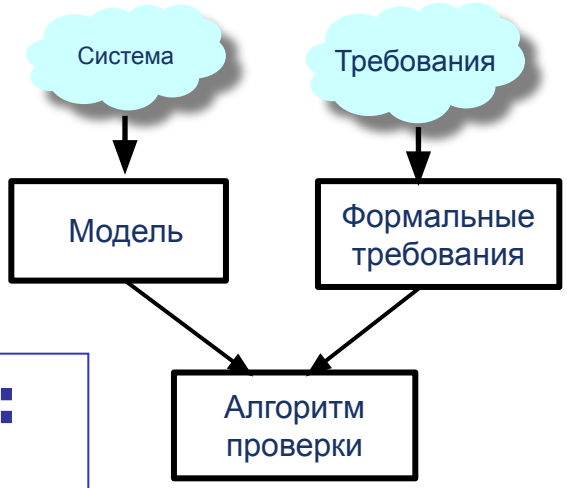
$$M \models \Phi?$$

- Обычно M – это абстрактная модель анализируемой системы (обычно конечная система переходов), а Ф – формула темпоральной или модальной логики, выражающая желаемое свойство поведения дискретной (программной или аппаратной) системы

Общий подход к верификации систем

Формально проверить можно только формальные объекты





Современный программист НЕ ЗНАЕТ:

- *какие модели использовать для формального представления реагирующей системы*
- *как формально выразить требования к работе системы*
- *как проверить, что эти требования выполняются*

Наша задача – рассмотреть все эти проблемы

Заключение. Вопросы по разделу

- Что такое реагирующие программы (reactive programs)?
  - Это специфический класс программных систем, целью которых является не преобразование входных данных в выходные, а поддержание специфических шаблонов взаимодействия со средой (другими системами, человеком, управляемыми объектами, другими модулями системы). Выделил этот класс программ Амир Пнуэли в 70-х гг. прошлого века. Это протоколы, операционные системы, системы логического управления
- Чем отличается реагирующая программа?
  - Она не останавливается, ее останов – ошибка. Такая система постоянно ждет событий от среды и реагирует на эти события в зависимости от своего состояния
- Почему верификация реагирующих программ является важной проблемой?
  - Все системы управления сложными техническими системами последнее время строятся на основе встроенных программных компонент. Ошибки в системах управления часто имеют катастрофические последствия. Поскольку обычно такие системы являются параллельными, ошибки в них очень сложно обнаружить
- Какие проблемы будут рассмотрены в курсе?
  - Как построить формальную модель реагирующей системы
  - Как формально выразить требования к поведению реагирующей системы
  - Каким алгоритмом можно проверить, что модель реагирующей системы удовлетворяет формальным требованиям к ее поведению

Спасибо за внимание