



CSCI 740 - Programming Language Theory

Lecture 32

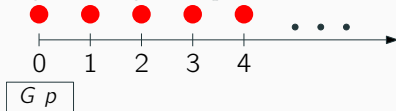
Temporal Logic

Instructor: Hossein Hojjat

November 17, 2017

Temporal Logic

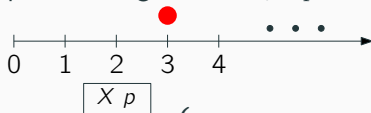
$G p$ is true for a computation path if p holds at all states (points of time)



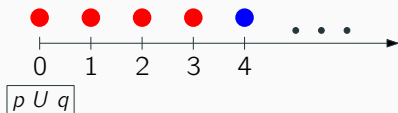
$F p$ is true for a computation path if p holds at some state along that path



$X p$ is true along a path starting in state s_i if p holds in the next state s_{i+1}



$p U q$ is true along a path starting at s $\left\{ \begin{array}{l} q \text{ is true in some state reachable from } s \\ p \text{ is true in all states from } s \text{ until } q \text{ holds} \end{array} \right.$



$p = \text{red dot}, q = \text{blue dot}$

Example

What do they mean?

- $G F p$
- $F G p$
- $G(p \rightarrow F q)$
- $F(p \rightarrow (X X q))$

Example

What do they mean?

- $\mathbf{G F p}$

p holds infinitely often

- $\mathbf{F G p}$

- $\mathbf{G(p \rightarrow F q)}$

- $\mathbf{F(p \rightarrow (X X q))}$

Example

What do they mean?

- $\mathbf{G F p}$

p holds infinitely often

- $\mathbf{F G p}$

Eventually, *p* holds henceforth

- $\mathbf{G(p \rightarrow F q)}$

- $\mathbf{F(p \rightarrow (X X q))}$

Example

What do they mean?

- $\mathbf{G F p}$

p holds infinitely often

- $\mathbf{F G p}$

Eventually, p holds henceforth

- $\mathbf{G(p \rightarrow F q)}$

Every p is eventually followed by a q

- $\mathbf{F(p \rightarrow (X X q))}$

Example

What do they mean?

- $\mathbf{G F p}$

p holds infinitely often

- $\mathbf{F G p}$

Eventually, p holds henceforth

- $\mathbf{G(p \rightarrow F q)}$

Every p is eventually followed by a q

- $\mathbf{F(p \rightarrow (X X q))}$

Every p is followed by a q two steps later

Temporal Operators & Relationships

G, F, X, U : All express properties along system traces

- Can you express $G p$ purely in terms of F, p , and Boolean operators?
- How about F in terms of U ?
- What about X in terms of G, F , or U ?

Temporal Operators & Relationships

G, F, X, U : All express properties along system traces

- Can you express $G p$ purely in terms of F, p , and Boolean operators?

$$G p = \neg F \neg p$$

- How about F in terms of U ?
- What about X in terms of G, F , or U ?

Temporal Operators & Relationships

G, F, X, U : All express properties along system traces

- Can you express $G p$ purely in terms of F, p , and Boolean operators?

$$G p = \neg F \neg p$$

- How about F in terms of U ?

$$F p = \text{true } U p$$

- What about X in terms of G, F , or U ?

Temporal Operators & Relationships

G, F, X, U : All express properties along system traces

- Can you express $G p$ purely in terms of F, p , and Boolean operators?

$$G p = \neg F \neg p$$

- How about F in terms of U ?

$$F p = \text{true } U p$$

- What about X in terms of G, F , or U ?

Cannot be done

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states
- In all states it is not the case that *read* and *write*
- At every state a *request* implies that there exists a future point where *grant* holds
- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point
- In all states, there is a future position where *enabled* holds
- There is a future position, from which all future positions have *enabled* holding

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states

$G \textit{ inv}$

- In all states it is not the case that *read* and *write*
- At every state a *request* implies that there exists a future point where *grant* holds
- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point
- In all states, there is a future position where *enabled* holds
- There is a future position, from which all future positions have *enabled* holding

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states

$$G \textit{ inv}$$

- In all states it is not the case that *read* and *write*

$$G \neg(\textit{read} \wedge \textit{write})$$

- At every state a *request* implies that there exists a future point where *grant* holds

- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point

- In all states, there is a future position where *enabled* holds

- There is a future position, from which all future positions have *enabled* holding

