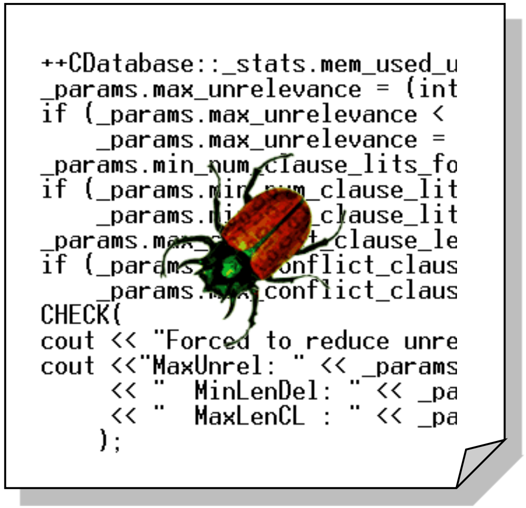
Model Checking and the Curse of Dimensionality

Edmund M. Clarke

School of Computer Science

Carnegie Mellon University



Turing's Quote on Program Verification

- "How can one check a routine in the sense of making sure that it is right?"

- "The programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole program easily follows."

Quote by A. M. Turing on 24 June 1949 at the inaugural conference of the EDSAC computer at the Mathematical Laboratory, Cambridge.

Temporal Logic Model Checking

- Model checking is an **automatic verification technique** for finite state concurrent systems.
- Developed independently **by Clarke and Emerson** and by **Queille and Sifakis** in early 1980's.
- The **assertions** written as **formulas** in **propositional temporal logic**. (Pnueli 77)
- Verification procedure is **algorithmic rather than deductive** in nature.

Main Disadvantage

Curse of Dimensionality:

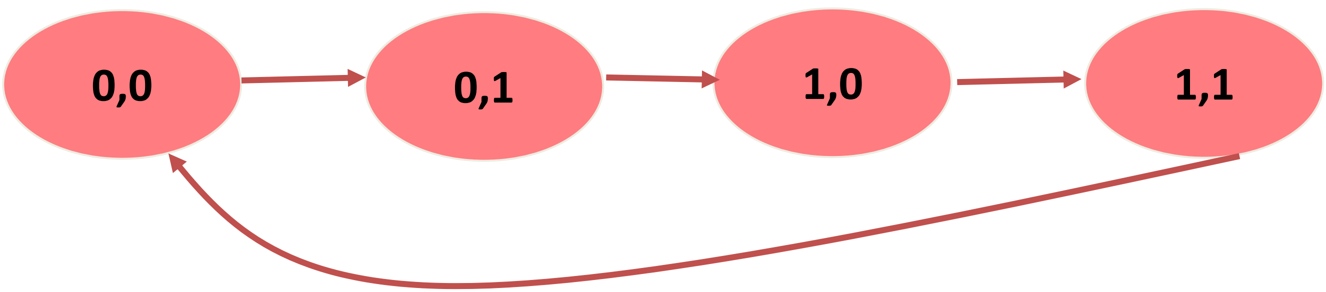
"In view of all that we have said in the foregoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is **the curse of dimensionality**, a malediction that has plagued the scientist from the earliest days."

Richard E. Bellman.
Adaptive Control Processes: A Guided Tour.
Princeton University Press, 1961.

Image courtesy Time Inc.
Photographer Alfred Eisenstaedt.

Main Disadvantage (Cont.)

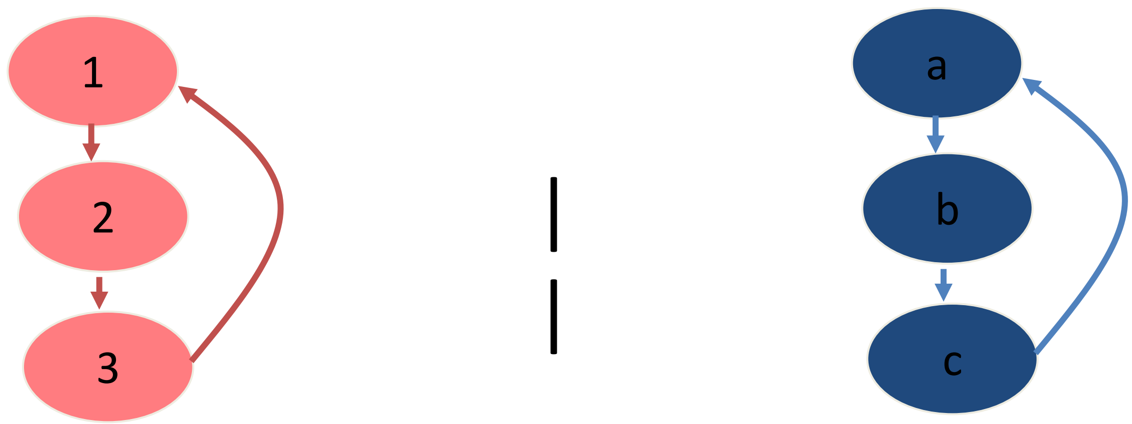
Curse of Dimensionality:



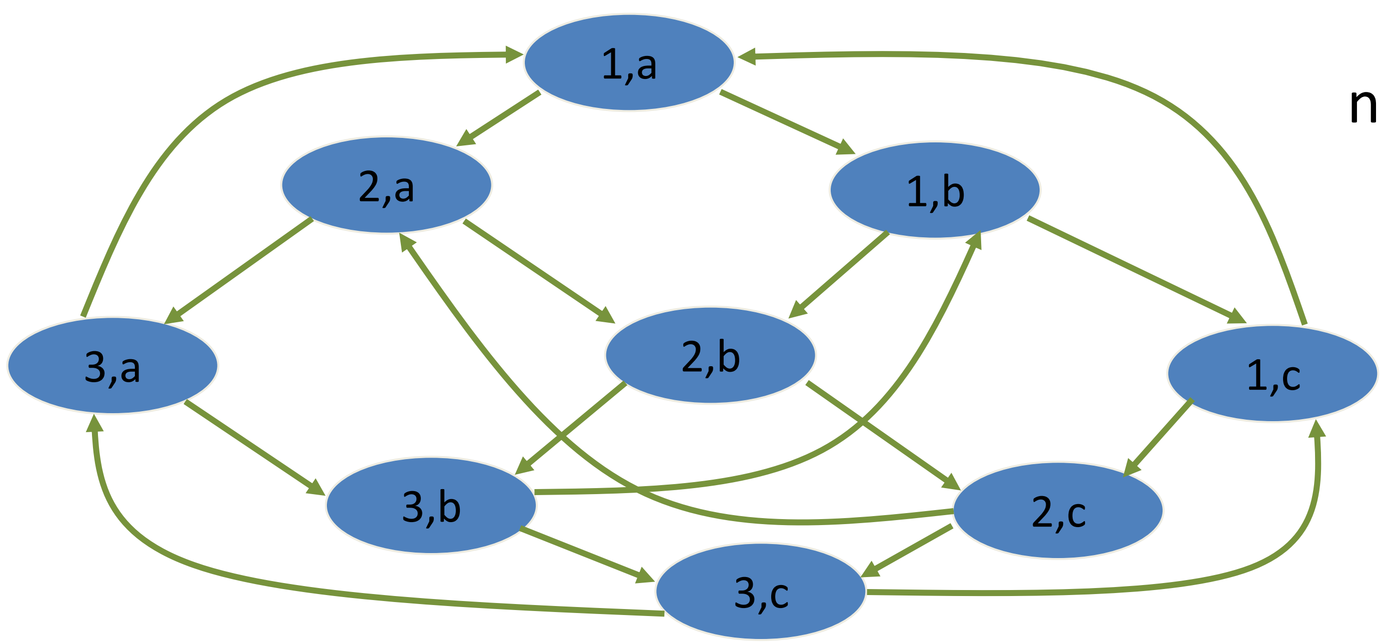
2-bit counter

n-bit counter has 2^n states

Main Disadvantage (Cont.)



n states,
m processes



n^m states

Main Disadvantage (Cont.)

