CS206 Lecture 2x
CTL Model Checking

G. Sivakumar
Computer Science Department
IIT Bombay
siva@iitb.ac.in
http://www.cse.iitb.ac.in/~siva

Thu, Apr 10, 2003

Plan for Lecture 2x

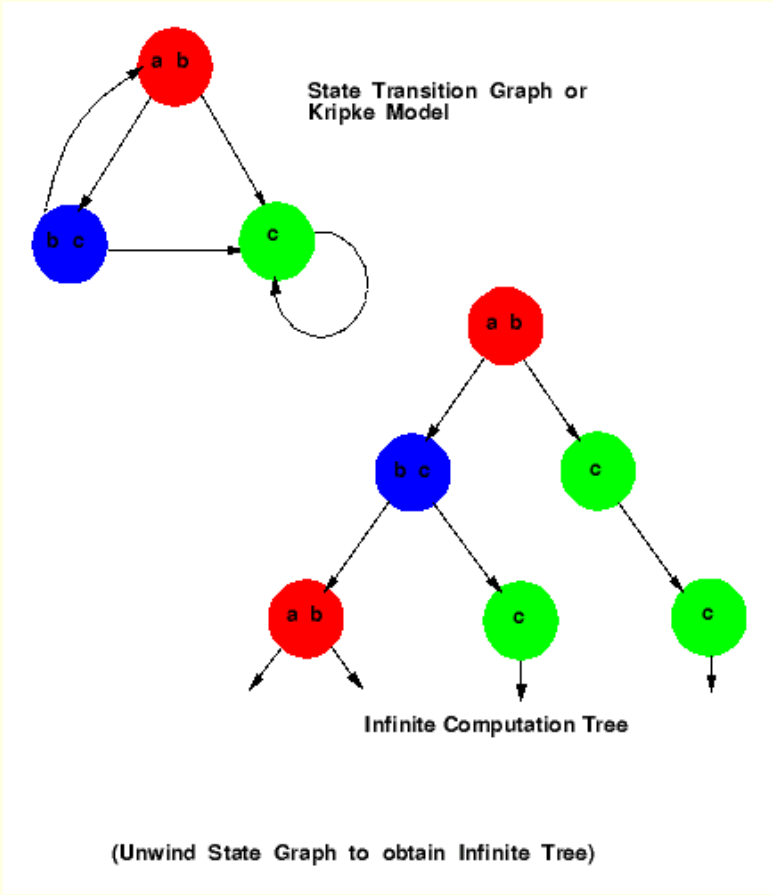
- Reducing Number of Operators
- Fixpoint Computation
- Simple Algorithm

Model Checking

- Input: Kripke structure M and CTL formula.
- Output: set of states where holds. (Can derive the desired yes/no answer.)
- Idea: label graph with subformulae known to be true, starting with smallest.
- Basically a fairly simple graph algorithm.

Note: M has finite number of states.

Example



In which states are the following true?

- **EG c**
- **EG EF a**

Functionally Complete Subset

All CTL formulas can be converted to equivalent one involving only

$$0, \neg, \wedge, \mathbf{AF}, \mathbf{EU}, \mathbf{EX}$$

- $1 = \neg 0$
- $\vee = \ldots$
- $\mathbf{AX\Phi} = \neg\mathbf{EX}\neg\mathbf{\Phi}$
- $\mathbf{EG\Phi} = \ldots$
- $\mathbf{EF\Phi} = \ldots$
- $\mathbf{AG\Phi} = \ldots$
- $\mathbf{A}(\mathbf{\Phi} \bigcup \mathbf{\Psi}) = \ldots$

How to Model Check

Simplistic Algorithm explained in 2 different ways. Given Kripke structure M and CTL formula Φ.

- Recursively check the formula.

  Check smallest subformula (atomic propositons), then slightly bigger formula ... finally full formula.

  Main Data Structure: For every subformula f use the set $Sat(f)$ to denote the states in which f holds.

- Recursively label the states.

  For every state s in the model, use the set $Label(s)$ to denote the subformales of Φ which hold in that state.

Model checking CTL: Explanation 1

How to check whether state s satisfies Φ?

- compute recursively the set $Sat(\Phi)$ of of states that satisfy Φ.
- check whether state s belongs to $Sat(\Phi)$

Recursive computation

- Determine the sub-formulas of Φ.
- Compute $Sat(p)$ for all atomic propositions p in Φ.
- Then check the smallest sub-formulas that contain each p
- Check the formulas that contain these sub-formulas.
- and so on..... until formula Φ is checked.

