Lecture 15

Model Checking CTL

Zvonimir Rakamarić
University of Utah

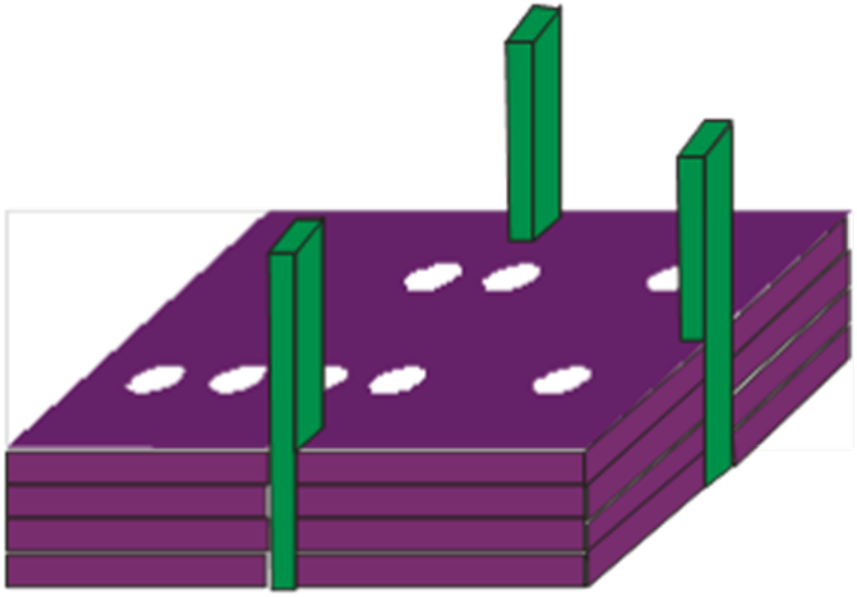
Announcement: Peter Neumann in Town

- Principal Scientist at SRI International
- Two lectures
  - A Personal History of Layered Trustworthiness
    Tue, 7:00PM, 220 Skaggs Biology Research Building
  - Clean-Slate Formally Motivated Hardware and Software for Highly Trustworthy Systems
    Wed, 3:40 PM, 102 L WEB

Homework 3 Assignment I

- Puzzle:
  - You are given a collection of cards and a card stand with three pegs.
  - Because of the stand pegs and the notches in the cards, each card fits on the stand in either of two ways.
  - Each card contains two columns of holes, some of which may not be punched out.
  - The puzzle is solved by placing all the cards on the stand so as to completely cover the stand (every hole position is blocked by at least one card that has no hole there).
- Let PUZZLE = {$(c_1,\ldots,c_k)$ such that each c_i is a card and this collection has a solution}
- Assignment: Prove that PUZZLE is NP-complete