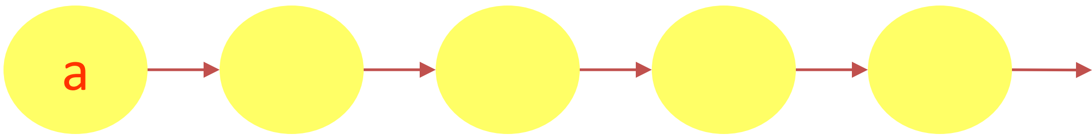
Curse of Dimensionality:

The number of states in a system grows **exponentially with its dimensionality** (i.e. number of variables or bits or processes). This makes the system harder to reason about.

Unavoidable in worst case, but steady progress over the past 30 years using clever algorithms, data structures, and engineering

LTL - Linear Time Logic (Pn 77)

Determines Patterns on Infinite Traces



Atomic Propositions

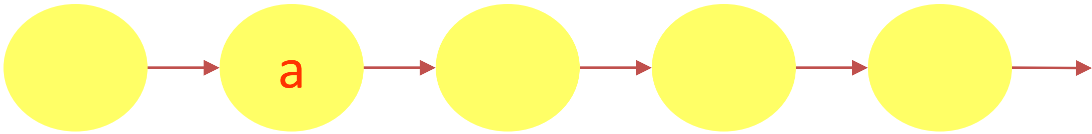
Boolean Operations

Temporal operators

a	"a is true now"
X a	"a is true in the neXt state"
F a	"a will be true in the Future"
G a	"a will be Globally true in the future"
a U b	"a will hold true Until b becomes true"

LTL - Linear Time Logic (Pn 77)

Determines Patterns on Infinite Traces



Atomic Propositions

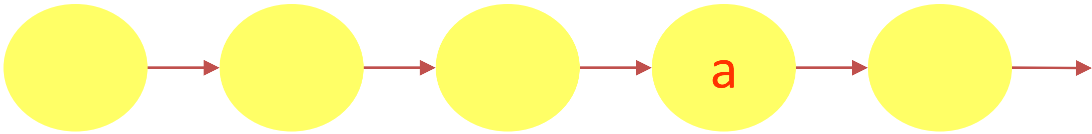
Boolean Operations

Temporal operators

a	"a is true now"
X a	"a is true in the neXt state"
F a	"a will be true in the Future"
G a	"a will be Globally true in the future"
a U b	"a will hold true Until b becomes true"

LTL - Linear Time Logic (Pn 77)

Determines Patterns on Infinite Traces



Atomic Propositions

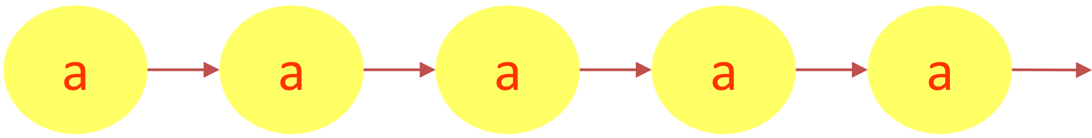
Boolean Operations

Temporal operators

a	"a is true now"
X a	"a is true in the neXt state"
F a	**"a will be true in the Future"**
G a	"a will be Globally true in the future"
a U b	"a will hold true Until b becomes true"

LTL - Linear Time Logic (Pn 77)

Determines Patterns on Infinite Traces



Atomic Propositions

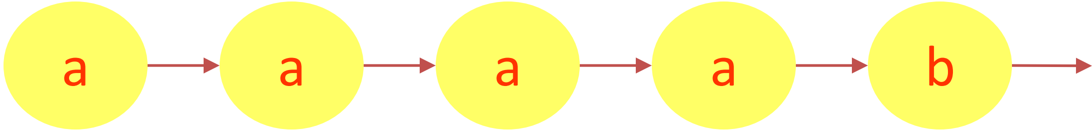
Boolean Operations

Temporal operators

a	"a is true now"
X a	"a is true in the neXt state"
F a	"a will be true in the Future"
G a	**"a will be Globally true in the future"**
a U b	"a will hold true Until b becomes true"

LTL - Linear Time Logic (Pn 77)

