CS5030

AUTOMATED PROGRAM VERIFICATION

WHO, WHERE AND WHEN

- Instructor : Kartik Nagar
- Online on Google Meet
- Slot B
    - Monday 9 AM, Tuesday 8 AM, Wednesday 12 PM, Friday 11 AM.
- Course Webpage : https://kartiknagar.github.io/courses/apv/
- Moodle : https://courses.iitm.ac.in/course/view.php?id=6286

WHAT IS PROGRAM VERIFICATION?

Ensuring that the program does what it is supposed to do.

Testing is the most common strategy used to ‘verify’ programs, from colleges to big companies.

DOES TESTING ACHIEVE PROGRAM VERIFICATION?

Ensuring that the program does what it is supposed to do on some inputs. ✓

Testing is the most common strategy used to 'verify' programs, from colleges to big companies.

DOES TESTING ACHIEVE PROGRAM VERIFICATION?

Ensuring that the program does **always** what it is supposed to do. ✗

Testing is the most common strategy used to 'verify' programs, from colleges to big companies.

WHY IS VERIFICATION NEEDED

- Bugs are rampant and can have catastrophic consequences

MOST (IN)FAMOUS BUGS IN HISTORY

THE ARIANE 5 DISASTER [1996]

- On June 4, 1996, the Ariane 5 Rocket began its flight and after 37 seconds, suddenly took a 90 degree flip and self-destructed in a gigantic explosion.
- The disaster cost $370 Million.
- Happened because of an integer overflow error!
- Widely acknowledged as the most expensive bug of all time.

MOST (IN)FAMOUS BUGS IN HISTORY

THERAC-25 KILLER BUG [1985-87]

- The Therac-25 was a radiation therapy machine used for treatment of cancer through concentrated doses of radiation.
- Between 1985 and 1987, the machine was the cause of six radiation-overdose accidents, resulting at least 2 deaths due to direct consequences of the overdose.
- Happened due to buggy synchronization between the software and the radiation hardware resulting in a race condition.

MORE CATASTROPHIC BUGS IN HISTORY…

- Boeing 737 Max Bug [2019]. Estimated Loss of $9.2 Billion.
- The DAO Smart Contract Hack [2017]. Loss of 3.6 Million Ether ($50 Million).
- Mars Climate Orbiter Crash [1999]. Loss of $235 Million.
- …

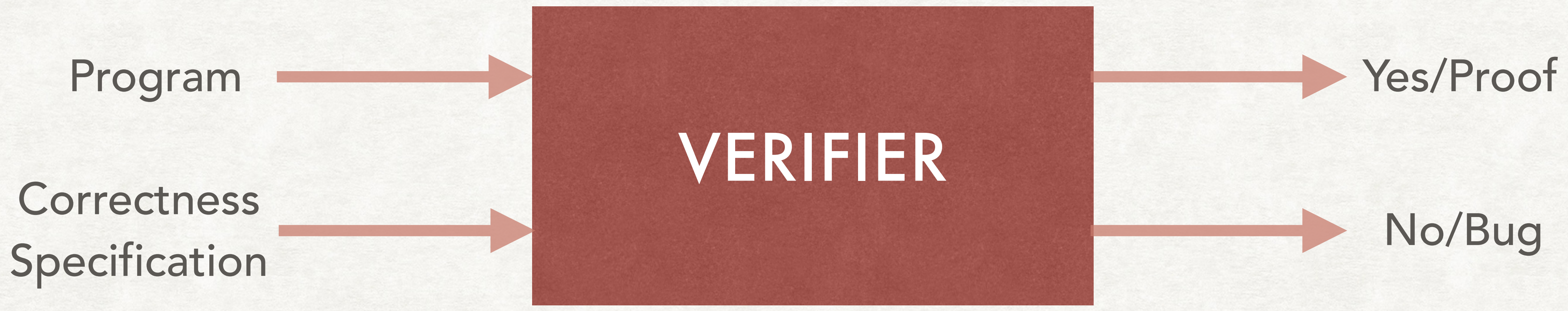
WHY IS VERIFICATION NEEDED

- Bugs are rampant and can have catastrophic consequences
- Bugs occur due to many reasons
  - Failure to understand the programming language semantics
  - Failure to take into account the behaviour of the underlying system (especially relevant for concurrent, distributed, cyber-physical systems)
  - Changes in the requirements/updates in the system
  - Higher volume of software with multiple developers
  - …