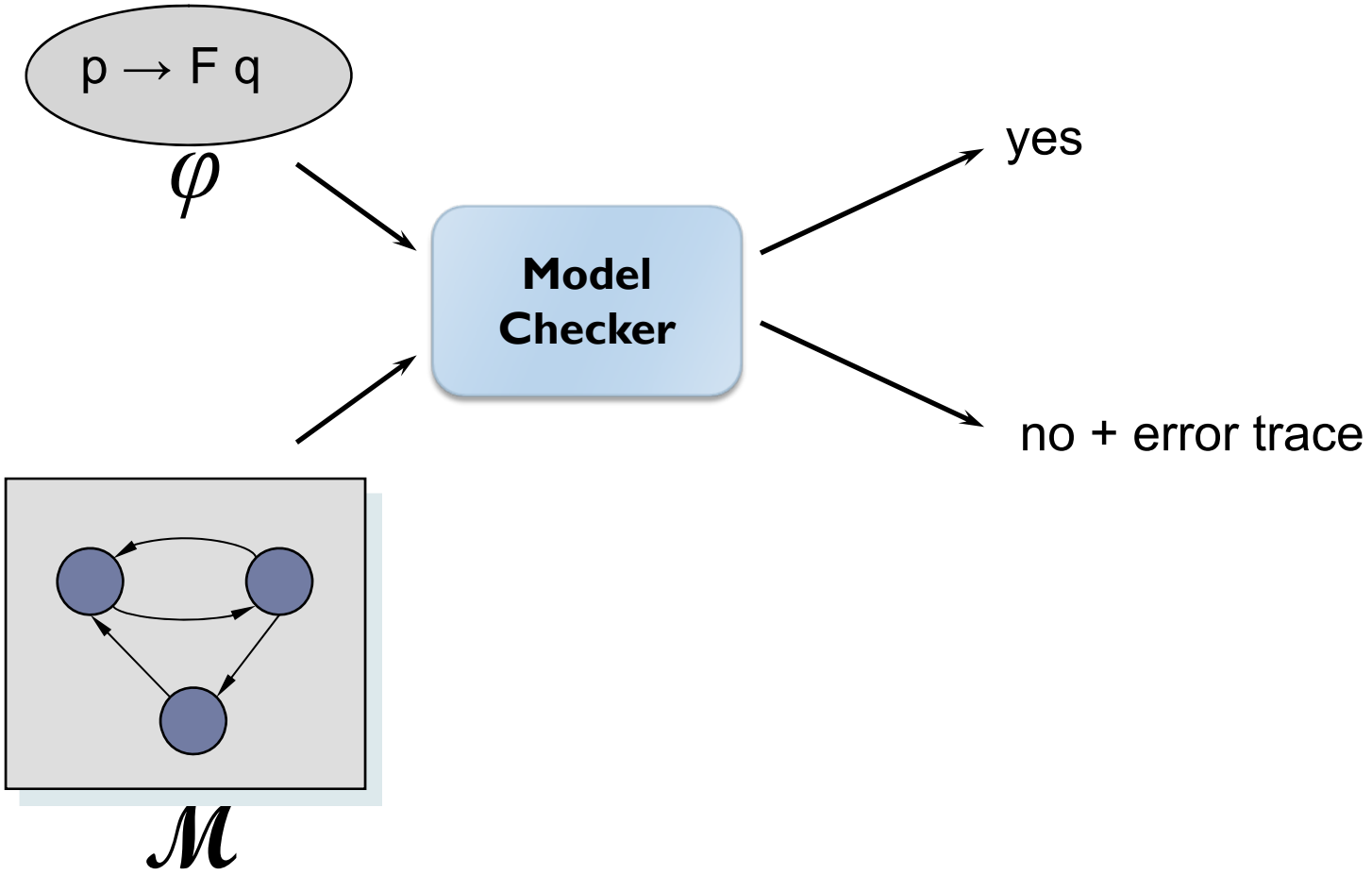
Homework 3 Assignment II



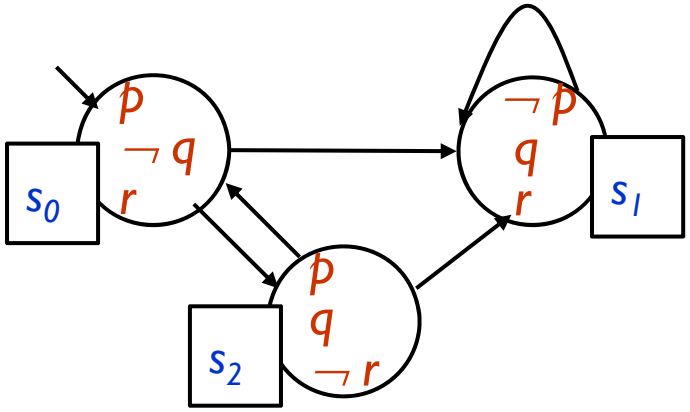
Last Time

- Introduction to model checking
  - Motivation: hunting bugs in software, hardware, protocols,…
- Basic model checking flow
- Kripke structures
- Computation Tree Logic (CTL)

