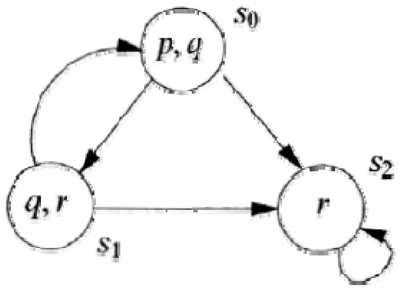
Temporal Logic
- Branching-time logic

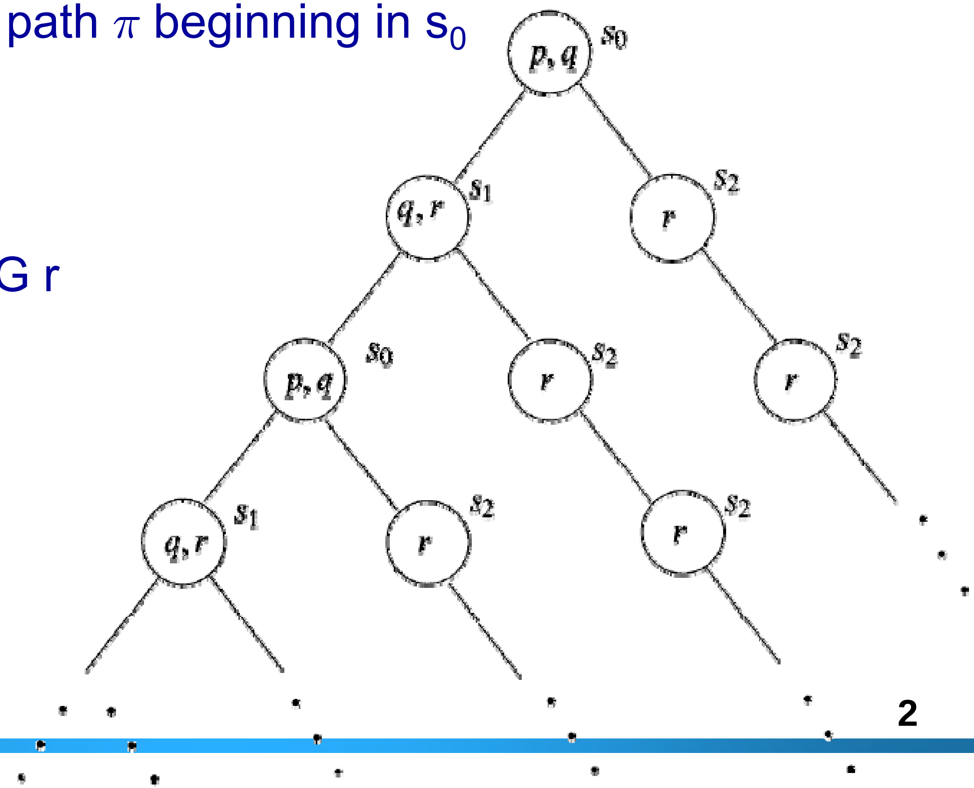
Moonzoo Kim
CS Division of EECS Dept.
KAIST

Semantics of LTL (3/3)

- Def 3.8 Suppose $\mathcal{M}$ = (S, $\rightarrow$, L) is a model, s $\in$ S, and ϕ an LTL formula. We write $\mathcal{M}$,s $\vDash \phi$ if for every execution path π of $\mathcal{M}$ starting at s, we have $\pi \vDash \phi$
  - If $\mathcal{M}$ is clear from the context, we write s $\vDash \phi$



