CMACS

Computational Modeling and Analysis for Complex Systems

Model checking

Christopher James Langmead

Department of Computer Science &

Lane Center for Computational Biology

Carnegie Mellon University

January 14, 2010

Outline

- Complex Systems
- Model checking
  - History, Concepts, Significance
- Model checking Biology?
  - Example: Cancer

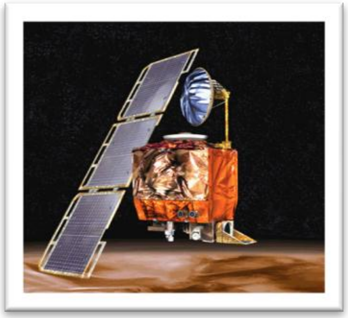
Complex Systems

- Even rocket scientists make mistakes

Ariane 5 (1996),
floating point conversion error

Mission Loss

Mars Climate Orbiter
(1999), unit confusion

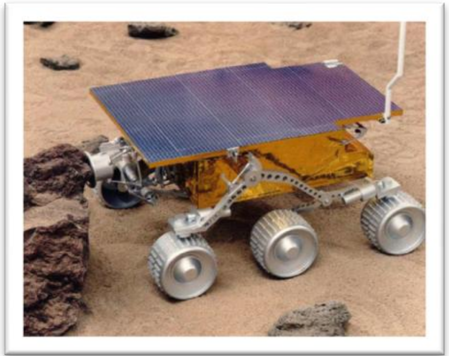


Mission Loss

Mars Polar Lander,
(1999) Landing logic error

Mission Loss

Mars PathFinder (1997),
priority inversion deadlock



Deep Space 1,
(1999) data race

Spirit Mars Rover,
(2004) file system error

Complex Systems

- … so do airplane designers

Airbus A330

2009: Crash off Brazil due to inaccurate airspeed indication

Complex Systems

- … and ship designers

USS Yorktown:
1997 database overflow caused its propulsion system to fail

Complex Systems

- … and circuit designers

Intel Pentium:
1994 FDIV bug:
  execute 4195835 – 4195835 / 3145727 * 3145727
Chip returns 256, instead of zero

Complex Systems

- … and financial "wizards"
  - Remember subprime mortgages?

Verification

- Theoretically, we cannot know whether a given design is free from *all* bugs
  - Because its impossible to anticipate <u>everything</u> that could possibly go wrong

- But, we *can* verify that the design satisfies properties we specify explicitly
  - E.g., “does the design satisfy property 1 *and* property 2 *and* … *and* property n?”

