# Model Checking

Abhik Roychoudhury
CS 5219
National University of Singapore



# Newsflash

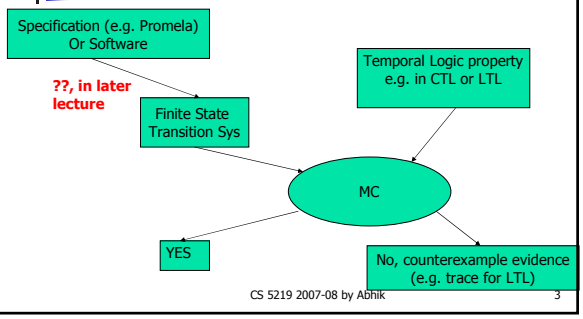
- Turing Award 2007 (announced Feb 08)
  - Clarke, Emerson, Sifakis
    - For their role in developing Model-Checking into a highly effective verification technology, widely adopted in the hardware and software industries.

http://awards.acm.org/homepage.cfm?srt=all&awd=140

**Award in 1996 for temporal logics to Amir Pnueli**



# Model Checking



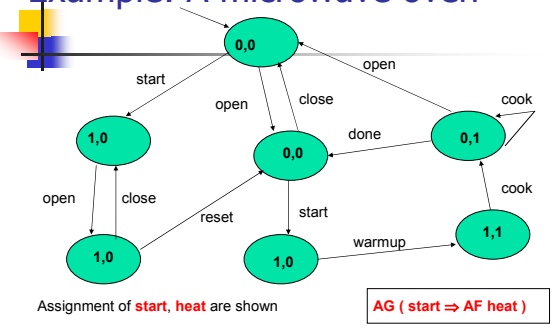



# CTL Model Checking

- Inputs:
  - Finite state Kripke Structure M = (S,I, $\rightarrow$, L)
    - S (set of states),
    - I $\subseteq$ S (set of initial states),
    - $\rightarrow$ (transition relation),
    - L (function labeling atomic propositions to states)
  - CTL formula $\varphi$ (with atomic propositions corresponding to those appearing in M)
- Output:
  - Whether for all s in I, we have M,s |= $\varphi$



# Example: A microwave oven

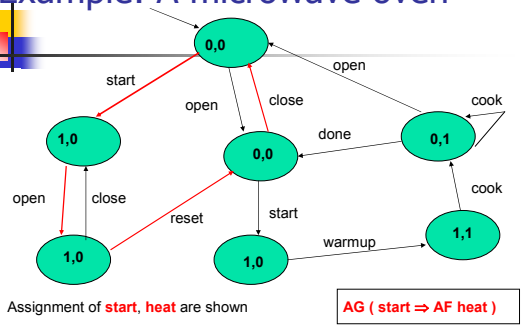





# Example

- AG (start $\Rightarrow$ AF heat)
  - For any reachable state, if start holds, then along all outgoing paths, heat eventually holds.
  - Violated if:
    - $\exists$ a reachable state s where start holds
    - $\exists$ an acyclic path from s to s' in which heat does not hold in any state
    - And there is a cycle containing s' such that heat does not hold in all states of the cycle.
- Can we find such a cycle in the given model ?

## Example: A microwave oven




Assignment of **start**, **heat** are shown

**AG ( start ⇒ AF heat )**



## CTL Model Checking

- A systematic way of doing the reasoning performed in our example for all CTL formulae.
- Recall set of all CTL formulae
- φ = Prop | ¬ φ | φ ∧ φ | AX φ | EX φ | AF φ | EF φ
- | EG φ | AG φ | E(φ U φ ) | A(φ U φ )
- | E (φ R φ) | A (φ R φ)
- Among the temporal operators, only consider
  - EX, EG, EU (along with ¬ and ∧)



## CTL temporal operators

- AX φ = ¬¬AXφ = ¬EX ¬φ
- AGφ = ¬¬AGφ = ¬EF ¬φ
- EFφ = E ( true U φ )
- AFφ = ¬EG ¬φ
- A(φ R Ψ ) = ¬¬ A(φ R Ψ ) = ¬ E (¬ φ U ¬ Ψ)
- A(φ U Ψ ) = ¬ E (¬ φ R ¬ Ψ)
- What about E (φ RΨ ) ??



## Model Checking Algorithm

- Pre-processing: Rewrite the CTL formula to be verified to contain EX,EG,EU,∧, ¬
- For all sub-formulae x of the re-written formula φ, find the set of states satisfying x in the given model M.
- The above step finally computes the set of states in M which satisfy φ, call it St(φ).
- Check whether all initial states of M are contained in St (φ).