- Example
  - $s_0 \vDash p \wedge q$ since $\pi \vDash p \wedge q$ for every path π beginning in s_0
  - $s_0 \vDash \neg r$, $s_0 \vDash \top$
  - $s_0 \vDash X\, r$, $s_0 \nvDash X\,(q \wedge r)$
  - $s_0 \vDash G\, \neg(p \wedge r)$, $s_2 \vDash G\, r$
  - For any s of $\mathcal{M}$, s $\vDash F(\neg q \wedge r) \rightarrow F\, G\, r$
    - Note that s_2 satisfies $\neg q \wedge r$
  - $s_0 \nvDash G\, F\, p$
    - $s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \ldots \vDash G\, F\, p$
    - $s_0 \rightarrow s_2 \rightarrow s_2 \rightarrow s_2 \ldots \nvDash G\, F\, p$
  - $s_0 \vDash G\, F\, p \rightarrow G\, F\, r$
  - $s_0 \nvDash G\, F\, r \rightarrow G\, F\, p$



Practical patterns of specification

- For any state, if a request occurs, then it will eventually be acknowledge
  - G(requested $\rightarrow$ F acknowledged)
- A certain process is enabled infinitely often on every computation path
  - G F enabled
- Whatever happens, a certain process will eventually be permanently deadlocked
  - F G deadlock
- If the process is enabled infinitely often, then it runs infinitely often
  - G F enabled $\rightarrow$ G F running
- An upwards traveling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor
  - G (fllor2 $\wedge$ directionup $\wedge$ ButtonPressed5 $\rightarrow$ (directionup U floor5)
- It is impossible to get to a state where a system has started but is not ready
  - ϕ = G $\neg$(started $\wedge$ $\neg$ready)
  - What is the meaning of (intuitive) negation of ϕ ?
    - For every path, it is possible to get to such a state (started $\wedge\neg$ready).
    - There exists a such path that gets to such a state.
      - we cannot express this meaning directly
- LTL has limited expressive power
  - For example, LTL cannot express statements which assert the existence of a path
    - From any state s, there exists a path π starting from s to get to a restart state
    - The lift can remain idle on the third floor with its doors closed
  - Computation Tree Logic (CTL) has operators for quantifying over paths and can express these properties

Summary of practical patterns

G p	always p	invariance
F p	eventually p	guarantee
$p \rightarrow (F\ q)$	p implies eventually q	response
$p \rightarrow (q\ U\ r)$	p implies q until r	precedence
G F p	always, eventually p	recurrence (progress)
F G p	eventually, always p	stability (non-progress)
$F\ p \rightarrow F\ q$	eventually p implies eventually q	correlation

Equivalences between LTL formulas

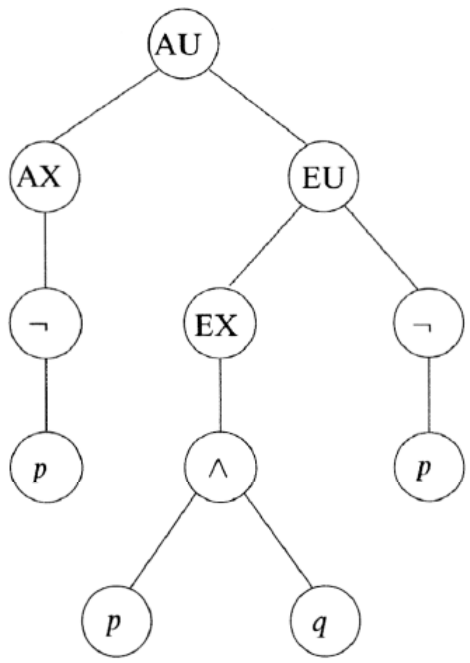
- Def 3.9 $\phi \equiv \psi$ if for all models $\mathcal{M}$ and all paths π in $\mathcal{M}$: $\pi \vDash \phi$ iff $\pi \vDash \psi$
- $\neg \mathrm{G}\ \phi \equiv \mathrm{F}\ \neg\phi$, $\neg \mathrm{F}\ \phi \equiv \mathrm{G}\ \neg\phi$, $\neg \mathrm{X}\ \phi \equiv \mathrm{X}\ \neg\phi$
- $\neg\ (\phi\ \mathrm{U}\ \psi) \equiv \neg\phi\ \mathrm{R}\ \neg\psi$, $\neg(\phi\ \mathrm{R}\ \psi) \equiv \neg\phi\ \mathrm{U}\ \neg\psi$
- $\mathrm{F}\ (\phi \vee \psi) \equiv \mathrm{F}\ \phi \vee \mathrm{F}\ \psi$
- $\mathrm{G}\ (\phi \wedge \psi) \equiv \mathrm{G}\ \phi \wedge \mathrm{G}\ \psi$
- $\mathrm{F}\ \phi \equiv \mathrm{T}\ \mathrm{U}\ \phi$, $\mathrm{G}\ \phi \equiv \perp \mathrm{R}\ \phi$
- $\phi\ \mathrm{U}\ \psi \equiv \phi\ \mathrm{W}\ \psi \wedge \mathrm{F}\ \psi$
- $\phi\ \mathrm{W}\ \psi \equiv \phi\ \mathrm{U}\ \psi \vee \mathrm{G}\ \phi$
- $\phi\ \mathrm{W}\ \psi \equiv \psi\ \mathrm{R}\ (\phi \vee \psi)$
- $\phi\ \mathrm{R}\ \psi \equiv \psi\ \mathrm{W}\ (\phi \wedge \psi)$