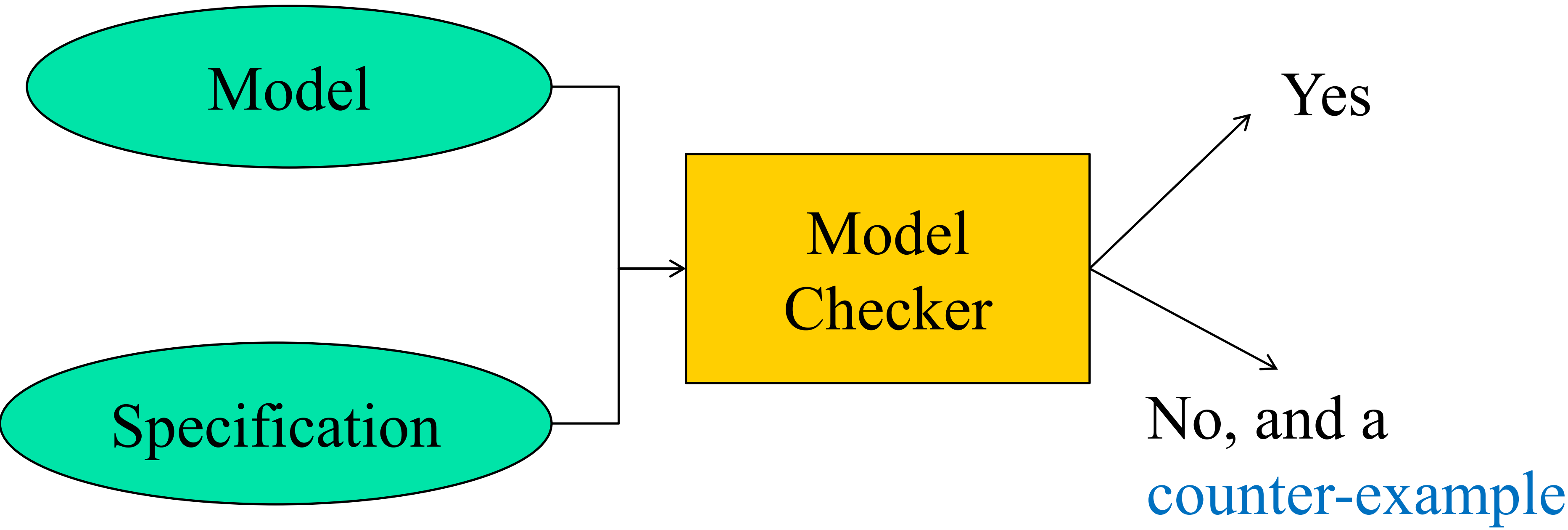
Formal Verification

- The field of Formal Verification develops clever techniques for computationally determining whether a given design satisfies a given set of properties

