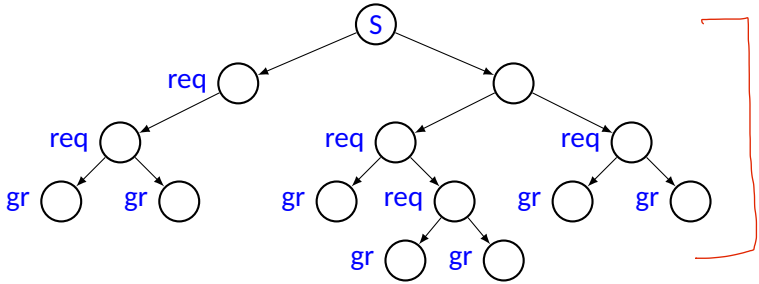
Artificial Intelligence: Foundations & Applications

Temporal Logic and Applications

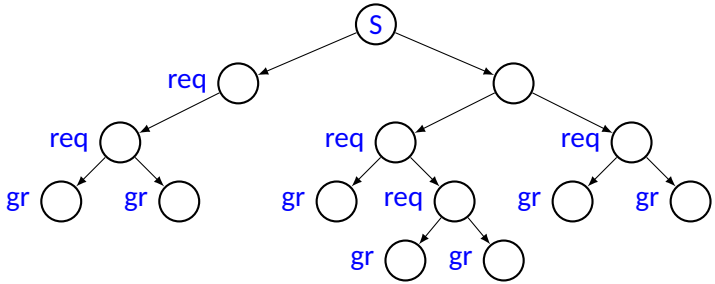
Prof. Partha P. Chakrabarti & Arijit Mondal
Indian Institute of Technology Kharagpur

CTL example



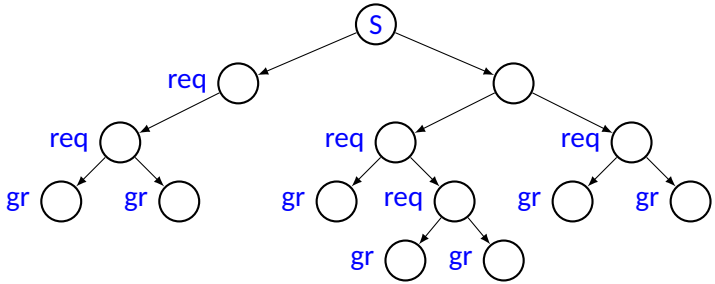
- **From *S* the system always makes a request in future:** A F req

CTL example



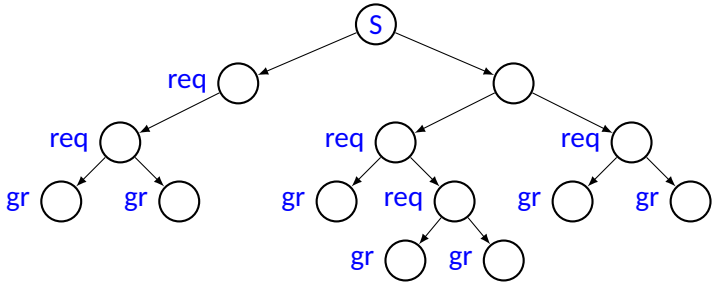
- **From *S* the system always makes a request in future:** *AF req*

CTL example



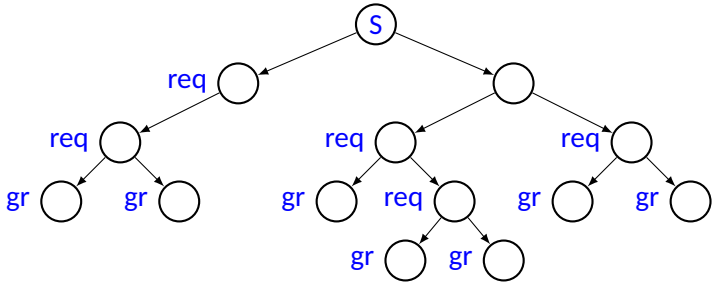
- From *S* the system always makes a request in future: $AF\ req$
- All requests are eventually granted: $AG(req \rightarrow AF\ gr)$

CTL example



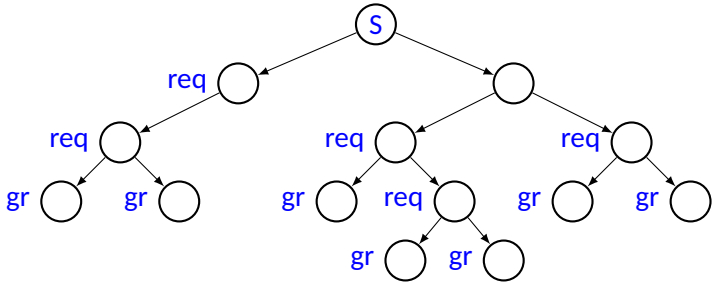
- **From S the system always makes a request in future:** $AF\ req$
- **All requests are eventually granted:** $AG(req \rightarrow AF\ gr)$

CTL example



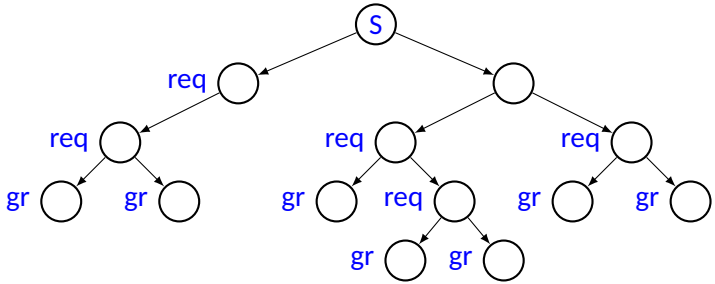
- From S the system always makes a request in future: $AF\ req$
- All requests are eventually granted: $AG(req \rightarrow AF\ gr)$
- Sometimes requests are immediately granted: $EF(req \rightarrow EX\ gr)$

CTL example



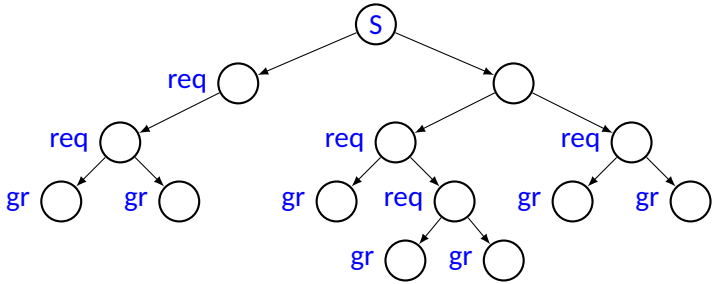
- **From S the system always makes a request in future:** $AF\ req$
- **All requests are eventually granted:** $AG(req \rightarrow AF\ gr)$
- **Sometimes requests are immediately granted:** $EF(req \rightarrow EX\ gr)$

CTL example



- **From S the system always makes a request in future:** $AF\ req$
- **All requests are eventually granted:** $AG(req \rightarrow AF\ gr)$
- **Sometimes requests are immediately granted:** $EF(req \rightarrow EX\ gr)$
- **Requests are held till grant is received:** $AG(req \rightarrow A(req\ U\ gr))$