WHERE IS VERIFICATION NEEDED

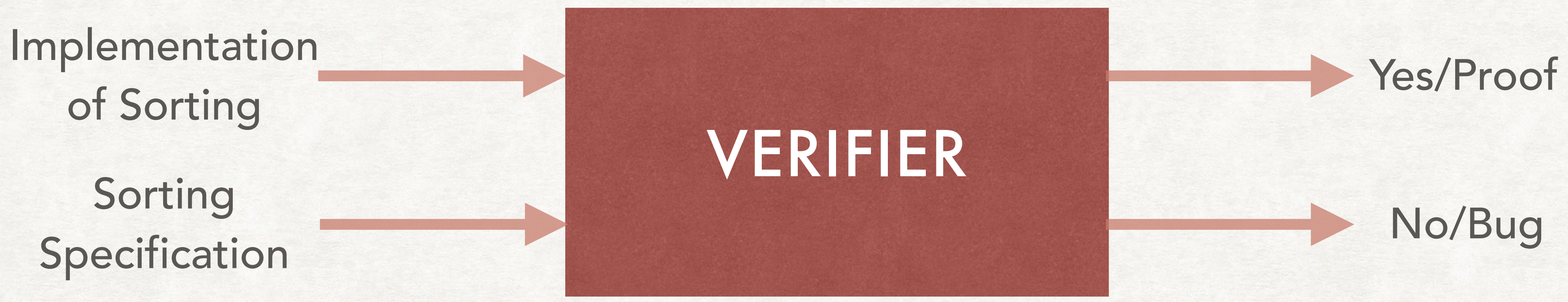
- There is a growing need for verified software in many areas
  - Aerospace, Avionics, Automobiles
  - Medical devices
  - Financial software
  - Operating Systems, Compilers, Software Libraries
  - Network Protocol Implementations
  - Any code ever written???

AUTOMATED VERIFICATION

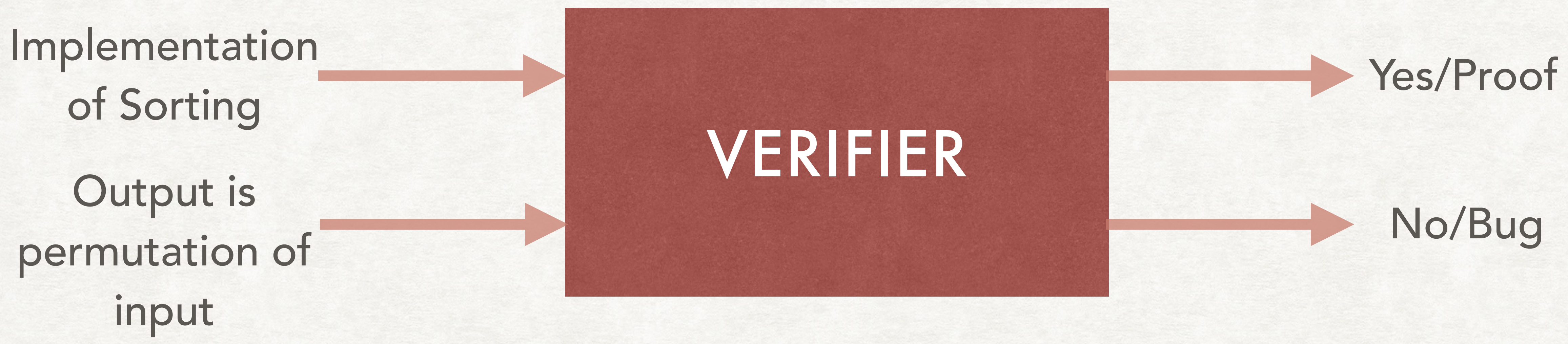


- Correctness is generally specified in the form of a formal mathematical specification
- The specification should hold for all executions of the program
- The verification is always with respect to the specification

AUTOMATED VERIFICATION



AUTOMATED VERIFICATION



QUESTIONS

QUESTIONS

- Is it possible that a program is verified but it still has bugs?

QUESTIONS

- Is it possible that a program is verified but it still has bugs?
- Is it possible that the verifier says the program is not correct but the program actually has no bugs?

QUESTIONS

- Is it possible that a program is verified but it still has bugs?
- Is it possible that the verifier says the program is not correct but the program actually has no bugs?
- Are there truly any bug-free programs? Conversely, can all program be verified to be correct?

UNDECIDABILITY OF VERIFICATION

- Automated verification of programs written in a Turing-complete language is undecidable.
  - Rice's Theorem: There is no Turing machine that can decide whether the language accepted by a given Turing machine has a non-trivial property.
- In practice, automated verifiers either restrict the space of programs, or the space of specifications.
  - Even then, verification is computationally expensive.

