EECS 294-98:
Introduction to Temporal Logic

Sanjit A. Seshia

EECS, UC Berkeley

Plan for Today's Lecture

- Linear Temporal Logic

- Signal Temporal Logic (by Alex Donze)

Behavior, Run, Computation Path

- Define in terms of states and transitions
- 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
  - “something bad must not happen”
  - E.g.: system should not crash
  - finite-length error trace

- Liveness property
  - “something good must happen”
  - E.g.: every packet sent must be received at its destination
  - infinite-length error trace

Examples: Safety or Liveness?

1. "No more than one processor (in a multi-processor system) should have a cache line in write mode"

2. "The grant signal must be asserted at some time after the request signal is asserted"

3. "Every request signal must receive an acknowledge and the request should stay asserted until the acknowledge signal is received"

Temporal Logic

- A logic for specifying properties over time
  - E.g., Behavior of a finite-state system
- Basic: *propositional* temporal logic
  - Other temporal logics are also useful:
    - e.g., real-time temporal logic, metric temporal logic, signal temporal logic, …

Atomic State Property (Label)

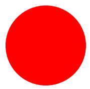
A Boolean formula over state variables

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

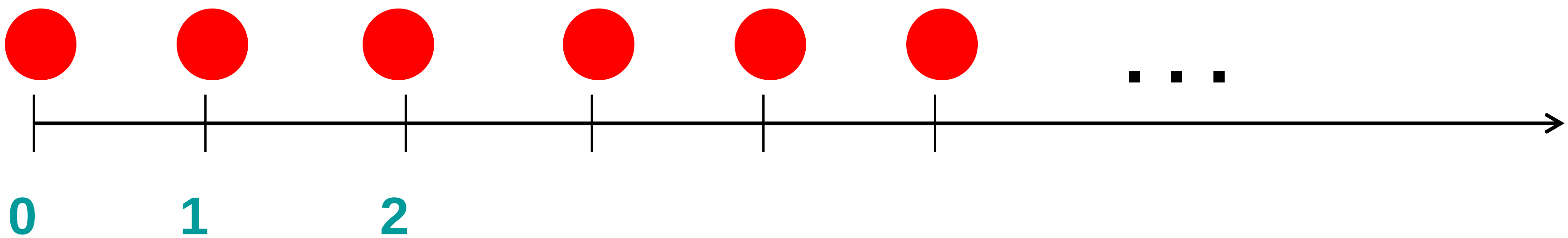
req   **req & !ack**

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**

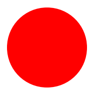
p = 

Suppose G p holds along the path below starting at s_0

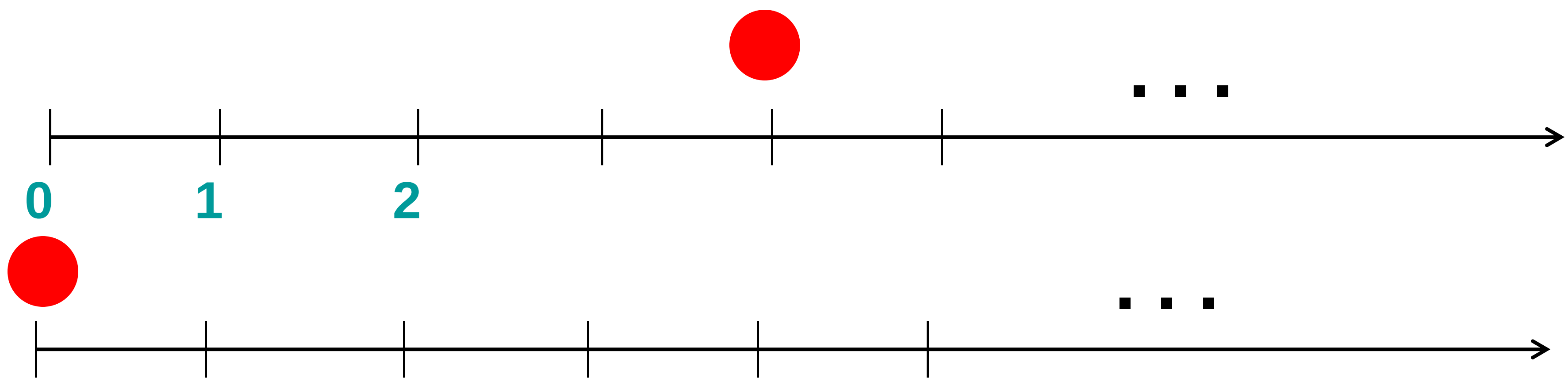


Eventually p: F p

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

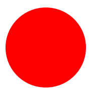
p = 

Does F p holds for the following examples?