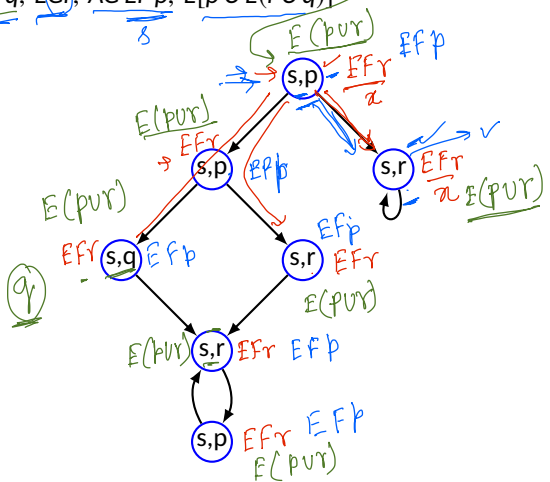
CTL example



- **From S the system always makes a request in future:** $AF\ req$
- **All requests are eventually granted:** $AG(req \rightarrow AF\ gr)$
- **Sometimes requests are immediately granted:** $EF(req \rightarrow EX\ gr)$
- **Requests are held till grant is received:** $AG(req \rightarrow A(req\ U\ gr))$

CTL example

- **True properties:** EFr, AFr, AGs, $AG\,EF\,r$, $E[p\,U\,q]$, $E[E(p\,U\,r)\,U\,q]$, $E[p\,U\,E(q\,U\,r)]$
- **False properties:** AFq, EGr, $AG\,EF\,p$, $E[p\,U\,E(r\,U\,q)]$



CTL Model Checking

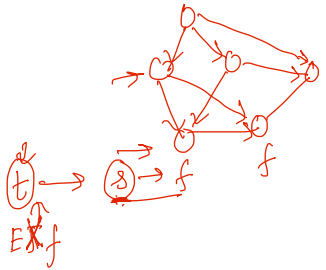
- It checks whether a given CTL formula f holds on a given Kripke structure M i.e., $M \models f$
- Need to have modalities for EX, EU and EG
  - Other modalities can be expressed using EX, EU and EG
    - $AF\, f \equiv \neg EG\, \neg f$
    - $AG\, f \equiv \neg EF\, \neg f$
    - $A(f\, U\, g) \equiv (\neg EG\, \neg g) \wedge (\neg E[\neg g\, U\, (\neg f \wedge \neg g)])$
- Basic procedure
  - The set Sat(f) of all states satisfying f is computed recursively
  - $M \models f$ if and only if $S_0 \subseteq Sat(f)$

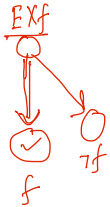
CTL Model Checking: *EX f*

function CheckEX(f)

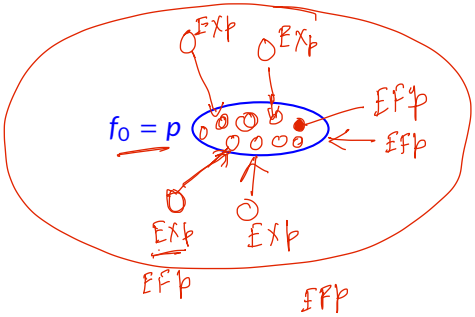
1. $S_f = \{s \in S \mid f \in L(s)\}$
2. **while** $S_f \neq \emptyset$
3. **Choose** $s \in S_f$
4. $S_f = S_f - \{s\}$
5. **for all** t **such that** $(t, s) \in T$
6. **if** $f \notin L(t)$
7. $L(t) = L(t) \cup \{EXf\}$
8. **endif**
9. **end for**
10. **end while**

$T \equiv (PS, NS)$

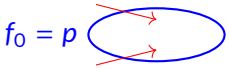




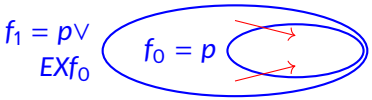
CTL Model Checking: *EF f*



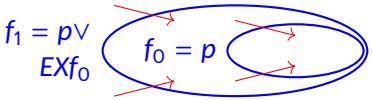
CTL Model Checking: *EF f*



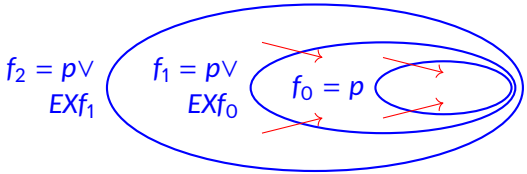
CTL Model Checking: *EF f*



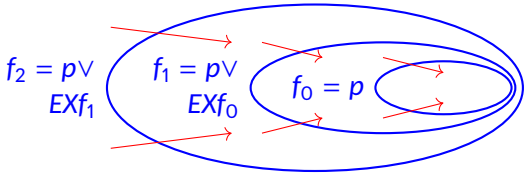
CTL Model Checking: *EF f*



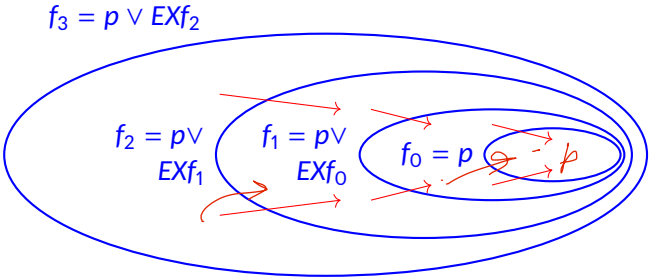
CTL Model Checking: *EF f*



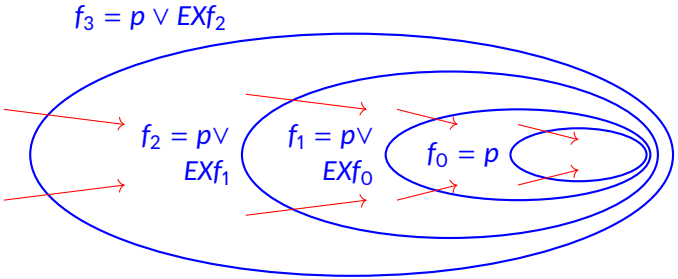
CTL Model Checking: *EF f*



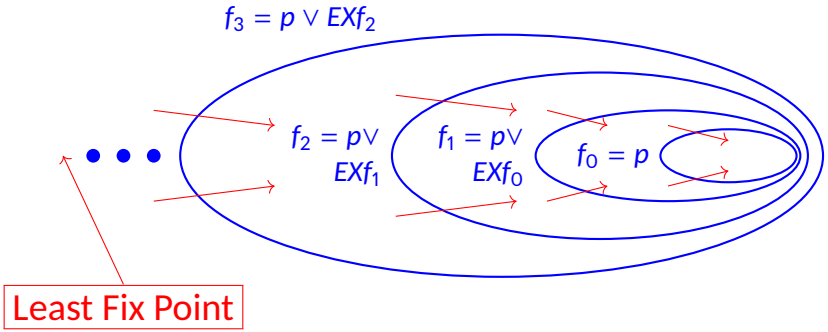
CTL Model Checking: *EF f*



CTL Model Checking: *EF f*



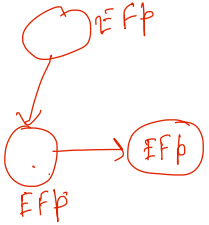
CTL Model Checking: *EF f*



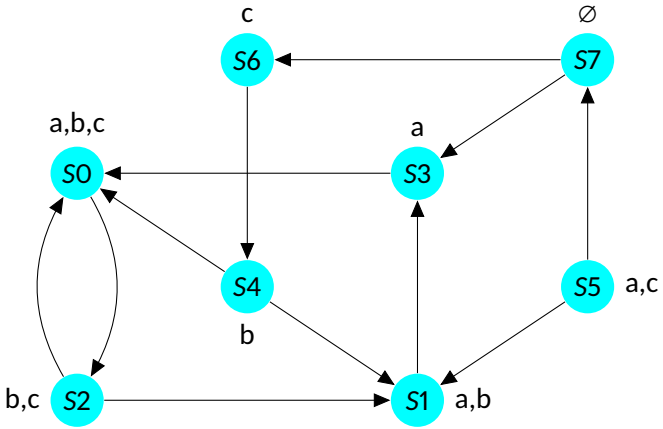
CTL Model Checking: *EF p*

function CheckEF(p)