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states

$$G \textit{ inv}$$

- In all states it is not the case that *read* and *write*

$$G \neg(\textit{read} \wedge \textit{write})$$

- At every state a *request* implies that there exists a future point where *grant* holds

$$G(\textit{request} \rightarrow F \textit{ grant})$$

- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point

- In all states, there is a future position where *enabled* holds

- There is a future position, from which all future positions have *enabled* holding

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states

$$G \textit{ inv}$$

- In all states it is not the case that *read* and *write*

$$G \neg(\textit{read} \wedge \textit{write})$$

- At every state a *request* implies that there exists a future point where *grant* holds

$$G(\textit{request} \rightarrow F \textit{ grant})$$

- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point

$$G(\textit{request} \rightarrow (\textit{request} U \textit{ grant}))$$

- In all states, there is a future position where *enabled* holds

- There is a future position, from which all future positions have *enabled* holding

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states

$$G \textit{ inv}$$

- In all states it is not the case that *read* and *write*

$$G \neg(\textit{read} \wedge \textit{write})$$

- At every state a *request* implies that there exists a future point where *grant* holds

$$G(\textit{request} \rightarrow F \textit{ grant})$$

- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point

$$G(\textit{request} \rightarrow (\textit{request} U \textit{ grant}))$$

- In all states, there is a future position where *enabled* holds

$$G F \textit{ enabled}$$

- There is a future position, from which all future positions have *enabled* holding

Exercise

Write a temporal logic formula for each of the given properties

- *inv* is true for all states

$$G \textit{ inv}$$

- In all states it is not the case that *read* and *write*

$$G \neg(\textit{read} \wedge \textit{write})$$

- At every state a *request* implies that there exists a future point where *grant* holds

$$G(\textit{request} \rightarrow F \textit{ grant})$$

- At every state a *request* implies that there exists a future point where *grant* holds, and *request* holds up until that point

$$G(\textit{request} \rightarrow (\textit{request} U \textit{ grant}))$$

- In all states, there is a future position where *enabled* holds

$$G F \textit{ enabled}$$

- There is a future position, from which all future positions have *enabled* holding

$$F G \textit{ enabled}$$

Liveness vs. Safety

Two terms you are likely to run into:

Safety

- Something bad will never happen

$$G \neg bad$$

- If it fails to hold, it's easy to produce a witness

Liveness

- Something good will eventually happen

$$F good$$

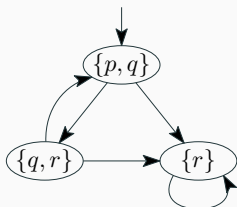
- What does a witness for this look like?

Temporal Logic Flavors

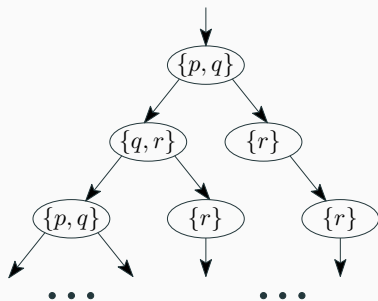
- What we have seen so far are properties expressed over a single computation path or run
- **Linear** Temporal Logic

Computation Tree Logic

- Properties expressed over a tree of all possible executions



Kripke structure



Infinite Computation Tree

Computation Tree Logic

- 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 extra path quantifiers A and E
- $A p$: for all paths f
- $E p$: for some path f
- Example:

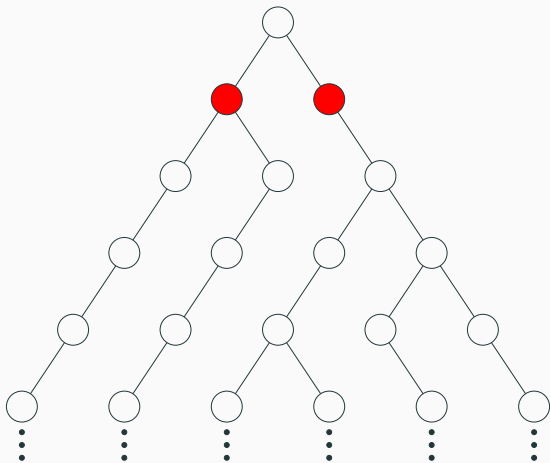
“The grant signal must always be asserted some time after the request signal is asserted”

$$AG(req \rightarrow AF grant)$$

Two important subsets:

- LTL : all formulas of the form $A p$
 - Example : $A(FG p)$
- CTL: there must be a path quantifier before every linear operator
 - Example : $AG(EF p)$

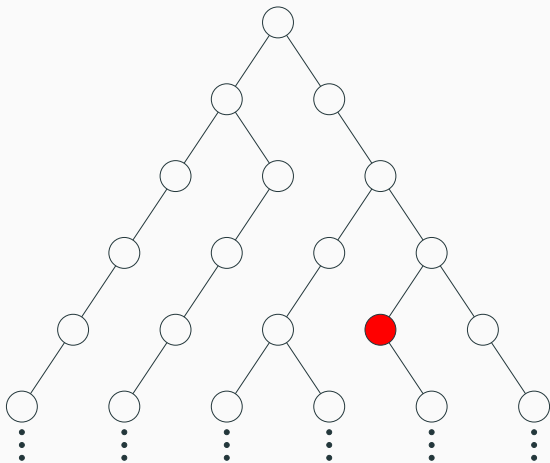
CTL Example



$AX p$

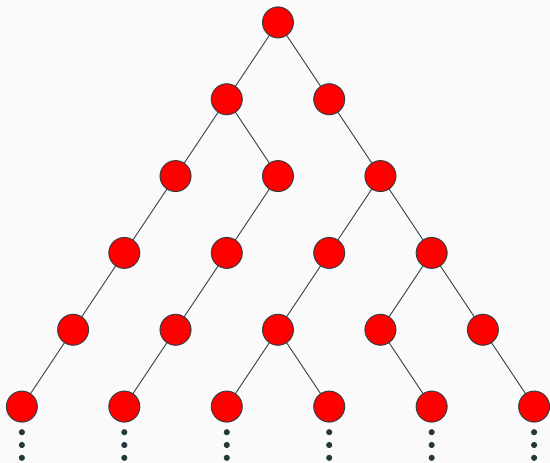
For every next state p holds

CTL Example



$EF p$

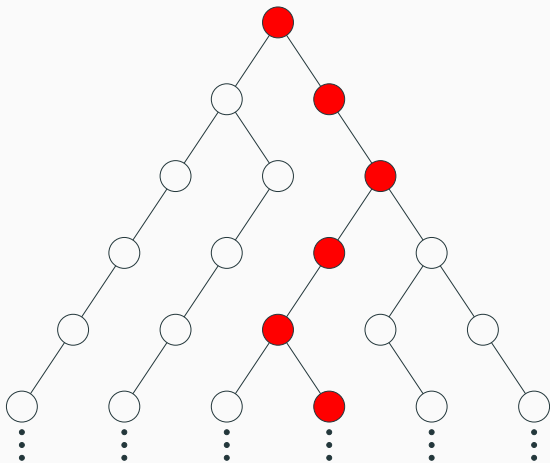
There exists a path with a future state where p holds



$AG p$

For all paths, for all states along them, p holds

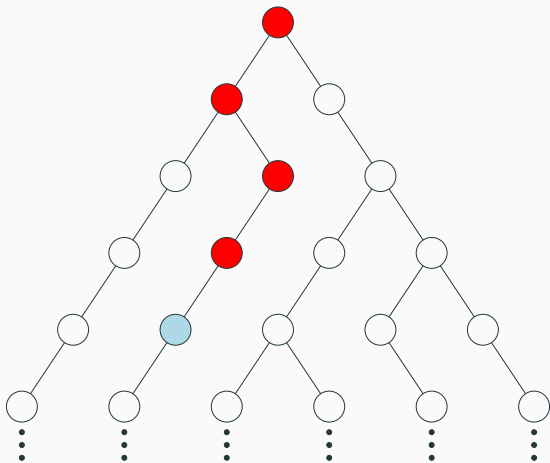
CTL Example



$EG p$

There exists a path such that, for all states along it, p holds

CTL Example



$$E(p \ U \ q)$$

Exists path where q eventually holds, and p holds at all states earlier

- CTL and LTL are not equivalent
- There are properties that can be expressed in LTL but cannot be expressed in CTL
 - For example: $F G p$
- There are properties that can be expressed in CTL but cannot be expressed in LTL
 - For example: $AG(EF p)$
- Hence, expressive power of CTL and LTL are not comparable

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