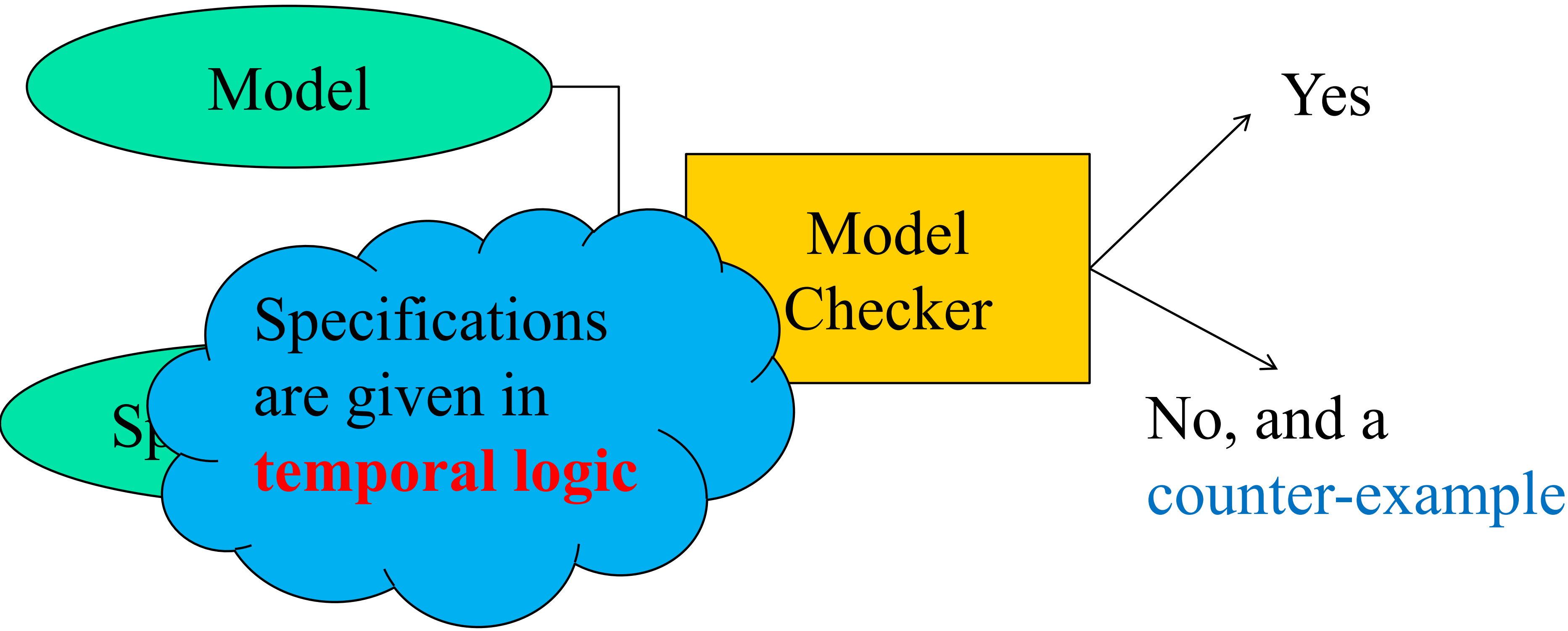
Verification Strategies

- Theorem Proving
  - Disadvantage: tedious and difficult
- Simulation/testing
  - Disadvantage: modern designs are too complicated to test exhaustively
- Model checking
  - Advantage:
    - Algorithmic (i.e., automated)
    - Has been successful in the “real world”
    - Can aid in debugging designs

The model checking problem



The model checking problem



Temporal Logic

- A logical notation for specifying logical relationships in time
  - E.g., “event p happens before event q”

- Two Types of Temporal Logics
  - Linear Time Logic (LTL)
  - Branching Time Logic
    - E.g., Computation Tree Logic (CTL)

Linear Temporal Logic (LTL)

- Temporal operators
  - G p — "henceforth p is true"
  - F p — "eventually p will be true"
  - X p — "p will be true next step"
  - p U q — "p is true until q is true"

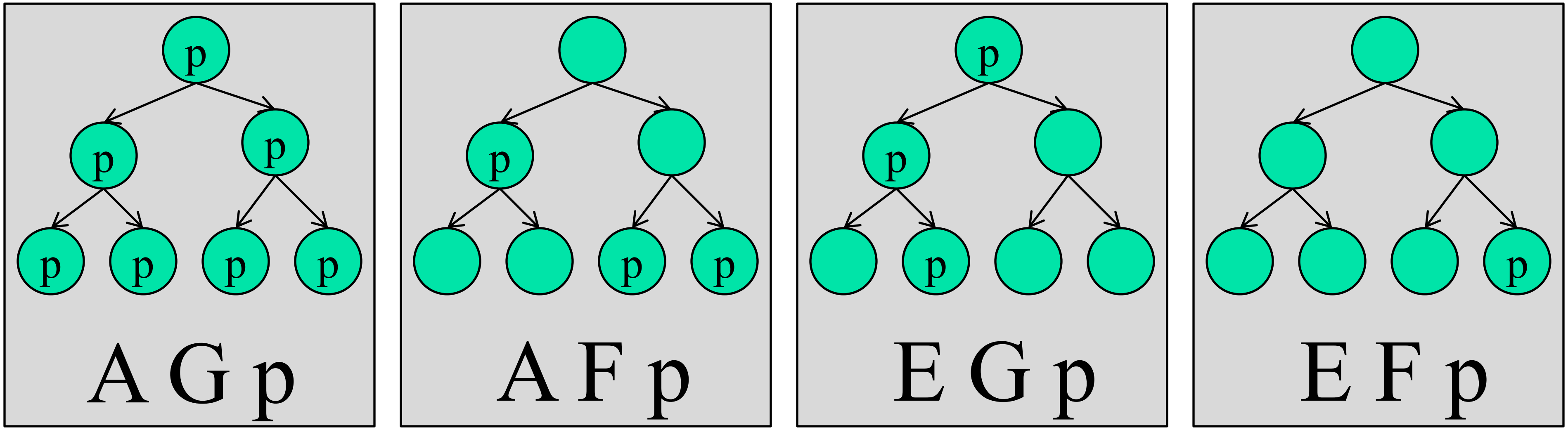
- An implicit path quantifier:
  - A ϕ — "property ϕ holds for all paths"
    - ϕ is 'G p', 'F p', 'X p', or 'p U q'

Typical LTL formulas

- Liveness (something ‘good’ will happen)
  - G F p
    - p eventually becomes true, for all paths
- Safety (nothing ‘bad’ will happen)
  - $G \neg (p \& q)$
    - p and q are never true at the same time
- Fairness
  - $(G F p) \rightarrow (G F q)$
    - If p becomes true, q eventually becomes true

Computation tree logic (CTL)

