CTL Model Checking

Abhik Roychoudhury
CS 5219
National University of Singapore

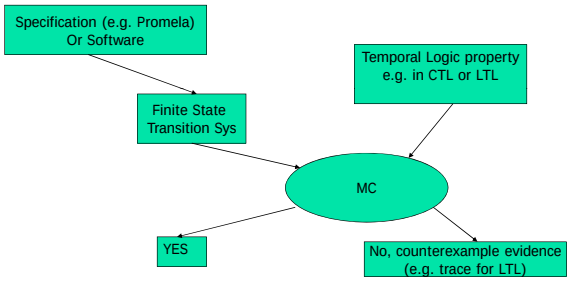
News

- 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

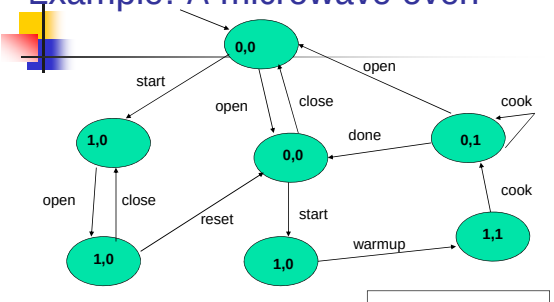
Model Checking



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 φ (with atomic propositions corresponding to those appearing in M)
- Output:
  - Whether for all s in I, we have M,s |= φ

Example: A microwave oven



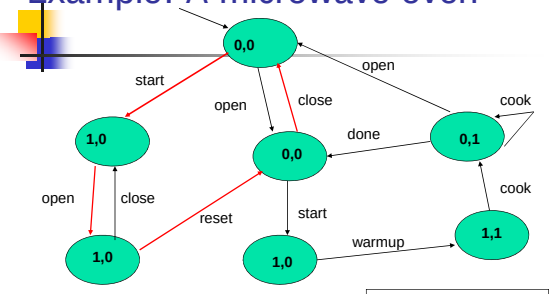
Assignment of **start**, **heat** are shown

AG (start ⇒ AF heat)

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 | $\neg \varphi$ | $\varphi \wedge \varphi$ | 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 $\neg$ and $\wedge$)

CTL temporal operators

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

Model Checking Algorithm

- Pre-processing: Rewrite the CTL formula to be verified to contain EX,EG,EU,$\wedge$, $\neg$
- 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

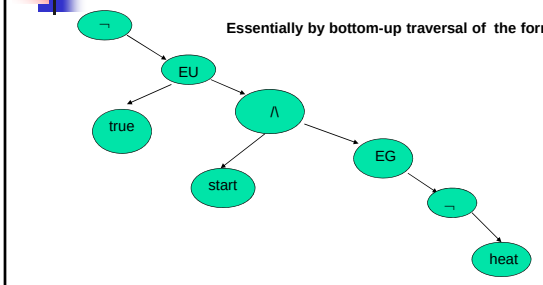
- The pre-processing of the formula is only being done to simplify the presentation of the MC algorithm
  - Reduce number of cases to consider.
- We can
  - Develop customized algorithms for each of the 10 CTL operators, and then
  - apply them in a bottom-up recursive fashion as we will be doing now.

Example

- M = the model of microwave oven given earlier
- φ = AG (start $\Rightarrow$ AF heat)
- = $\neg$ EF $\neg$($\neg$start $\vee$ AF heat)
- = $\neg$ EF (start $\wedge$ $\neg$ AF heat)
- = $\neg$ EF (start $\wedge$ EG $\neg$ heat)
- = $\neg$ E(true U (start $\wedge$ EG $\neg$ heat))
- Now, how to compute the set of states in M satisfying this transformed formula ?