LTL vs. CTL

- LTL implicitly quantifies universally over paths
  - a state of a system satisfies an LTL formula if all paths from the given state satisfy it
  - properties which use both universal and existential path quantifiers cannot in general be model checked using LTL.
    - property ϕ which use only universal path quantifiers can be checked using LTL by checking $\neg\phi$
- Branching-time logic solve this limitation by quantifying paths explicitly
  - There is a reachable state satisfying q: EF q
    - Note that we can check this property by checking LTL formula ϕ=G $\neg$q
      - If ϕ is true, the property is false. If ϕ is false, the property is true
  - From all reachable states satisfying p, it is possible to maintain p continuously until reaching a state satisfying q: AG (p $\rightarrow$ E (p U q))
  - Whenever a state satisfying p is reached, the system can exhibit q continuously forevermore: AG (p $\rightarrow$ EG q)
  - There is a reachable state from which all reachable states satisfy p: EF AG p

Syntax of Computation Tree Logic (CTL)

- Def 3.12 $\phi = \bot \mid \top \mid p \mid \neg \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid$ AX ϕ | EX ϕ | AF ϕ | EF ϕ | AG ϕ | EG ϕ | A (ϕ U ϕ) | E (ϕ U ϕ)
  - A: along all paths
  - E: along at least one path
- Precedence
  - AG, EG, AF, EF,AX, EX, $\wedge$, $\vee$, $\rightarrow$, AU, EU
- Note that the following formulas are not well-formed CTL formulas
  - EF G r
  - A $\neg$G $\neg$ p
  - F (r U q)
  - EF (r U q)
  - AEF r
  - A ((r U q) $\wedge$ (p U r))



A [(AX $\neg$p) U (E [(EX p$\wedge$q) U $\neg$p)]]

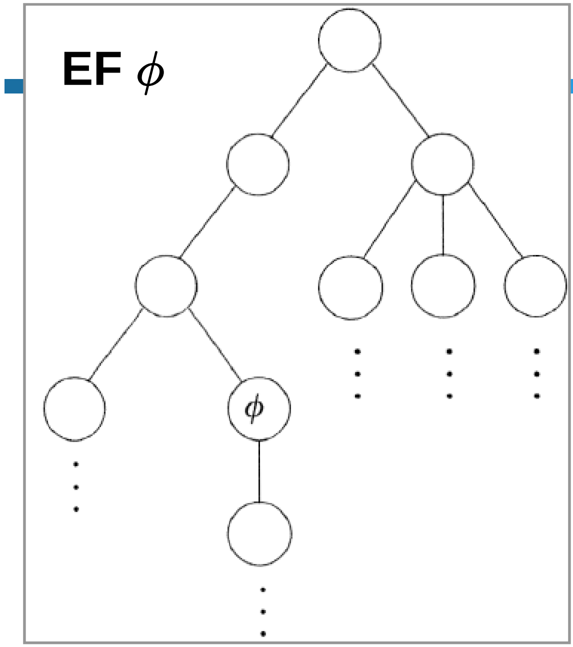
Semantics of CTL (1/2)

