Lecture 18

Symbolic Model Checking II

Zvonimir Rakamarić
University of Utah

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

Quantified Boolean Formulas

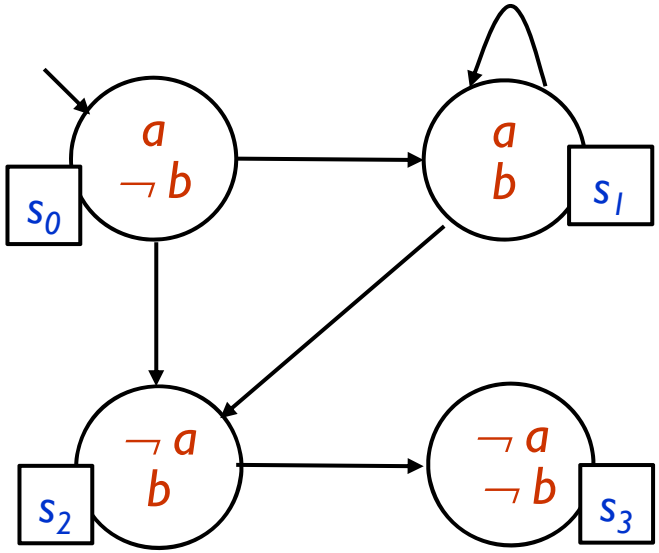
- Let F denote a Boolean formula, and v denote one or more Boolean variables
- A quantified Boolean formula φ is obtained as: $\varphi ::= F \mid \exists v.\varphi \mid \forall v.\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$
- Quantifiers $\exists v.\varphi$ and $\forall v.\varphi$ can be expressed in terms of φ's cofactors and standard boolean operators

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

  $\forall v.\varphi = \varphi|_{v \leftarrow 0} \wedge \varphi|_{v \leftarrow 1}$

Representing Models Symbolically I

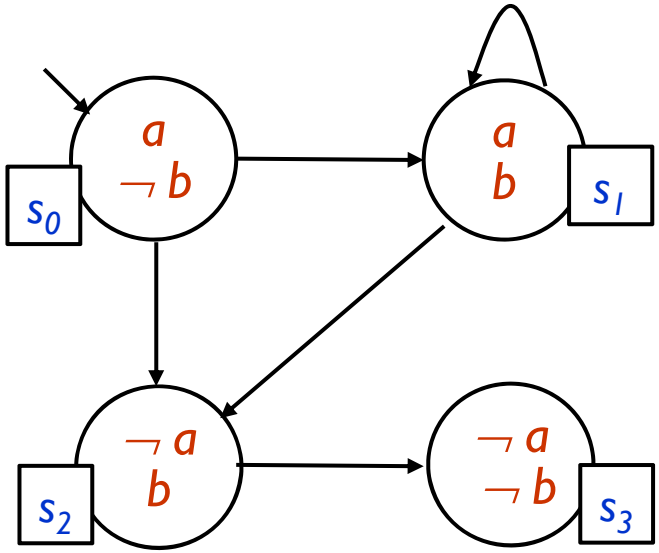
- A system state represents an interpretation (truth assignment) for a set of propositional variables *V*



- 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 $SAT_{EG}(\varphi_1)$

Symbolic Model Checking EX p

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

Using relations:

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

Using Boolean formulas:

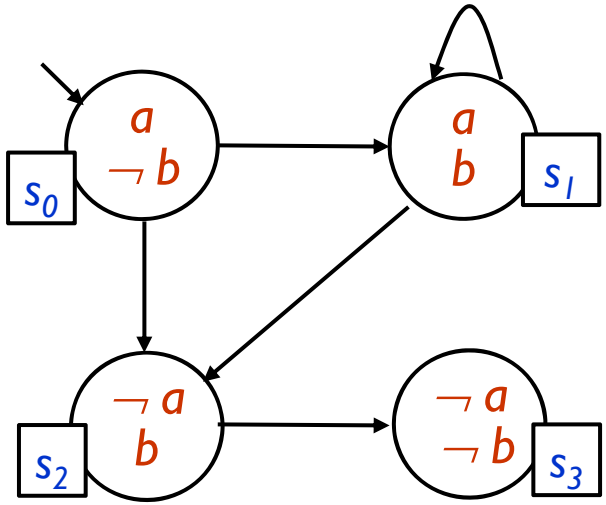
return $\exists s' \,.\, p' \wedge R(s, s')$

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

Simple Example

$R = (a \wedge b') \vee (\neg a \wedge b \wedge \neg a' \wedge \neg b')$

$||EX\ b|| = ???$



Symbolic Model Checking AG p

- Given: Set of initial states S_0, transition relation R
- Check property AG p
- How symbolic model checking does this:
  - Compute S_0, S_1, S_2,... where S_i is the set of states reachable from some initial state in at most i steps
    - When do we stop?
  - After computing each S_i, check whether any element of S_i satisfies ¬p
    - How?

Reachability Analysis

- The process of computing the set of states reachable from some initial state in 0 or more steps
- Often characterized as checking (AG true)
- The resulting set is called “reachable set” or “set of reachable states”

Implementing Reachability Analysis

- How is S_i related to S_{i+1}?
- $v \in S_{i+1}$ iff $v \in S_i$ or there is a state $x \in S_i$ such that R(x,v)
- $S_{i+1}(v) = S_i(v) \vee \exists x \{S_i(x) \wedge R(x,v)\}$
- $S_{i+1}(v) = S_i(v) \vee (\exists v \{S_i(v) \wedge R(v,v')\})[v/v']$
  - F[x/y] means that we substitute y with x in F

Implementing Reachability Analysis

```
i := 0;
do {
  i++;
  S_i(v) = S_{i-1}(v) ∨ (∃v{S_{i-1}(v) ∧ R(v,v')})[v/v']
} while (S_i(v) != S_{i-1}(v))
```

$S_i(v)$ is the set of reachable states

BDD Issues

- Remember that S_i and R are represented as BDDs
- How large they grow determines the space and time usage of the algorithm

Backward Reachability Analysis

- Suppose we want to verify AG p
- The formula ¬p characterizes all error states
- We can search backwards for a path to an error state from some initial state
  - $E_0 = \neg p$
  - Compute E_0, E_1, E_2, … as states reachable from the error states in at most 0, 1, 2,… steps going backwards
  - How to express E_{i+1} in terms of E_i?

    $E_{i+1}(v) = E_i(v) \vee (\exists v \{E_i(v) \wedge R(v',v)\})[v/v']$
- Why would we want to do backwards reachability analysis? Is it always better?

Verification of AG p

- Using forward reachability analysis:
  - Check if some $S_i \wedge \neg p$ is true
- Using backward reachability analysis:
  - Set $E_0 = \neg p$
  - Check if $E_k \wedge S_0$ is true for any k

Next Time

- More on symbolic model checking
- Summary of topics we covered this term