Model Checker



Kripke Structures

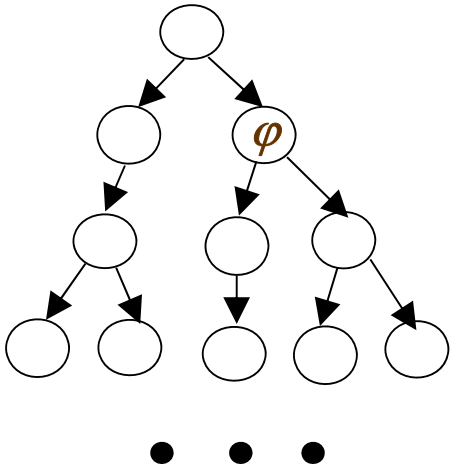


- Conventional state machines
  - $M = \langle S, A, s_0, I, R \rangle$
  - S is a (finite) set of states
  - A is a (finite) set of propositional variables
  - s_0 is a unique initial state ($s_0 \in S$)
  - $I: S \rightarrow 2^A$ is a labeling function that maps each state to the set of propositional variables that hold in it
  - $R \subseteq S \times S$ is a (total) transition relation

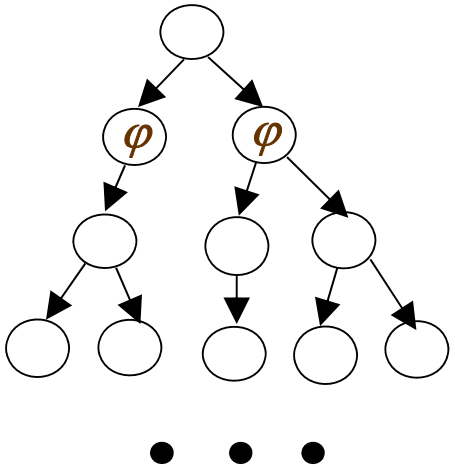
Syntax of CTL

- Propositional temporal logic
  - Allows explicit quantification over possible futures
- Syntax:
  - *True* (T) and *False* ($\perp$) are CTL formulae;
  - propositional variables are CTL formulae;
  - if ϕ and ψ are CTL formulae, then so are: $\neg \phi$, $\phi \wedge \psi$, $\phi \vee \psi$
  - $EX\ \phi$ - ϕ holds in some next states;
  - $EF\ \phi$ - along some path, ϕ is true in a future state;
  - $E[\phi\ U\ \psi]$ - along some path, ϕ holds until ψ holds;
  - $EG\ \phi$ - along some path, ϕ holds in every state
  - Universal quantification: $AX\ \phi$, $AF\ \phi$, $A[\phi\ U\ \psi]$, $AG\ \phi$

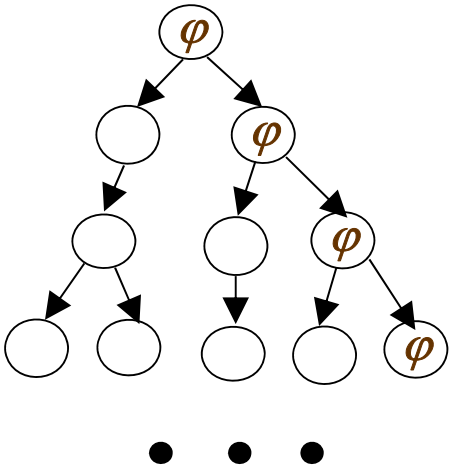
Temporal Operators I



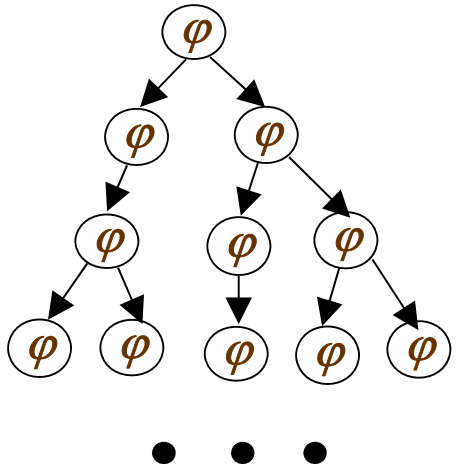
EX (exists next)