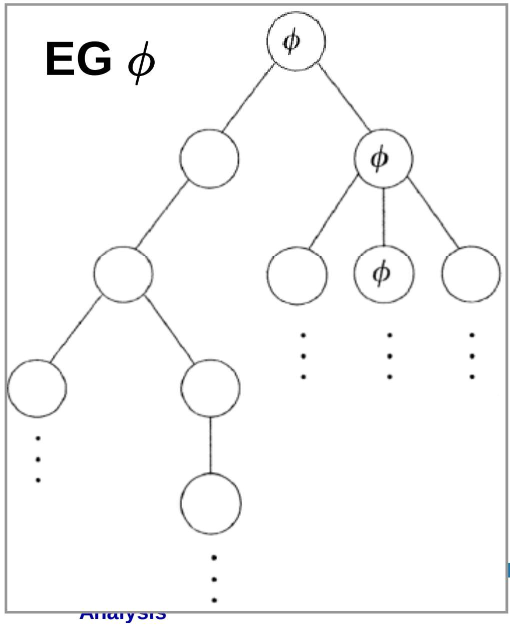
- Def 3.15 Let $\mathcal{M}$ = (S, $\rightarrow$, L) be a model for CTL, s in S, ϕ a CTL formula. The relation $\mathcal{M}$,s $\models \phi$ is defined by structural induction on ϕ. We omit $\mathcal{M}$ if context is clear.
  - $\mathcal{M}$,s $\models \top$ and $\mathcal{M}$,s $\not\models \bot$
  - $\mathcal{M}$,s $\models$ p iff p $\in$ L(s)
  - $\mathcal{M}$,s $\models \neg\, \phi$ iff $\mathcal{M}$,s $\not\models \phi$
  - $\mathcal{M}$,s $\models \phi_1 \wedge \phi_2$ iff $\mathcal{M}$,s $\models \phi_1$ and $\mathcal{M}$,s $\models \phi_2$
  - $\mathcal{M}$,s $\models \phi_1 \vee \phi_2$ iff $\mathcal{M}$,s $\models \phi_1$ or $\mathcal{M}$,s $\models \phi_2$
  - $\mathcal{M}$,s $\models \phi_1 \rightarrow \phi_2$ iff $\mathcal{M}$,s $\not\models \phi_1$ or $\mathcal{M}$,s $\models \phi_2$
  - $\mathcal{M}$,s $\models$ AX ϕ iff for all s_1 s.t. s $\rightarrow$ s_1 we have $\mathcal{M}$, $s_1 \models \phi$. Thus AX says "in every next state"
  - $\mathcal{M}$,s $\models$ EX ϕ iff for some s_1 s.t. s $\rightarrow$ s_1 we have $\mathcal{M}$, $s_1 \models \phi$. Thus EX says "in some next state"
  - $\mathcal{M}$,s $\models$ AX ϕ iff for all s_1 s.t. s $\rightarrow$ s_1 we have $\mathcal{M}$, $s_1 \models \phi$. Thus AX says "in every next state"
  - $\mathcal{M}$,s $\models$ EX ϕ iff for some s_1 s.t. s $\rightarrow$ s_1 we have $\mathcal{M}$, $s_1 \models \phi$. Thus EX says "in some next state"

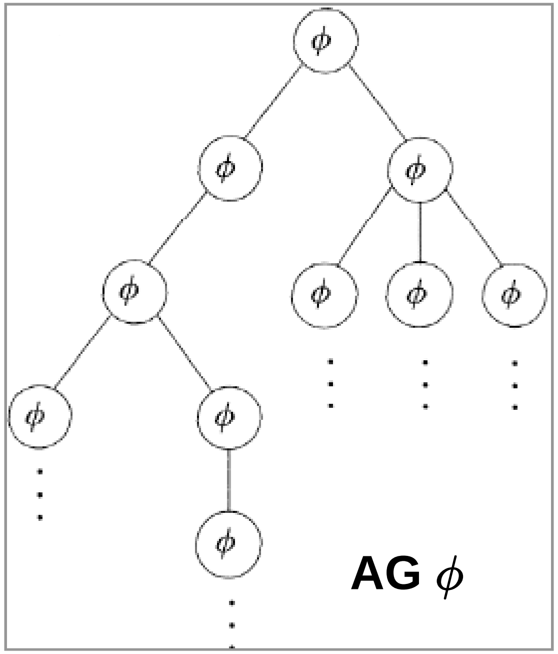
Semantics of CTL (2/2)