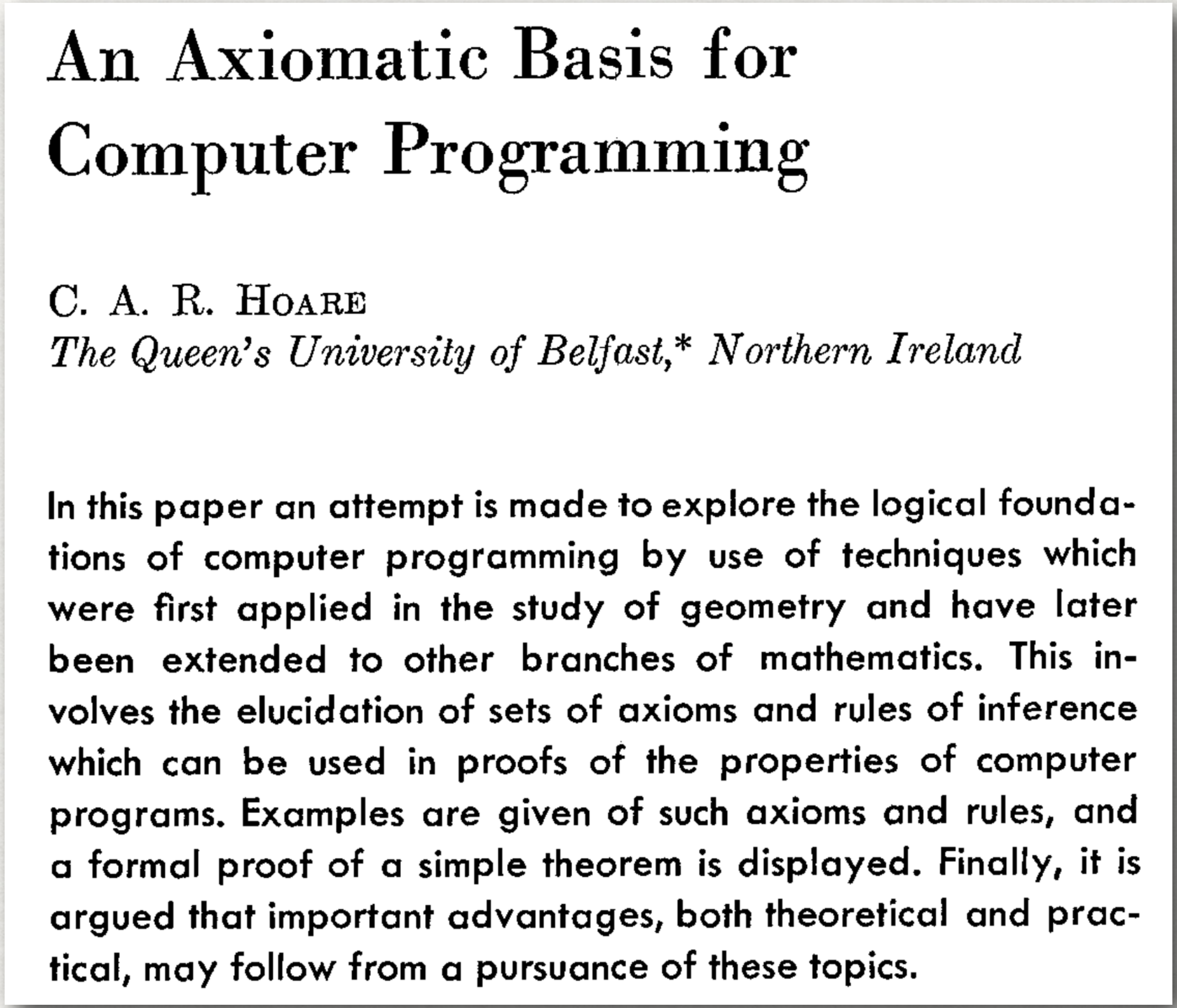
A BRIEF HISTORY OF VERIFICATION

- Three main threads of Program Verification
  - Proofs of Programs [1970-]
  - Model Checking [1980-]
  - Constraint solving [1990-]

THREAD 1: PROOFS OF PROGRAMS [1970-]

DEDUCTIVE VERIFICATION

- Pioneered by the seminal paper by Tony Hoare in 1969.
- Deductive Verification uses a set of inference rules and axioms and applies them to construct the proof of correctness.
- Several semi-automated tools (also called Theorem provers) such as Coq, Isabelle, ACL2, etc. are widely used today and continue this mode of verification.



