Lecture 16

Model Checking CTL II

Zvonimir Rakamarić
University of Utah

Announcements

- Traveling next week
  - April 1 and April 3 classes are cancelled
- Project nne-on-nnes on April 8
- Guest lecture by Prof. Ganesh Gopalakrishnan on April 10

Homework Assignment 4

- 3 cannibals and 3 missionaries are on the left side of a river. There is 1 boat that can carry two people. (The boat of course needs to be ferried by at least one person.) If at any point, there are more cannibals than missionaries on one bank, the cannibals eat the missionaries.
- Construct a model of this problem in NuSMV and then reduce the question "can all 3 missionaries and 3 cannibals safely cross to the right side?" to model checking problem by stating the question as an CTL property.

Skeleton of my Solution

```
VAR
  m_onleft : 0 .. 3;
  c_onleft : 0 .. 3;
  m_inboat : 0 .. 2;
  c_inboat : 0 .. 2;
  boatonleft : boolean ;

DEFINE
  m_onright := 3 - m_onleft;
  c_onright := 3 - c_onleft;
  boatonright := !boatonleft;

INVAR ...
INVAR ...

ASSIGN ...

SPEC AG !(m_onright = 3 & c_onright = 3)
```

Last Time

- Reviewing Computation Tree Logic (CTL)
- NuSMV model checker

Simple NuSMV Example

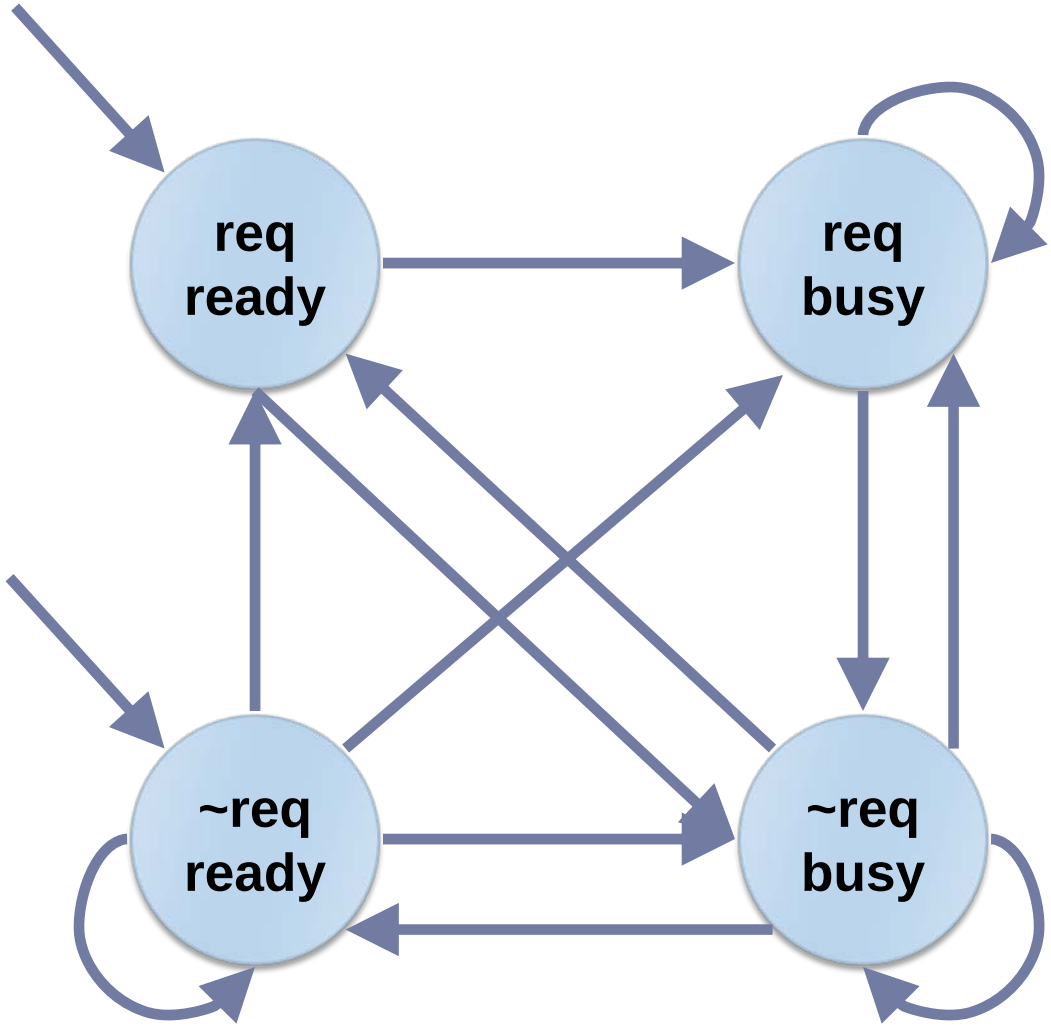
```
MODULE main

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



This Time

- Model checking algorithm for CTL

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

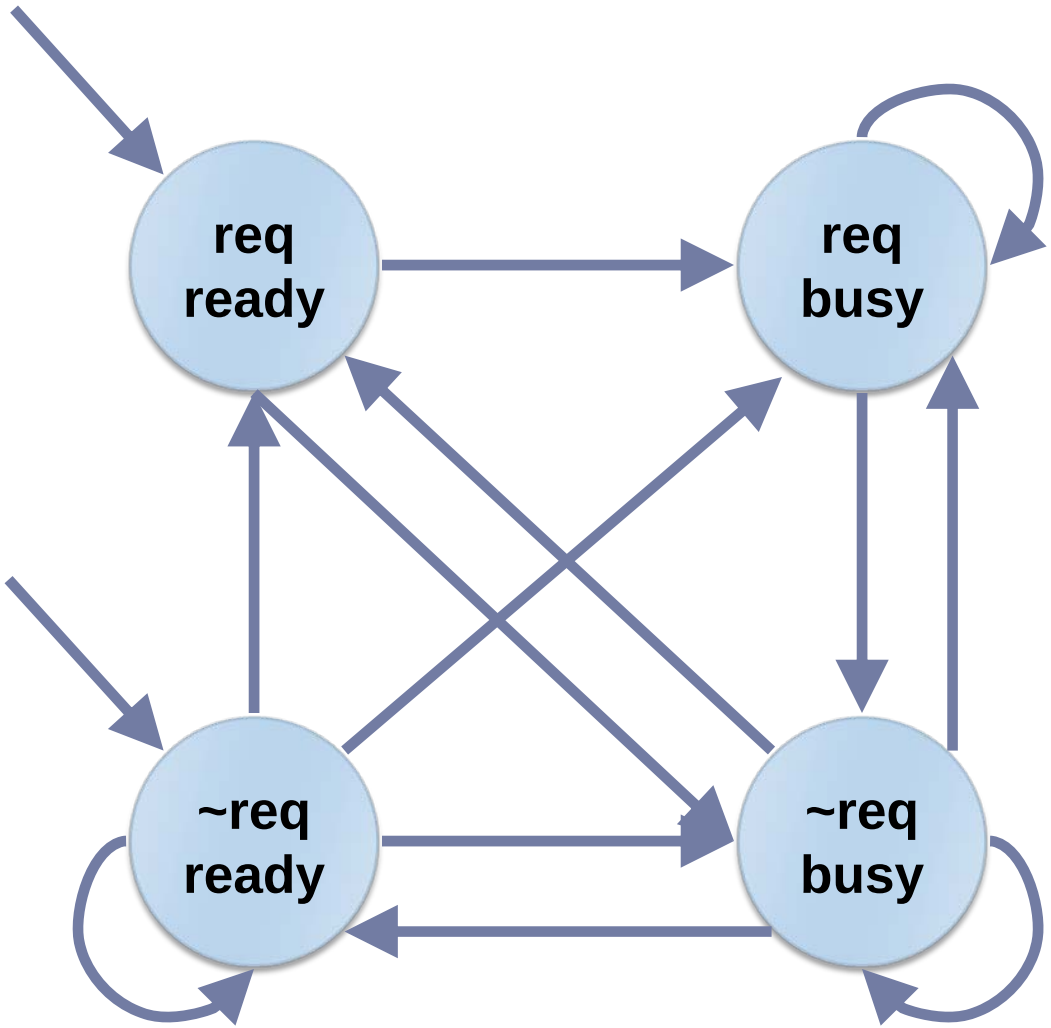
- A[g_1 U g_2]:
  - If a state s is labeled with g_2, label it with A[g_1 U g_2]
  - Repeat until there is no change:
    label any state with A[g_1 U g_2] if it is labeled with g_1
    and all of its successors are labeled with A[g_1 U g_2]

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