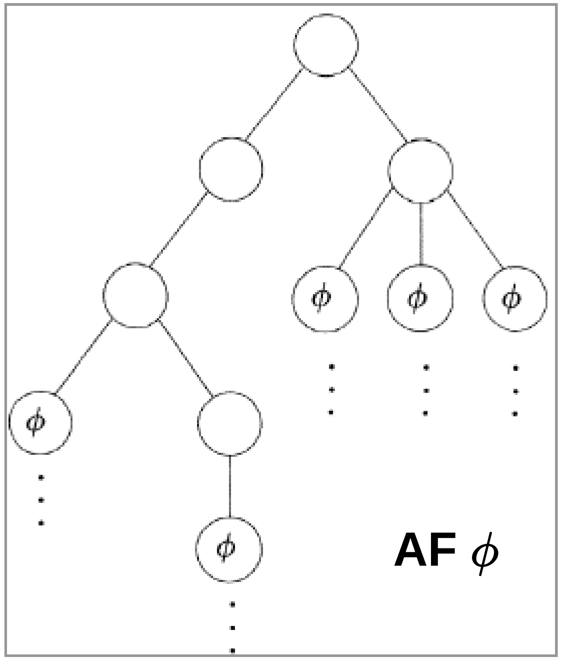
- Def 3.15 Let $\mathcal{M}$ = (S, $\rightarrow$, L) be a model for CTL, s in S, ϕ a CTL formula. The relation $\mathcal{M}$,s $\vDash \phi$ is defined by structural induction on ϕ. We omit $\mathcal{M}$ if context is clear.
  - $\mathcal{M}$,s $\vDash$ AG ϕ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, and all s_i along the path, we have $\mathcal{M}, s_i \vDash \phi$.
  - $\mathcal{M}$,s $\vDash$ EG ϕ iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, and all s_i along the path, we have $\mathcal{M}, s_i \vDash \phi$.
  - $\mathcal{M}$,s $\vDash$ AF ϕ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, and there is some s_i s.t. $\mathcal{M}, s_i \vDash \phi$.
  - $\mathcal{M}$,s $\vDash$ EF ϕ iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, and there is some s_i s.t. $\mathcal{M}, s_i \vDash \phi$.
  - $\mathcal{M}$,s $\vDash$ A [ϕ_1 U ϕ_2] iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, that path satisfies ϕ_1 U ϕ_2
  - $\mathcal{M}$,s $\vDash$ E [ϕ_1 U ϕ_2] iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, that path satisfies ϕ_1 U ϕ_2

Example (1/2)



