Model Checking: Concepts, Techniques and Tools

Lect 1: Course Introduction

Paritosh Pandya

Indian Institute of Technology, Bombay
Course URL: http://www.tcs.tifr.res.in/∼pandya/grad/CS738.html

Autumn, 2020

Model Checking in Nutshell

Model Checker

A program that analyzes whether a given program behaves correctly for all possible inputs.

Alan Turing

Alan Turing: Halting Problem

It is impossible to construct a program (TM) which analyses whether a given program (TM) on a given input halts.

In spite of this impossibility, with intense research, we have discovered analysis algorithms for many classes of programs and many forms of correctness properties.

Correctness in Software

Improving Reliability of Programs.

Techniques

- Software metrics
- Testing
- Code walkthrough
- Managing Software Development Process (CMM Maturity Levels)
- n-version programming

Software Crisis Nato Software Engineering Conference in 1960s.

Basis of Correctness in Engineering

- Mathematical Modelling of Designs
- Analysis and Synthesis Techniques
- Rigorous Mathematical Specification of Products

Basis of Correctness in Engineering

- Mathematical Modelling of Designs
- Analysis and Synthesis Techniques
- Rigorous Mathematical Specification of Products

Software: Engineering or Craft?

- No rigorous specification of the intended behaviour of the system.
- No mathematical analysis of designs for correctness. Validation is by testing.
- Correctness is implicit and uncertain. Products are unreliable.

State of Affairs

Disasters:

- Intel Pentium SRT Division Bug (1995) (Cost 1 Billion $).
- Arian 5 Launch Failure (1996) (Cost 850 Million $).
- Mars Polar lander (1999). Incomplete requirements.
- Indian PSLV Failed Launch

Changing Face of Hardware and Software

- Multi core processors,new memory models.
- New architectures: clusters, Grids, Multi-agent systems, Service oriented computing
- IoT, Concurrent asynchronous modules with mediated interaction

Formal Methods in Programming

- Mathematical Models for program behaviours. Automata.
- Logical notations for specifying properties of programs. Predicate and Temporal logics
- Methods for checking that program meets its desired specification.

Verification Problem

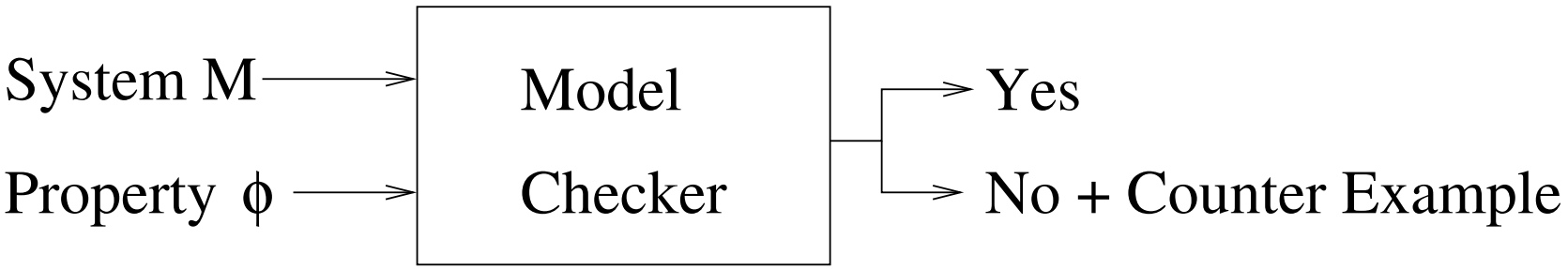
To check whether $M \models \phi$

- M system model in programming/modelling language.
- ϕ property in specification notation.

Must check that all behaviours of M satisfy ϕ.

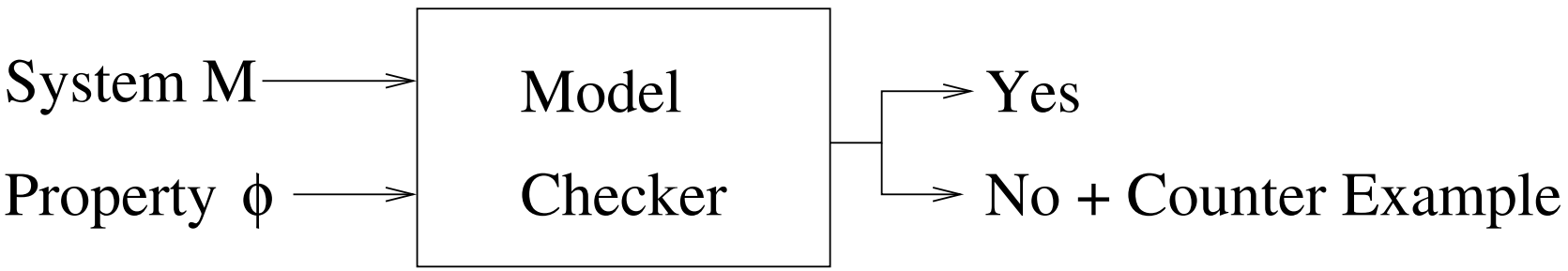
Model Checking: Algorithmic Verification