- 2 (explicit) path quantifiers
  - A = “for all paths”
  - E = “for some path”

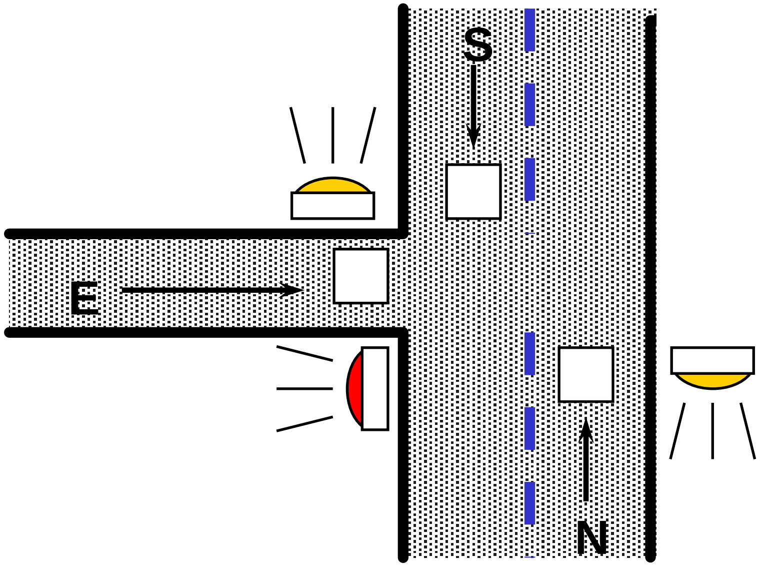


Typical CTL formulas

- $EF(p \wedge \neg q)$
  - It is possible to get to a state where p holds, but q does not
- $AG(p \Rightarrow AF\ q)$
  - If p occurs, q will eventually occur
- AG(AF p)
  - p occurs infinitely often, along all paths
- AG(EF p)
  - It is possible to reach state a state were p holds, regardless of where you begin

Model checking: Example

- Traffic Light Controller
  - 3 sensors (N,S,E)
  - If car present, and light red, sensor requests a light change
  - A lock controls access to the lights



- Can we:
  - Guarantee no collisions
  - Guarantee eventual service

Specifications

- Safety (no collisions)

  $AG \neg (E\_Go \wedge (N\_Go \mid S\_Go));$

- Liveness

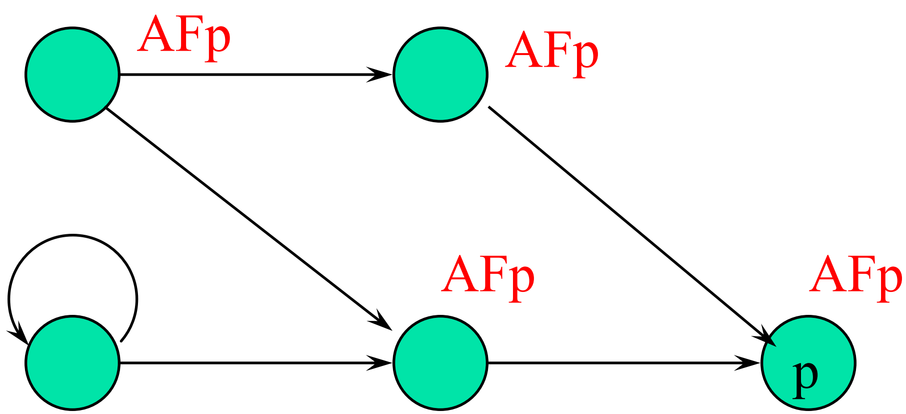
  $AG (\neg N\_Go \wedge N\_Sense \Rightarrow AF\ N\_Go);$

  $AG (\neg S\_Go \wedge S\_Sense \Rightarrow AF\ S\_Go);$

  $AG (\neg E\_Go \wedge E\_Sense \Rightarrow AF\ E\_Go);$

CTL model checking algorithm

- Example: AF p = "inevitably p"