## Word of caution

- The pre-processing of the formula is only being done to simplify the presentation of the MC algorithm
  - Reduce # of cases to consider.
- We can
  - Develop customized algorithms for each of the 10 CTL operators, and then
  - apply them in a bottom-up recursive fashion as we will be doing now.

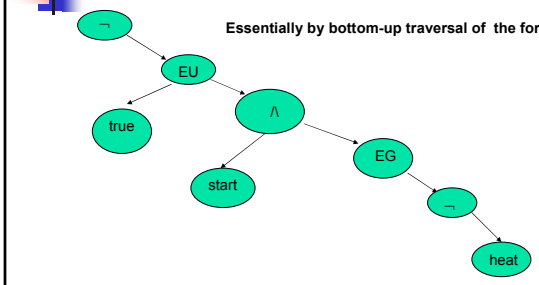


## Example

- M = the model of microwave oven given earlier
- φ = AG (start ⇒ AF heat)
- = ¬ EF ¬( ¬start ∨ AF heat)
- = ¬ EF (start ∧ ¬ AF heat)
- = ¬ EF (start ∧ EG ¬ heat)
- = ¬ E( true U (start ∧ EG ¬ heat))
- Now, how to compute the set of states in M satisfying this horrible formula ?

## Example

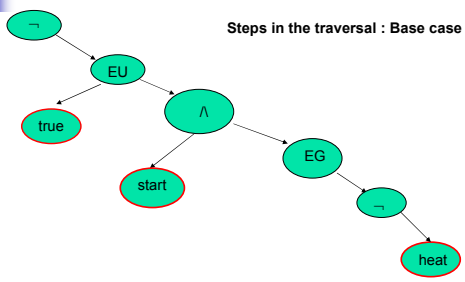
Essentially by bottom-up traversal of the formula






## Example

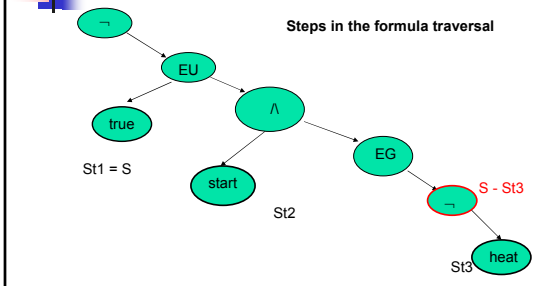
Steps in the traversal : Base case






## Example

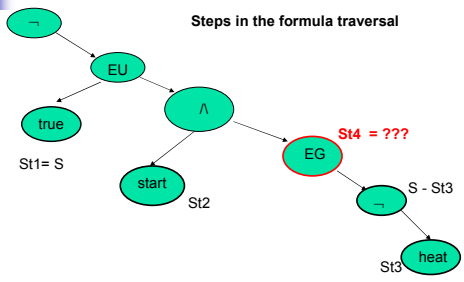
Steps in the formula traversal






## Example

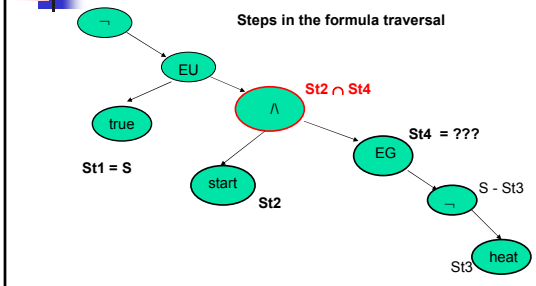
Steps in the formula traversal






## Example

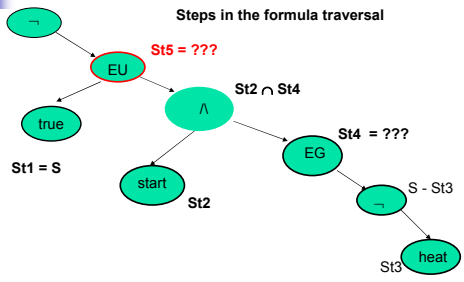
Steps in the formula traversal



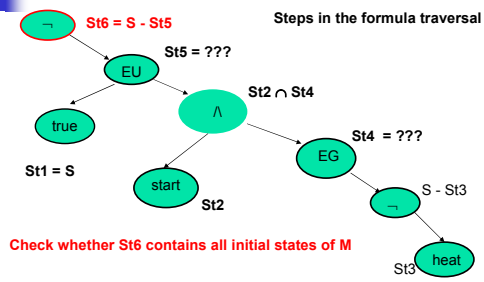



## Example

Steps in the formula traversal



## Example




Check whether St6 contains all initial states of M

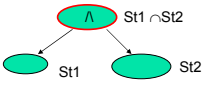


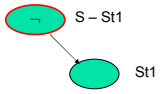
## Bottom-up formula traversal

**Base case**: Atomic propositions


The set of states satisfying p in M = (S,S0,R,L), is found from L, the labeling function of M