An Axiomatic Basis for Computer Programming

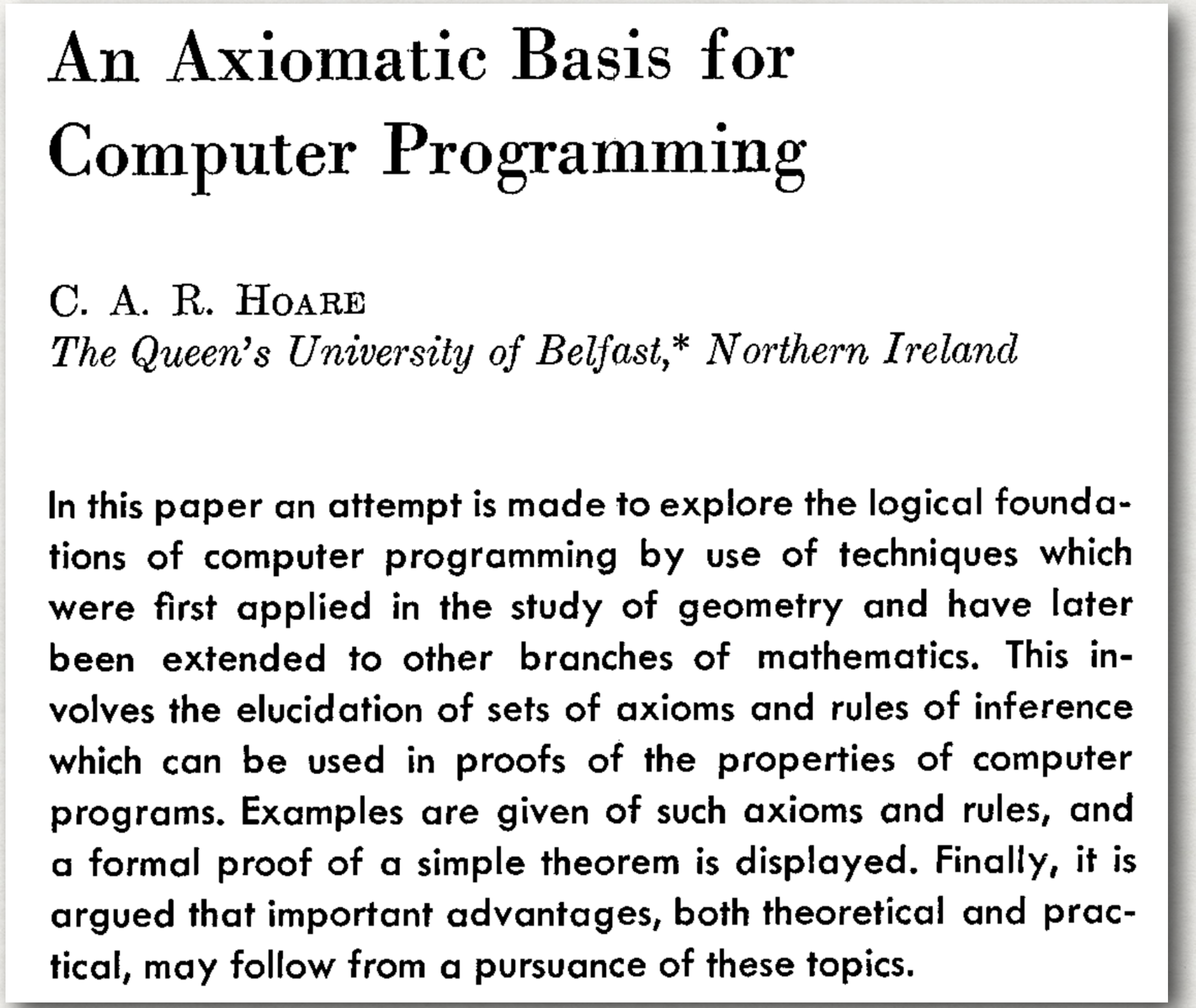
C. A. R. HOARE
The Queen's University of Belfast, Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

THREAD 1: PROOFS OF PROGRAMS [1970-]

DEDUCTIVE VERIFICATION

- Pioneered by the seminal paper by Tony Hoare in 1969.
- Deductive Verification uses a set of inference rules and axioms and applies them to construct the proof of correctness.
- Several semi-automated tools (also called Theorem provers) such as Coq, Isabelle, ACL2, etc. are widely used today and continue this mode of verification.



