Lecture 2:

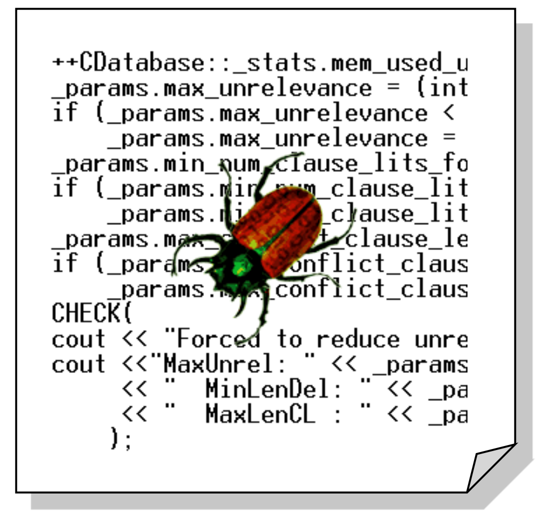
Model Checking

My 30 Year Quest to Conquer the State Explosion Problem

Edmund M. Clarke

School of Computer Science

Carnegie Mellon University



Intel Pentium FDIV Bug

- Try 4195835 – 4195835 / 3145727 * 3145727.
  In 94' Pentium, it doesn't return 0, but 256.
- Intel uses the SRT algorithm for floating point division. Five entries in the lookup table are missing.
- Cost: $400 - $500 million
- Xudong Zhao's Thesis on Word Level Model Checking

Temporal Logic Model Checking

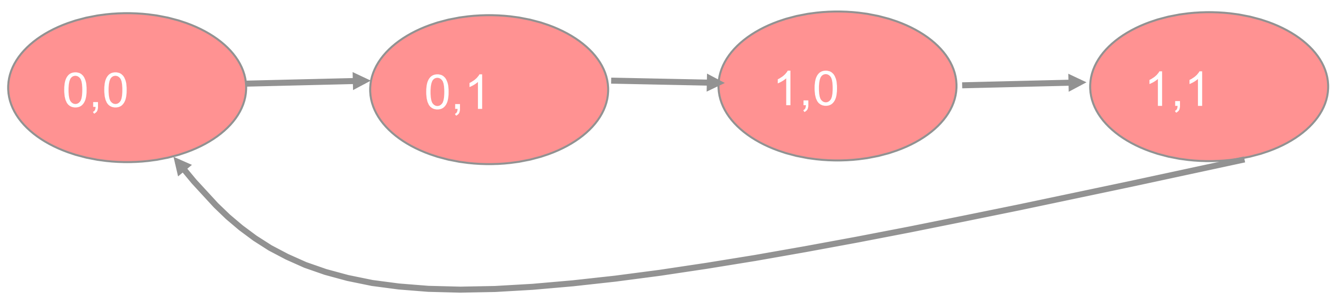
- Model checking is an automatic verification technique for finite state concurrent systems.
- Developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980's.
- Specifications are written in propositional temporal logic. (Pnueli 77)
- Verification procedure is an intelligent exhaustive search of the state space of the design.

Advantages of Model Checking

- No proofs!!! (Algorithmic rather than Deductive)
- Fast (compared to other rigorous methods such as theorem proving)
- Diagnostic counterexamples
- No problem with partial specifications
- Logics can easily express many concurrency properties

Main Disadvantage

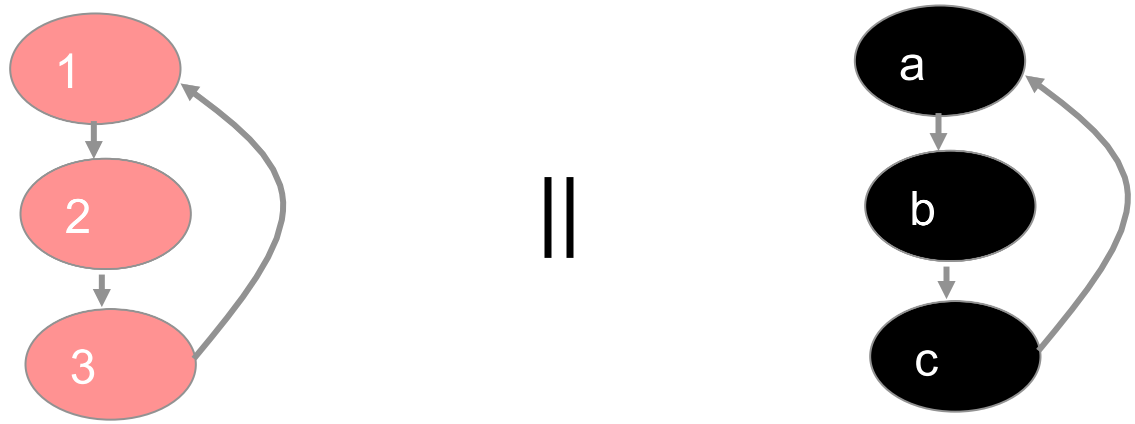
State Explosion Problem:



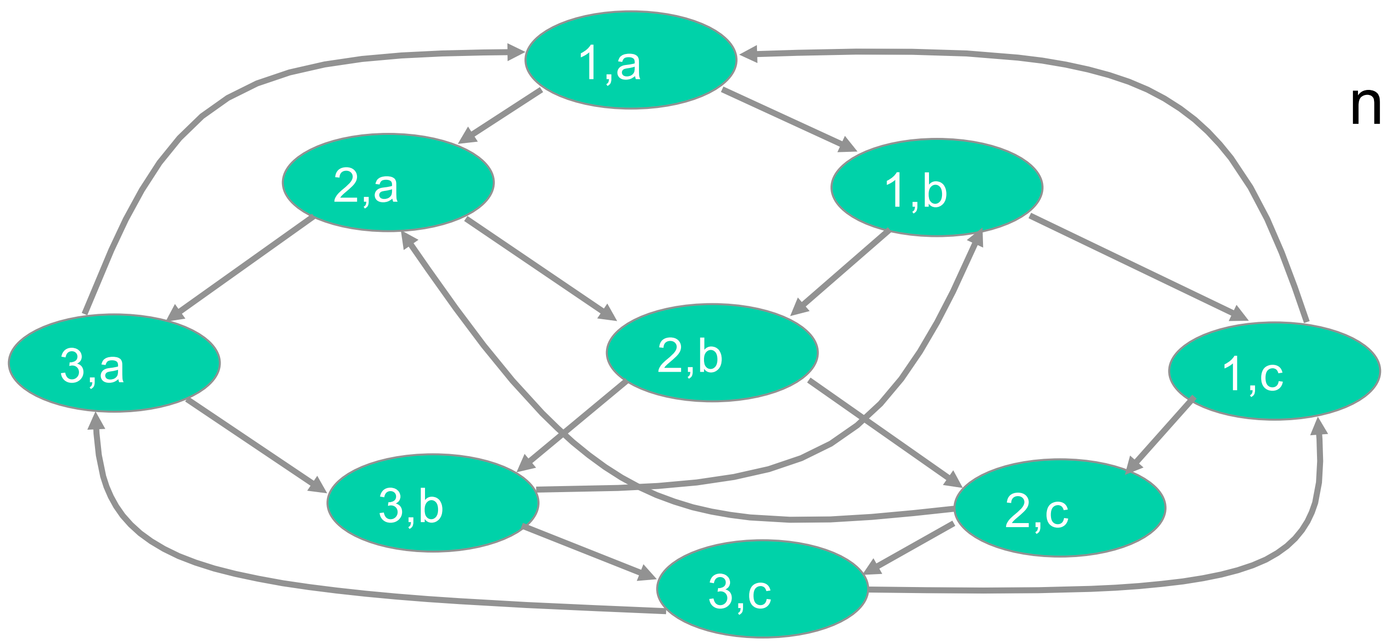
2-bit counter

n-bit counter has 2^n states

Main Disadvantage (Cont.)



n states,
m processes



n^m states

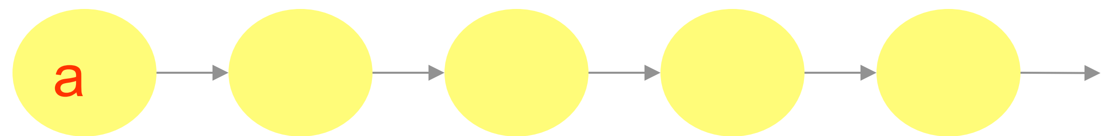
Main Disadvantage (Cont.)