AX (all next)

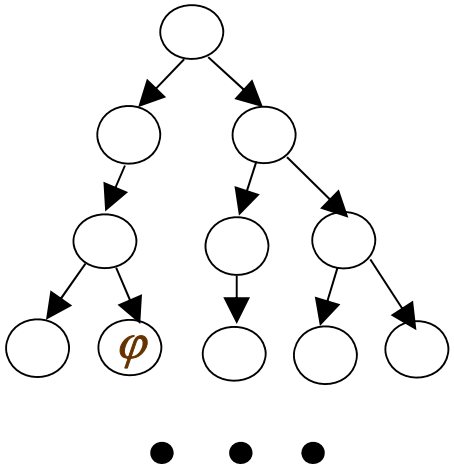


EG (exists global)

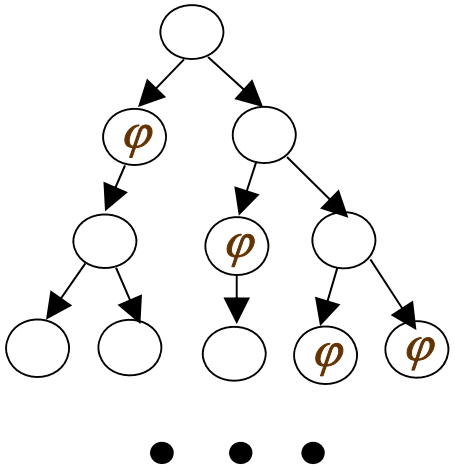


AG (all global)

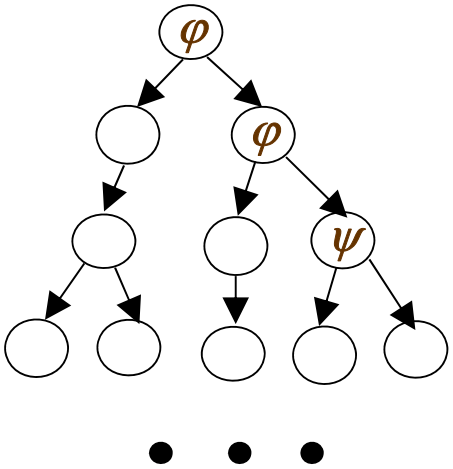
Temporal Operators II



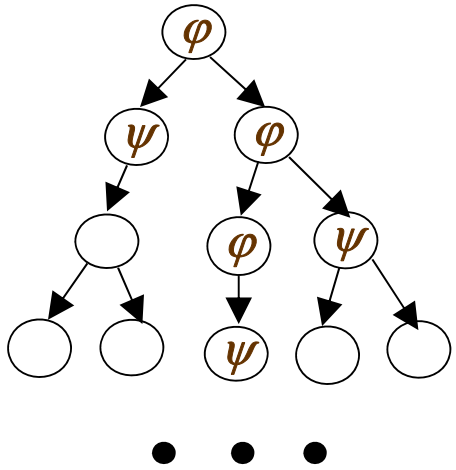
EF (exists future)



AF (all future)