An Axiomatic Basis for Computer Programming

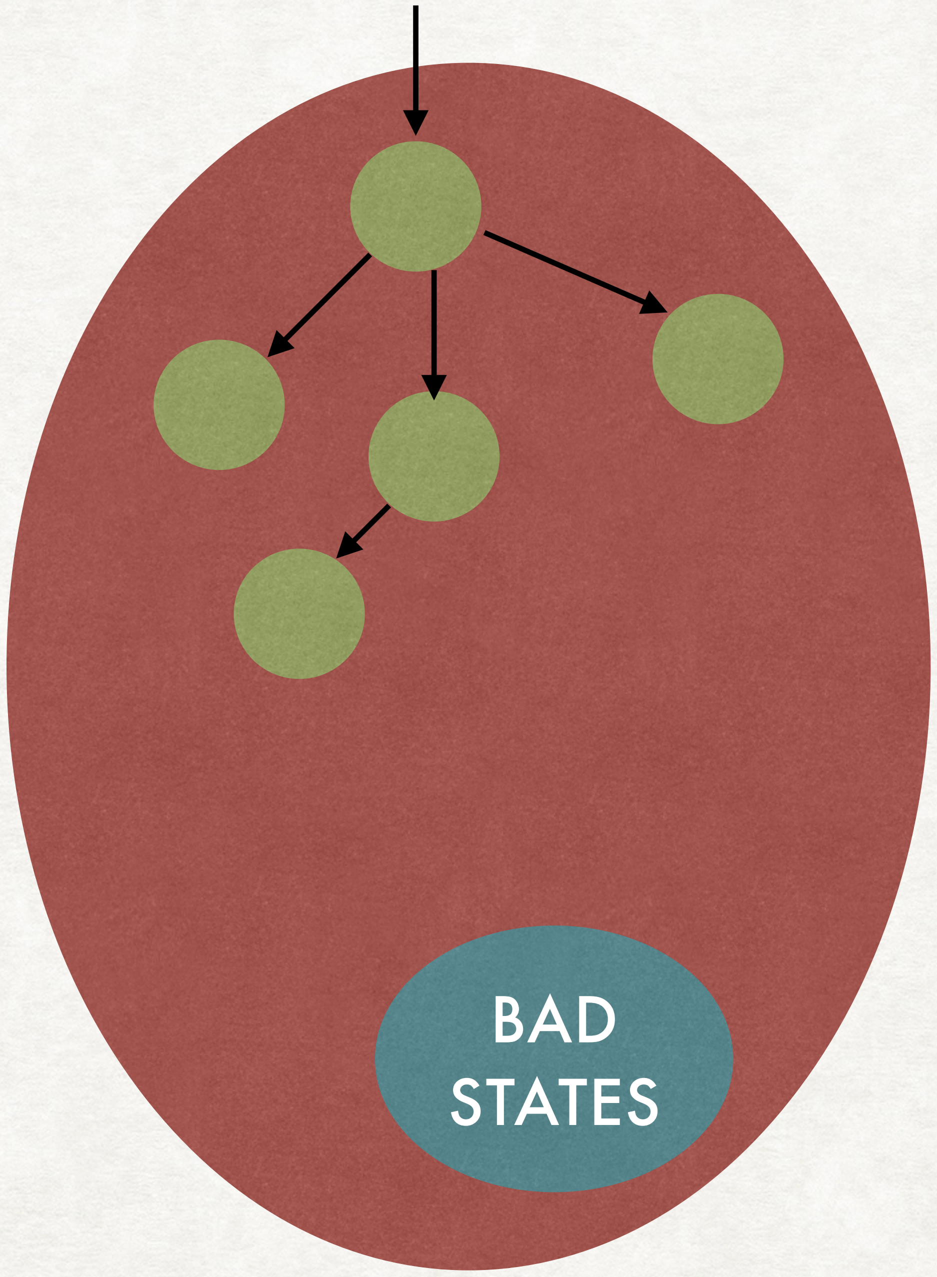
C. A. R. Hoare
The Queen's University of Belfast, Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

TONY HOARE WON THE TURING AWARD IN 1980 "FOR FUNDAMENTAL CONTRIBUTIONS TO THE DEFINITION AND DESIGN OF PROGRAMMING LANGUAGES"