State Explosion Problem:

Unavoidable in worst case, but steady progress over the past 28 years using clever algorithms, data structures, and engineering

LTL - Linear Time Logic (Pn 77)

Determines Patterns on Infinite Traces



Atomic Propositions

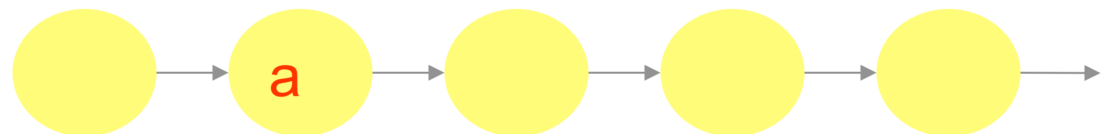
Boolean Operations

Temporal operators

a	“a is true now”
X a	“a is true in the neXt state”
Fa	“a will be true in the Future”
Ga	“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

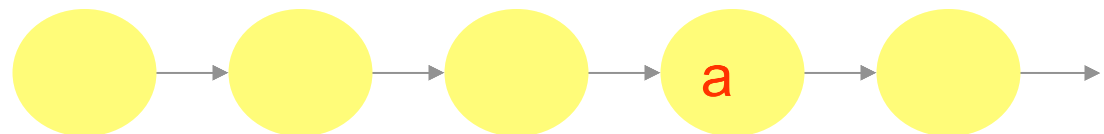
Boolean Operations

Temporal operators

a	“a is true now”
X a	“a is true in the neXt state”
Fa	“a will be true in the Future”
Ga	“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

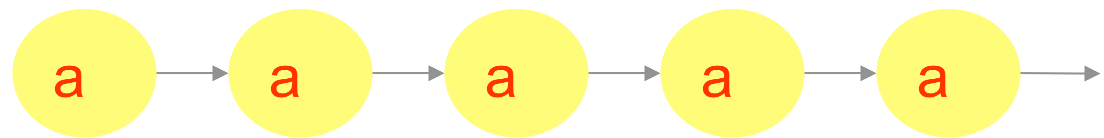
Boolean Operations

Temporal operators

a	“a is true now”
X a	“a is true in the neXt state”
Fa	**“a will be true in the Future”**
Ga	“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

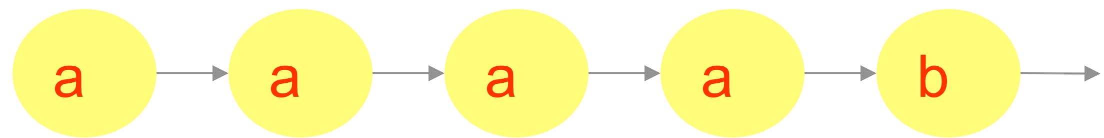
Boolean Operations

Temporal operators

a	"a is true now"
X a	"a is true in the neXt state"
Fa	"a will be true in the Future"
Ga	**"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



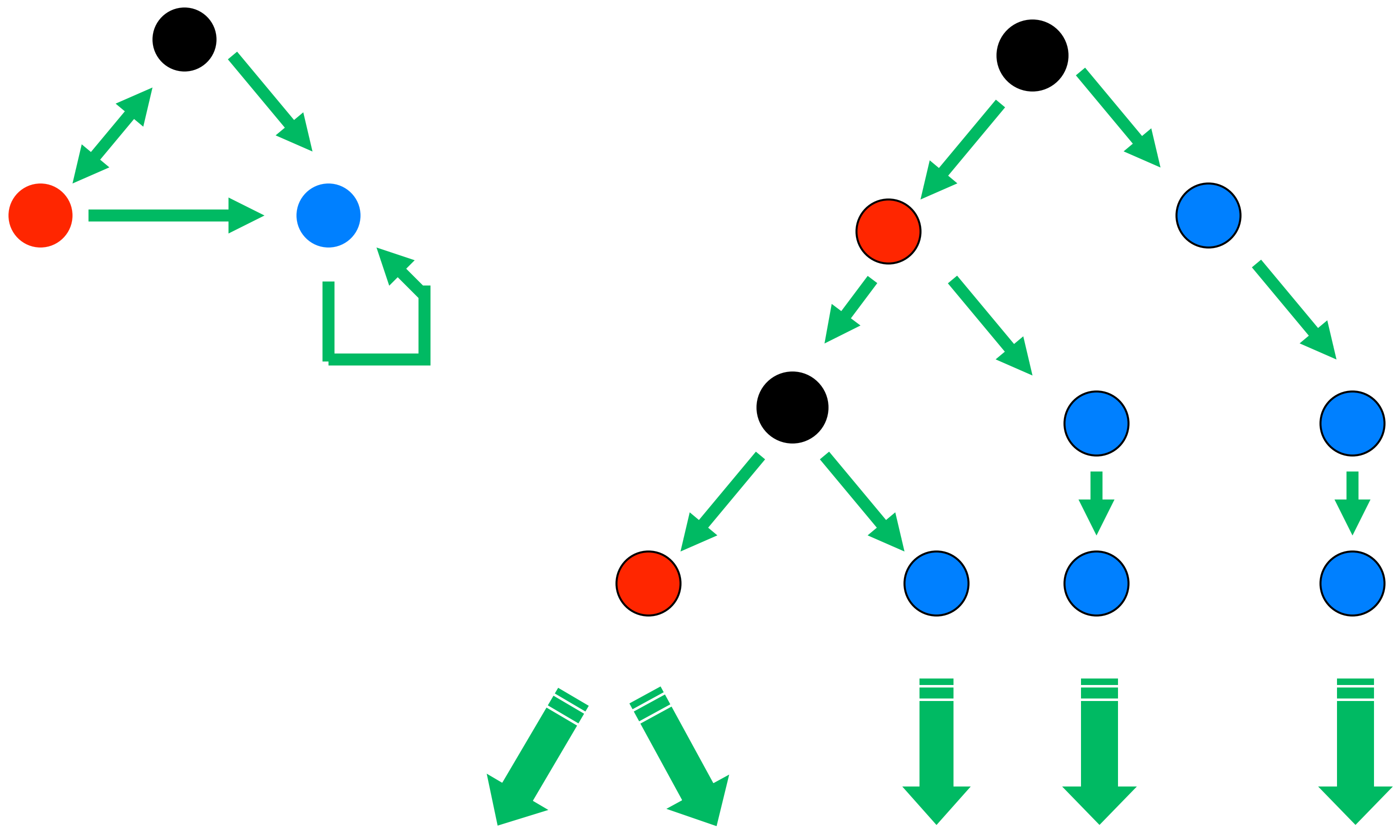
Atomic Propositions

Boolean Operations

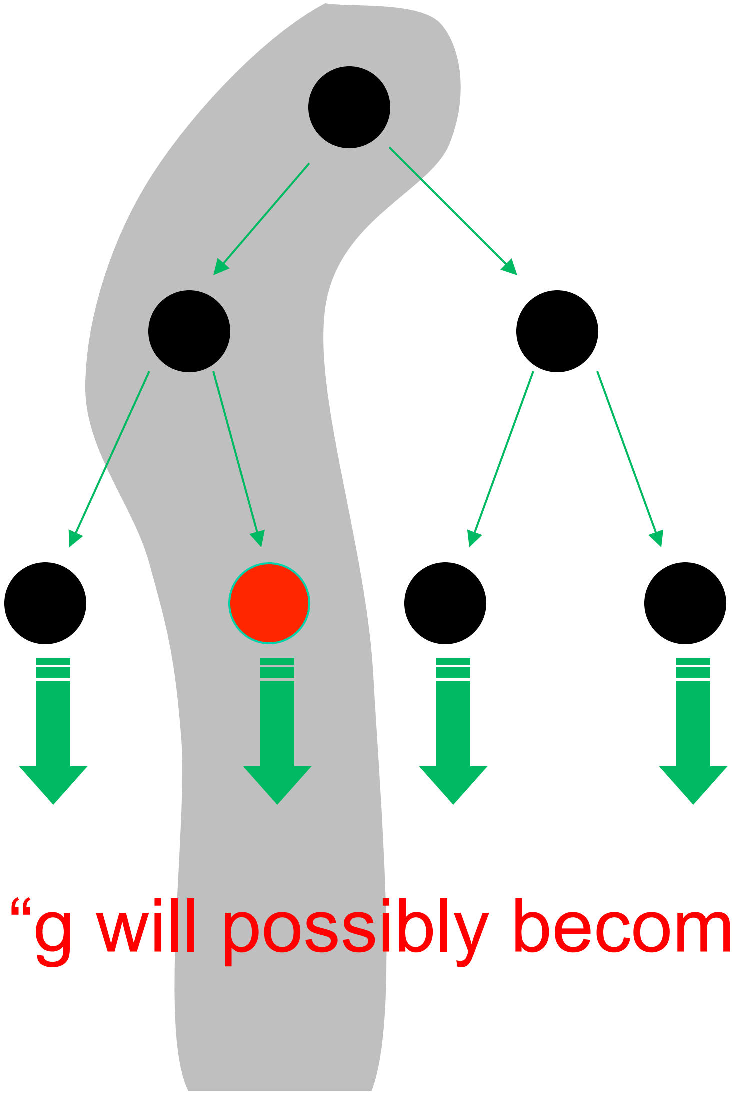
Temporal operators

a	“a is true now”
X a	“a is true in the neXt state”
Fa	“a will be true in the Future”
Ga	“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)

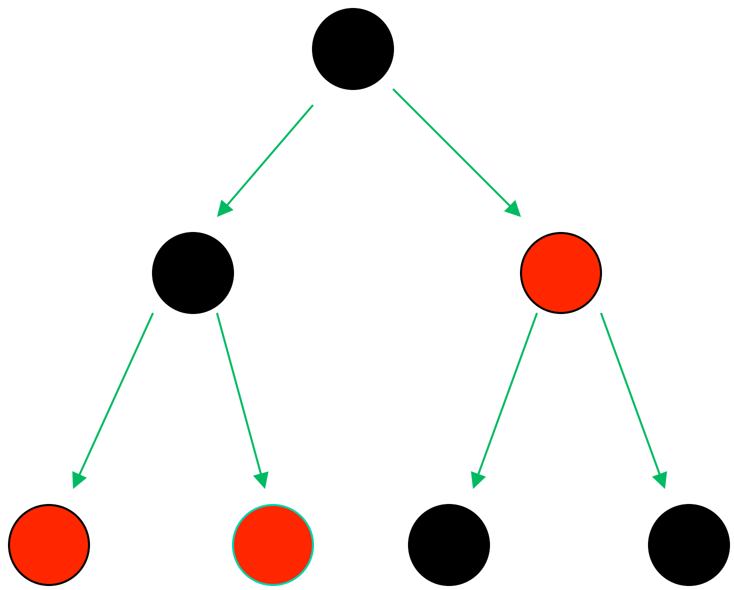


CTL: Computation Tree Logic



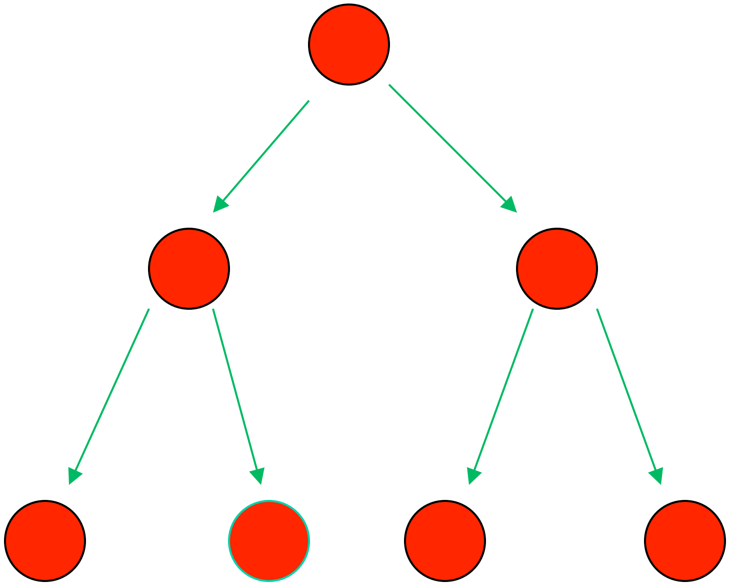
EF g “g will possibly become true”

