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 \(s\)

\[\begin{align*}
p &= \bullet, \ q &= \bigcirc\end{align*}\]
Example

What do they mean?

- \( G \ F \ p \)
- \( F \ G \ p \)
- \( G(p \rightarrow F \ q) \)
- \( F(p \rightarrow (X \ X \ q)) \)
What do they mean?

- \( \text{G F p} \)
  - \( p \) holds infinitely often

- \( \text{F G p} \)

- \( \text{G(p \rightarrow F q)} \)

- \( \text{F(p \rightarrow (X X q))} \)
What do they mean?

- $G F p$
  \[\text{$p$ holds infinitely often}\]
- $F G p$
  \[\text{Eventually, $p$ holds henceforth}\]
- $G(p \rightarrow F q)$
- $F(p \rightarrow (X X q))$
Example

What do they mean?

- $G F p$
  
  $p$ holds infinitely often

- $F G p$
  
  Eventually, $p$ holds henceforth

- $G(p \rightarrow F q)$
  
  Every $p$ is eventually followed by a $q$

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

What do they mean?

- **G F p**
  
  $p$ holds infinitely often

- **F G p**
  
  Eventually, $p$ holds henceforth

- **G(p → F q)**
  
  Every $p$ is eventually followed by a $q$

- **F(p → (X X q))**
  
  Every $p$ is followed by a $q$ two steps later
$G, F, X, U$: 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 $U$?
\( G, F, X, U \): All express properties along system traces

- Can you express \( Gp \) purely in terms of \( F, p, \) and Boolean operators?
  \[
  Gp = \neg F \neg p
  \]

- How about \( F \) in terms of \( U? \)

- What about \( X \) in terms of \( G, F, \) or \( U? \)
$G, F, X, U$: 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 $U$?
$G, F, X, U$: 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 $G, F, \text{ or } U$?

  Cannot be done
Exercise

Write a temporal logic formula for each of the given properties

- \( \text{inv} \) is true for all states

- In all states it is not the case that \( \text{read} \) and \( \text{write} \)

- At every state a \( \text{request} \) implies that there exists a future point where \( \text{grant} \) holds

- At every state a \( \text{request} \) implies that there exists a future point where \( \text{grant} \) holds, and \( \text{request} \) holds up until that point

- In all states, there is a future position where \( \text{enabled} \) holds

- There is a future position, from which all future positions have \( \text{enabled} \) holding
Exercise

Write a temporal logic formula for each of the given properties

- $\text{inv}$ is true for all states
  
  $G \text{inv}$

- In all states it is not the case that $\text{read}$ and $\text{write}$
  
  $G \neg (\text{read} \land \text{write})$

- At every state a $\text{request}$ implies that there exists a future point where $\text{grant}$ holds
  
  $G (\text{request} \rightarrow F \text{grant})$

- At every state a $\text{request}$ implies that there exists a future point where $\text{grant}$ holds, and $\text{request}$ holds up until that point
  
  $G (\text{request} \rightarrow (\text{request} \mathcal{U} \text{grant}))$

- In all states, there is a future position where $\text{enabled}$ holds
  
  $G F \text{enabled}$

- There is a future position, from which all future positions have $\text{enabled}$ holding
  
  $F G \text{enabled}$
Exercise

Write a temporal logic formula for each of the given properties

- $inv$ is true for all states

\[ G \ inv \]

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

\[ G \neg (read \land write) \]

- 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
Write a temporal logic formula for each of the given properties

- \( \text{inv} \) is true for all states
  \[ G \text{ inv} \]

- In all states it is not the case that \text{read} and \text{write}
  \[ G \neg (\text{read} \land \text{write}) \]

- At every state a \text{request} implies that there exists a future point where \text{grant} holds
  \[ G (\text{request} \rightarrow F \text{ grant}) \]

- At every state a \text{request} implies that there exists a future point where \text{grant} holds, and \text{request} holds up until that point

- In all states, there is a future position where \text{enabled} holds

- There is a future position, from which all future positions have \text{enabled} holding
Write a temporal logic formula for each of the given properties

- $inv$ is true for all states
  
  $$G \ inv$$

- In all states it is not the case that $read$ and $write$
  
  $$G \neg (read \land write)$$

- At every state a $request$ implies that there exists a future point where $grant$ holds
  
  $$G (request \rightarrow F \ grant)$$

- At every state a $request$ implies that there exists a future point where $grant$ holds, and $request$ holds up until that point
  
  $$G (request \rightarrow (request U grant))$$

- 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

- \( \text{inv} \) is true for all states
  
  \[ G \text{ inv} \]

- In all states it is not the case that \( \text{read} \) and \( \text{write} \)
  
  \[ G \neg (\text{read} \land \text{write}) \]

- At every state a \( \text{request} \) implies that there exists a future point where \( \text{grant} \) holds
  
  \[ G(\text{request} \rightarrow F \text{ grant}) \]

- At every state a \( \text{request} \) implies that there exists a future point where \( \text{grant} \) holds, and \( \text{request} \) holds up until that point
  
  \[ G(\text{request} \rightarrow (\text{request} \mathbin{U} \text{ grant})) \]

- In all states, there is a future position where \( \text{enabled} \) holds
  
  \[ G F \text{ enabled} \]

- There is a future position, from which all future positions have \( \text{enabled} \) holding
Exercise

Write a temporal logic formula for each of the given properties

• $inv$ is true for all states
  
  $G \ inv$

• In all states it is not the case that $read$ and $write$
  
  $G \neg(read \land write)$

• At every state a $request$ implies that there exists a future point where $grant$ holds
  
  $G(request \rightarrow F\ grant)$

• At every state a $request$ implies that there exists a future point where $grant$ holds, and $request$ holds up until that point
  
  $G(request \rightarrow (request U grant))$

• In all states, there is a future position where $enabled$ holds
  
  $G\ F\ enabled$

• There is a future position, from which all future positions have $enabled$ holding
  
  $FG\ enabled$
Two terms you are likely to run into:

**Safety**
- Something bad will never happen
  \[ G \neg \text{bad} \]
- If it fails to hold, it’s easy to produce a witness

**Liveness**
- Something good will eventually happen
  \[ F \text{good} \]
- 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 (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 : $AG(EF\ p)$
For every next state $p$ holds

AX $p$
There exists a next state where $p$ holds

$EX \ p$
For all paths, there exists a future state where $p$ holds
There exists a path with a future state where $p$ holds
For all paths, for all states along them, $p$ holds
There exists a path such that, for all states along it, $p$ holds

$EG\ p$
For all paths, $q$ eventually holds, and $p$ holds at all states earlier
$E(p U q)$

Exists path where $q$ eventually holds, and $p$ holds at all states earlier
CTL vs. LTL

- 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