- Model checker is an algorithm which, given a system model M and property ϕ, determines whether $M \models \phi$.
- A good model checker will produce a counter example if $M \not\models \phi$.



Model Checking: Algorithmic Verification

- Model checker is an algorithm which, given a system model M and property ϕ, determines whether $M \models \phi$.
- A good model checker will produce a counter example if $M \not\models \phi$.



Applicability

Digital circuit (RTL design) verification
Verification of Embedded systems controllers
Protocol verification (networks, IoT, mobile telephony, computer architecture)

Reactive Systems

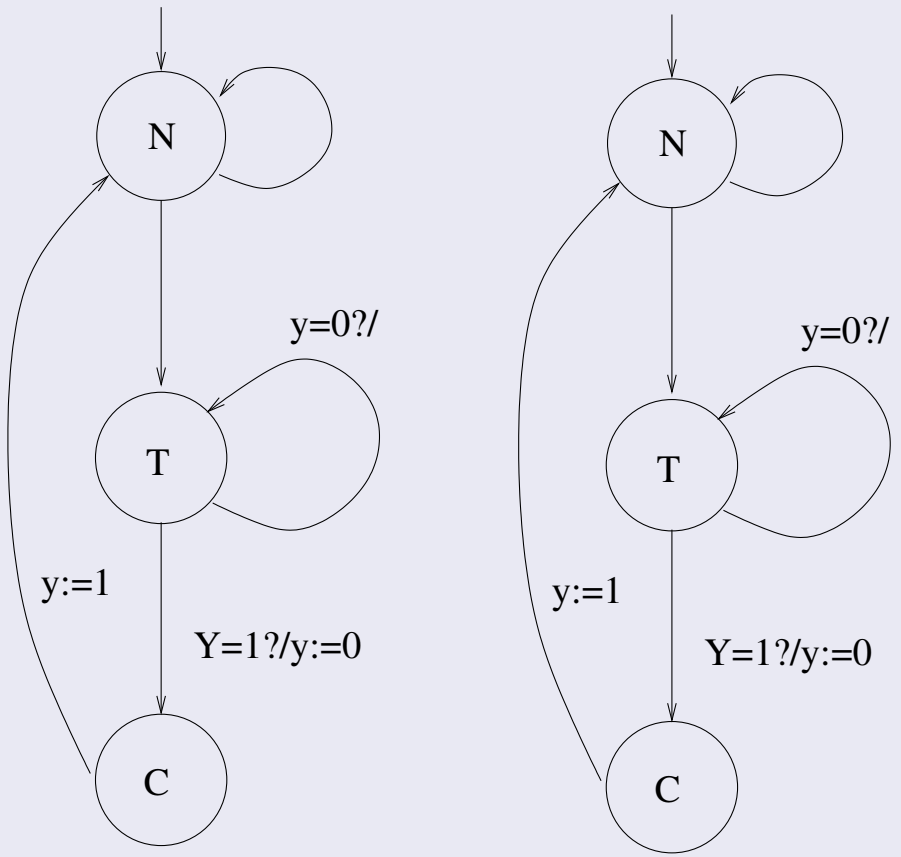
Reactive systems are complex requiring verification.

- Reactive: Output depends on interaction with environment.
- Temporal: Current output depends on past *sequence* of inputs.
- Global: Output from a component may depend on all other components due to interaction.
- Safety Critical Often used in applications requiring high degree of reliability.
- Difficult to test.
  - Failure is not repeatable.
  - Cannot collect enough test data.

Verification question: Do all behaviours of M satisfy ϕ ?

Mutual Exclusion

Initially $y := 1$



- Asynchronous parallelism
- Guarded assignments.

Properties

- Mutual exclusion: System will not reach a state where both processes are in critical region.
- In *each execution*, whenever either process is in critical region the value of $y = 0$.
- Starvation-freedom: If process 1 is trying to enter the critical region, it will eventually succeed.

Logical notations such as Temporal logics CTL and LTL are used to specify these properties rigorously.