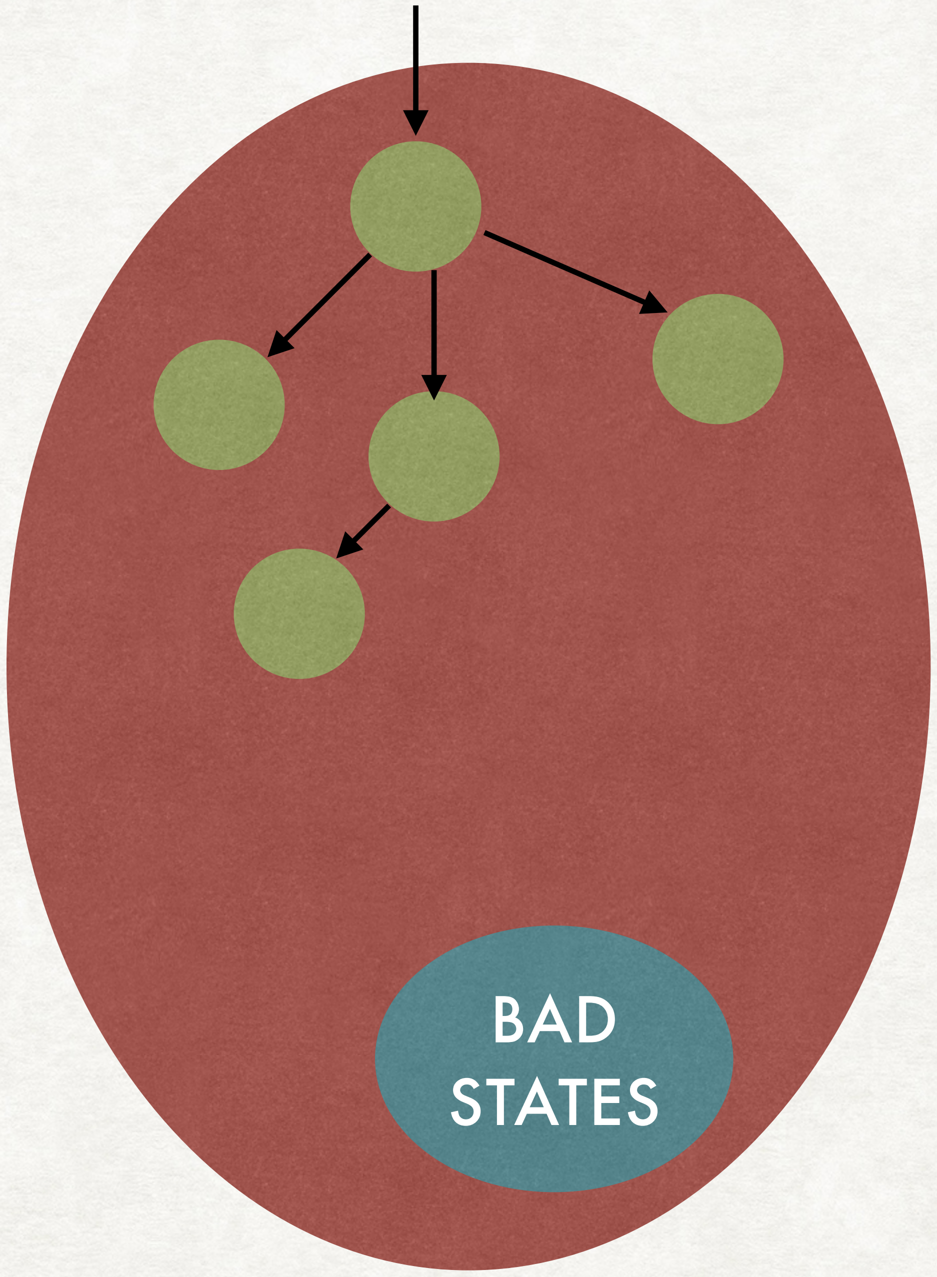
THREAD 2: MODEL CHECKING[1980-]

- For finite-state systems, Model Checking reduces the verification problem to the reachability problem in transition systems.
- Pioneered by Clarke, Emerson, Quielle and Sifakis in the 1980s.
- Initially used for concurrent program verification and protocol analysis.



THREAD 2: MODEL CHECKING[1980-]

- For finite-state systems, Model Checking reduces the verification problem to the reachability problem in transition systems.
- Pioneered by Clarke, Emerson, Quielle and Sifakis in the 1980s.
- Initially used for concurrent program verification and protocol analysis.



CLARKE, EMERSON AND SIFAKIS WON THE TURING AWARD IN 2007 "FOR THEIR SEMINAL WORK FOUNDING AND DEVELOPING THE FIELD OF MODEL CHECKING"

THREAD 2: MODEL CHECKING[1980-]

- Linear Temporal Logic (LTL) and Computational Tree Logic (CTL) were developed for expressing complex properties over transition systems.
- Symbolic Model Checking was developed in the 1990s for verification of hardware.
  - States are valuations of a set of boolean variables, a transition is a formula in Propositional Logic relating new values with old values.
  - The symbolic reachability problem is reduced to that of satisfiability of PL formulae.
  - Drove a lot of research in efficient procedures for the SAT problem (e.g. Binary Decision Diagrams).
  - Successfully used by IBM, Intel, Cadence, Synopsis…

THREAD 2: MODEL CHECKING[1980-]

SOFTWARE MODEL CHECKING

- Predicate Abstraction: Given a program P, build another program A which only consists of boolean variables, such that A over-approximates behaviours of P.
    - If A is safe, then P is also safe.
    - If A is unsafe, P may or may not be safe. Leads to Counter Example Guided Abstraction Refinement (CEGAR).
- Successfully used in the SLAM Project by Microsoft for verification of device drivers for Windows OS.