1. $S_p = \{s \in S \mid p \in L(s)\}$
2. **for all** $s \in S_p$ **do** $L(s) = L(s) \cup \{EFp\}$
3. **while** $S_p \neq \emptyset$
4. **Choose** $s \in S_p$
5. $S_p = S_p - \{s\}$
6. **for all** t **such that** $(t\ s) \in T$
7. **if** $\{EFp\} \notin L(t)$
8. $L(t) = L(t) \cup \{EFp\}$
9. $S_p = S_p \cup t$
10. **endif**
11. **end for**
12. **end while**



Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



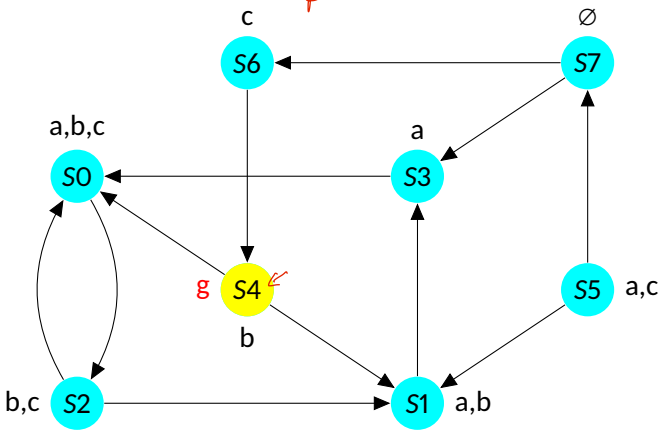
Let $p = (\overline{(a \oplus c)} \wedge (a \oplus b))$

$S_p = \{\}$

$\{\} \vdash g$

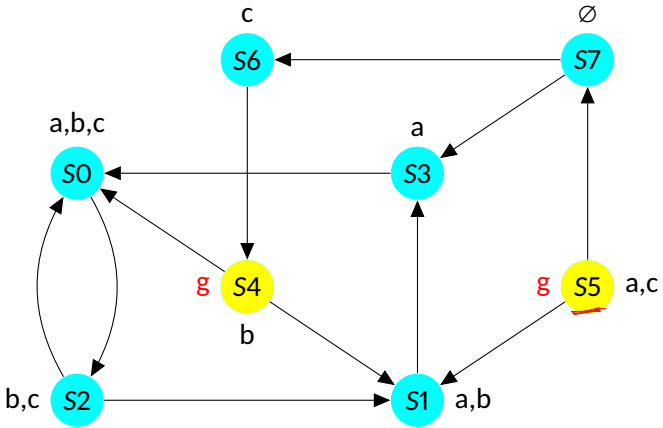
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$

p

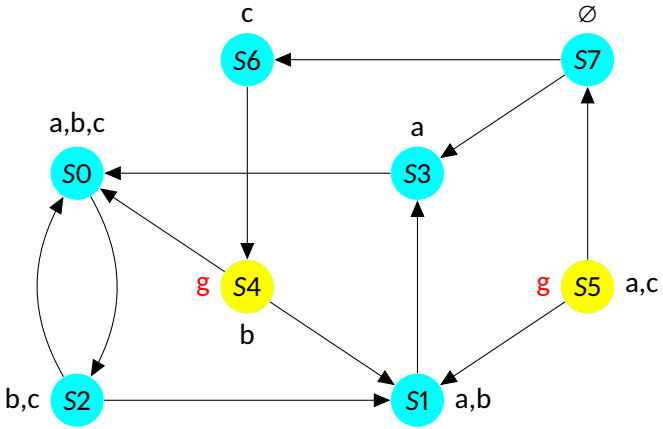


$S4 \models g.$ $S_p = \{S4\}$ $\{S4\} \models g$

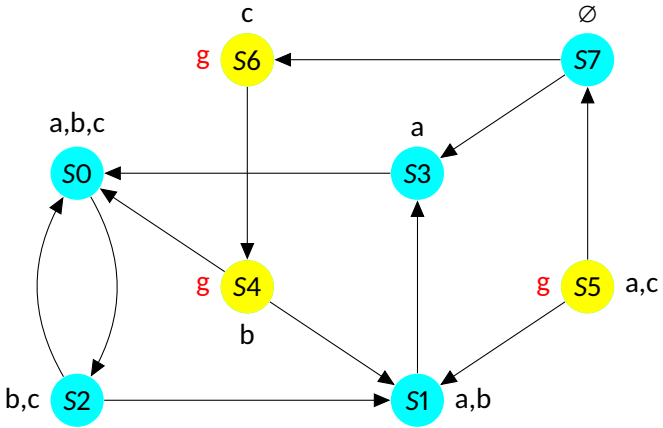
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



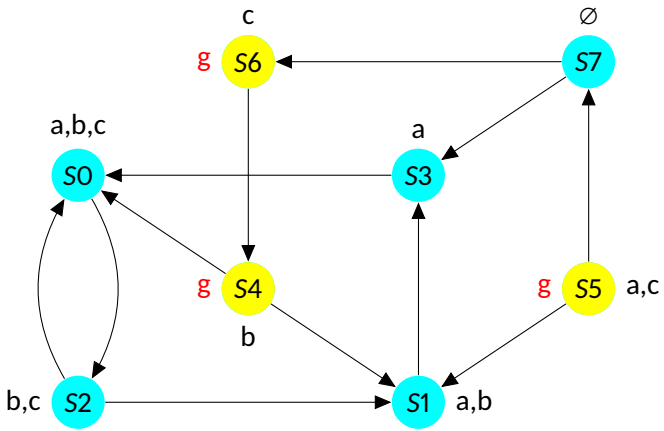
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



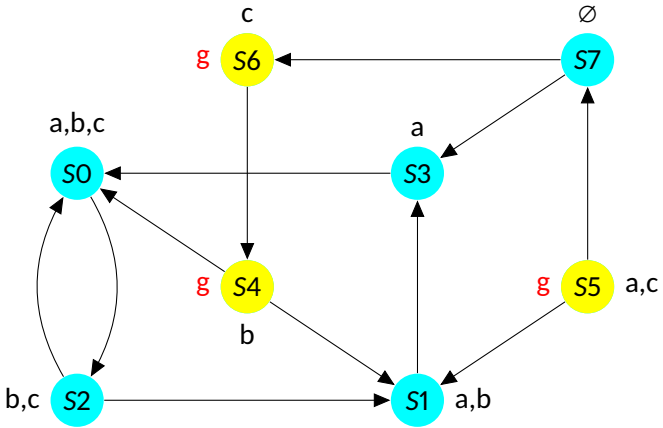
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