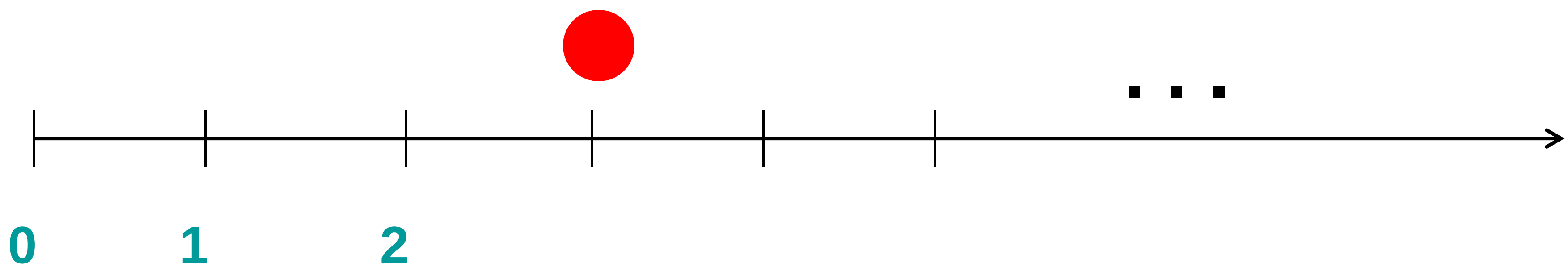


Next p: X p

- 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}

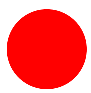
p = 

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:

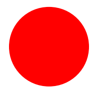
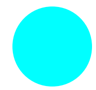
G	□
F	◊
X	∘

Examples: What do they mean?

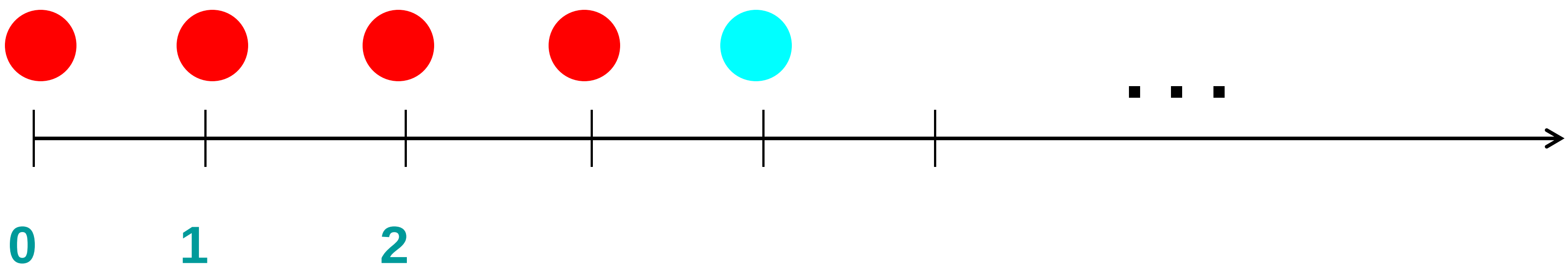
- G F p
- F G p
- G(p → F q)
- F(p → (X X q))

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

p =  **q =** 

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. "Every request signal must receive an acknowledge and the request should stay asserted until the acknowledge signal is received"
   - Signals: req, ack

