Lecture 17

Symbolic Model Checking I

Zvonimir Rakamarić
University of Utah

Announcements

- Homework 4 is due tonight
- Project presentations on Apr 24 in class
  - I will post the schedule this week
- Final project report due on Apr 29

Last 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*

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]

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

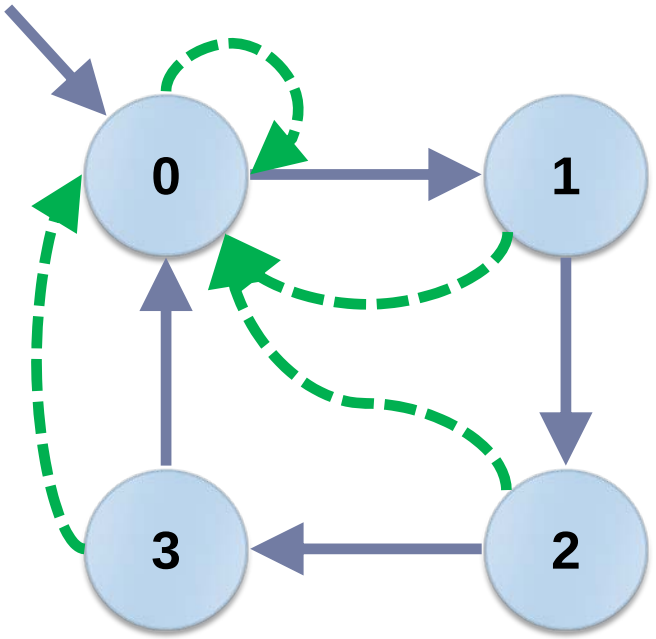
This Time

- Simple NuSMV exercise
- Symbolic model checking

NuSMV Exercise

- Modulo 4 counter with reset
  - The counter can be reset by an external reset signal

```
MODULE main
VAR
   b0 : boolean;
   b1 : boolean;
   reset : boolean;
   out : 0..3;
ASSIGN
   …
SPEC EF out = 3
```



Symbolic Model Checking

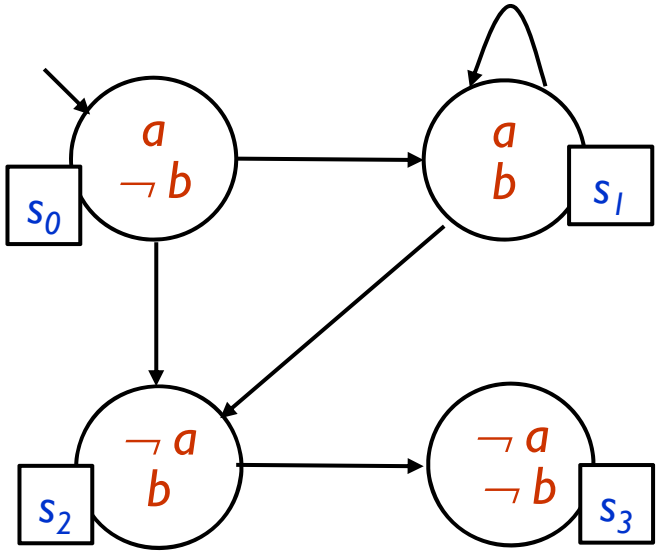
- Why?
  - Saves us from constructing a model's state space explicitly
  - Effective ”cure” for state space explosion problem
- How?
  - Sets of states and the transition relation are represented by formulas
  - Set operations are defined in terms of formula manipulations
- Data structures used
  - ROBDDs – allow for efficient storage and manipulation of logic formulas

Sets as Boolean Formulas

- Every finite set can be represented as a Boolean formula
- How?
  - Suppose the set has N elements
  - Each element is encoded as a string of at least log N bits
  - Characteristic Boolean formula is the one whose satisfying assignments are those strings
  - Empty set is simply “False”
- Every finite set can be represented as a BDD

Representing Models Symbolically I

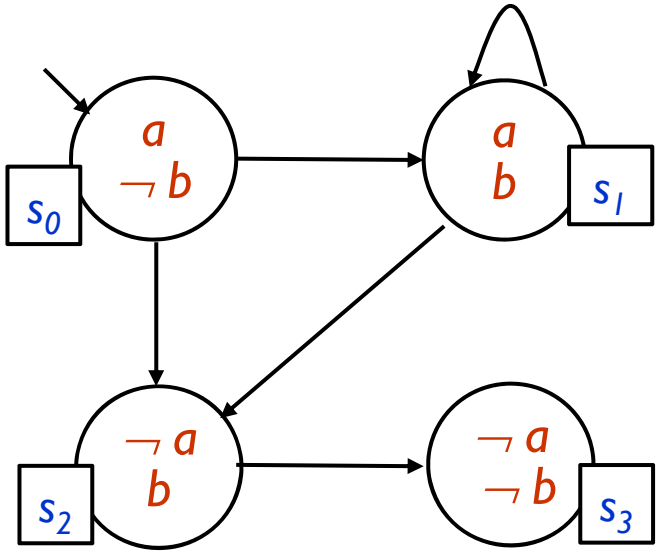
- A system state represents an interpretation (truth assignment) for a set of propositional variables *V*



