EECS 219C: Computer-Aided Verification

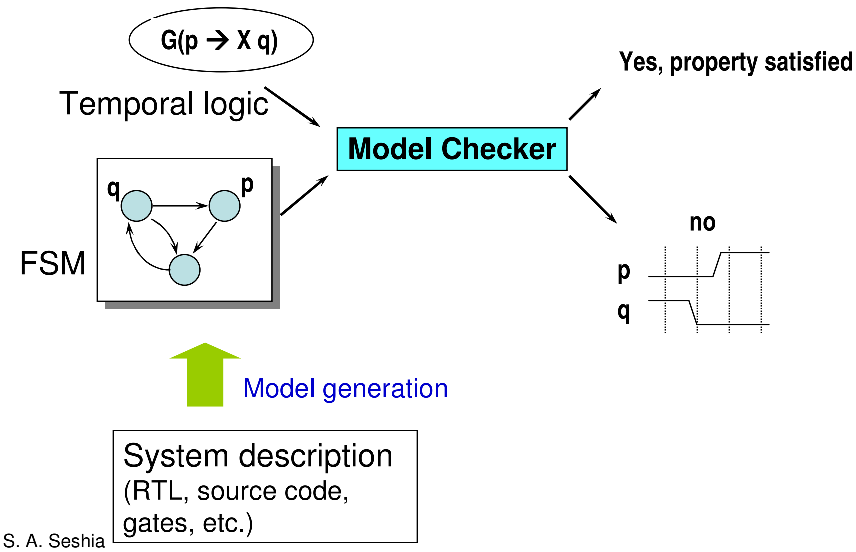
Models and Properties: Temporal Logic

Sanjit A. Seshia
EECS, UC Berkeley

Announcements

- Project topics due by e-mail to me next Monday
  - Include a short 1 paragraph description of the project

Finite-State Model Checking



Recap

- We're verifying closed systems
- Modeled as Kripke structures (S, S_0, R, L)
  - Represents the product of the "system" with its "environment"

System Behavior

- A sequence of states, starting with an initial state
  - $s_0\ s_1\ s_2 \ldots$ such that $R(s_i, s_{i+1})$ is true
- Also called “run”, or “(computation) path”
- Trace: sequence of observable parts of states
  - Sequence of state labels

Safety vs. Liveness

- Safety property
  - Error trace is finite
- Liveness property
  - Error trace is infinite

Temporal Logic

- A logic for specifying properties over time
  - E.g., Behavior of a finite-state system

- We will study *propositional* temporal logic
  - Other temporal logics exist:
    - e.g., real-time temporal logic

Atomic State Property (Label)

A Boolean formula over state variables

We will denote each unique Boolean formula by
• a distinct color
• a name such as p, q, …

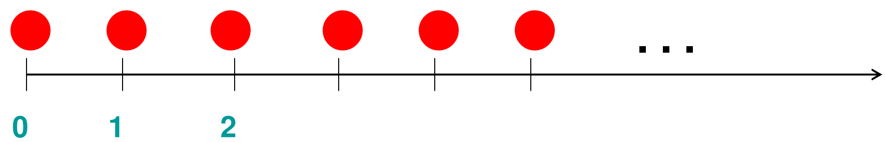
req **req & !ack**

Globally (Always) p: G p

G p is true for a computation path if p holds at all states (points of time) along the path

p = 

Suppose G p holds along the path below

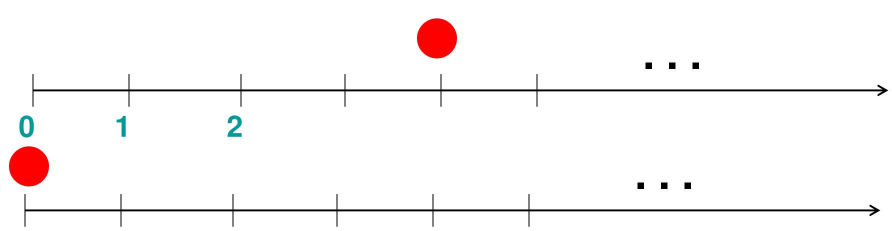


Eventually p: F p

- F p is true for a path if p holds at some state along that path

p = 

Does F p holds for the following examples?