CTL: Computation Tree Logic



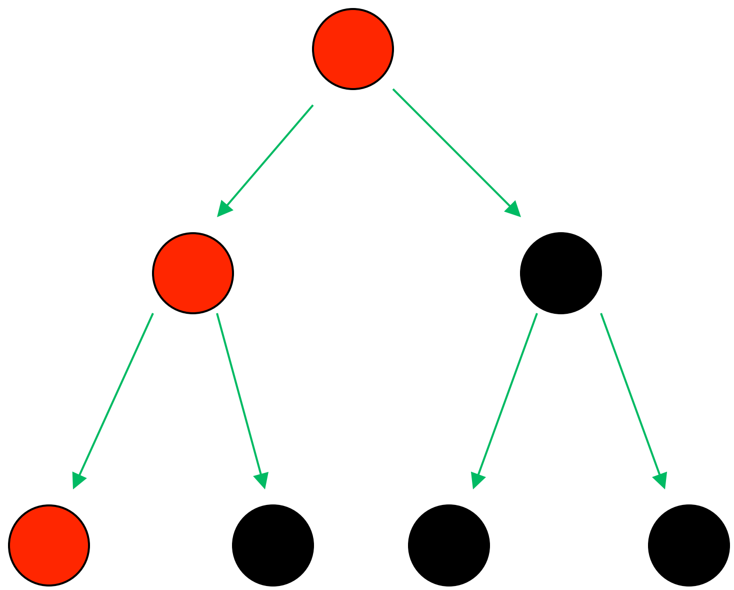
AF g "g will necessarily become true"

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 (CES83-86) uses the temporal operators

AX, AG, AF, AU

EX, EG, EF, EU

CTL* allows complex nestings such as

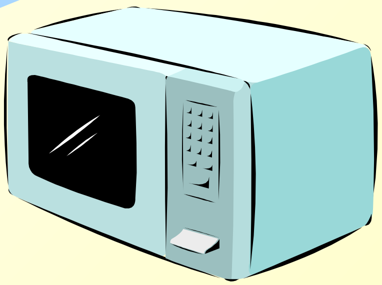
AXX, AGX, EXF, ...

Model Checking Problem

- Let M be a state-transition graph.
- Let f be the specification in temporal logic.
- Find all states s of M such that $M, s \models f$.

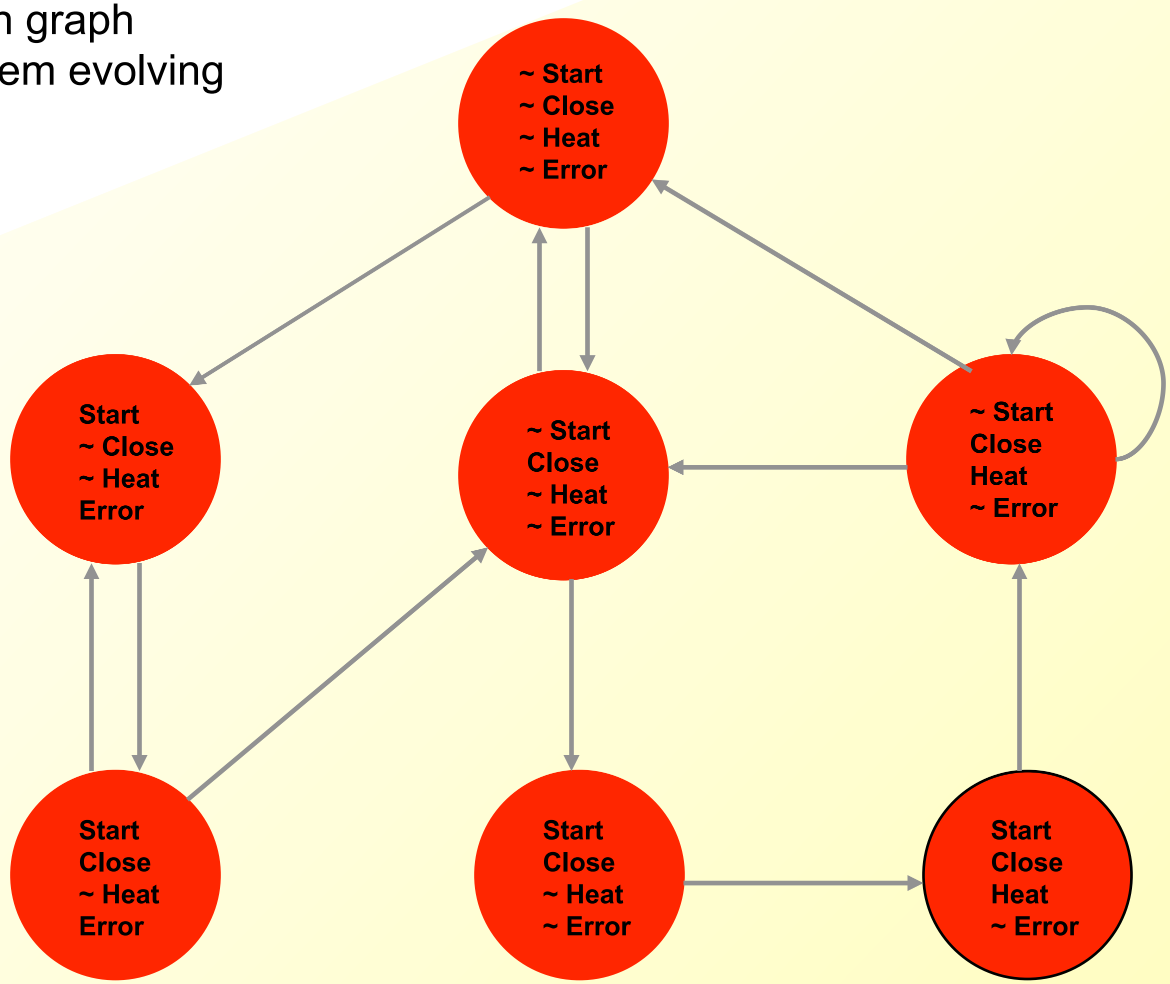
- **CTL Model Checking: CE 81; CES 83/86; QS 81/82.**
- **LTL Model Checking: LP 85.**
- **Automata Theoretic LTL Model Checking: VW 86.**
- **CTL* Model Checking: EL 85.**

Trivial Example



Microwave Oven

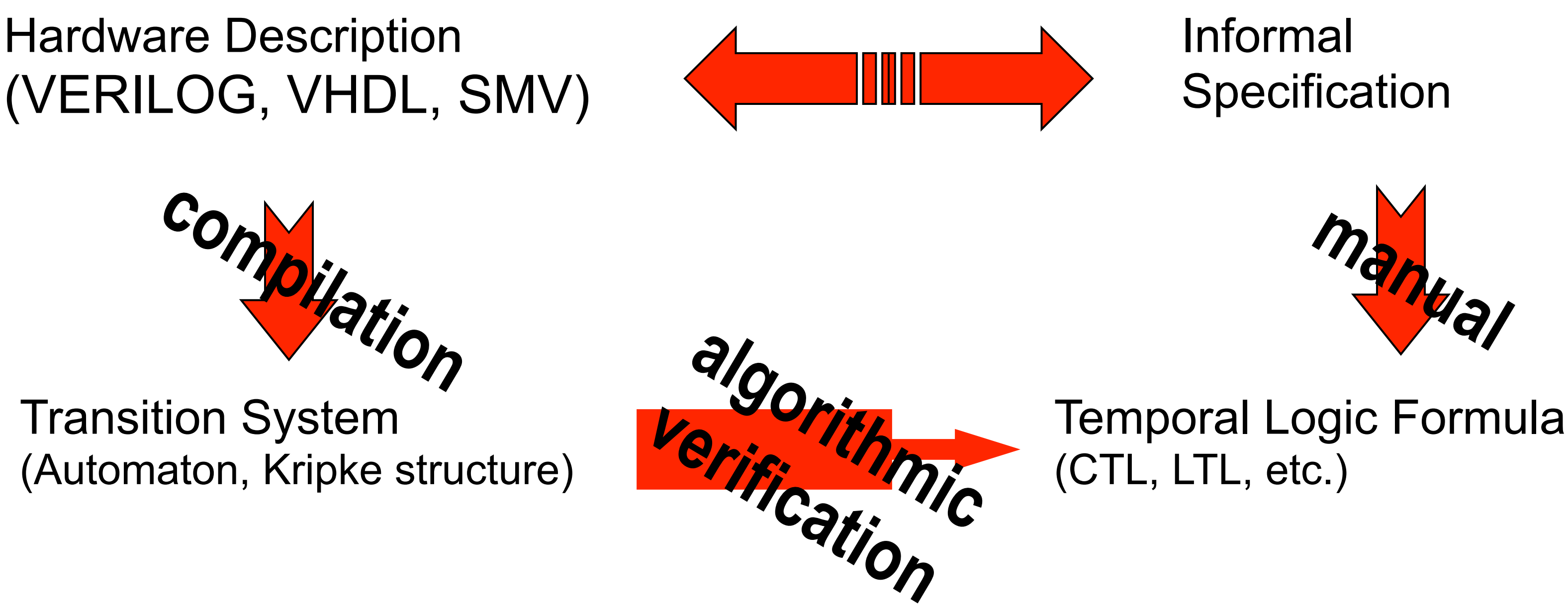
State-transition graph describes system evolving over time.