Determines Patterns on Infinite Traces



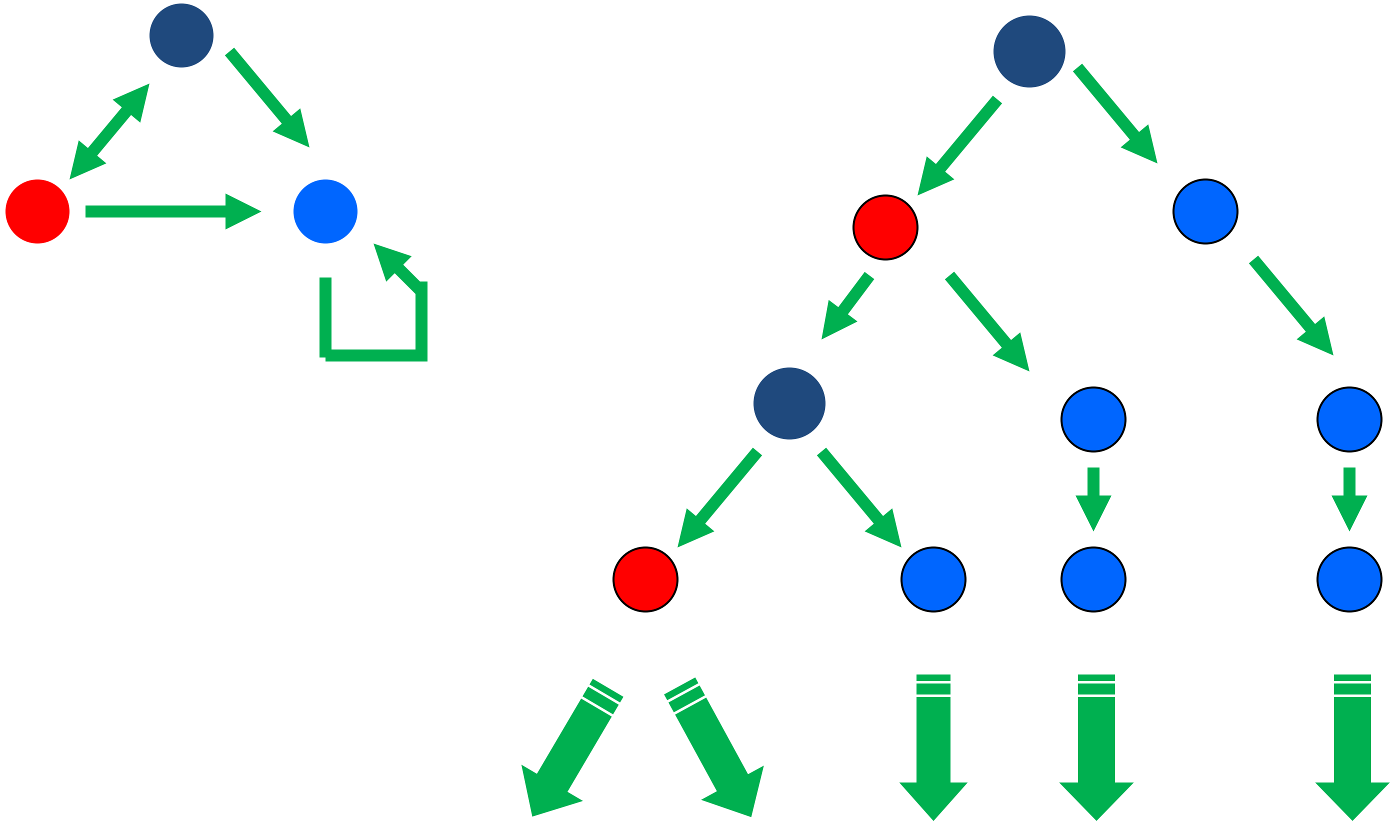
Atomic Propositions

Boolean Operations

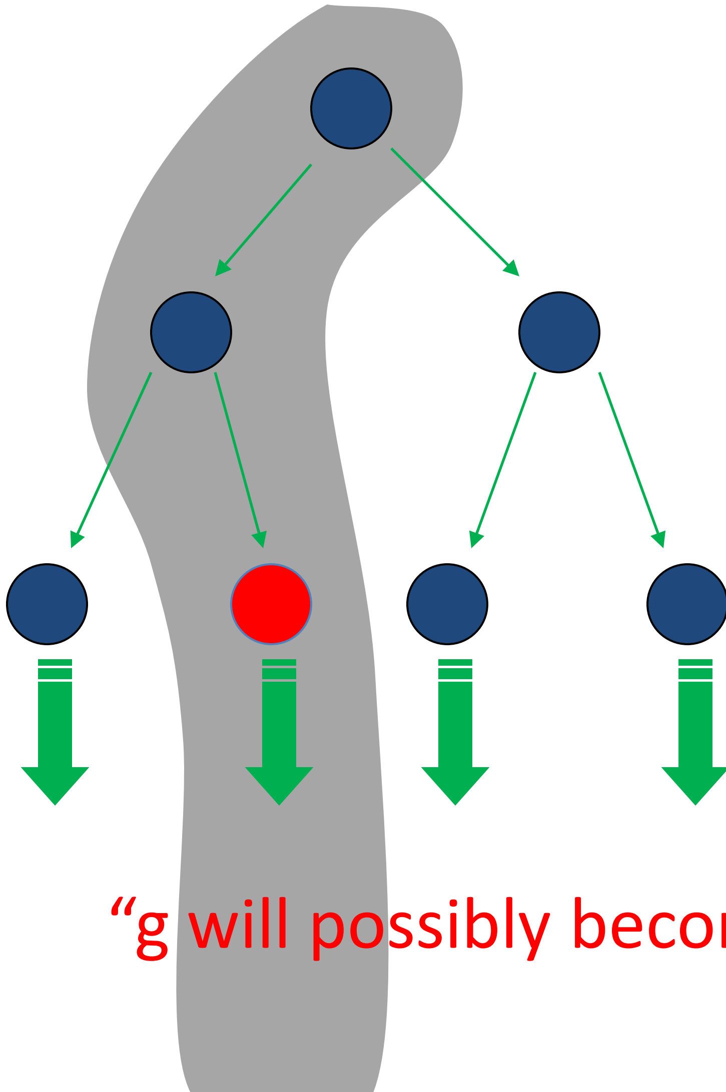
Temporal operators

a	"a is true now"
X a	"a is true in the neXt state"
F a	"a will be true in the Future"
G a	"a will be Globally true in the future"
a U b	"a will hold true Until b becomes true"

Branching Time (EC 80, BMP 81)

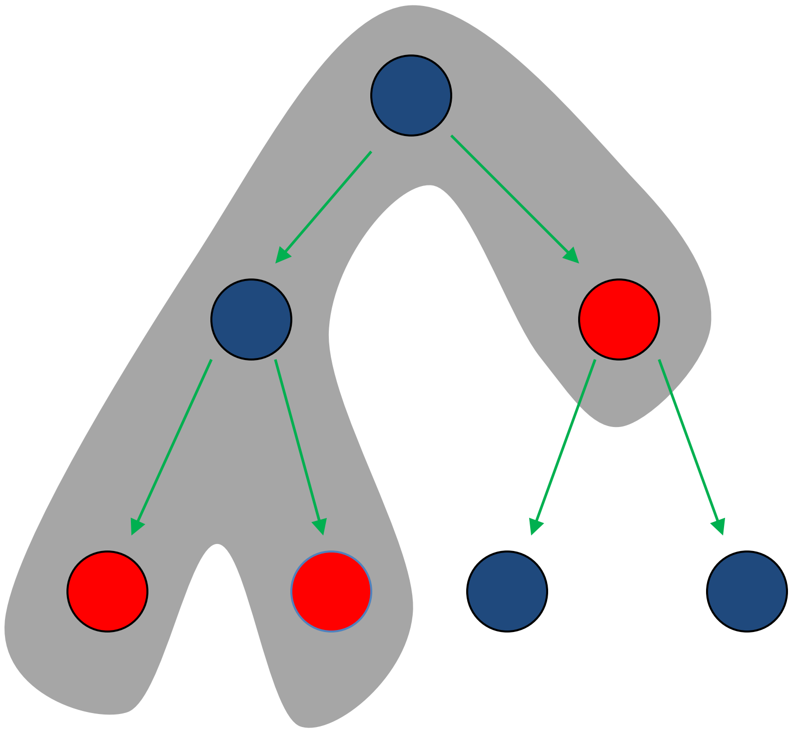


CTL: Computation Tree Logic



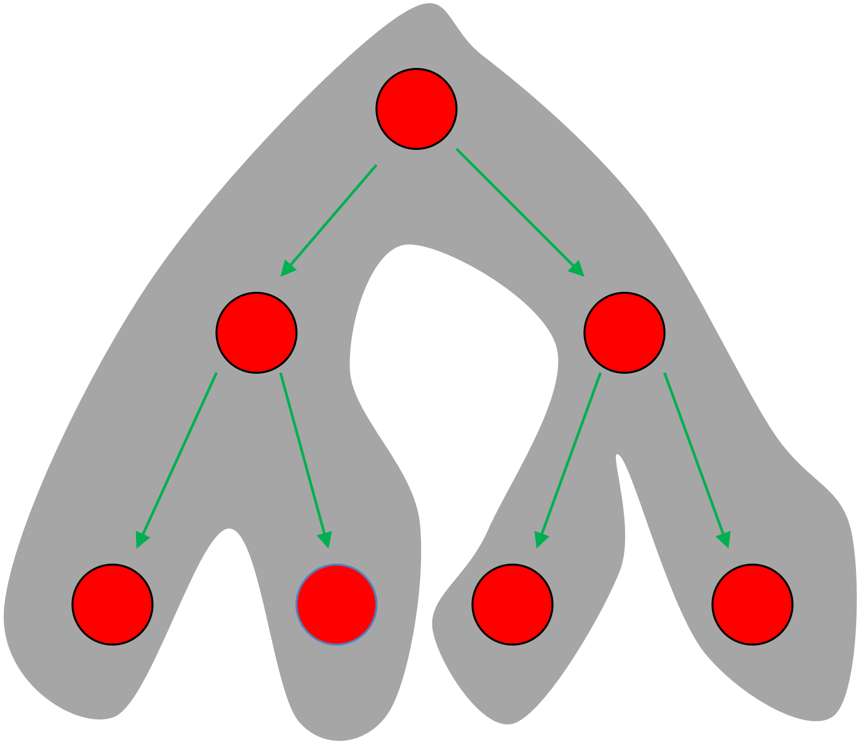
EF g "g will possibly become true"