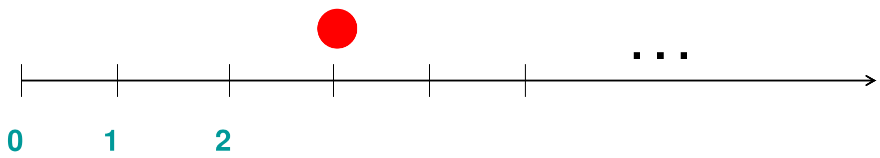


Next p: X p

- X p is true along a path starting in state s_i (suffix of the main path) if p holds in the next state s_{i+1}

p = 

Suppose X p holds along the path starting at state s_2



Nesting of Formulas

- p need not be just a Boolean formula.
- It can be a temporal logic formula itself!

p = 

"X p holds for all suffixes of a path"

How do we draw this?

How can we write this in temporal logic?

Write down formal definitions of Gp, Fp, Xp

Notation

- Sometimes you'll see alternative notation in the literature:

  G $\square$

  F $\diamond$

  X $\circ$

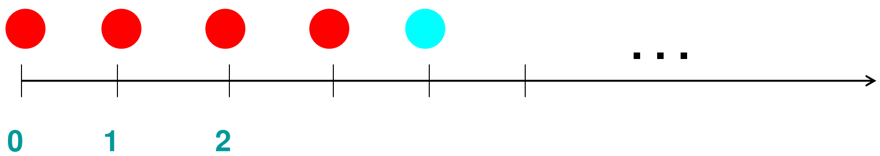
Examples: What do they mean?

- G F p
- F G p
- G(p → F q)
- F(p → (X X q))

p Until q: p U q

- p U q is true along a path starting at s if
  - q is true in some state reachable from s
  - p is true in all states from s until q holds

Suppose p U q holds for the path below



Temporal Operators & Relationships

- G, F, X, U: All express properties along paths
- Can you express G p purely in terms of F, p, and Boolean operators ?
- How about G and F in terms of U and Boolean operators?
- What about X in terms of G, F, U, and Boolean operators?

Examples in Temporal Logic

1. "No more than one processor (in a 2-processor system) should have a cache line in write mode"
   - wr_1 / wr_2 are respectively true if processor 1 / 2 has the line in write mode
2. "The grant signal must be asserted at some time after the request signal is asserted"
   - Signals: grant, req
3. "A request signal must receive an acknowledge and the request should stay asserted until the acknowledge signal is received"
   - Signals: req, ack

Examples in Temporal Logic

4. "From any state, it is possible to return to the reset state along some execution"
   - Signal indicating reset state: reset
5. "The grant signal must always be asserted 3 cycles after the request signal is asserted"
   - Signals: grant, req

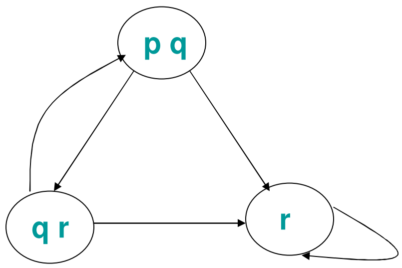
Linear Temporal Logic

- What we've seen so far are properties expressed over a single computation path or run
  – LTL

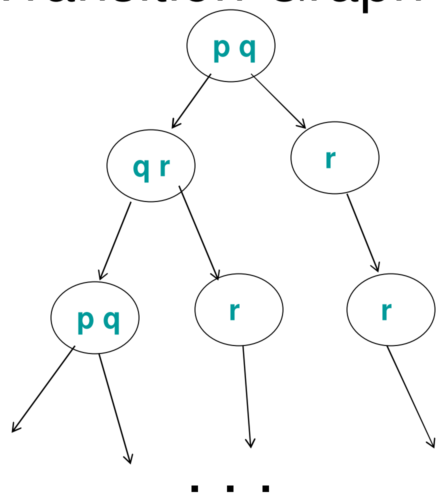
Temporal Logic Flavors

- Linear Temporal Logic
- Computation Tree Logic
  – Properties expressed over a tree of all possible executions
  – Where does this "tree" come from?