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



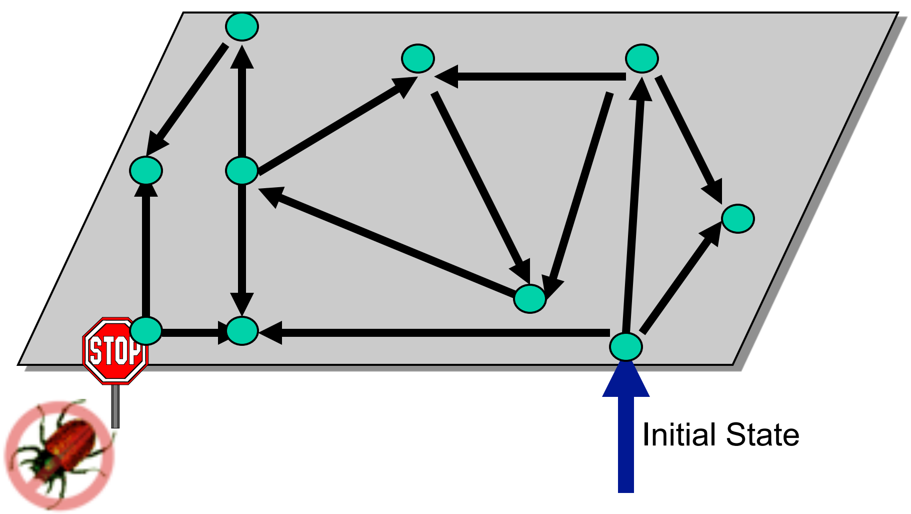
Counterexamples

Program or circuit

Informal Specification

Transition System

Temporal Logic Formula
(CTL, LTL, etc.)



Safety Property:

bad state 

unreachable:

satisfied

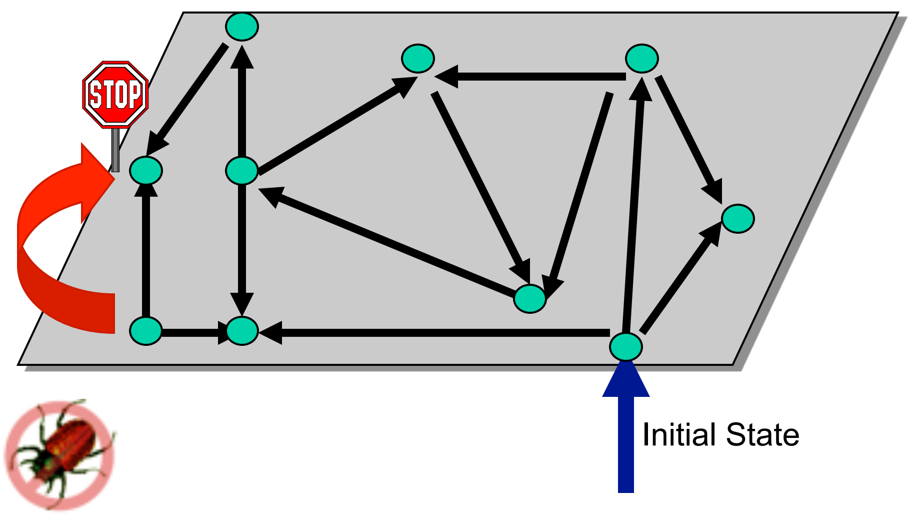
Counterexamples

Program or circuit

Informal Specification

Transition System

Temporal Logic Formula
(CTL, LTL, etc.)



Safety Property:
bad state 

unreachable

Counterexample

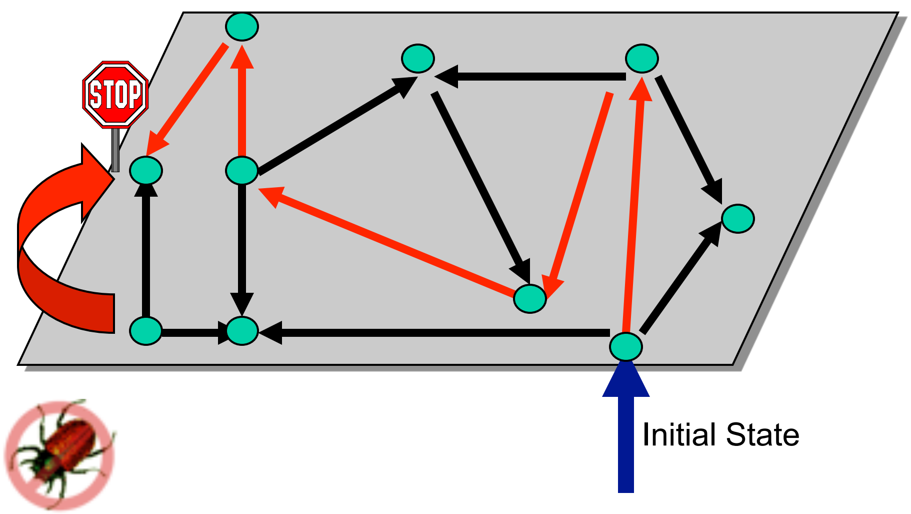
Counterexamples

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 Future + 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 on State Space Explosion Problem!

- **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 on State Space Explosion Problem (Cont.)



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

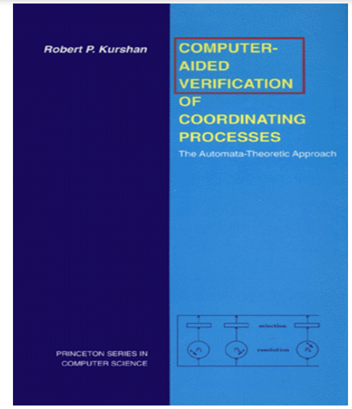
Can the given property fail in k-steps?

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

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 on State Space Explosion Problem (Cont.)

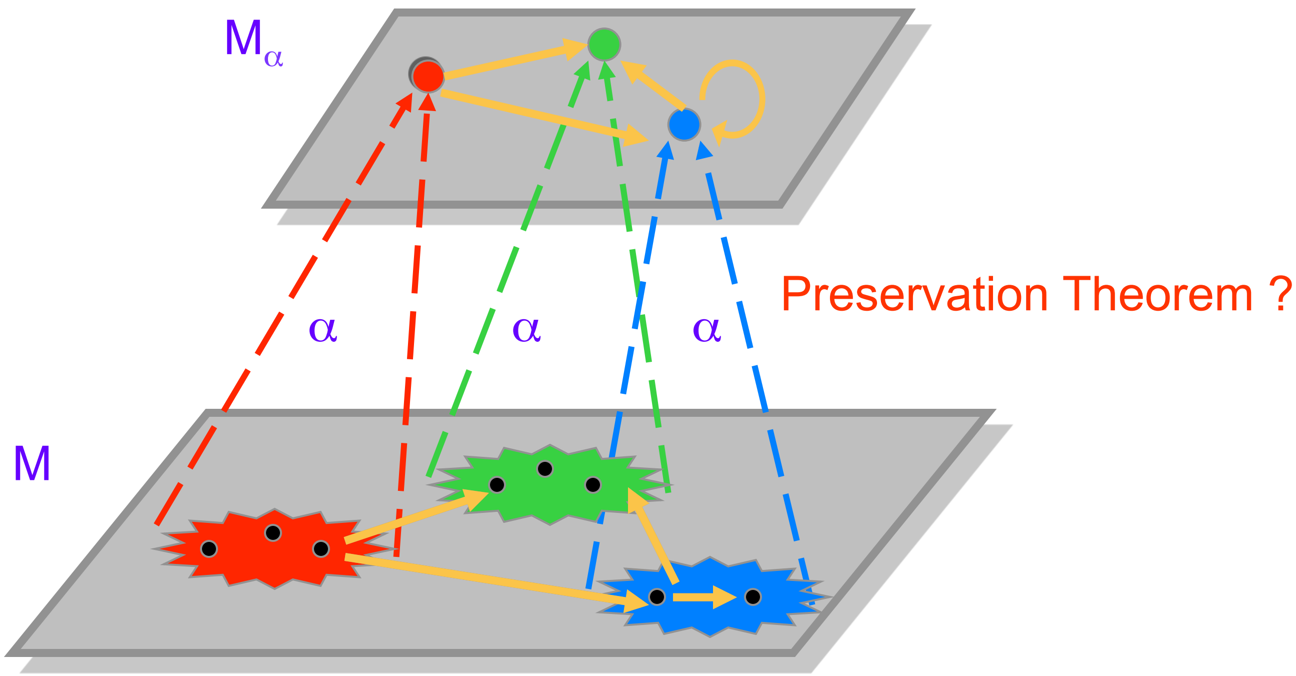


- **Localization Reduction**
  - Bob Kurshan 1994

- **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:

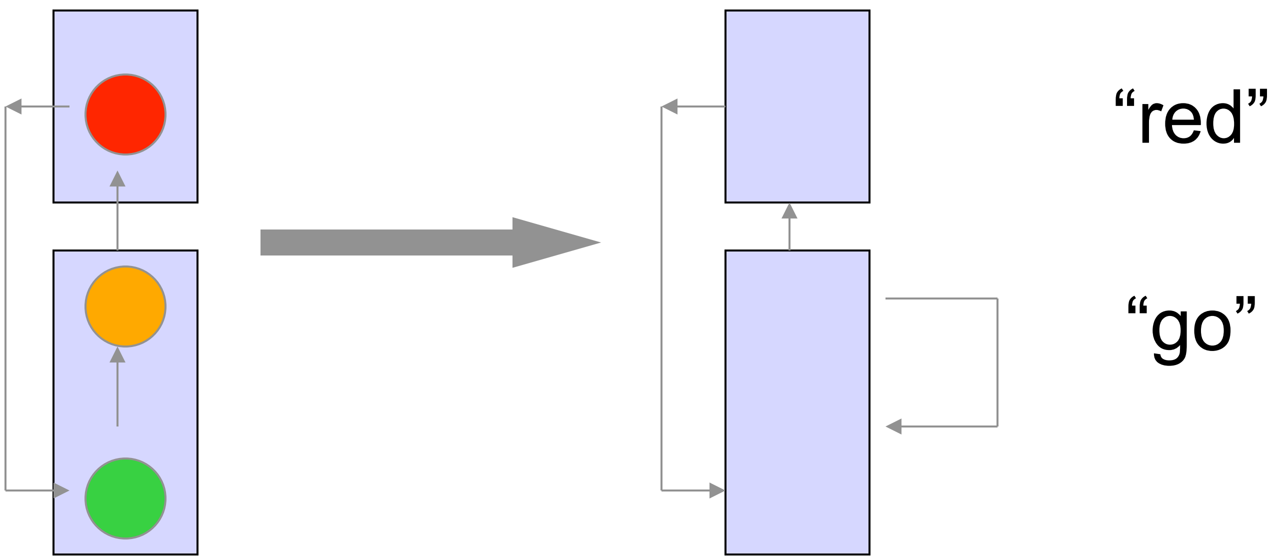


Preservation Theorem

- 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



AGAF red

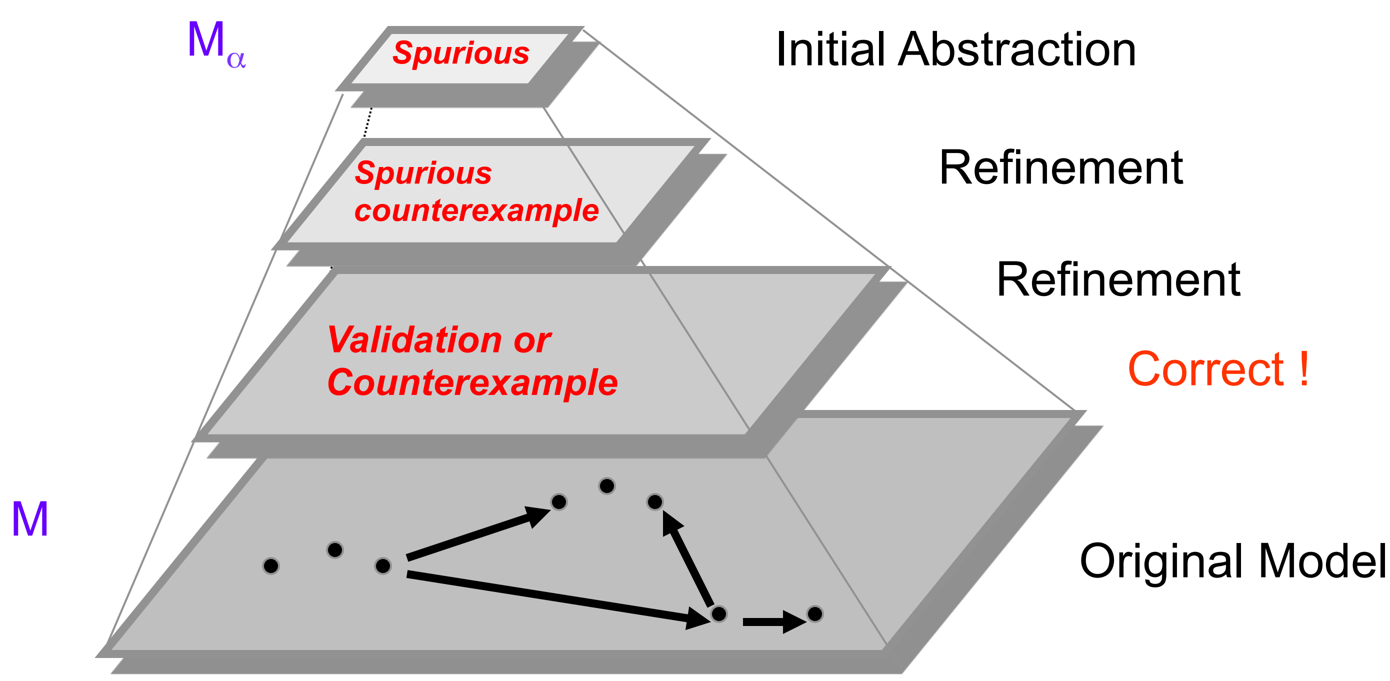
"Every path necessarily leads back to red."

Spurious Counterexample:

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

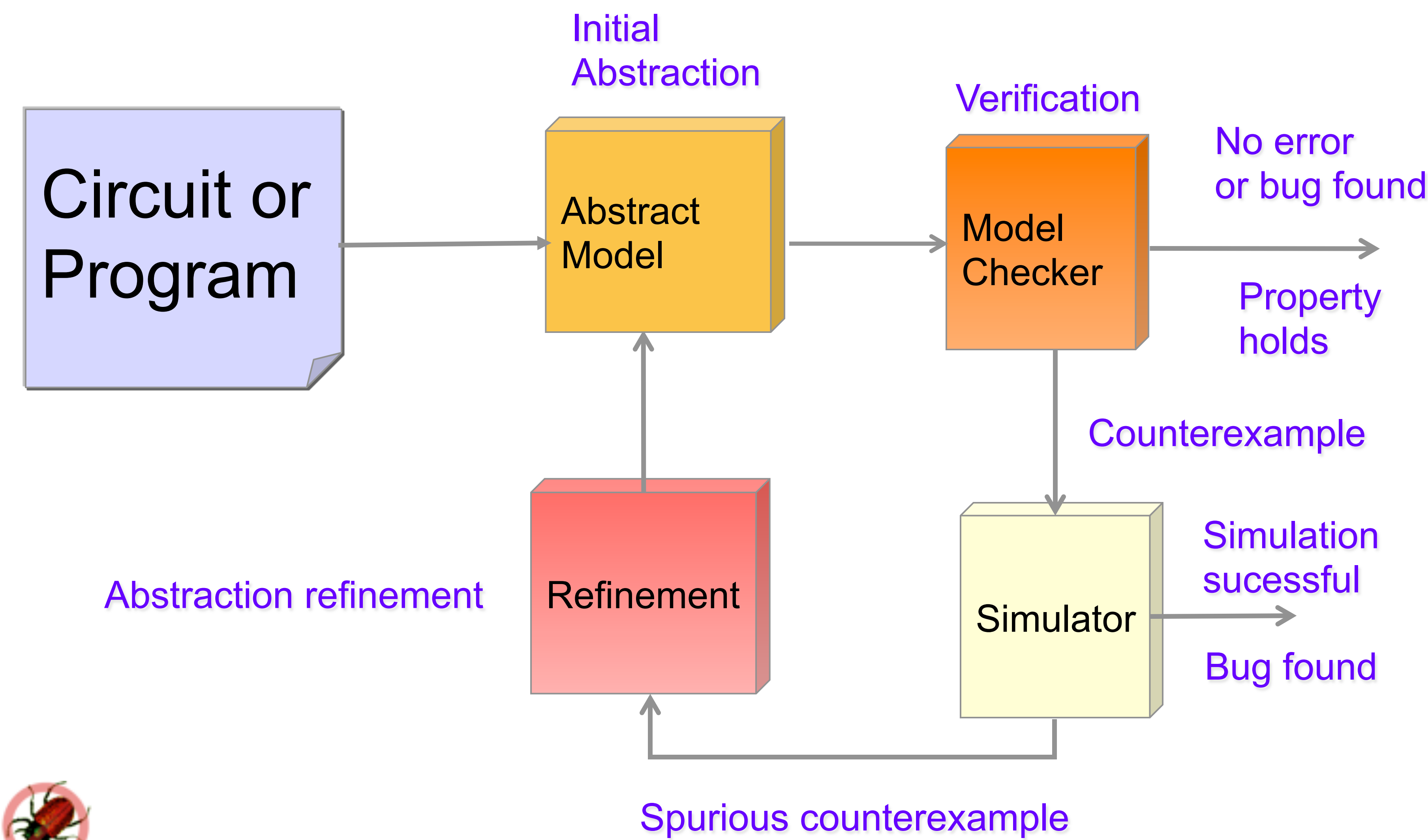
Artifact of the abstraction !

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?

1. Combine static analysis and model checking

Use static analysis to extract a model K from a boolean 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)

What Does It Mean to Model Check Software?