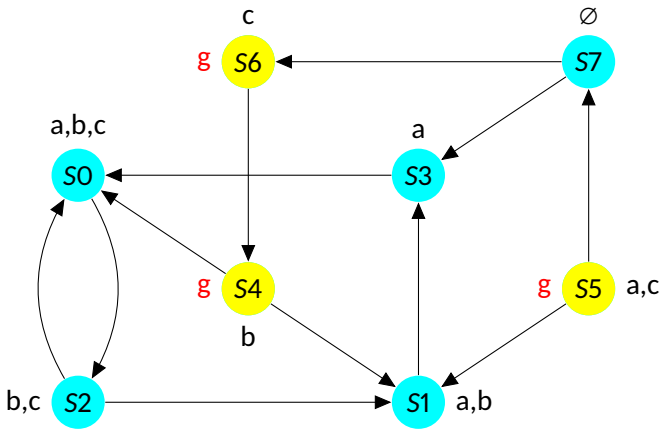
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



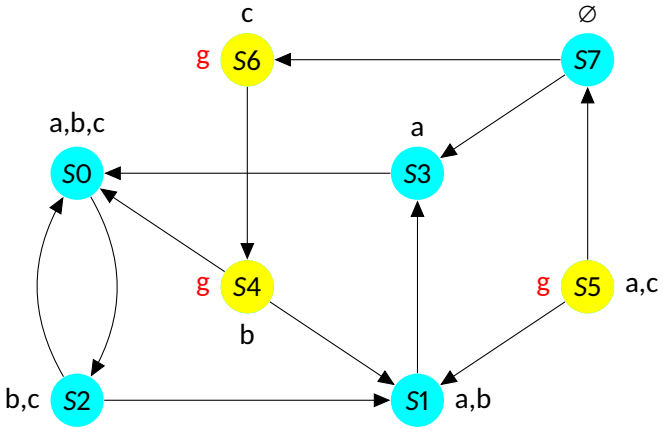
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



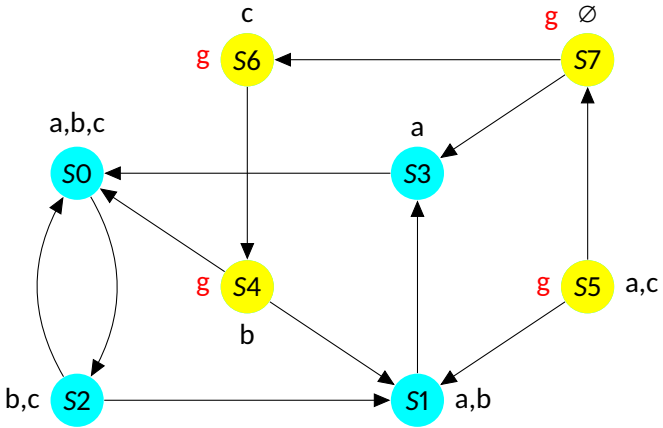
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



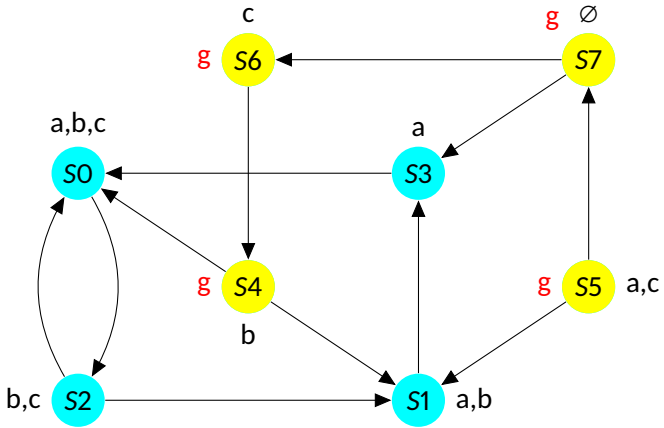
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



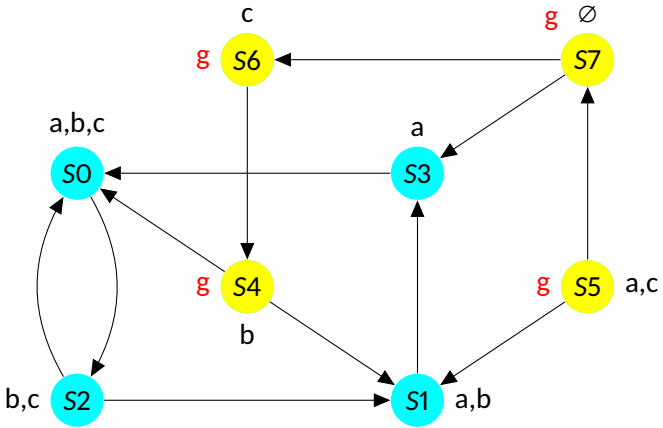
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



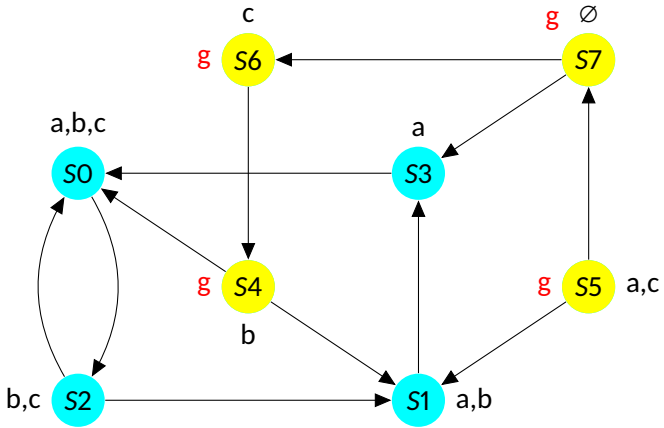
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



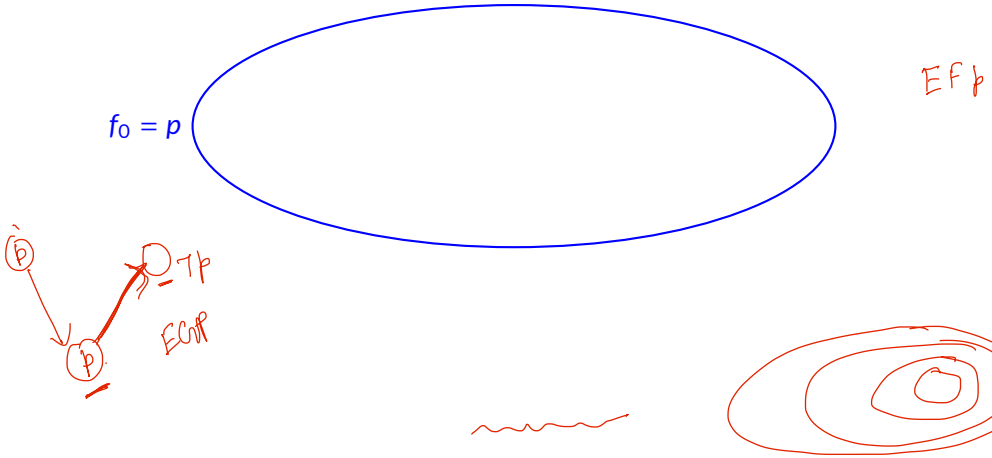
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