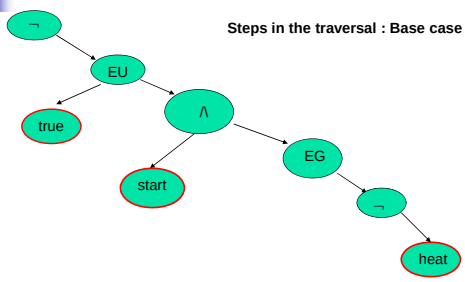
Example

Essentially by bottom-up traversal of the formula



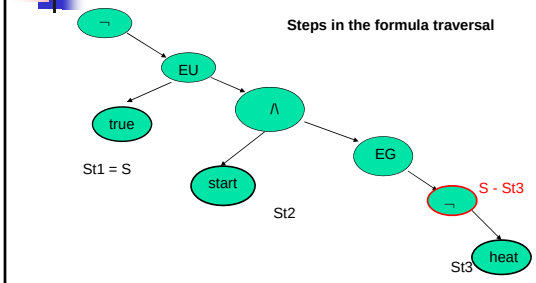
Example

Steps in the traversal : Base case



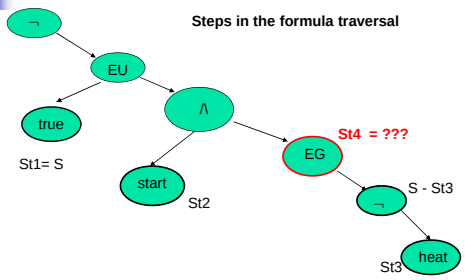
Example

Steps in the formula traversal



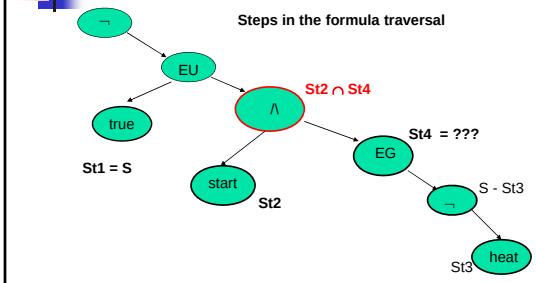
Example

Steps in the formula traversal



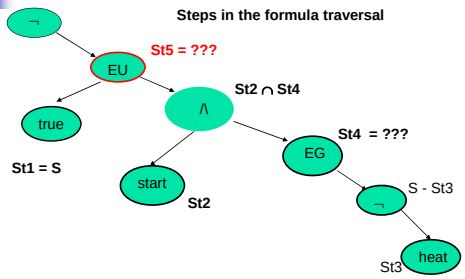
Example

Steps in the formula traversal

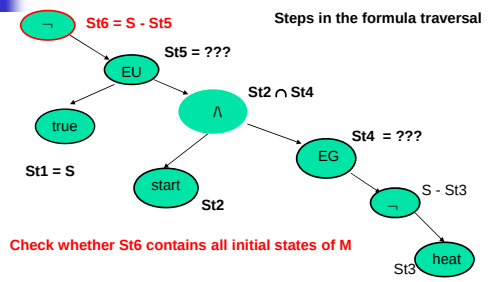


Example

Steps in the formula traversal



Example



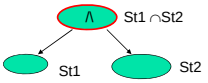
Check whether St6 contains all initial states of M

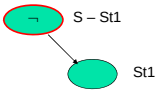
Bottom-up formula traversal

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

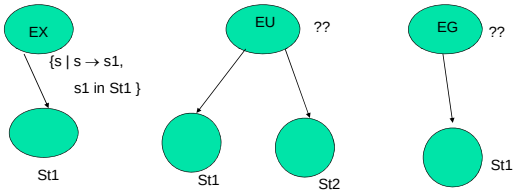
Boolean operators: ∧, ¬





Questions Remaining

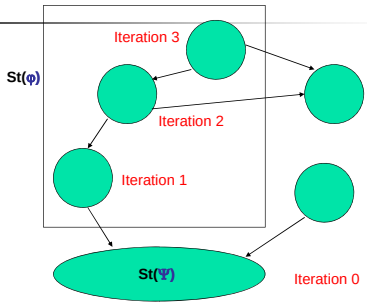
Temporal operators: EX, EU, EG



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



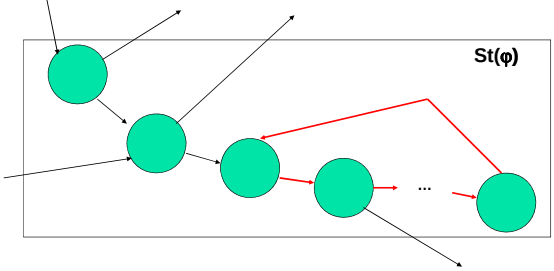
E(φ U Ψ) : Algorithm

- 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 $St_{EG\varphi} = St_\varphi$
  - For each state in St_φ, we check the out-edges. Many of the destination states are not in St_φ, so cannot satisfy EGφ
- It suffices to consider a reduced Kripke Structure M' constructed from M such that
  - All states of M which satisfy φ are retained.
  - All other states and transitions are deleted.
- For any s, we have M,s |= EGφ 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_φ
- Output: $St_{EG\varphi}$
- Technique:
  - Compute M'=(S',I',→',L') from M by keeping only nodes in St_φ
  - Temp := $St_{EG\varphi}$:= All nodes in nontrivial SCCs of M'
  - while Temp ≠ empty do
  - pick s ∈ Temp; Temp := Temp – {s};
  - $St_{EG\varphi}$:= $St_{EG\varphi}$ ∪ { t | t →' s ∧ t ∉ $St_{EG\varphi}$ };
  - Temp := Temp ∪ { t | t →' s ∧ t ∉ $St_{EG\varphi}$};
  - 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.