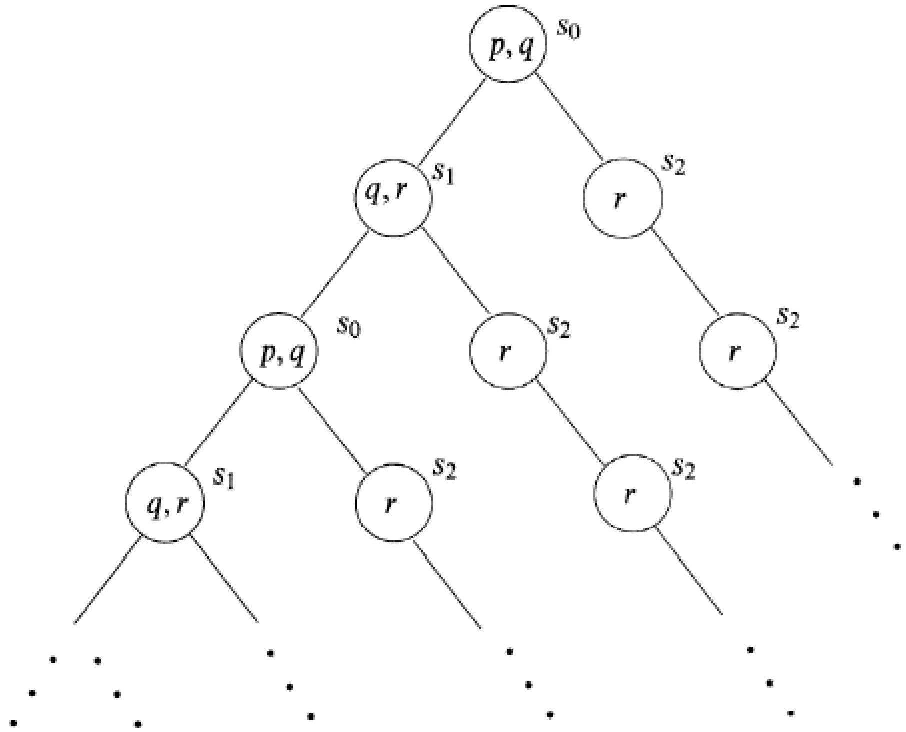


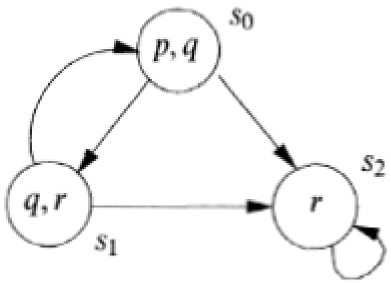




Example (2/2)

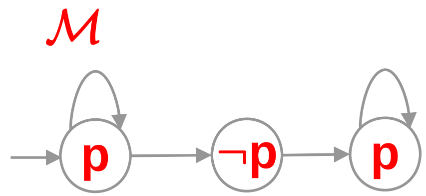
- $\mathcal{M}, s_0 \models p \wedge q$, $\mathcal{M}, s_0 \models \neg r$, $\mathcal{M}, s_0 \models$
- $\mathcal{M}, s_0 \models$ EX $(q \wedge r)$
- $\mathcal{M}, s_0 \models \neg$AX$(q \wedge r)$
- $\mathcal{M}, s_0 \models \neg$EF$(p \wedge r)$
- $\mathcal{M}, s_2 \models$ EG r
- $\mathcal{M}, s_0 \models$ AF r
- $\mathcal{M}, s_0 \models$ E $[(p \wedge q)$ U $r]$
- $\mathcal{M}, s_0 \models$ A $[p$ U $r]$
- $\mathcal{M}, s_0 \models$ AG $(p \vee q \vee r \rightarrow$ EF EG r)