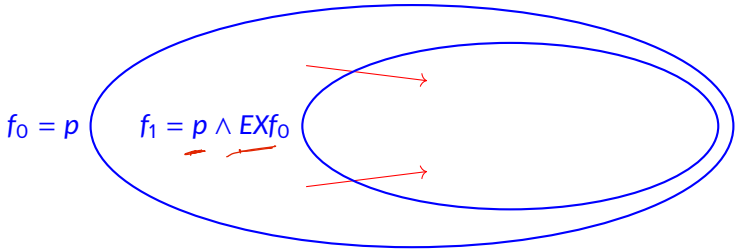
Example: $g = EF(\overline{(a \oplus c)} \wedge (a \oplus b))$



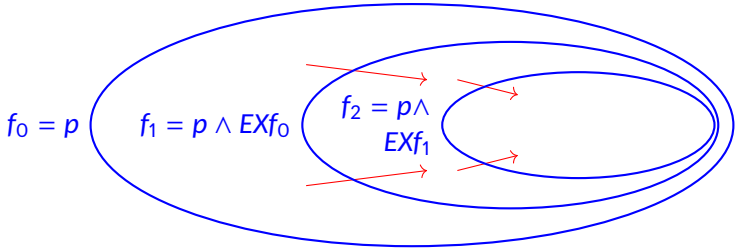
CTL Model Checking: *EG p*



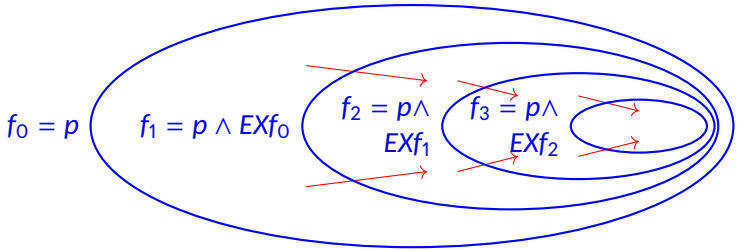
CTL Model Checking: *EG p*



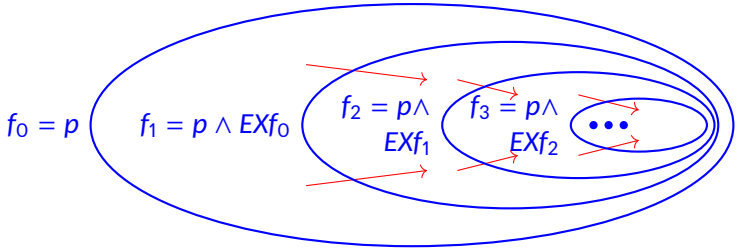
CTL Model Checking: *EG p*



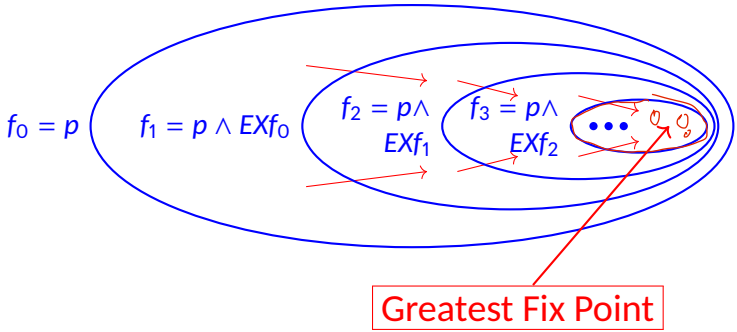
CTL Model Checking: *EG p*



CTL Model Checking: *EG p*



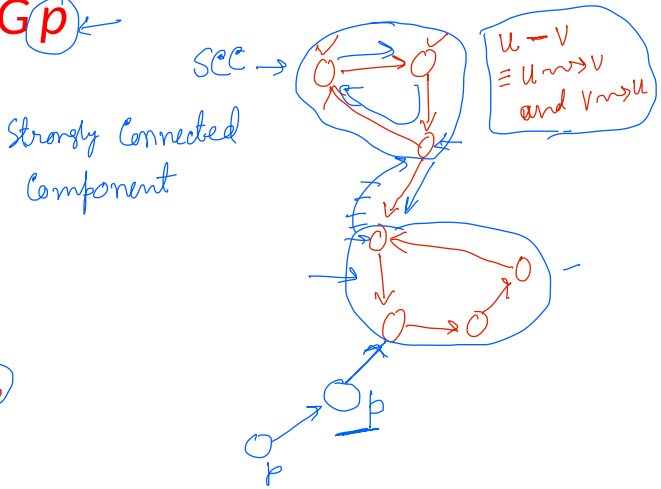
CTL Model Checking: *EG p*



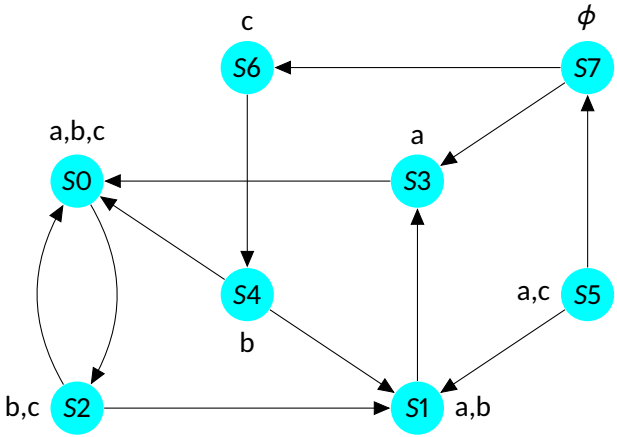
CTL Model Checking: $EG\,p$

function CheckEG(p)

1. $S_p = \{s \in S \mid p \in L(s)\}$
2. $SCC = \{C \mid$ **C is nontrivial SCC of** $S_p\}$
3. $R = \bigcup_{C \in SCC} \{s \mid s \in C\}$
4. **for all** $s \in R$ **do** $L(s) = L(s) \cup \{EGp\}$
5. **while** $R \neq \emptyset$
6. **Choose** $s \in R$
7. $R = R - \{s\}$
8. **for all** t **such that** $(t, s) \in T$ **and** $t \in S_p$
9. **if** $\{EGp\} \notin L(t)$
10. $L(t) = L(t) \cup \{EGp\}$
11. $R = R \cup \{t\}$
12. **endif**
13. **end for**
14. **end while**