Labelled State Transition Graph



"Kripke structure"



Infinite Computation Tree

Temporal Logic Flavors

- Linear Temporal Logic (LTL)
- Computation Tree Logic (CTL, CTL*)
  - Properties expressed over a tree of all possible executions
  - CTL* gives more expressiveness than LTL
  - CTL is a subset of CTL* that is easier to verify than arbitrary CTL*

Computation Tree Logic (CTL*)

- Introduce two new operators A and E called "Path quantifiers"
  - Corresponding properties hold in states (not paths)
  - A p : Property p holds along all computation paths starting from the state where A p holds
  - E p : Property p holds along at least one path starting from the state where E p holds
- Example:

  "The grant signal must always be asserted some time after the request signal is asserted"

  A G (req → A F grant)

- Notation: A sometimes written as ∀, E as ∃

S. A. Seshia 23

CTL

- Every F, G, X, U must be immediately preceded by either an A or a E
  - E.g., Can't write A (FG p)
- LTL is just like having an "A" on the outside

S. A. Seshia 24

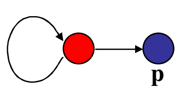
Why CTL?

- Verifying LTL properties turns out to be computationally harder than CTL
- But LTL is more intuitive to write
- Complexity of model checking
  - Exponential in the size of the LTL expression
  - linear for CTL
- For both, model checking is linear in the size of the state graph

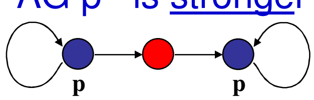
S. A. Seshia 25

CTL as a way to approximate LTL

- AG EF p is <u>weaker</u> than G F p **Good for finding bugs...**



- AF AG p is <u>stronger</u> than F G p



Good for verifying correctness...

Why? And what good is this approximation?

S. A. Seshia 26

More CTL

- "From any state, it is possible to get to the reset state along some path"

A G (E F reset)

S. A. Seshia 27

CTL vs. LTL Summary

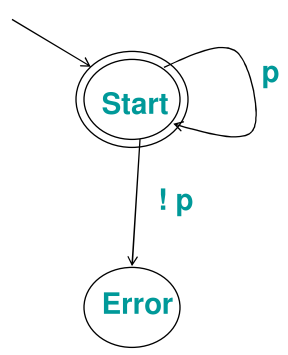
- Have different expressive powers
- Overall: LTL is easier for people to understand, hence more commonly used in property specification languages

S. A. Seshia 28

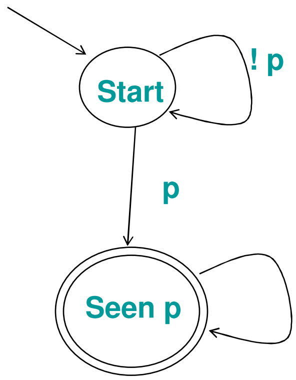
From Temporal Logic to Monitors

- A monitor for a temporal logic formula
  - is a finite state machine (automaton)
  - Accepts exactly those behaviors that satisfy the temporal logic formula
    - “Accepts” means that the accepting state is visited infinitely often
- Properties are often specified as automata

Monitor for G p, p a Boolean formula



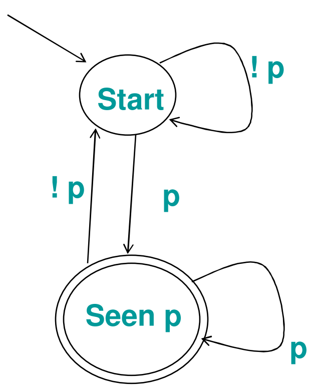
Monitor for F p, p a Boolean formula ?



S. A. Seshia

31

Monitor for GFp, p a Boolean formula ?



S. A. Seshia

32

Summary

- What we did today: Properties in Temporal Logic, LTL, CTL, CTL*
- Next: Start model checking algorithms