2. Simulate program along all paths in computation tree

- Java PathFinder (NASA Ames)
- Source code + backtracking (e.g., Verisoft)
- Source code + symbolic execution + backtracking (e.g., MS/Intrinsa Prefix)

3. Use finite-state machine to look for patterns in control-flow graph [Engler]

What Does It Mean to Model Check Software?

4. Design with Finite-State Software Models

Finite state software models can act as “missing link” between transition graphs and complex software.

- **Statecharts**
- **Esterel**

What Does It Mean to Model Check Software?

5. Use Bounded Model Checking and SAT [Kroening]

- Problem: How to compute set of reachable states? Fixpoint computation is too expensive.
- Restrict search to states that are reachable from initial state within fixed number n of transitions
- Implemented by unwinding program and using SAT solver

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?

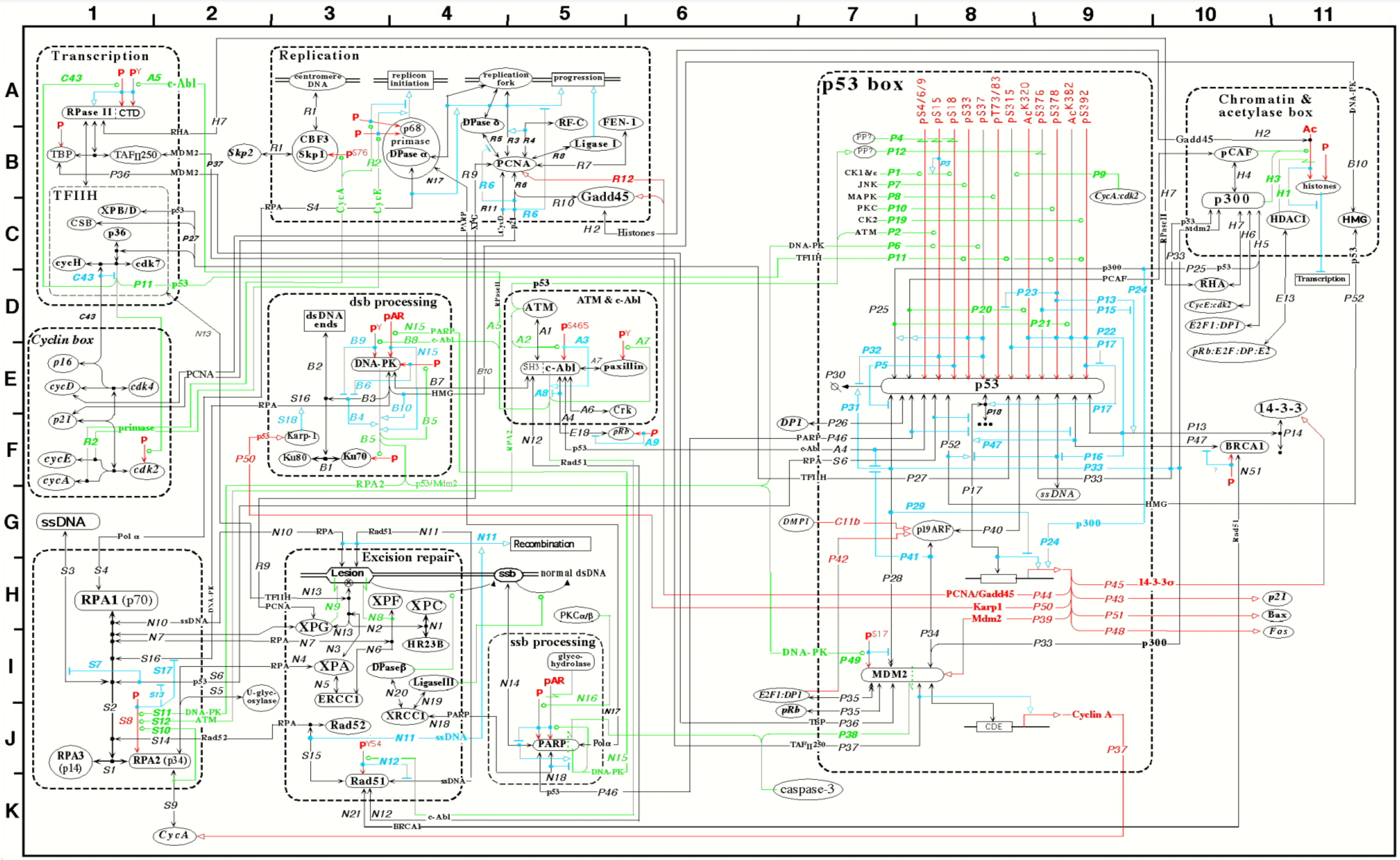


Figure 6B: The p53-Mdm2 and DNA repair regulatory network (version 2p - May 19, 1999)

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."

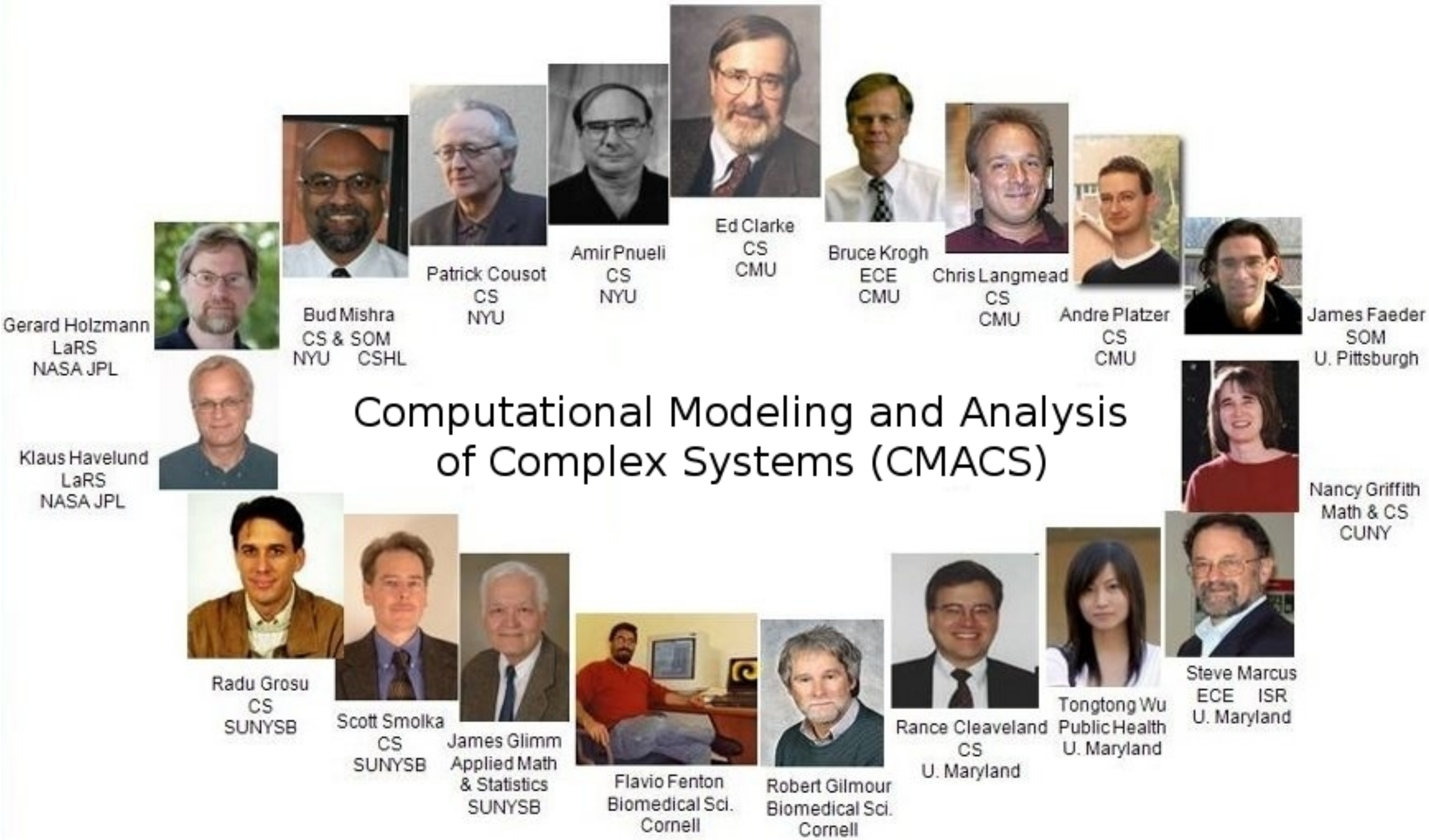
(A. Vazquez, E. Bond, A. Levine, G. Bond. The genetics of the p53 pathway, apoptosis and cancer therapy. Nat Rev Drug Discovery 2008 Dec;7(12):979-87.)