CTL: Computation Tree Logic



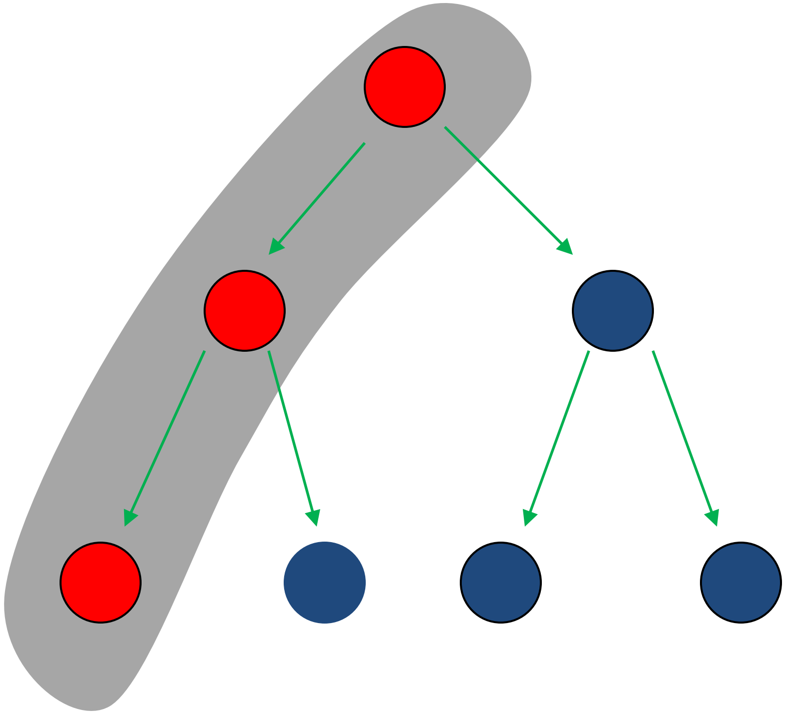
AF g "g will necessarily become true"

CTL: Computation Tree Logic



AG g "g is an invariant"

CTL: Computation Tree Logic



EG g "g is a potential invariant"

CTL: Computation Tree Logic

CTL (CES 83-86) uses the temporal operators

AX, AG, AF, AU
EX, EG, EF, EU

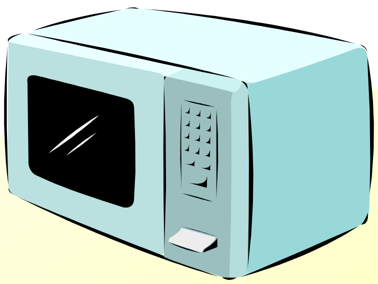
CTL* allows complex nestings such as
AXX, AGX, EXF, ...

Model Checking Problem

- Let *M* be a state-transition graph.
- Let *f* be an assertion or specification in temporal logic.
- Find all states *s* of *M* such that *M, s satisfies f*.

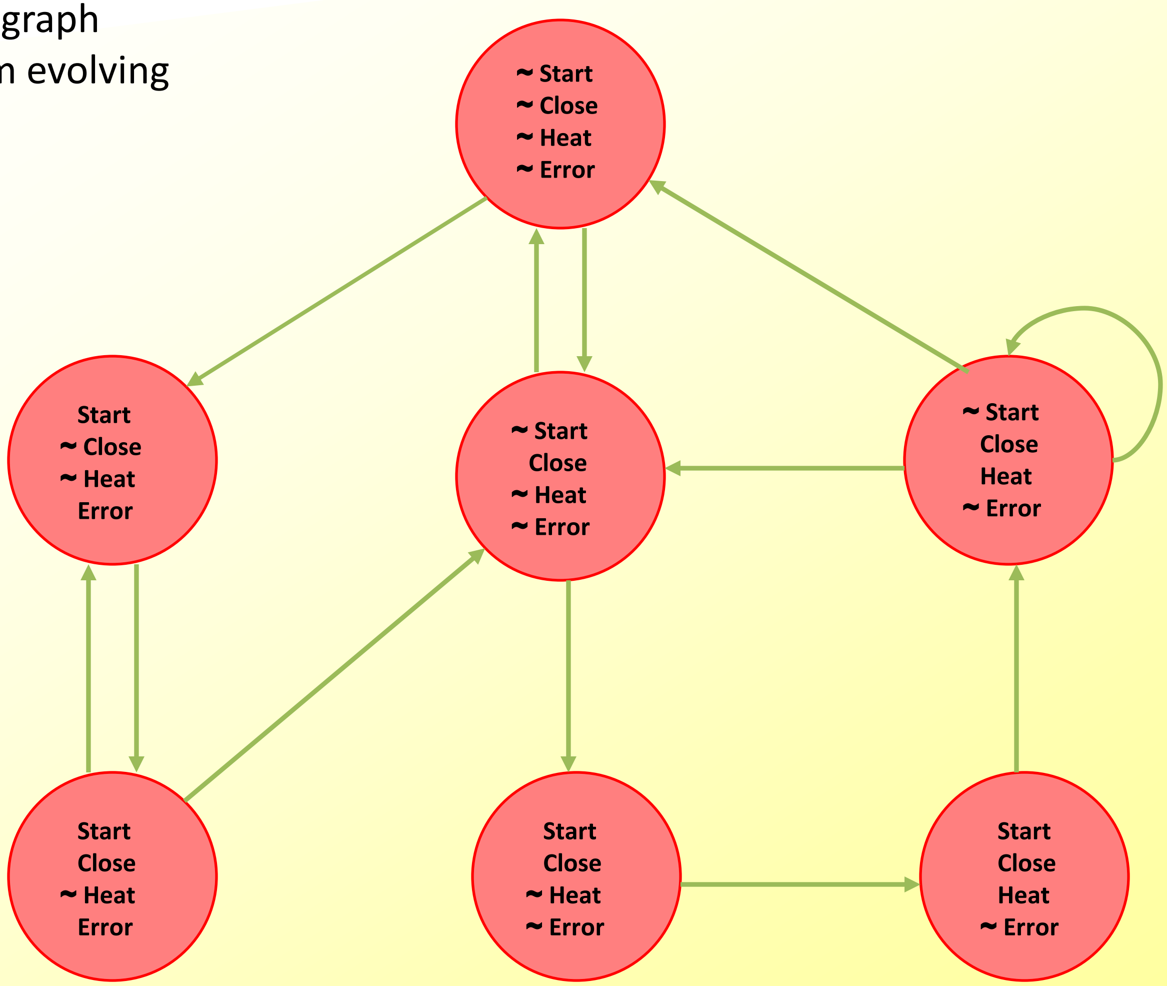
- CTL Model Checking: CE 81; CES 83/86; QS 81/82.
- LTL Model Checking: LP 85.
- Automata Theoretic LTL Model Checking: VW 86.
- CTL* Model Checking: EL 85.