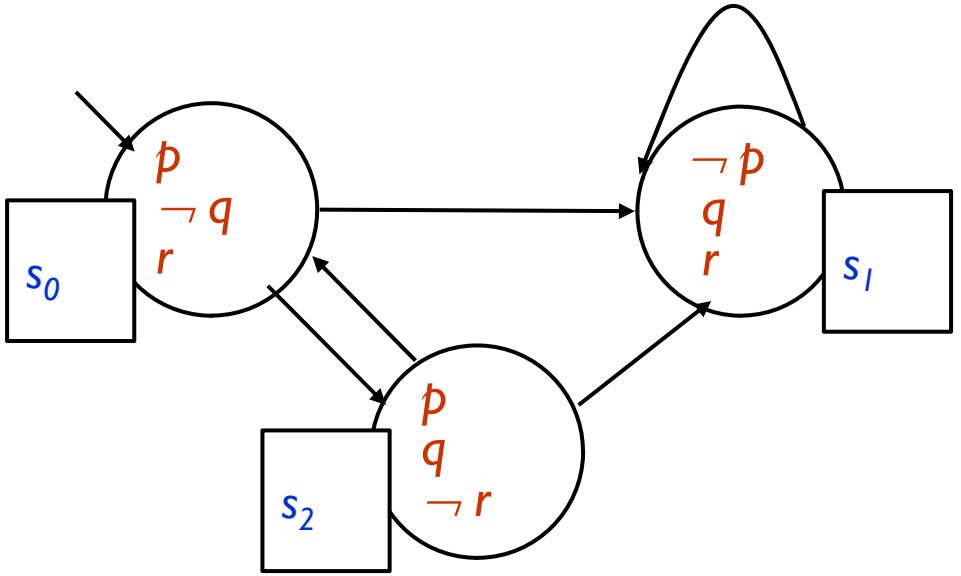
EU (exists until)



AU (all until)

CTL Formula Examples

- Properties that hold:
  - $(EX\ p)(s_0)$
  - $(A[p\ U\ q])(s_0)$
  - $(\ EX\ AF\ p\ )(s_0)$
- Properties that fail:
  - $(A[\neg p\ U\ q])(s_0)$



This Time

- Model checking algorithm for CTL
- NuSMV model checker

Simple NuSMV Example

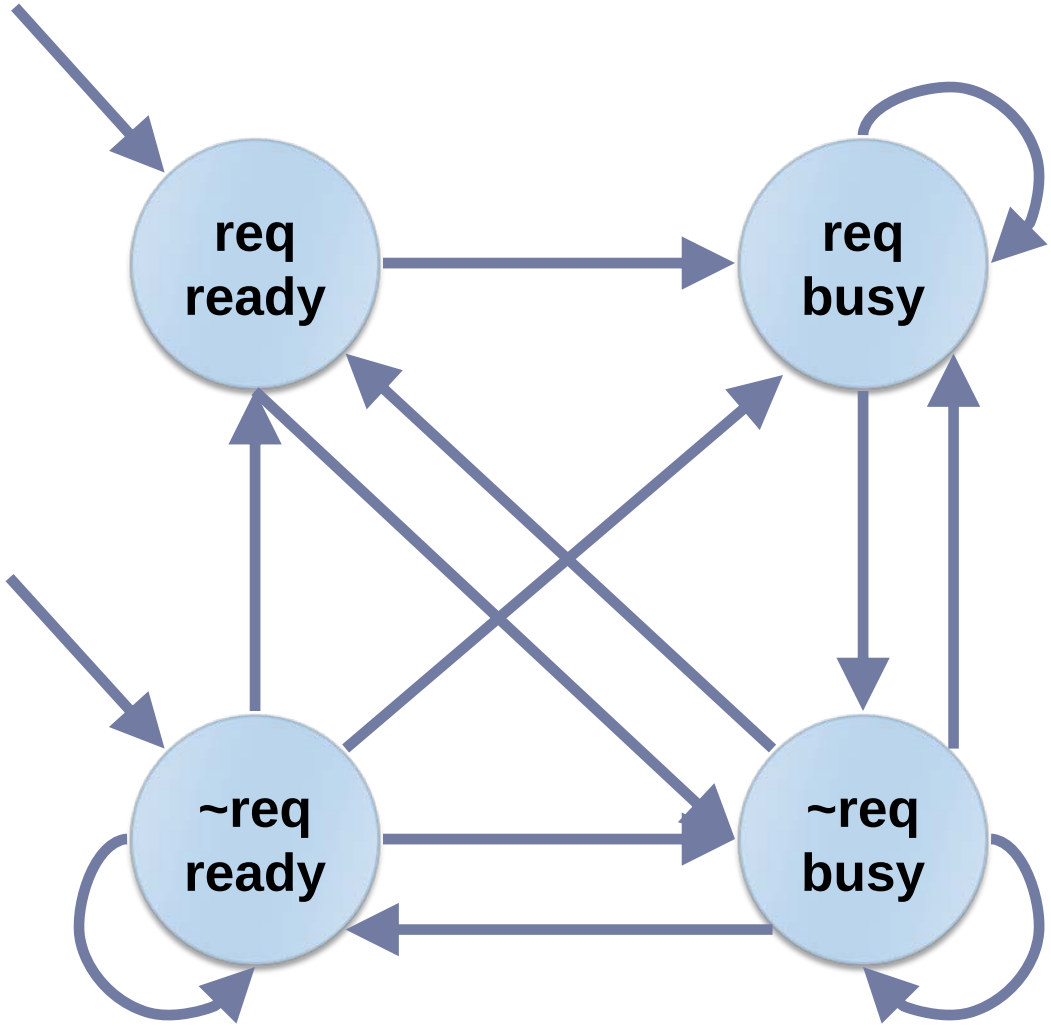
```
MODULE main

VAR
  request: boolean;
  state: {ready, busy};

ASSIGN
  init(request) := {TRUE, FALSE};
  init(state) := ready;
  next(request) := {TRUE, FALSE};
  next(state) :=
    case
      request : busy;
      TRUE : {ready, busy};
    esac;

SPEC AG(request -> AF(state = busy))
```