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

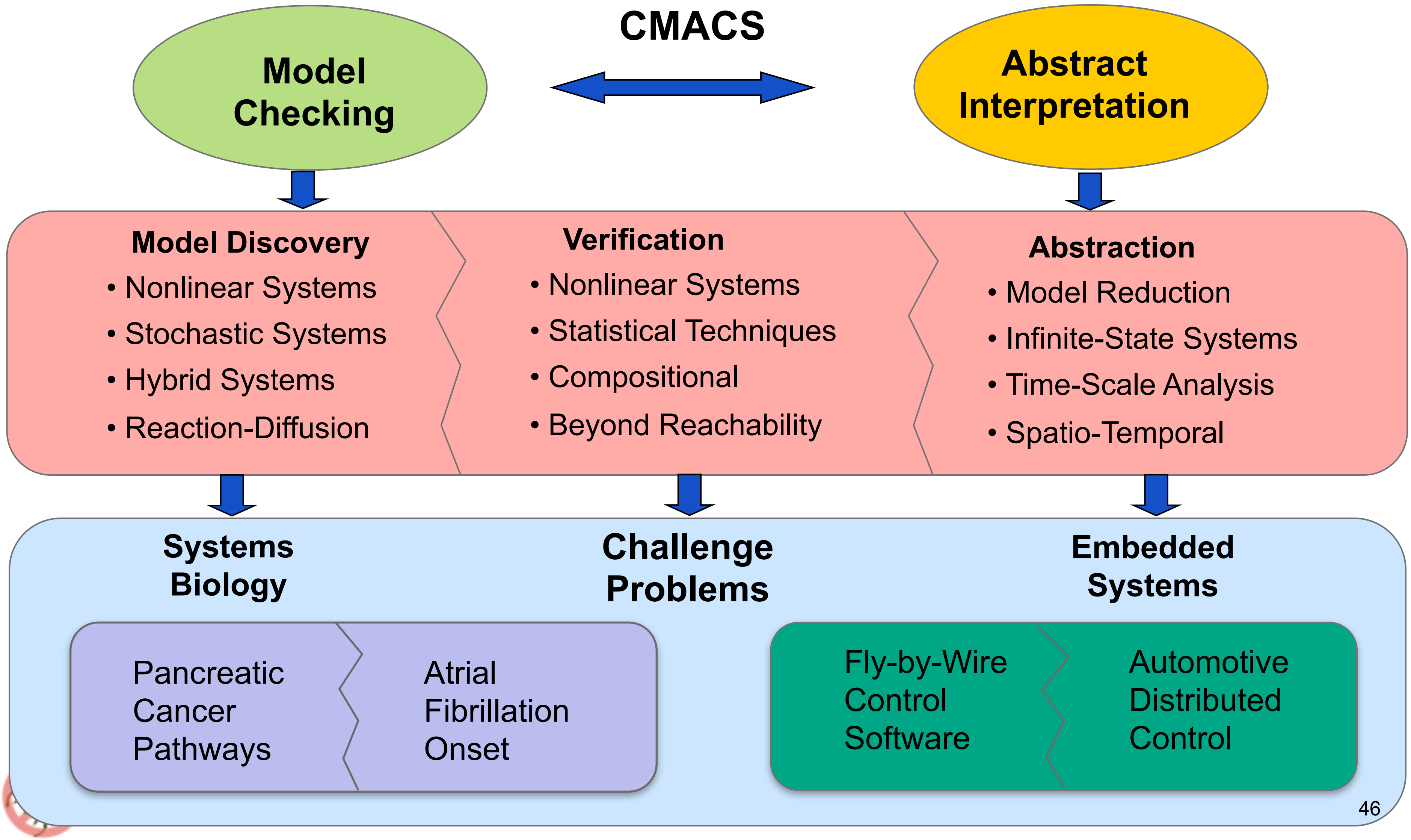
In 1993, **p53** was voted ***molecule of the year*** by **Science Magazine**.

New NSF Expedition Grant

Next-Generation Model Checking and Abstract Interpretation with a Focus on Systems Biology and Embedded Systems



CMACS Strategic Plan



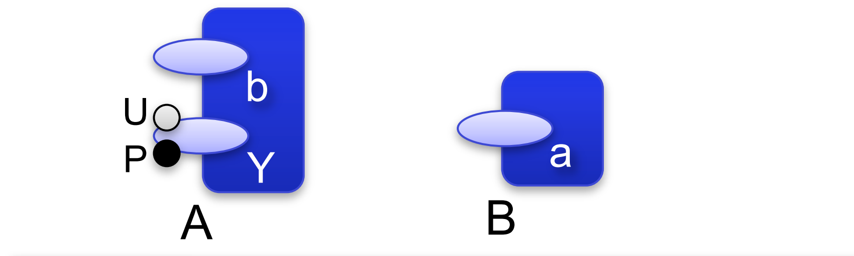
The BioNetGen Language

Jim Faeder, UPMC

begin molecule types

```
A(b,Y~U~P)
B(a)
```

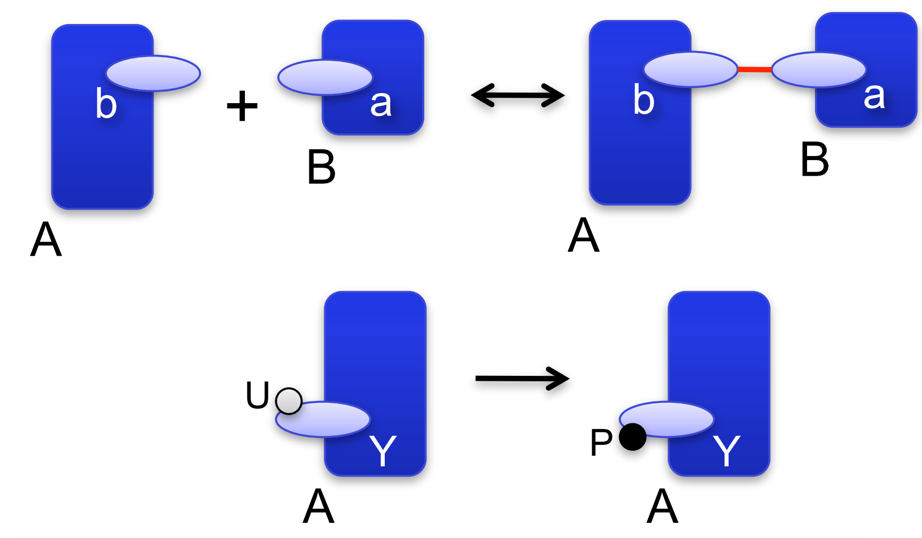
end molecule types



begin reaction rules

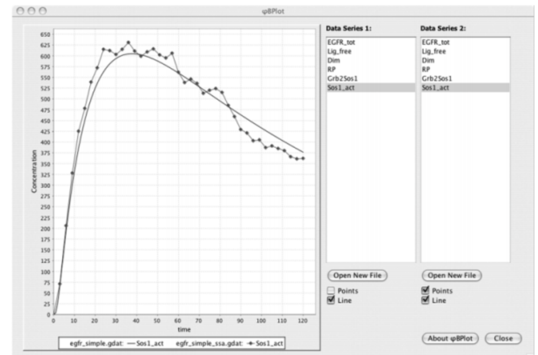
```
A(b)+ B(a)<-> A(b!1).B(a!1)

A(Y~U) -> A(Y~P)
```

end reaction rules



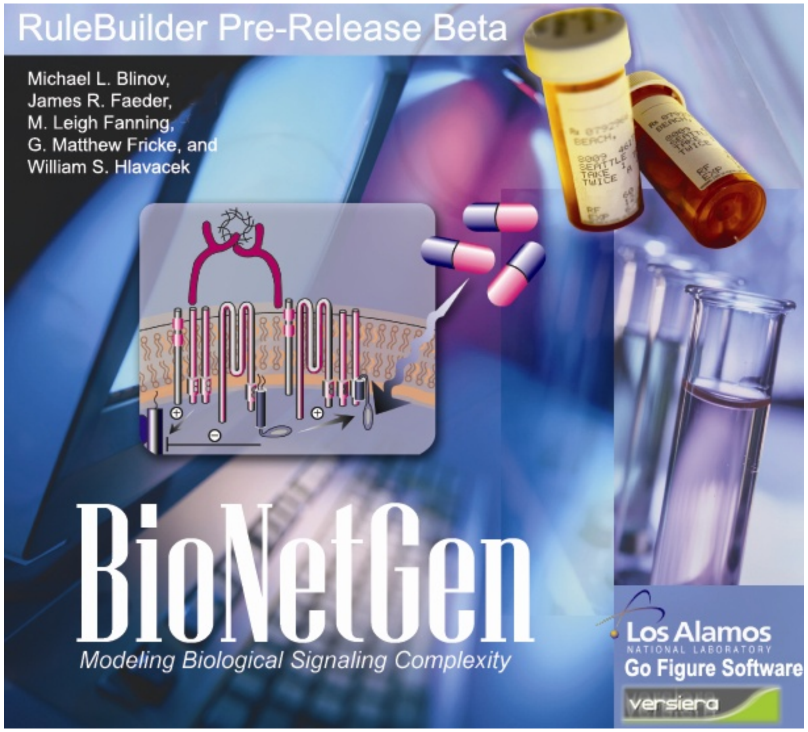
Faeder JR, Blinov ML, Hlavacek WS **Rule-Based Modeling of Biochemical Systems with BioNetGen.** In *Methods in Molecular Biology: Systems Biology*, (2009).

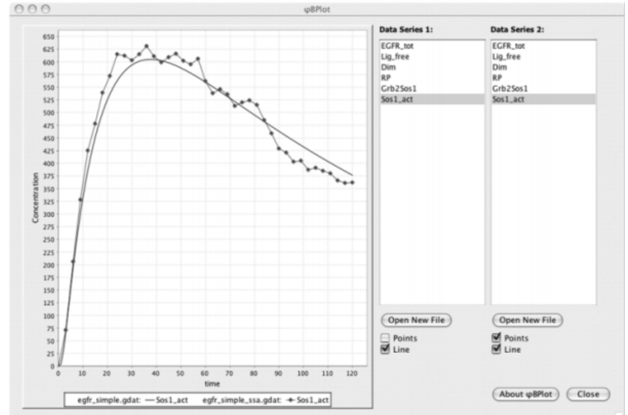
Existing Approach: Manual Analysis



Many simulation traces need to be carefully analyzed !

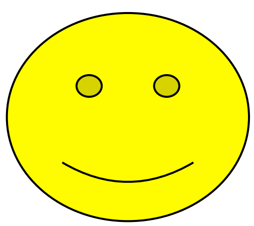
Model Checking Approach





Automated

Analysis !