Trivial Example



Microwave Oven

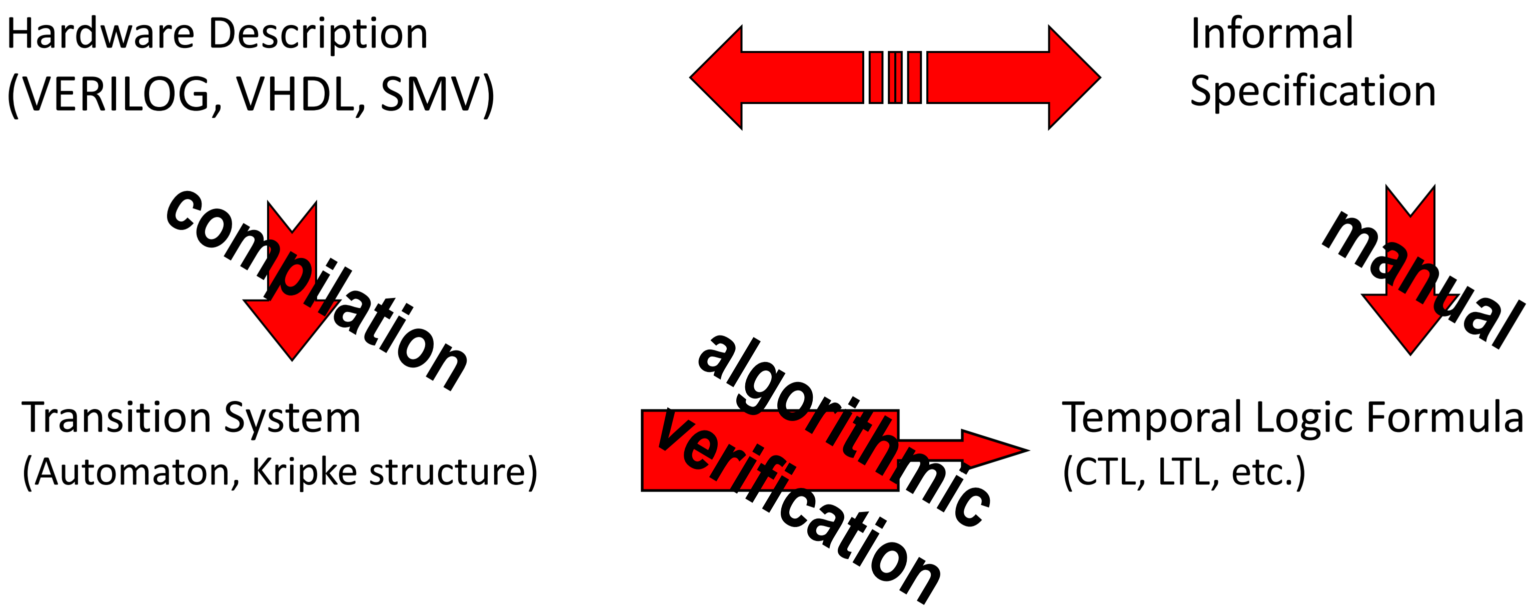
State-transition graph describes system evolving over time.



Temporal Logic and Model Checking

- The oven doesn't heat up until the door is closed.
- Not heat_up holds until door_closed
- (~ heat_up) U door_closed

Model Checking



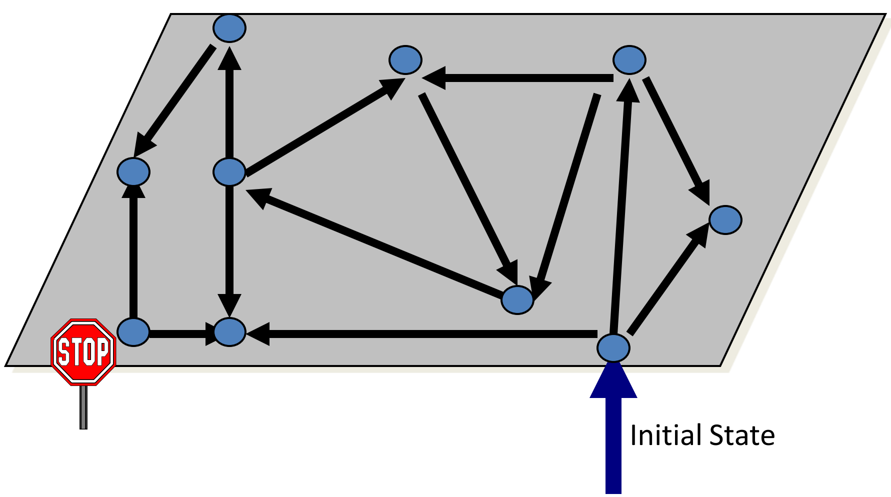
Counterexamples

Program or circuit

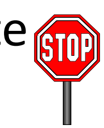
Informal Specification

Transition System

Temporal Logic Formula
(CTL, LTL, etc.)



Safety Property:

bad state  unreachable:

satisfied

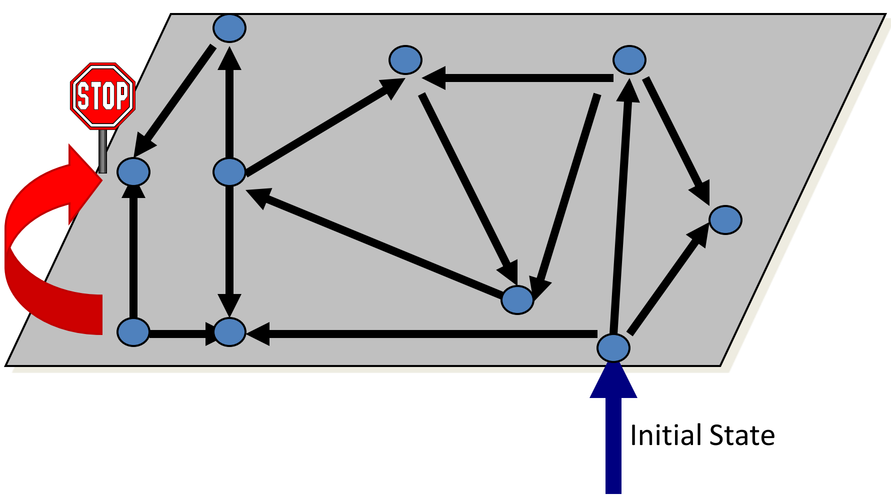
Counterexamples

Program or circuit

Informal Specification

Transition System

Temporal Logic Formula
(CTL, LTL, etc.)



Safety Property:
bad state  unreachable

Counterexample

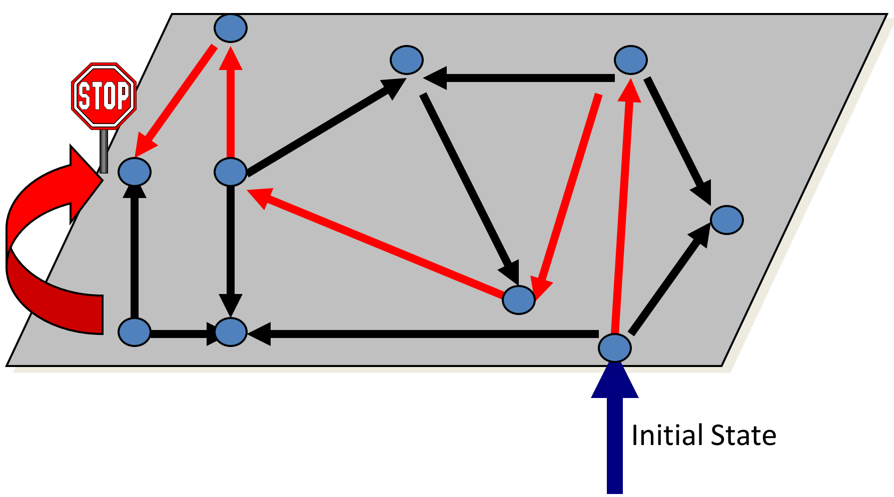
Counterexamples

Program or circuit

Informal Specification

Transition System

Temporal Logic Formula
(CTL, LTL, etc.)