Top-level Algorithm for Method 1

- $Sat(p)$ is the set of states labelled with atomic proposition p.
- $Sat(\Phi \vee \Psi)$ is $Sat(\Phi) \cup Sat(\Psi)$
- $Sat(\neg\Phi)$ is $S - Sat(\Phi)$ (complement)
- $Sat(\mathbf{EX}\Phi)$ is the set of states that can directly move to $Sat(\Phi)$.

  That is, $s_1 \in Sat(\mathbf{EX}\Phi)$ if $s_2 \in Sat(\Phi)$ and from s_1 we can go to s_2 in 1-step in the reachability relation of M.
- $\mathbf{EU}$ (next slide)
- $\mathbf{AF}$ (homework)

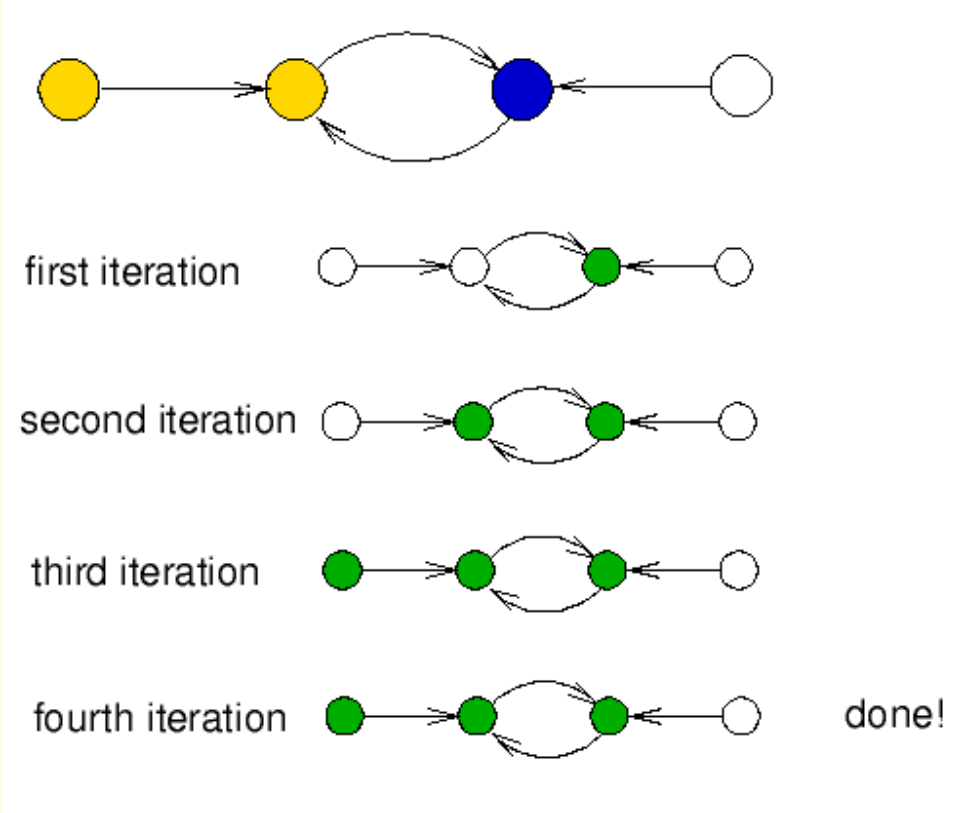
Checking Until Formula

Example of **Fixpoint** computation (you will see this often in CS later!)
Let $f = \mathbf{E}(\Phi \bigcup \Psi)$. $Sat(f)$ is computed as follows.

- $Sat^0(f) = Sat(\Psi)$
- $Sat^1(f) = Sat^0(f) \cup \{\Phi-$states from where we can move to $Sat^0(f)\}$
  That is, $s_1 \in Sat^1(f)$ if $s_1 \in Sat^0(f)$ or Φ holds in s_1 and there is $s_2 \in Sat^0(f)$ and from s_1 we can reach s_2 in 1-step.
- ...
- $Sat^{i+1}(f) = Sat^i(f) \cup \{\Phi-$states from where we can move to $Sat^i(f)\}$
- ...
- until $Sat^{i+1}(f) = Sat^i(f)$ fixpoint!

Example

Compute $Sat(\mathbf{E}(yellow \bigcup blue))$



Method 2: Top-level Algorithm

ModelCheck(Φ ,M):

- Translate Φ so it mentions only $\mathbf{0}, \neg, , \mathbf{AF}, \mathbf{EU}, \mathbf{EX}$ and variables.
- For each state s of M, initialize Label(s) to be the empty set.
  Label(s) is the set of subformulae of known to be true in state s.
- Let F be a list of all subformulae of Φ sorted in nondecreasing order of size.
- For each $f \in F$, call the procedure $addLabel(f)$.
- Return the set of all s such that $\Phi \in Label(s)$.

How to $addLabel$? (homework).