Bounded Linear Temporal Logic

- Bounded Linear Temporal Logic (BLTL): Extension of LTL with time bounds on temporal operators.
- Let $\sigma = (s_0, t_0), (s_1, t_1), \ldots$ be an execution of the model
  - along states $s_0, s_1, \ldots$
  - the system stays in state s_i *for time* t_i
- σ^i: Execution trace starting at state i.
- $V(\sigma, i, x)$: Value of the variable x at the state s_i .
- A natural model for BioNetGen traces.

Bounded Linear Temporal Logic

- Bounded Linear Temporal Logic (BLTL): Extension of LTL with **time bounds** on temporal operators.
- Let $\sigma = (s_0, t_0), (s_1, t_1), \ldots$ be an execution of the model
  - along states $s_0, s_1, \ldots$
  - the system stays in state s_i *for time* t_i
- A natural model for BioNetGen traces.
- Example: (Yeast Heterotrimec G Protein Cycle) does the G protein stay above 6000 for 2 time units and fall below 6000 before 20 time units?
  - $G^2\,(GProtein > 6000) \wedge F^{20}\,(GProtein < 6000)$

Semantics of BLTL

The semantics of the **timed Until** operator:

- "within time t, Φ_2 will be true and Φ_1 will hold until then "
- σ^k: Execution trace starting at state k.
- $\sigma^k \models \Phi_1 \,\mathcal{U}^t\, \Phi_2$ iff there exists natural n such that
  1) $\sigma^{k+n} \models \Phi_2$
  2) $\Sigma_{i<n}\, t_{k+i} \leq t$
  3) for each $0 \leq j < n$, $\sigma^{k+j} \models \Phi_1$

- In particular: $F^t\, \Phi = true\ U^t\, \Phi$, $G^t\, \Phi = \neg F^t\, \neg\Phi$

Semantics of BLTL

The semantics of BLTL for a trace σ^k:

- $\sigma^k \models x \sim c$ iff $V(\sigma, k, x) \sim c$, where $\sim$ is in $\{\leq, \geq, =\}$
- $\sigma^k \models \Phi_1 \vee \Phi_2$ iff $\sigma^k \models \Phi_1$ or $\sigma^k \models \Phi_2$
- $\sigma^k \models \neg\Phi$ iff $\sigma^k \models \Phi$ does not hold
- $\sigma^k \models \Phi_1 \,\mathcal{U}^t\, \Phi_2$ iff there exists natural i such that
  1) $\sigma^{k+i} \models \Phi_2$
  2) $\Sigma_{j<i}\, t_{k+j} \leq t$
  3) for each $0 \leq j < i$, $\sigma^{k+j} \models \Phi_1$

Probabilistic Model Checking

- Given a stochastic model $\mathcal{M}$ such as
  - a Discrete or Continuous Markov Chain, or
  - a stochastic differential equation
- a BLTL property ϕ and a probability threshold $\theta \in (0,1)$.
- Does $\mathcal{M}$ satisfy ϕ with probability at least θ?

$$\mathcal{M} \models P_{\geqslant\theta}(\phi)$$

- Numerical techniques compute precise probability of $\mathcal{M}$ satisfying ϕ:
  - Does **NOT** scale to large systems.

Wait a minute!

Isn't ***Statistical Model Checking*** an oxymoron?

I thought so for the first 28 years of my quest.

Much easier to simulate a complex biological system than to build the transition relation for it.

Moreover, we can bound the probability of error.

Statistical Model Checking

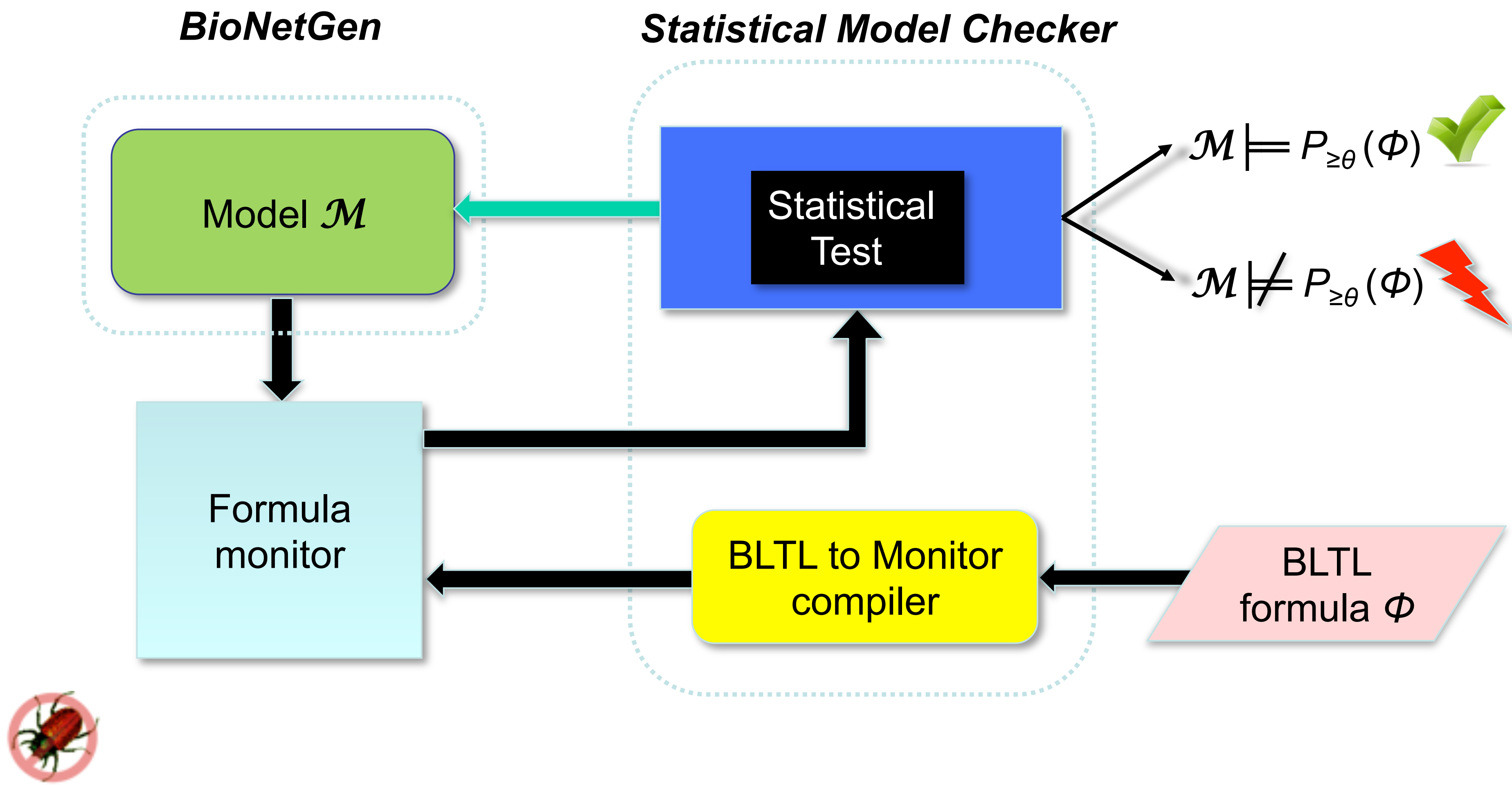
- Decides between two mutually exclusive hypotheses:
  - Null Hypothesis $H_0 : \mathcal{M} \models P_{\geqslant\theta}(\phi)$
  - Alternate Hypothesis $H_1 : \mathcal{M} \models P_{<\theta}(\phi)$

- Statistical tests can determine the true hypothesis:
  - based on sampling the traces of system $\mathcal{M}$
  - answer may be wrong, but error probability is bounded.

- ***Statistical Hypothesis Testing* → *Model Checking!***

BioLab 2.0

Model Checking Biochemical Stochastic models: $\mathcal{M} \models P_{\geq\theta}(\Phi)$?



Motivation - Scalability

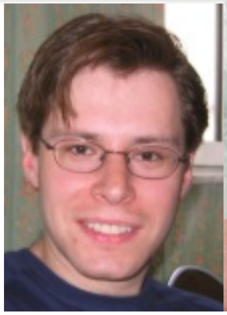
- State Space Exploration often infeasible for complex systems.
  - May be relatively easy to simulate a system
- Our Goal: Provide probabilistic guarantees using fewer simulations
  - How to generate each simulation run?
  - How many simulation runs to generate?
- Applications: BioNetGen, Stateflow / Simulink

BioLab: A Statistical Model Checker for BioNetGen Models.
E. Clarke, C. Langmead, J. Faeder, L. Harris, A. Legay and S. Jha. (*International Conference on Computational Methods in System Biology, 2008*)

Motivation – Parallel Model Checking

- Some success with explicit state Model Checking
- More difficult to distribute Symbolic MC using BDDs.
- Learned Clauses in SAT solving are not easy to distribute.
- Multiple simulations can be easily parallelized.
- Next Generation Model Checking should exploit
  - multiple cores
  - commodity clusters

Existing Work




- [Younes and Simmons 02-06] use Wald's SPRT
  - SPRT: Sequential Probability Ratio Test
- [Hérault et al. 04] use Chernoff bound:
  - Estimate the probability that $\mathcal{M} \models \Phi$
- [Sen et al. 04-05] use *p-value*:
  - Approximates the probability that the null hypothesis $\mathcal{M} \models P_{\geq\theta}(\Phi)$ is true
- [Clarke et al. 09] Bayesian approach
  - Both hypothesis testing and estimation
  - Faster (fewer samples required)

The End

Questions?