Safety Property:
bad state  unreachable

Counterexample

Hardware Example: IEEE Futurebus^{+}

- In 1992 we used Model Checking to verify the IEEE Futurebus+ cache coherence protocol.
- Found a number of previously undetected errors in the design.
- First time that a formal verification tool was used to find errors in an IEEE standard.
- Development of the protocol began in 1988, but previous attempts to validate it were informal.

Four Big Breakthroughs in Model Checking!

- **Symbolic Model Checking**

  Burch, Clarke, McMillan, Dill, and Hwang 90;

  Ken McMillan's thesis 92



- **The Partial Order Reduction**

  Valmari 90

  Godefroid 90

  Peled 94

  Gerard Holzmann's SPIN



Four Big Breakthroughs in Model Checking!

- **Symbolic Model Checking**

  Burch, Clarke, McMillan, Dill, and Hwang 90;

  Ken McMillan's thesis 92

10^{20} states

- **The Partial Order Reduction**

  Valmari 90

  Godefroid 90

  Peled 94

  Gerard Holzmann's SPIN

Four Big Breakthroughs in Model Checking!

- **Symbolic Model Checking**

  Burch, Clarke, McMillan, Dill, and Hwang 90;

  Ken McMillan's thesis 92



10^{100} states

- **The Partial Order Reduction**

  Valmari 90

  Godefroid 90

  Peled 94

  Gerard Holzmann's SPIN



Four Big Breakthroughs in Model Checking!

- **Symbolic Model Checking**
  Burch, Clarke, McMillan, Dill, and Hwang 90;
  Ken McMillan's thesis 92



10^{120} states

- **The Partial Order Reduction**
  Valmari 90
  Godefroid 90
  Peled 94
  Gerard Holzmann's SPIN



Four Big Breakthroughs in Model Checking (Cont.)

- **Bounded Model Checking**
  - Biere, Cimatti, Clarke, Zhu 99
  - Using Fast SAT solvers
  - Can handle thousands of state elements

Can the given property fail in k-steps?

$$I(V_0) \wedge T(V_0,V_1) \wedge \ldots \wedge T(V_{k-1},V_k) \wedge (\neg P(V_0) \vee \ldots \vee \neg P(V_k))$$

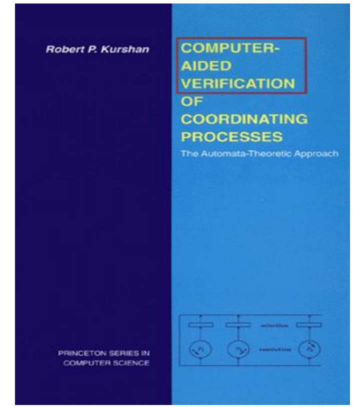
Initial state — **k-steps** — **Property fails in some step**

BMC in practice: Circuit with 9510 latches, 9499 inputs

BMC formula has 4×10^6 variables, 1.2×10^7 clauses

Shortest bug of length 37 found in 69 seconds

Four Big Breakthroughs in Model Checking (Cont.)

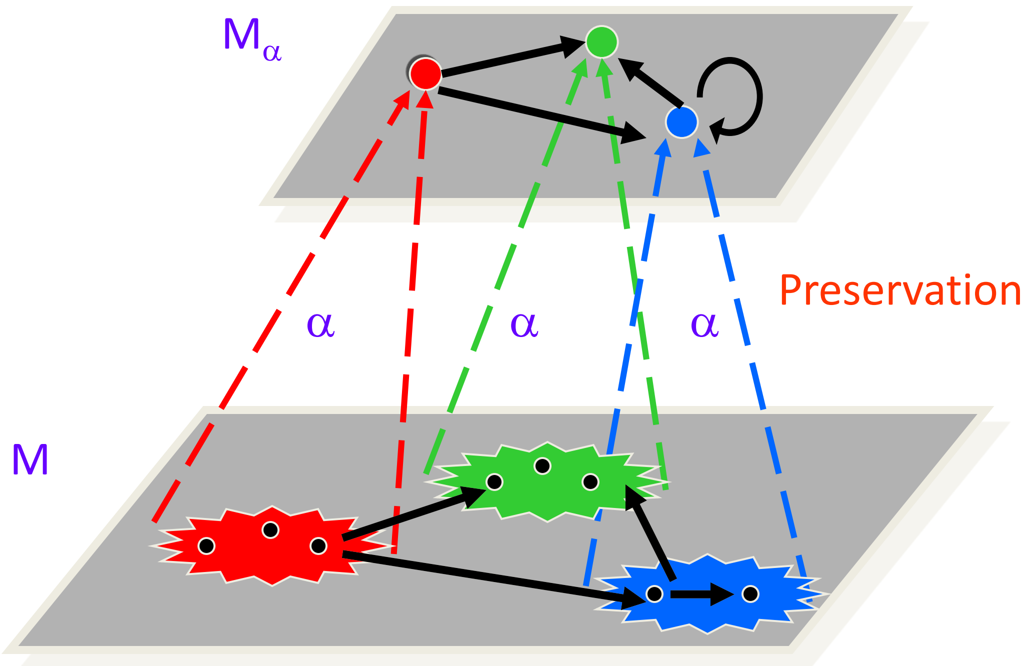


- **Localization Reduction**
  - Bob Kurshan 1994

- **Counterexample Guided Abstraction Refinement (CEGAR)**
  - Clarke, Grumberg, Jha, Lu, Veith 2000
  - Used in most software model checkers

Existential Abstraction

Given an abstraction function $\alpha : S \rightarrow S_\alpha$, the concrete states are grouped and mapped into abstract states:

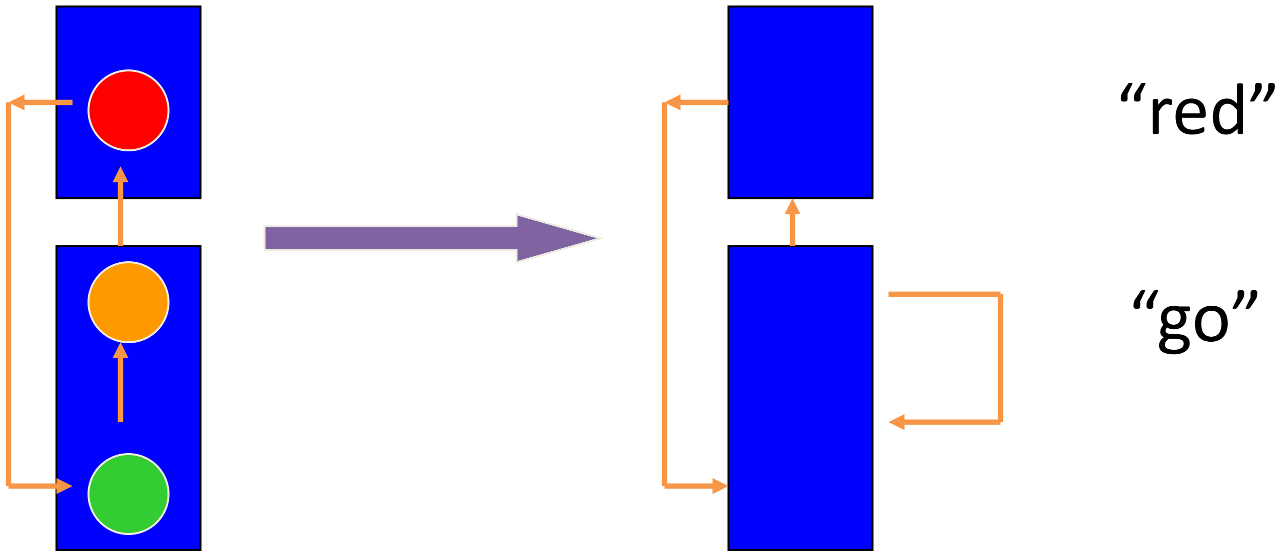


Preservation Theorem ?

Preservation Theorem

- Theorem (Clarke, Grumberg, Long): If property holds on abstract model, it holds on concrete model
- Technical conditions
  - Property is universal i.e., no existential quantifiers
  - Atomic formulas respect abstraction mapping
- Converse implication is not true !

Spurious Behavior



AG AF red

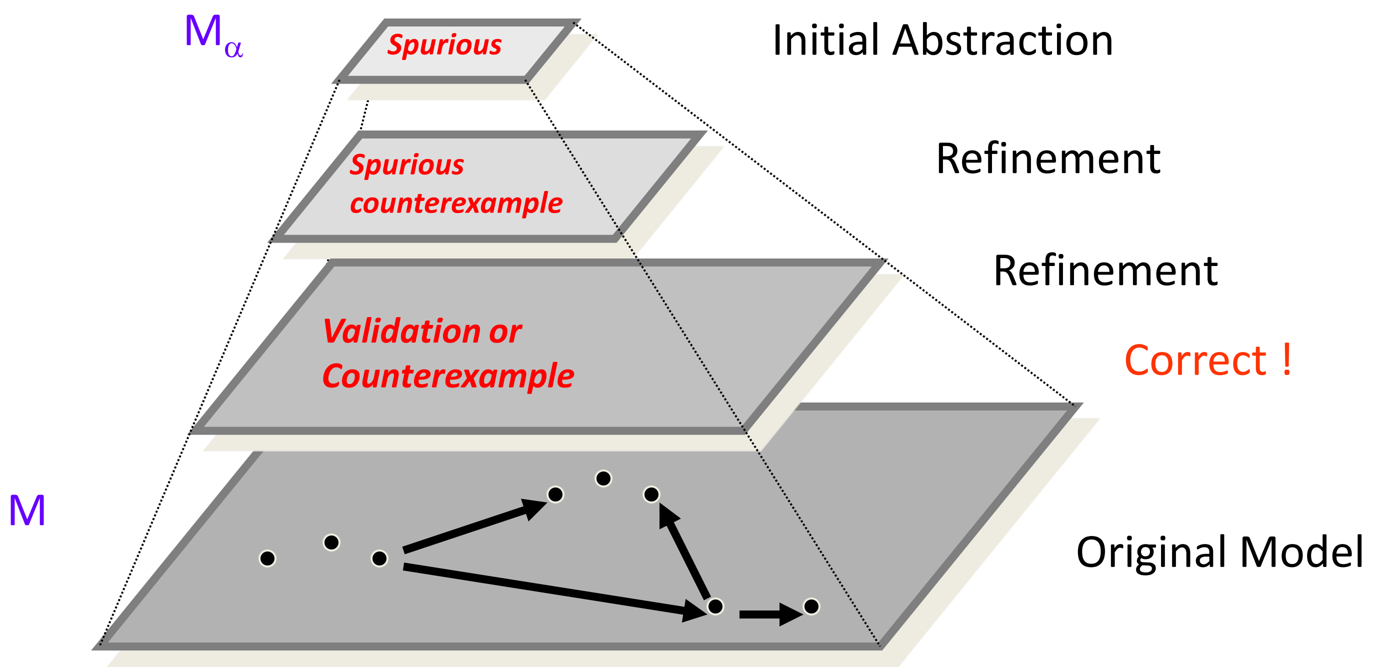
"Every path necessarily leads back to red."

Spurious Counterexample:

<go><go><go><go> ...

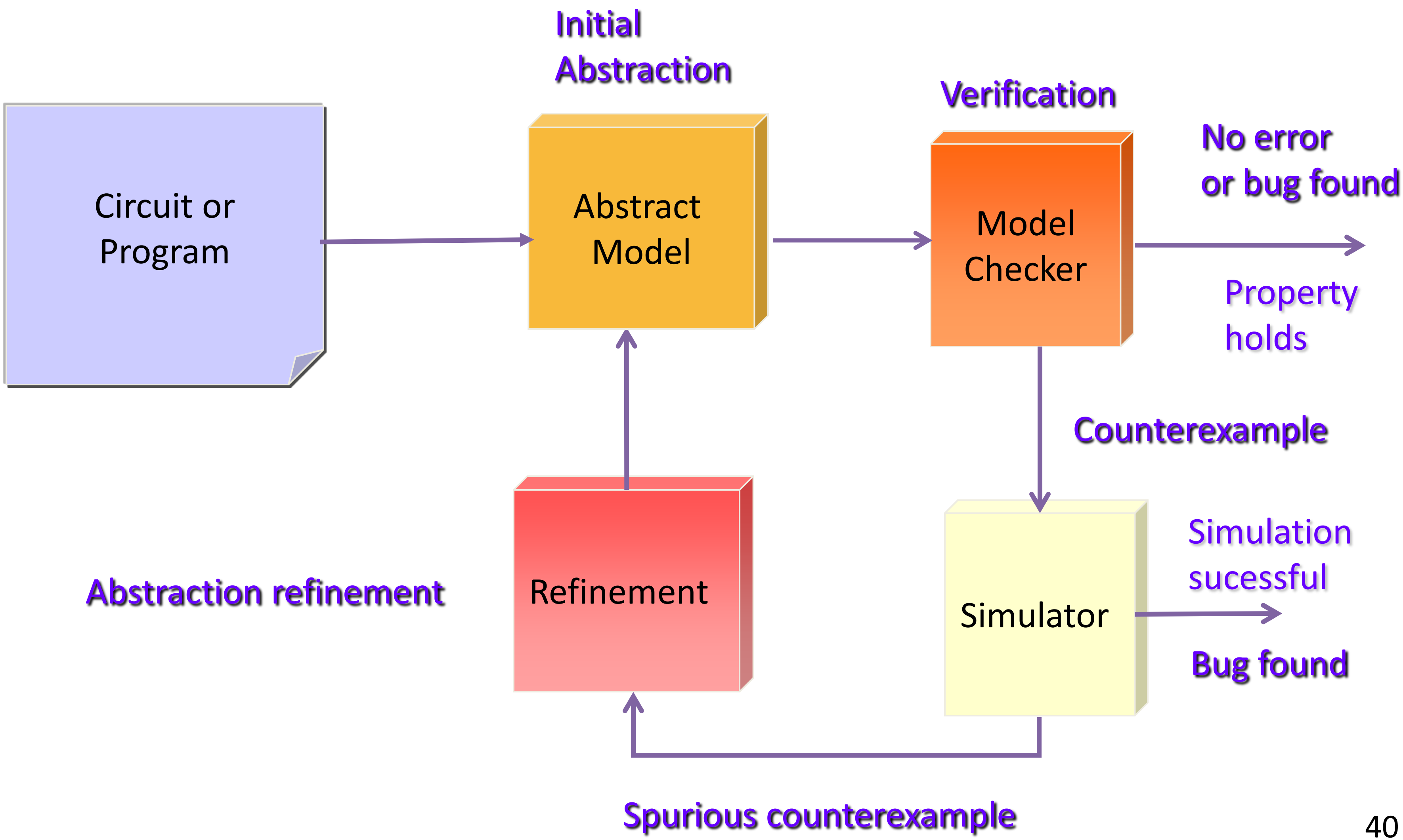
Artifact of the abstraction !