Model for Simple NuSMV Example



CTL Model Checking

- Problem: Determine whether formula *f* is true in a finite structure *M*
- Algorithm overview
  1. Create a parse tree for *f*
     - A subformula of a CTL formula *f* is any formula *g* whose parse tree is a subtree of *f*'s parse tree
  2. Label the states of *M* with the subformulas of *f* that are satisfied there and work outwards towards *f*
  3. If starting state s_0 is labeled with *f*, then *f* holds on *M*

Example Parse Tree

f: AG(request -> AF(state = busy))

Labeling Algorithm

- Suppose g is a subformula of f and states satisfying all the immediate subformulas of g have already been labeled.
- We want to determine which states to label with g.
- If g is:
  - $\bot$: then no states are labeled with $\bot$
  - p (prop. formula): label s with p if p $\in$ I(s)
  - $g_1 \wedge g_2$: label s with $g_1 \wedge g_2$ if s is already labeled both with g_1 and g_2
  - $\neg g_1$: label s with $\neg g_1$ if s is not already labeled with g_1

Labeling Algorithm for AX

- AX g_1: label any state with AX g_1 if all of its successors are labeled with g_1

Labeling Algorithm for EX

- EX g_1: label any state with EX g_1 if one of its successors is labeled with g_1

Labeling Algorithm for AG

- AG g_1:
  - If a state s is labeled with g_1, label it with AG g_1
  - Repeat until there is no change:
    delete label AG g_1 from a state if one of its successors is not labeled with AG g_1

Labeling Algorithm for EG

- EG g_1:
  - If a state s is labeled with g_1, label it with EG g_1
  - Repeat until there is no change:
    delete the label EG g_1 from a state if none of its successors is labeled with EG g_1

Labeling Algorithm for AF

- AF g_1:
  - If a state s is labeled with g_1, label it with AF g_1
  - Repeat until there is no change (fix-point): label a state with AF g_1 if all successor states are labeled with AF g_1

Labeling Algorithm for EF

- EF g_1:
  - If a state s is labeled with g_1, label it with EF g_1
  - Repeat until there is no change (fix-point): label a state with EF g_1 if at least one of its successors is labeled with EF g_1