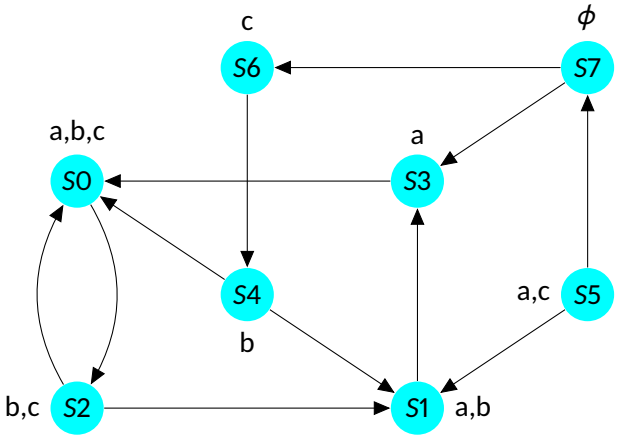


Example: $g = EG\, b$



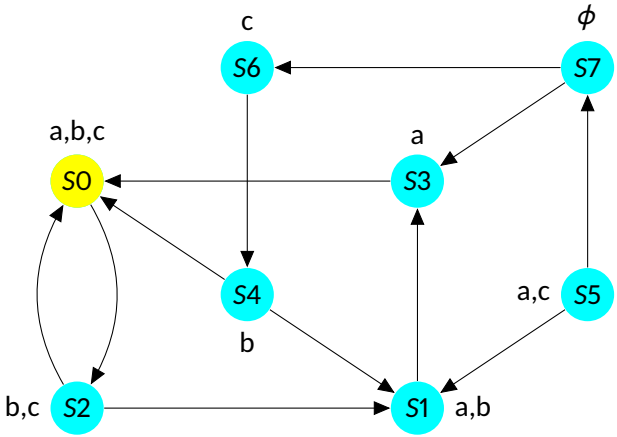
Let $p = b$

Example: $g = EG\, b$



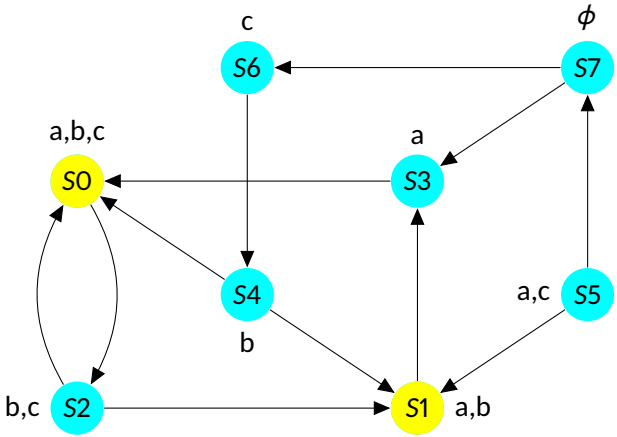
Find states satisfying b.

Example: $g = EG\, b$



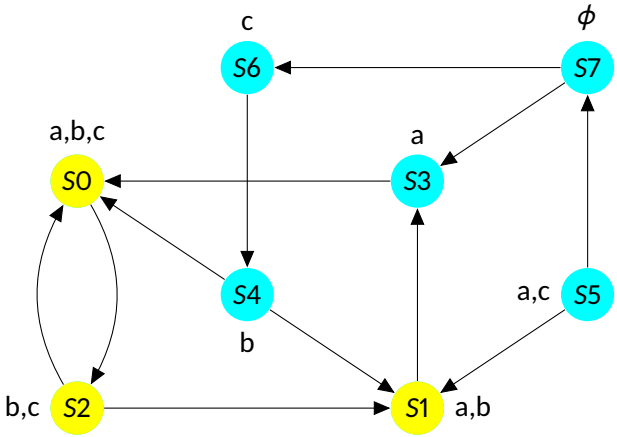
$S_p = \{S0\}$

Example: $g = EG\, b$



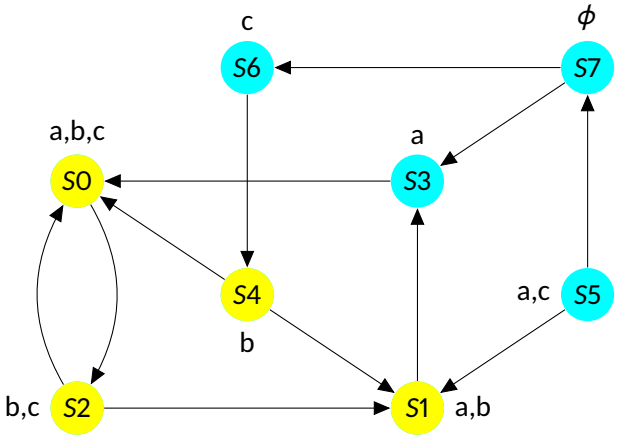
$S_p = \{S0, S1\}$

Example: $g = EG\, b$



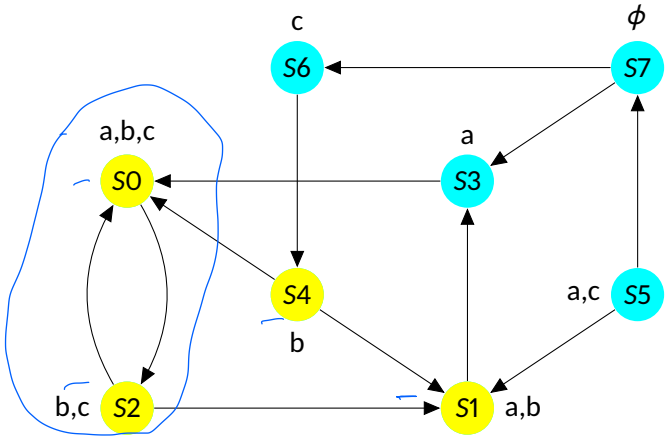
$S_p = \{S0, S1, S2\}$

Example: $g = EG\, b$



$S_p = \{S0, S1, S2, S4\}$

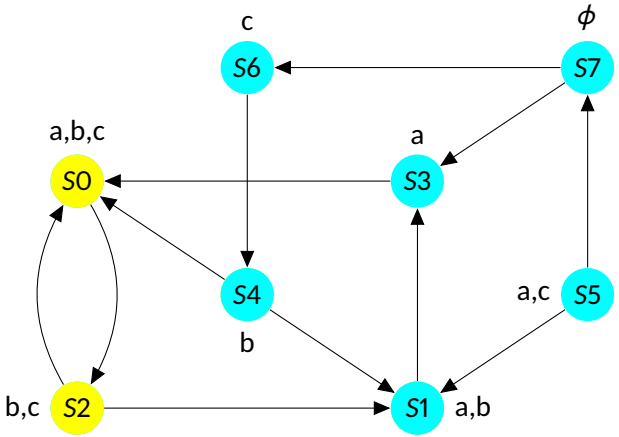
Example: $g = EG\, b$



Find SCC using S_p.

$S_p = \{S0, S1, S2, S4\}$

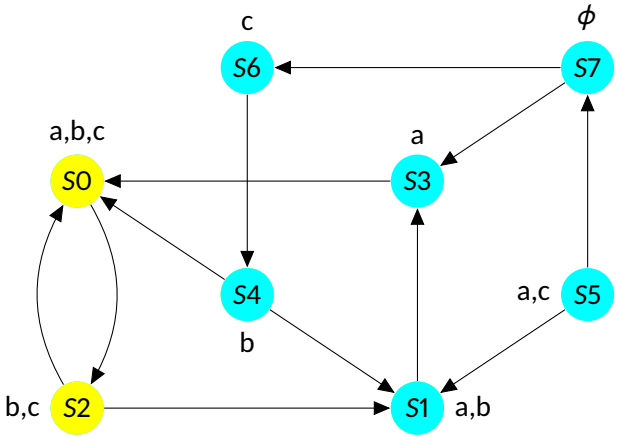
Example: $g = EG\, b$



Find SCC using S_p.