THREAD 2: MODEL CHECKING[1980-]

SOFTWARE MODEL CHECKING

- Predicate Abstraction: Given a program P, build another program A which only consists of boolean variables, such that A over-approximates behaviours of P.
    - If A is safe, then P is also safe.
    - If A is unsafe, P may or may not be safe. Leads to Counter Example Guided Abstraction Refinement (CEGAR).
- Successfully used in the SLAM Project by Microsoft for verification of device drivers for Windows OS.

THINGS LIKE EVEN SOFTWARE VERIFICATION, THIS HAS BEEN THE HOLY GRAIL OF COMPUTER SCIENCE FOR MANY DECADES BUT NOW IN SOME VERY KEY AREAS, FOR EXAMPLE, DRIVER VERIFICATION WE'RE BUILDING TOOLS THAT CAN DO ACTUAL PROOF ABOUT THE SOFTWARE AND HOW IT WORKS IN ORDER TO GUARANTEE THE RELIABILITY.

– BILL GATES (2002)

THREAD 3: CONSTRAINT SOLVERS [1990-]

THREAD 3: CONSTRAINT SOLVERS [1990-]

- SAT is the original NP-Complete Problem, but decades of research in SAT solving has led to efficient solvers which can easily handle thousands of variables and millions of clauses.

THREAD 3: CONSTRAINT SOLVERS [1990-]

- SAT is the original NP-Complete Problem, but decades of research in SAT solving has led to efficient solvers which can easily handle thousands of variables and millions of clauses.
- This was complemented by rise of Satisfiability Modulo Theories (SMT).

THREAD 3: CONSTRAINT SOLVERS [1990-]

- SAT is the original NP-Complete Problem, but decades of research in SAT solving has led to efficient solvers which can easily handle thousands of variables and millions of clauses.
- This was complemented by rise of Satisfiability Modulo Theories (SMT).
    - Specialised Algorithms for SMT for various theories such as Linear Arithmetic, Bit Vector Logic, etc.

THREAD 3: CONSTRAINT SOLVERS [1990-]

- SAT is the original NP-Complete Problem, but decades of research in SAT solving has led to efficient solvers which can easily handle thousands of variables and millions of clauses.
- This was complemented by rise of Satisfiability Modulo Theories (SMT).
    - Specialised Algorithms for SMT for various theories such as Linear Arithmetic, Bit Vector Logic, etc.
    - Z3, CVC4, …

THREAD 3: CONSTRAINT SOLVERS [1990-]

- SAT is the original NP-Complete Problem, but decades of research in SAT solving has led to efficient solvers which can easily handle thousands of variables and millions of clauses.
- This was complemented by rise of Satisfiability Modulo Theories (SMT).
  - Specialised Algorithms for SMT for various theories such as Linear Arithmetic, Bit Vector Logic, etc.
  - Z3, CVC4, …
- A number of verifiers (deductive, model-checking, combination…) use constraint solvers in the backend.

THREAD 3: CONSTRAINT SOLVERS [1990-]

- SAT is the original NP-Complete Problem, but decades of research in SAT solving has led to efficient solvers which can easily handle thousands of variables and millions of clauses.
- This was complemented by rise of Satisfiability Modulo Theories (SMT).
  - Specialised Algorithms for SMT for various theories such as Linear Arithmetic, Bit Vector Logic, etc.
  - Z3, CVC4, …
- A number of verifiers (deductive, model-checking, combination…) use constraint solvers in the backend.
  - CBMC, Dafny, Seahorn, VCC, Sage…

IN THIS COURSE...

- We will mostly focus on **Deductive Verification using Constraint Solvers**.
- We will also have a brief look at model-checking based approaches.
- Note that we will just scratch the surface in the area of verification
  - Topics **not** covered: Verification for concurrent and distributed systems (a vast area, suitable for another course), Verification for Heap-based programs, Language-based Verification, Program Synthesis, Verification of Hybrid/Cyber-physical systems,…
  - Many topics to choose from for Project!

COURSE LOGISTICS

- Grading Policy (tentative)
    - Project - 40%
    - Assignments (2) - 30%
    - Final Exam - 30%
- Assignment Dates (tentative)
    - Assignment 1: Out on Sep 28, Due Oct 7
    - Assignment 2: Out on Oct 19, Due Oct 28
- Textbook
    - The Calculus of Computation: Decision Procedures with Applications to Verification. Aaron R. Bradley and Zohar Manna.
    - Chapters will be uploaded to Moodle.