Properties

- Mutual exclusion: System will not reach a state where both processes are in critical region.

  $$\mathbf{AG}\ \neg(pc1 = C \wedge pc2 = C)$$

- In *each execution*, whenever either process is in critical region the value of $y = 0$.

  $$\mathbf{AG}\ ((pc1 = C \vee pc2 = C)\ \Rightarrow\ y = 0)$$

- Starvation-freedom: If process 1 is trying to enter the critical region, it will eventually succeed.

  $$\mathbf{AG}\ (pc1 = T\ \Rightarrow\ (\mathbf{F}\ pc1 = C))$$

Logical notations such as Temporal logics CTL and LTL are used to specify these properties rigorously.

Algorithmically verifying properties

Invariance Properties

$AG\ P$ states that all states in all executions satisfy predicate P.

How to verify?

- Construct Global state graph.
- Check that all reachable states satisfy P.

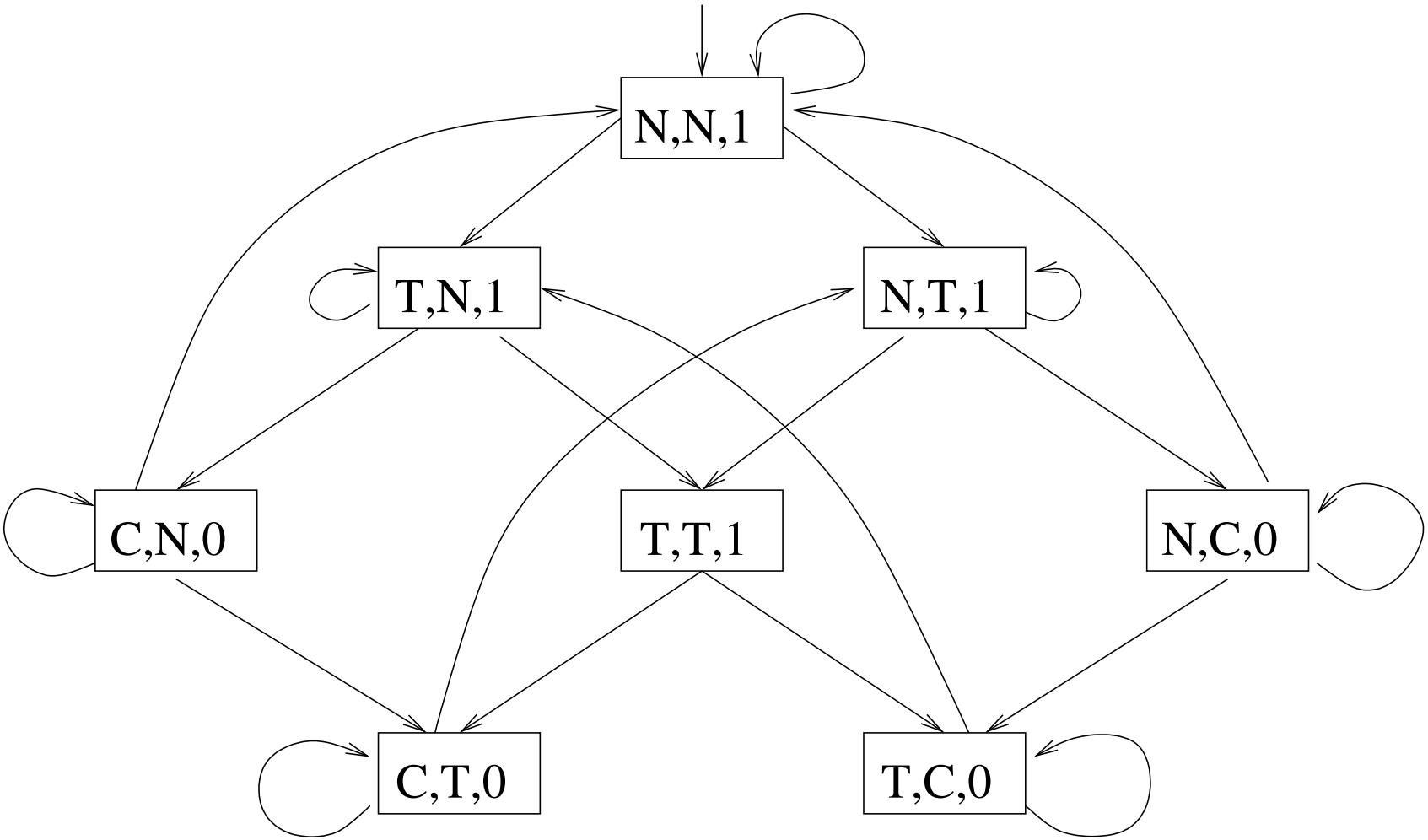
Liveness Properties

System eventually reaches a desired good state from every trigger in any infinite execution. Stated in Temporal logic

How to verify?