Automatic Abstraction



CEGAR

CounterExample-Guided Abstraction Refinement



Future Challenge

Is it possible to model check software?

According to ***Wired News*** on Nov 10, 2005:

“When Bill Gates announced that the technology was under development at the 2002 Windows Engineering Conference, he called it the Holy Grail of computer science”

What Makes Software Model Checking Different ?

- Large/unbounded base types: int, float, string
- User-defined types/classes
- Pointers/aliasing + unbounded #'s of heap-allocated cells
- Procedure calls/recursion/calls through pointers/dynamic method lookup/overloading
- Concurrency + unbounded #'s of threads

What Makes Software Model Checking Different ?

- Templates/generics/include files
- Interrupts/exceptions/callbacks
- Use of secondary storage: files, databases
- Absent source code for: libraries, system calls, mobile code
- Esoteric features: continuations, self-modifying code
- Size (e.g., MS Word = 1.4 MLOC)

What Does It Mean to Model Check Software?

Combine static analysis and model checking

Use static analysis to extract a model K from an abstraction of the program.

Then check that f is true in K (K |= f),
where f is the specification of the program.

- SLAM (Microsoft)
- Bandera (Kansas State)
- MAGIC, SATABS (CMU)
- BLAST (Berkeley)
- F-Soft (NEC)

Software Example: Device Driver Code

Also according to ***Wired News***:

"Microsoft has developed a tool called Static Device Verifier or SDV, that uses 'Model Checking' to analyze the source code for Windows drivers and see if the code that the programmer wrote matches a mathematical model of what a Windows device driver should do. If the driver doesn't match the model, the SDV warns that the driver might contain a bug."

(Ball and Rajamani, Microsoft)

Future Challenge

Can We Debug This Circuit?

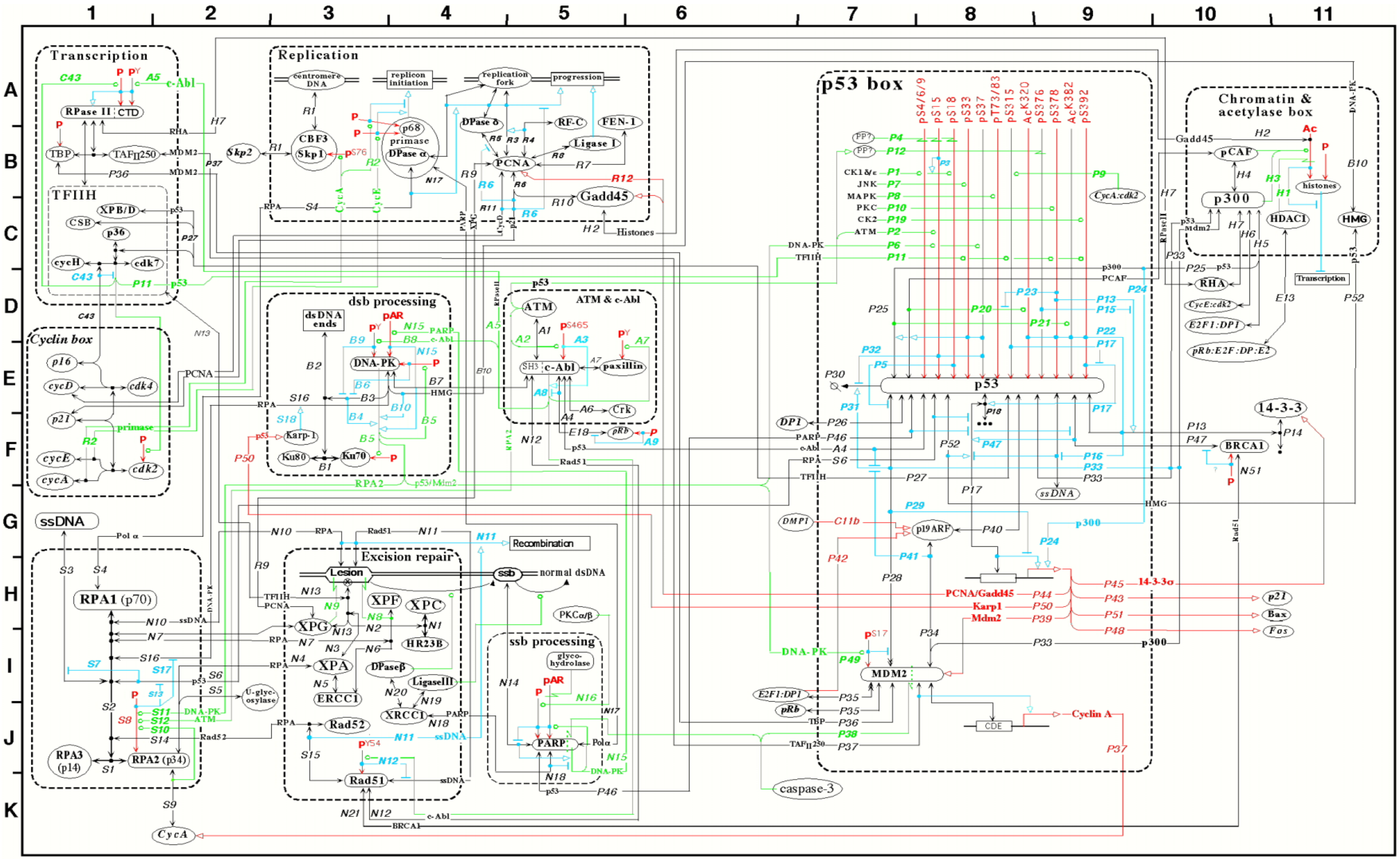


Figure 6B: The p53-Mdm2 and DNA repair regulatory network (version 2p - May 19, 1999)

Kurt W. Kohn, Molecular Biology of the Cell 1999

P53, DNA Repair, and Apoptosis

"The p53 pathway has been shown to mediate cellular stress responses; p53 can initiate DNA repair, cell-cycle arrest, senescence and, importantly, apoptosis. These responses have been implicated in an individual's ability to suppress tumor formation and to respond to many types of cancer therapy."

(A. Vazquez, E. Bond, A. Levine, G. Bond. The genetics of the p53 pathway, apoptosis and cancer therapy. Nat Rev Drug Discovery 2008 Dec;7(12):979-87.)

The protein **p53** has been described as the **guardian of the genome** referring to its role in preventing genome mutation.

In 1993, **p53** was voted ***molecule of the year*** by **Science Magazine**.