COURSE PROJECT

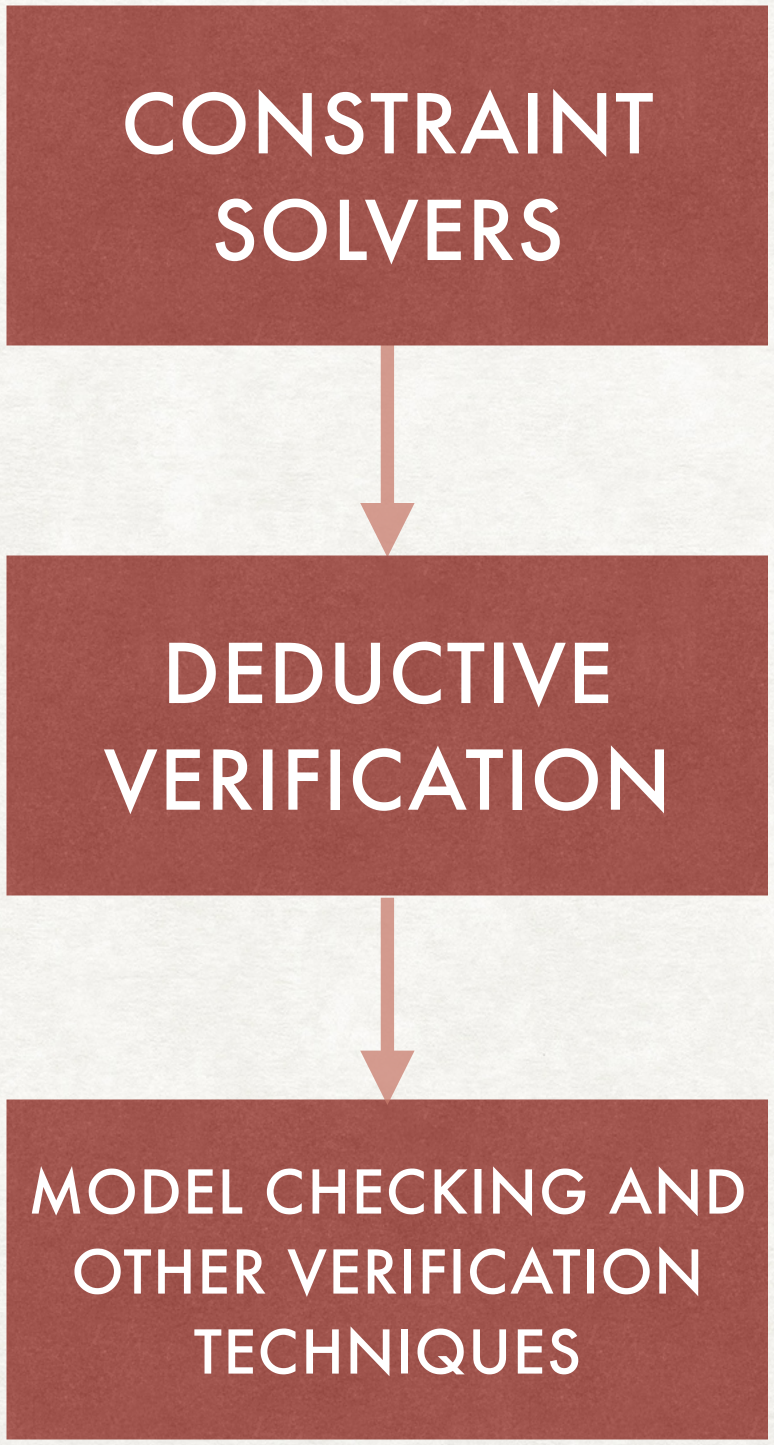
COURSE PROJECT

- Research-oriented project
  - Could be an in-depth survey of a sub-area of Verification, re-implementing a verification technique, applying verification techniques to new areas, etc.
  - Browse through recent editions of conferences like CAV, POPL, PLDI, ESOP, OOPSLA, etc.

COURSE PROJECT

- Research-oriented project
  - Could be an in-depth survey of a sub-area of Verification, re-implementing a verification technique, applying verification techniques to new areas, etc.
  - Browse through recent editions of conferences like CAV, POPL, PLDI, ESOP, OOPSLA, etc.
- Deliverables
  - Project Proposal: Should be a self-contained document introducing the project and providing the exact list of tasks to be performed and final deliverables [2-4 Pages]. Due Date - October 14.
  - Presentation/Demo: Last week of the semester.
  - Project Report [10 Pages]: Due after the presentations.

COURSE STRUCTURE



- Propositional Logic, SAT solving, DPLL
- First-Order Logic, SMT
- First-Order Theories

- Operational Semantics
- Strongest Post-condition, Weakest Pre-condition
- Hoare Logic

- Predicate Abstraction, CEGAR
- Abstract Interpretation
- Property-directed Reachability
- …