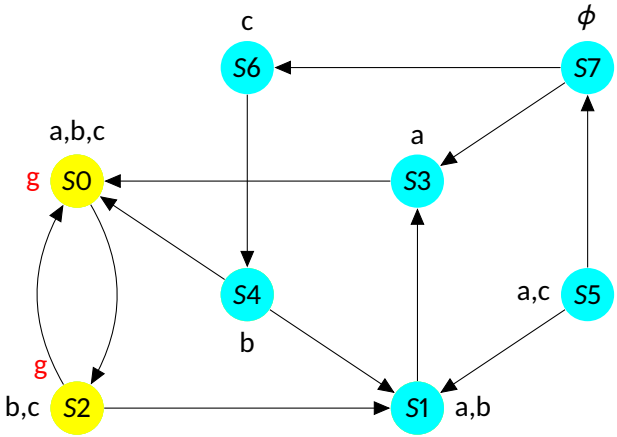
$S_p = \{S0, S1, S2, S4\}$

Example: $g = EG\, b$



$R = \{S0, S2\}$

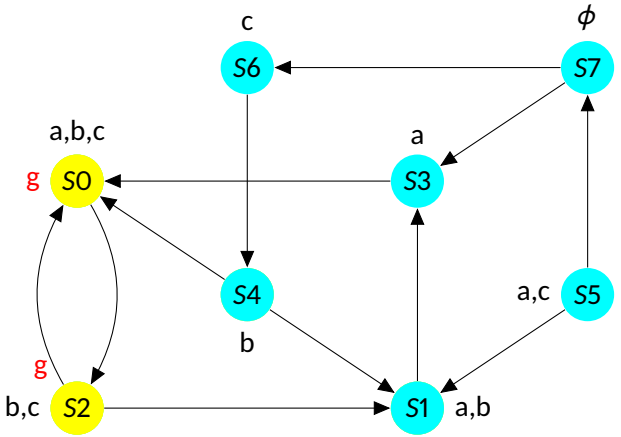
Example: $g = EG\, b$



$R = \{S0, S2\}$

$S_g = \{S0, S2\}$

Example: $g = EG\, b$

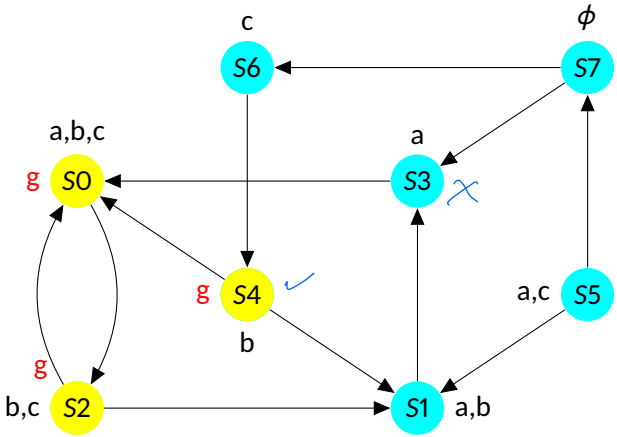


Choose S0.

$R = \{S2\}$

$S_g = \{S0, S2\}$

Example: $g = EG\, b$

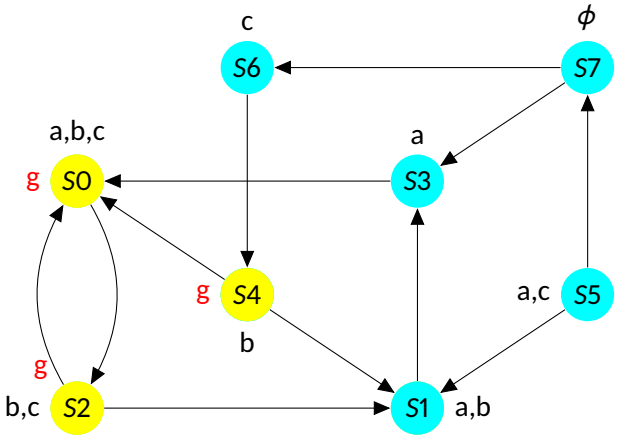


$S4 \models g.$

$R = \{S2\}$

$S_g = \{S0, S2\}$

Example: $g = EG\, b$

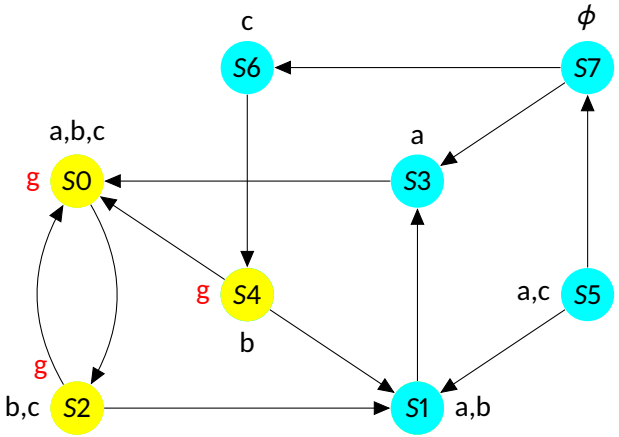


$S4 \models g.$

$R = \{S2, S4\}$

$S_g = \{S0, S2, S4\}$

Example: $g = EG\, b$

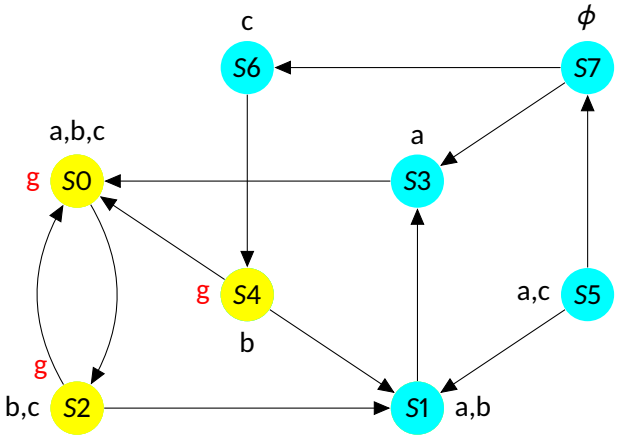


Choose S2.

$R = \{S4\}$

$S_g = \{S0, S2, S4\}$

Example: $g = EG\, b$

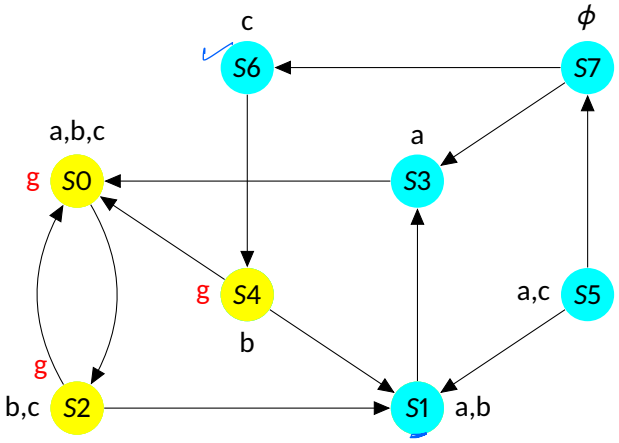


Nothing to explore.

$R = \{S4\}$

$S_g = \{S0, S2, S4\}$

Example: $g = EG\, b$

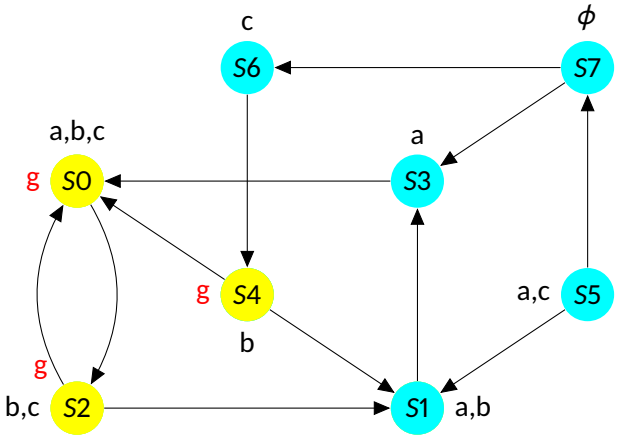


Choose S4.