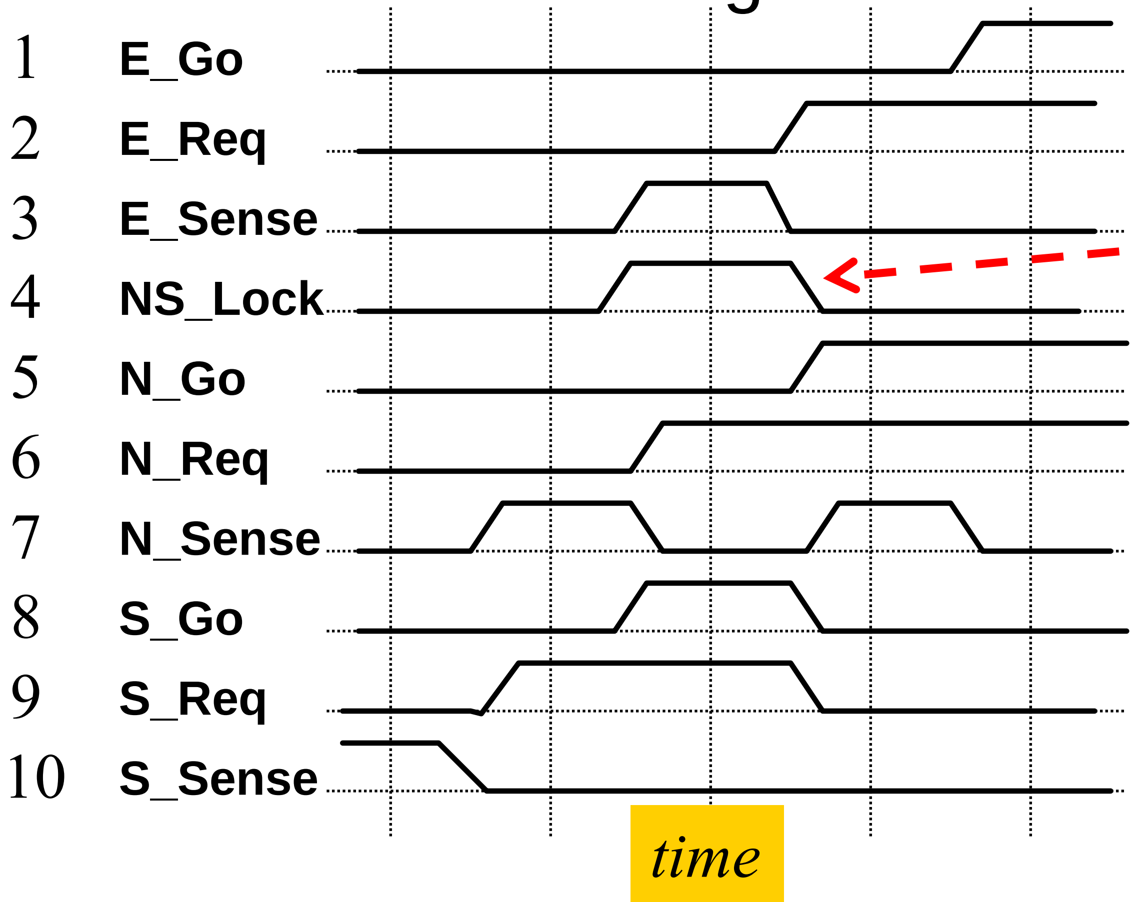


- Complexity
  - linear in size of model (FSM)
  - linear in size of specification formula

Note: corresponding LTL MC algorithm is exponential in formula size

Counterexample

- AG ¬ (E_Go ∧ (N_Go | S_Go)) is false
  - Ex. East and North lights on at same time...



S releases NS lock, just as N light goes on.