Labeling Algorithm for AU

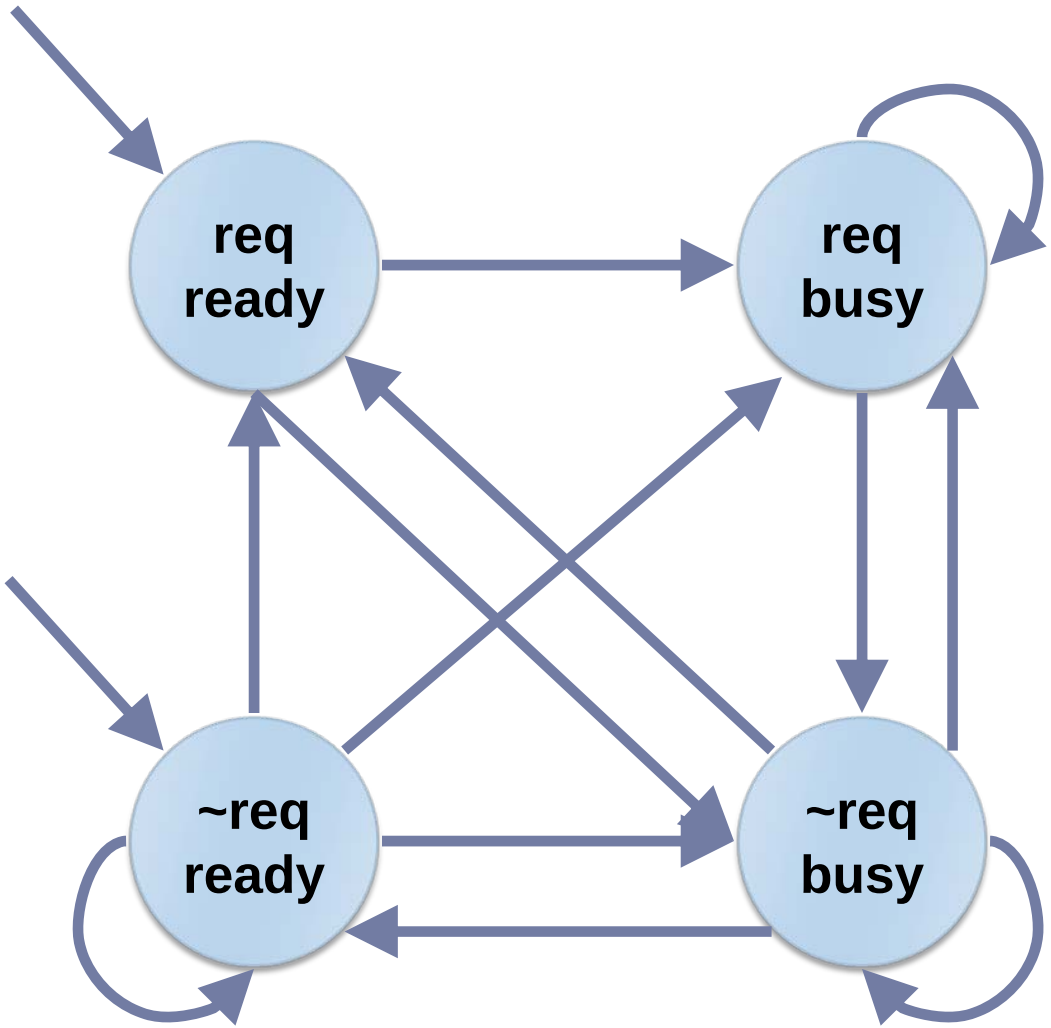
- A[g_1 U g_2]:
  - If a state s is labeled with g_2, label it with A[g_1 U g_2]
  - Repeat until there is no change:
    label any state with A[g_1 U g_2] if it is labeled with g_1
    and all of its successors are labeled with A[g_1 U g_2]

Labeling Algorithm for EU

- E[g_1 U g_2]:
  - If a state s is labeled with g_2, label it with E[g_1 U g_2]
  - Repeat until there is no change:
    label any state with E[g_1 U g_2] if it is labeled with g_1 and at least one of its successors is labeled with E[g_1 U g_2]

Model Checking Example

f: `AG(request -> AF(state = busy))`



Complexity

- Polynomial time with respect to the size of the input Kripke structure and CTL formula
- However, Kripke structure size is exponential in the number of propositional variables!
  - State explosion problem

Next Time

- Symbolic model checking