- Reduce temporal property to "recurrence" of some good states in infinite executions.
- Analyze cycles in the global state graph for desired recurrence.

Global Automaton for Mutual Exclusion Problem



Modelling Complex Interacting Automata

- UML State Charts: Finite State Automata with Hiearchy, Concurrency, Synchronization
- Synchronous Programming languages: Esterel, SCADE/Lustre.
- Verilog, VHDL for RTL designs.
- The SMV language.

Verifying large finite state systems

State space explosion

Global state graph grows expoentially with increase in number of concurrent components.

Techniques:

- Efficient representation and exploration of state graph. (Tool SPIN – 10^9 states). ACM Software award 2001.
- Symbolic Verification. Represent large graph symbolically with formulas of predicate logic. Explore graph by mathematical reasoning on logical formulas. Can some times analyse very large finite state systems. (Tool NuSMV – 10^{120} states.)
- Bounded verification: Explores evolution of system upto bounded depth using predicate logic. Generalizes the results using induction. (Tool NuSMV).

Beyond Model checking of Finite State Systems

- Infinite state systems: Data, Recursion, Dynamic networks. Symbolic and Bounded Verification techniques are applicable.
- Real-time systems: Reason about latencies of events in systems.
  - Timed automata (Alur,Dill) and reachabilty.
  - Symbolic zone based reachability analysis using simple linear constraint solving. (Tool UPPAAL).
  - Timed logics *MTL*, *TCTL* and their model checking.
- Probabilistic systems: Reason about probability of occurrence of good conditions in probabilistic programs.
  - Probabilistic choice: An "if" statement where branches are chosen according to given probabilities.
  - Model failure in components, as well as distribution of inputs.
  - Logic PCTL allows specification of probabilities of temporal properties.
  - Symbolic Reachability Analysis. (Tool PRISM)

These advanced topics will be covered based on the interest and the available time.

Leading academic Model Checking Tools

Finite state systems, Systems with Data

SPIN (Bell Labs), SMV *(CMU)*, VIS *(UC Berkeley, UC Colorado)*, NuSMV *(IRST, Trento)*, UCLID *(CMU)*, SAL *(SRI,Stanford)*.

Real-time Systems

UPPAAL *(Universities of Uppsala and Aalborg)*, KRONOS *(University of Grenoble)*.

Probabilistic Systems

PRISM *(University of Oxford)*, MRMC *(University of Aachen)*.

Industrial Use

IEEE standard temporal logic PSL/Sugar. Supported by leading EDT.
Hardware and Computer Architecture Companies have inhouse model checking tools. Microsoft (Z3), Intel, IBM (RuleBase), Motorola (ACL2), Siemens, Cadence, Synopsis.

- Intel announced formal verification of floating point unit of Pentium Pro processor (1999). Used formal verification to check parts of design of Pentium IV processor.
- Siemens used Equivalence Checking in validation of ASICs with upto a Million Gates.
- Microsoft provides driver development kit where every third party driver is model checked for health.
- Intel, Motorola, IBM routinely use Model Checking in their design process.

Some model checkers from India (incomplete list)

- DCVALID *(TIFR)* (Released since 1997)
- Word Level ABC verifier *(IITB)*
- Bebop and SLAM *Microsoft Research India*
- DCSYNTH controller synthesis tool *(TIFR)* (Released 2019)
- TChecker Timed Systems Model checker *(CMI, Univ Bordeaux)*

CS738: Course Details

Objectives

This course is a practical introduction to model checking tools and the science behind them. It will provide a hands-on introduction to modelling and verifying the correctness of reactive systems (digital circuits, embedded controllers, communication protocols) using some leading model checkers.

The course will explore the concepts and algorithms underlying these model checking tools.

Depending on the interst and the available time, the course will also explore extensions of model checking techniques to the verification of real-time systems as well as probabilistic systems.

CS783: Course Details (cont)

Course Web Page
`http://www.tcs.tifr.res.in/~pandya/grad/CS738.html`
All lecture slides and links to video lectures can be found there.

Syllabus, Textbooks and Resources These can be found on the course web site.

Assignments The course will involve assignments. These include manual assignments as well as modelling and verification exercises using NuSMV and UPPAAL. These are actually the fun part of the course (where the computer does the thinking for you!).