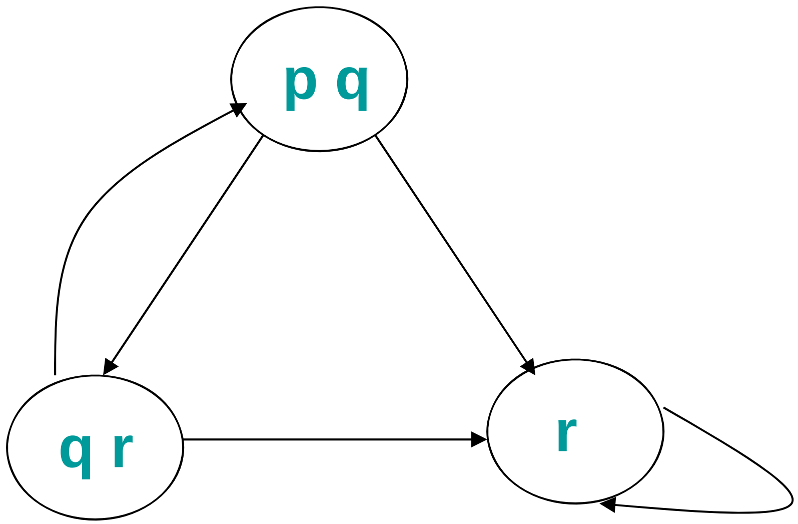
Linear Temporal Logic

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

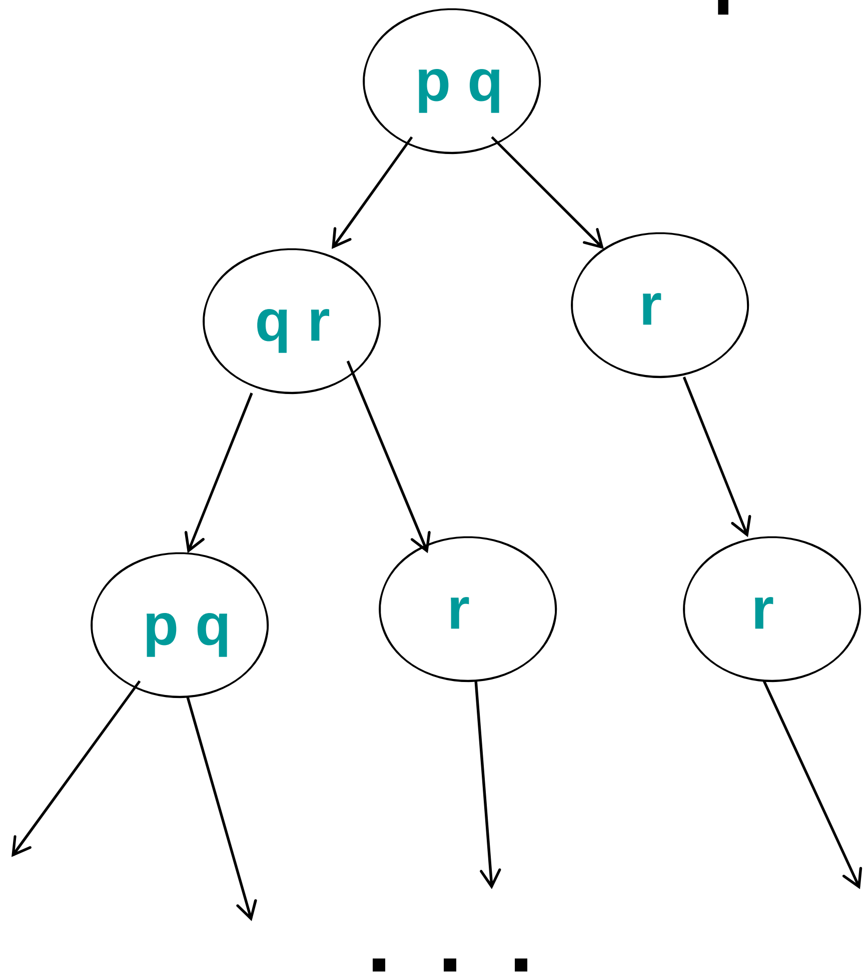
Temporal Logic Flavors

- 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 assented some time after the request signal is asserted"

  A G (req → A F grant)

- Notation: A sometimes written as 8, E as 9

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)

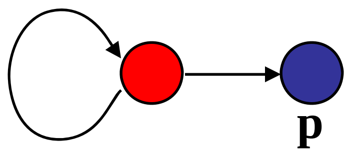
- LTL is just like having an “A” on the outside

Why CTL?

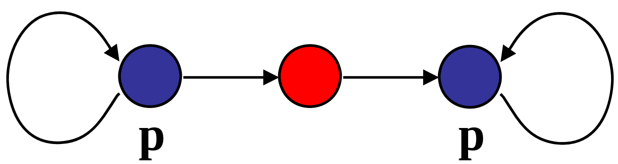
- Verifying LTL properties turns out to be computationally harder than CTL
- But LTL is more intuitive to write
- Complexity of model checking
  - Exponential in the size of the LTL expression
  - linear for CTL
- For both, model checking is linear in the size of the state graph

CTL as a way to approximate LTL

– AG EF p is weaker than G F p **Useful for finding bugs...**



– AF AG p is stronger than F G p



Useful for verifying correctness...

Why? And what good is this approximation?

More CTL

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

A G (E F reset)

CTL vs. LTL Summary

- Have different expressive powers

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

Some Remarks on Temporal Logic

- The vast majority of properties are safety properties
- Liveness properties are useful abstractions of more complicated safety properties (such as real-time response constraints)

(Absence of) Deadlock

- An oft-cited property, especially people building distributed / concurrent systems
- Can you express it in terms of
  – a property of the state graph (graph of all reachable states)?
  – a CTL property?
  – a LTL property?