E thus gets lock (by mistake), and turns on, while N is still on

State explosion problem

- Explicit state model checkers are only applicable to small systems
  - ~10^9 states

- Unfortunately, most real-world designs have **MUCH** larger state spaces

State explosion problem

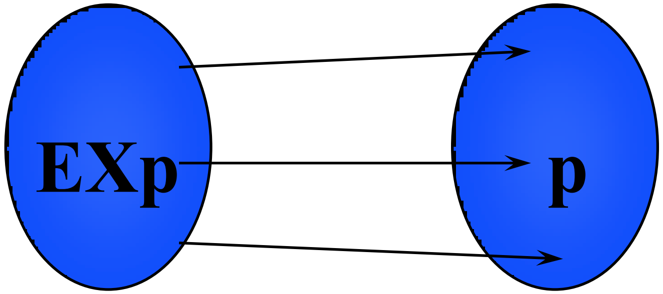
- The MC community has devised a number of clever approaches to (partially) dealing with this problem
  - Abstraction
  - “Symbolic” methods
  - “Partial order” methods

Symbolic model checking

- Basic idea:
  - Don’t represent states explicitly. Use clever data structures or formulas to implicitly represent sets of states, and the transitions between them
  - Model checking can then be performed using set operations

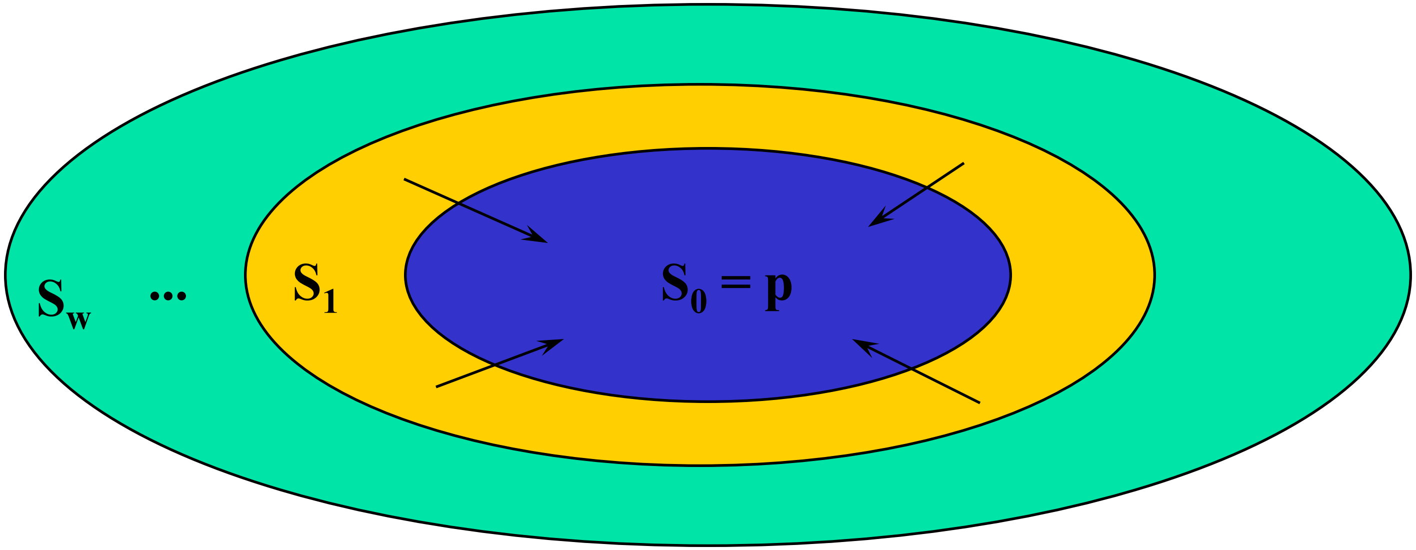
Image computation

- EX p = states that can reach p in one step



Fixed point iteration

- EF p = states that can reach p



Model checking: History

- Early 1980s: model checking invented
- 1990s: first commercial applications
- Late 1990s/Early 2000s: first applications to Biology

Model Checking: Significance

- Symbolic approaches have been used to perform model checking on systems with more than 10^{120} states

- Industrial Applications
  - Hardware Design
  - Avionics
  - Chemical plant control
  - Nuclear Storage facilities

Outline

- Complex Systems
- Model checking
  - History, Significance, Concepts
- Model checking Biology?
  - Example: Cancer

Model Checking for Biology?