- Formulas represent sets of states that satisfy it
  - False – ∅, True – *S*
  - a – set of states in which a is true – ({s_0, s_1})
  - b – set of states in which b is true – ({s_1, s_2})
  - a ∨ b = {s_0, s_1} ∪ {s_1, s_2} = {s_0, s_1, s_2}

Representing Models Symbolically II

- State transitions are described by relations over two sets of variables
  - V – source state
  - V' – destination state



- Transition from s_2 to s_3 is described by $\neg a \wedge b \wedge \neg a' \wedge \neg b'$
- Transition from s_0 to s_1 and s_2, and from s_1 to s_2 and to itself is described by $a \wedge b'$
- Then the whole state relation R is described by $(a \wedge b') \vee (\neg a \wedge b \wedge \neg a' \wedge \neg b')$

Sets of States and Transitions

- Set of states
  - Each state s is bit-string comprising values of state variables
- Set of transitions
  - Transition is a state pair (s, s')
  - View the pair as a combined bit-string
- From now, we will view sets of states and the transition relation *R* as Boolean formulas over vector of current state variables v and next state variables v' – S(v), R(v, v')

Model Checking using Sets of States I

Computing $||\varphi||$

- φ is $\top$: return S
- φ is $\bot$: return $\emptyset$
- φ is atomic : return $\{s \in S \mid \varphi \in L(s)\}$
- φ is $\neg\varphi_1$: return $S \setminus ||\varphi_1||$
- φ is $\varphi_1 \wedge \varphi_2$: return $||\varphi_1|| \cap ||\varphi_2||$
- φ is $\varphi_1 \vee \varphi_2$: return $||\varphi_1|| \cup ||\varphi_2||$

Model Checking using Sets of States II

Computing $||\varphi||$

- φ is AX φ_1 : return $||\neg EX \neg \varphi||$
- φ is EX φ_1 : return $SAT_{EX}(\varphi_1)$
- φ is EU φ_1 : return $SAT_{EU}(\varphi_1)$
- φ is EG φ_1 : return $SAT_{EG}(\varphi_1)$

Model Checking using Sets of States III

- function $SAT_{EX}(\varphi)$:
  return $\{s \in S \mid s \rightarrow s_1 \text{ for some } s_1 \in ||\varphi||\}$

Using relations:

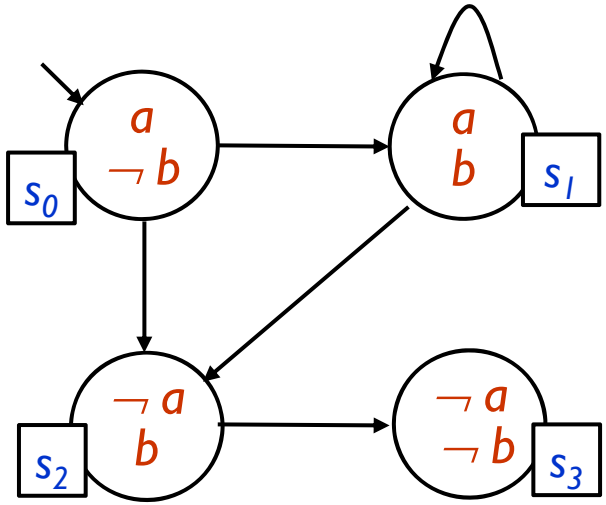
return $\{s \mid \exists s' \in ||\varphi|| \wedge (s, s') \in R\}$

Using boolean formulas:

return $\exists s' \,.\, \varphi' \wedge R(s, s')$

where $\exists x.\varphi = \varphi|_{x \leftarrow 0} \vee \varphi|_{x \leftarrow 1}$

Simple Example



R = (a∧b′)∨(¬a∧b∧¬a′∧¬b′)

||EX b||

= ∃a′,b′.*R*∧b′

= ∃a′,b′.((a∧b′)∨(¬a∧b∧¬a′ ∧¬b′))∧b′

= ∃a′,b′.((a∧b′)∧b′)∨((¬a∧b∧¬a′ ∧¬b′)∧b′)

= ∃a′,b′.(a∧b′)∨0

= ∃a′,b′.(a∧b′)

= ∃b′.(a∧b′)

= (a∧1)∨(a∧0)

= a

That is, ||EX b|| is true in state s iff s |= a.

Next Time

- More on symbolic model checking