**Boolean operators**: ∧, ¬

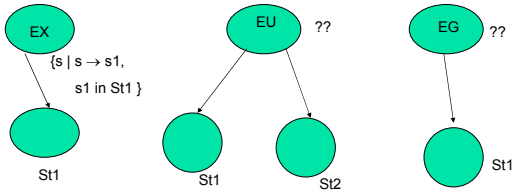





## Questions Remaining

**Temporal operators: EX, EU, EG**



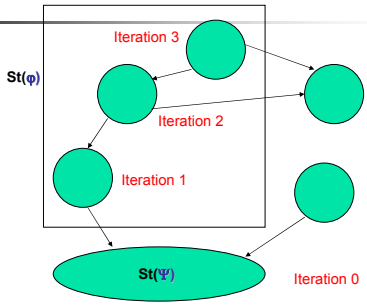



## EU

- Inputs:
  - Kripke Structure M = (S, I, →, L).
  - CTL formulae φ and Ψ.
  - St(φ), set of states satisfying φ in M.
  - St(Ψ), set of states satisfying Ψ in M.
- Output:
  - Set of states satisfying E(φ U Ψ) in M.
- Technique:
  - ***Traversing the states (and transitions) of M.***



## E(φ U Ψ) : Intuition






## E(φ U Ψ) : Algorithm

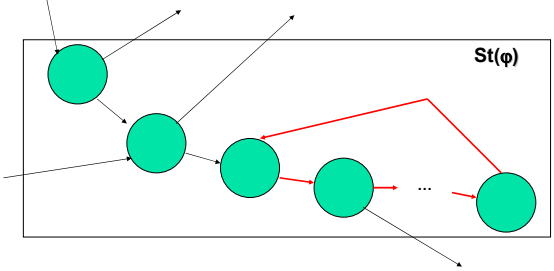
- Result := St(Ψ);
- Temp := St(Ψ);
- while Temp ≠ empty do
- pick s ∈ Temp; Temp := Temp – {s};
- Backstep:= {s1 | s1 → s, and s1 ∈ St(φ)};
- Temp := Temp ∪ Backstep;
- Result := Result ∪ Backstep;
- endwhile;
- return Result;

## EG

- Inputs:
  - Kripke Structure M = (S, I, →, L).
  - CTL formulae φ.
  - St(φ), set of states satisfying φ in M.
- Output:
  - Set of states satisfying EG φ in M.
- Technique:
  - ***Traversing the states (and transitions) of M.***



## EG φ : Intuition






## EG φ : Algorithm

- Result := St(φ);
- **repeat**
- Temp := { s | s ∈ Result , and
- ∀s1. s → s1 ⇒ s1 ∉ Result } ;
- Result := Result – Temp;
- **until** Temp = empty;
- **return** Result;



## How to make it more efficient

- We initialize $St_{EGf} = St_f$
  - For each state in $St_f$, we check the out-edges. Many of the destination states are not in $St_f$, so cannot satisfy EGf
- It suffices to consider a reduced Kripke Structure M' constructed from M such that
  - All states of M which satisfy f are retained.
  - All other states and transitions are deleted.
- For any s, M,s |= EG f if and only if
  - s is a state in M'
  - s reaches a state s' in M' where s' loops back to itself.



## Efficient computation

- Input: M = (S,I,→,L) , $St_f$
- Output: $St_{EGf}$
- Technique:
  - Compute M'=(S',I',→',L') from M by keeping only nodes in $St_f$
  - Temp := $St_{EGf}$ := All nodes in nontrivial SCCs of M'
  - while Temp ≠ empty do
  - pick s ∈ Temp; Temp := Temp – {s};
  - $St_{EGf}$ := $St_{EGf}$ ∪ { t | t →' s ∧ t ∉ $St_{EGf}$ };
  - Temp := Temp ∪ { t | t →' s ∧ t ∉ $St_{EGf}$};
  - endwhile

SCC = Strongly connected component = Maximal sub-graph where every node has a path to every other node.



## Summary, Exercises

- We have only presented model checking as a decision procedure.
- Other issues such as counter-example computation not shown.
- Direct iterative algorithms given only for EU, EG
  - What about EF, AF, AG etc. ?
- Algorithmic complexity of the iterative algorithms discussed in today's lecture.