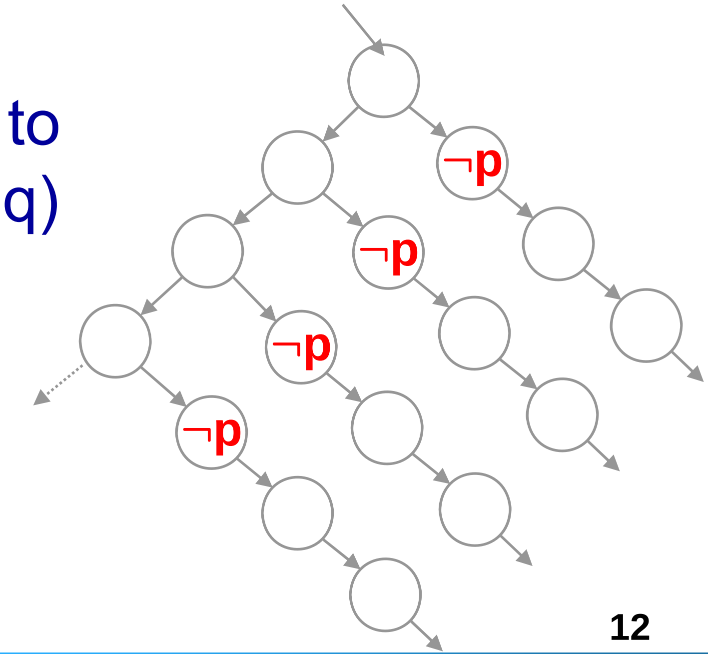




CTL is not more expressive than LTL

- CTL cannot select a range of paths
  - F G p in LTL is not equivalent to AF AG p
    - $\mathcal{M}, s_0 \models$ F G p but $\mathcal{M}, s_0 \not\models$ AF AG p
    - AF AG p is strictly stronger than F G p
    - AF EG p is strictly weaker than F G p
- Similarly, F p $\rightarrow$ F q is not equivalent to AF p $\rightarrow$ AF q, neither to AG (p $\rightarrow$ AF q)
- Remark
  - F X p ≡ X F p in LTL
  - AF AX p is not equivalent to AX AF p