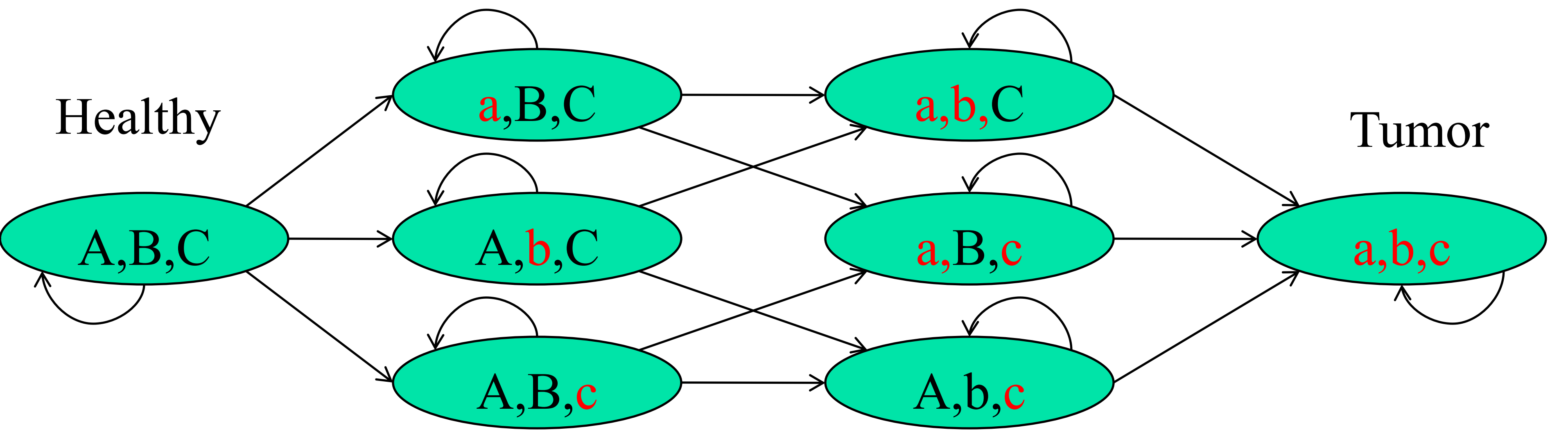
- Biological systems are significantly different than engineered systems
  - Stochastic Dynamics
  - Larger state spaces (some are infinite-state)
  - Biological systems evolve in time *and* space
- Biologists have different needs than engineers
  - E.g., a biologist will often want to know the *probability* of an event occuring, not simply whether an event can occur

Probabilistic model checking

- There are specialized temporal logics for reasoning about the probability a specification is true
  - Ex. $\text{Pr}_{\geq\rho}$ F p
    - The probability that p will be true in the future is greater than, or equal to ρ
- Specialized model checking algorithms (e.g., BioLab) can be used to determine whether the model satisfies the property
  - These algorithms rely on extensive simulations and statistics

Example

- Cancer Modeling
  - Tumor development is a complex process involving many genetic changes
  - These genetic changes occur over time
  - Are there preferred mutation sequences?



Example: Properties

- $\Pr_{\geq\rho}\ \neg$A_mut U (A_mut $\wedge$ B_mut $\wedge$ C_mut)
  - "The probability that mutations *B* and *C* occur before mutation *A* is at least ρ"
- A_mut $\rightarrow \Pr_{\geq\rho}$(G F A_mut $\wedge$ B_mut $\wedge$ C_mut)
  - "If *A* is mutated, the probability that a tumor will develop is at least ρ"
- A_mut $\wedge$ D $\rightarrow \Pr_{\leq\rho}$(G F A_mut $\wedge$ B_mut $\wedge$ C_mut)
  - "If *A* is mutated, but we use drug D, the probability that a tumor will develop is no more than ρ"

This workshop

- You will be modeling specific signaling pathways in BioNetGen that are known to be altered in many tumor types
- You will be using model checking to verify properties of the BioNetGen models

Summary

- Model checking is useful way to verify properties of complex systems
- Historically, model checking was invented to verify properties of engineered systems
- More recently, new model checking algorithms have been developed for studying biological systems