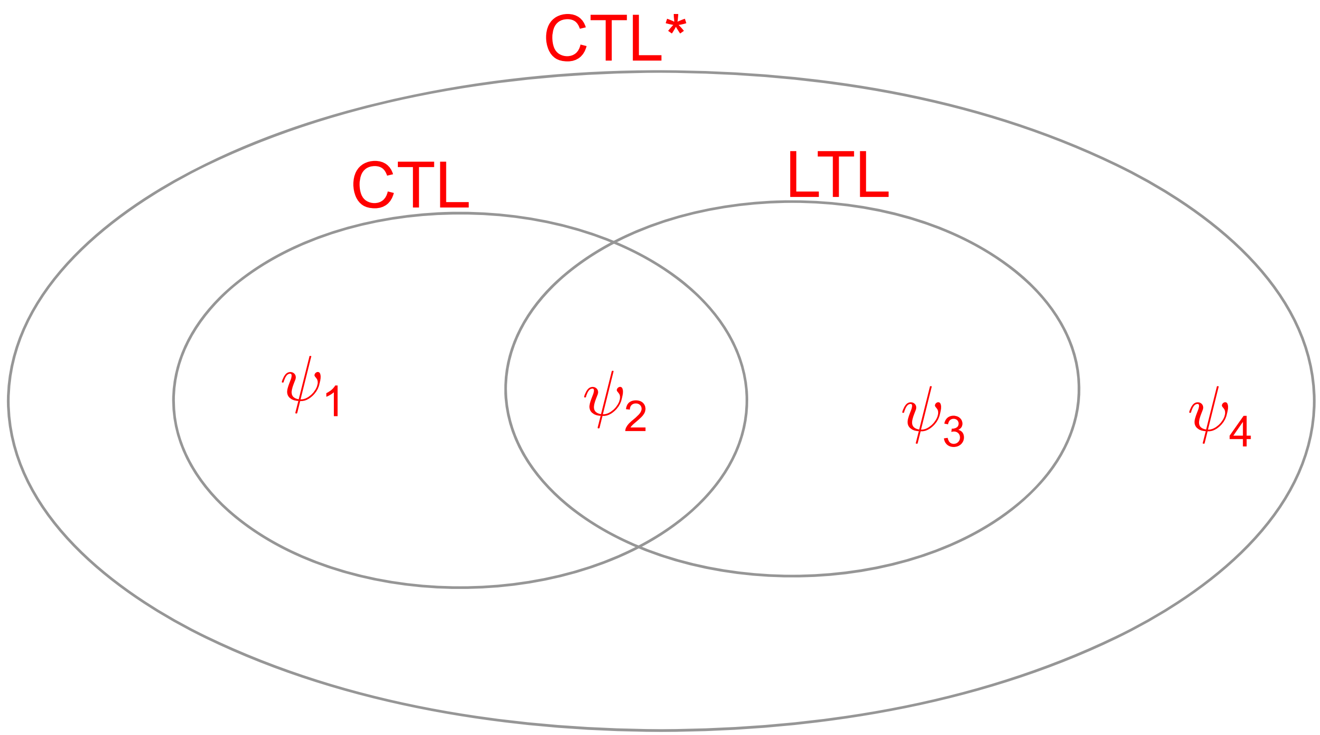




CTL*

- CTL* combines the expressive powers of LTL and CTL
- Syntax of CTL*
  - State formula ϕ ::= T | p | $\neg\ \phi$ | $\phi \wedge \phi$ | A [α] | E[α]
  - Path formula α ::= ϕ | $\neg\ \alpha$ | $\alpha \wedge \alpha$ | α U α | G α | F α | X α
- LTL is a subset of CTL*
  - LTL formula α is equivalent to A[α] in CTL*
- CTL is a subset of CTL*
  - We restrict α ::= ϕ U ϕ | G ϕ | F ϕ | X ϕ
    - No boolean connectives in path formula
      - Not real limitation. See page 6
    - No nesting of the path modalities X,F, and G

Relationship between LTL,CTL, and CTL*



GCTL Formulas in CWB-NC

- tt, ff : true, false
- {act_list} is satisfied by an action a if a appears in act_list
- {- act_list} is satisfied by an action a if a is not included in act_list
- ~ p is true if p is false
- Example
  - prop can_deadlock = E F ~{- }
  - prop recv_guarantee = A G ({send} -> F{'receive})
  - prop fair_recv_guarantee =
    A ((G F {- t}) -> (G {send} -> F {'receive}))

Peterson's Mutual Exclusion Protocol

```
proc Sys = (P1|P2|Flag_1_0|Flag_2_0|Turn1|NCRIT0)
\{flag1_0,flag1_1,flag2_0,flag2_1,turn1,turn2,test_flag1_0,test_flag1_1,t
      est_flag2_0,test_flag2_1,test_turn1,test_turn2,inc_cnt,dec_cnt}
proc P1   = flag1_1.turn1.P1'
proc P1'  = test_flag2_0.P1'' + test_turn2.P1'' + test_flag2_1.P1' +
      test_turn1.P1'
proc P1'' = inc_cnt.dec_cnt.flag1_0.P1
proc P2   = flag2_1.turn2.P2'
proc P2'  = test_flag1_0.P2'' + test_turn1.P2'' + test_flag1_1.P2' +
      test_turn2.P2'
proc P2'' = inc_cnt.dec_cnt.flag2_0.P2
proc UpdateF1 = 'flag1_0.Flag_1_0 + 'flag1_1.Flag_1_1
proc Flag_1_0 = 'test_flag1_0.Flag_1_0 + UpdateF1
proc Flag_1_1 = 'test_flag1_1.Flag_1_1 + UpdateF1
proc UpdateF2 = 'flag2_0.Flag_2_0 + 'flag2_1.Flag_2_1
proc Flag_2_0 = 'test_flag2_0.Flag_2_0 + UpdateF2
proc Flag_2_1 = 'test_flag2_1.Flag_2_1 + UpdateF2
proc UpdateT = 'turn1.Turn1 + 'turn2.Turn2
proc Turn1   = 'test_turn1.Turn1 + UpdateT
proc Turn2   = 'test_turn2.Turn2 + UpdateT
proc NCRIT0  = 'inc_cnt.cnt_1.NCRIT1
proc NCRIT1  = 'inc_cnt.cnt_2.NCRIT2 + 'dec_cnt.cnt_0.NCRIT0
proc NCRIT2  = 'dec_cnt.cnt_1.NCRIT1
```

*** Verification through equivalence**
  *** obseq, trace inclusion**

proc Spec = cnt_1.cnt_0.Spec

*** Verification through model checking**

```
prop ab1 =
 A G ({cnt_1} -> X ( {t} W {cnt_0}) )

prop ab2 =
 A G ({cnt_0} -> X ( {t} W {cnt_1}) )

prop ab3 =
 A G ~{cnt_2}

prop REQ = ab1 /\ ab2 /\ ab3
```