$R = \{\}$

$S_g = \{S0, S2, S4\}$

Example: $g = EG\, b$

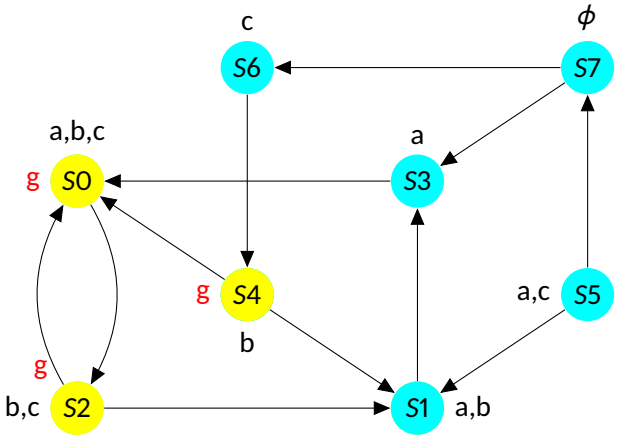


Nothing to explore.

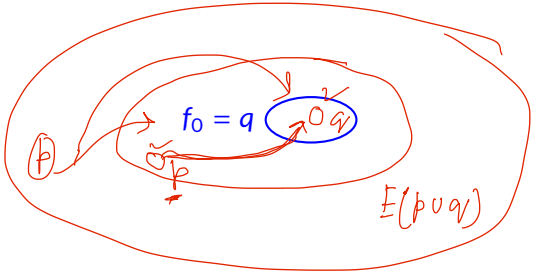
$R = \{\}$

$S_g = \{S0, S2, S4\}$

Example: $g = EG\, b$

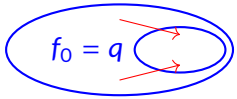


CTL Model Checking: $E(p\,U\,q)$

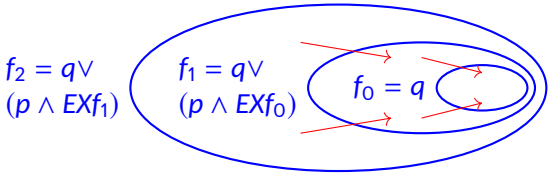


CTL Model Checking: $E(p\,U\,q)$

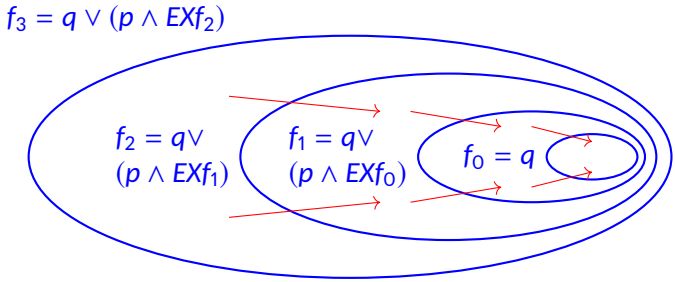
$f_1 = q \vee (p \wedge EXf_0)$



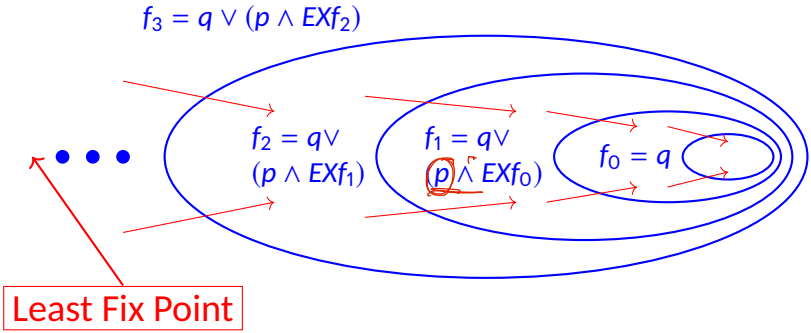
CTL Model Checking: $E(p\,U\,q)$



CTL Model Checking: $E(p\,U\,q)$



CTL Model Checking: $E(p\,U\,q)$

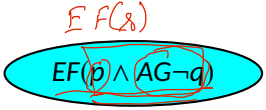


CTL Model Checking: $E(p\,U\,q)$

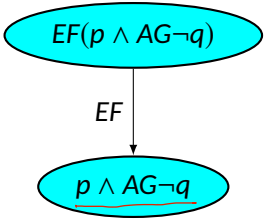
function CheckEU(p,q)

1. $S_q = \{s \in S \mid q \in L(s)\}$
2. **for all** $s \in S_q$ **do** $L(s) = L(s) \cup \{E(p\,U\,q)\}$
3. **while** $S_q \neq \emptyset$
4. **Choose** $s \in S_q$
5. $S_q = S_q - \{s\}$
6. **for all** t **such that** $(t, s) \in T$
7. **if** $\{E(p\,U\,q)\} \notin L(t)$ **and** $p \in L(t)$
8. $L(t) = L(t) \cup \{E(p\,U\,q)\}$
9. $S_q = S_q \cup \{t\}$
10. **endif**
11. **end for**
12. **end while**

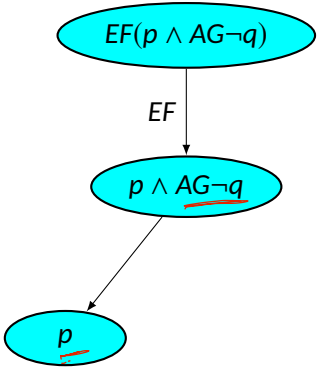
Nested CTL query



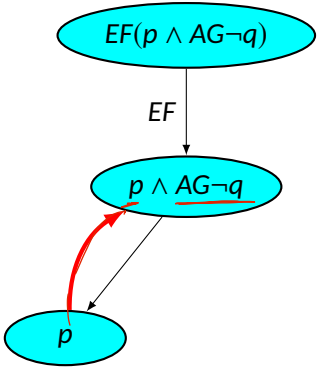
Nested CTL query



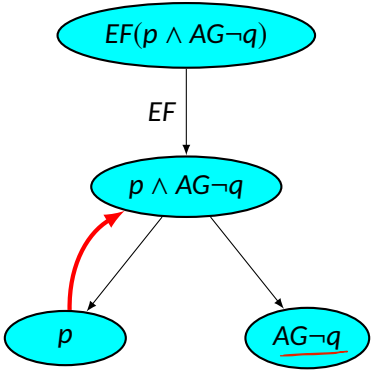
Nested CTL query



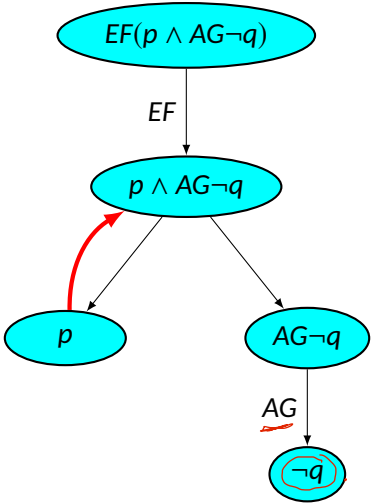
Nested CTL query



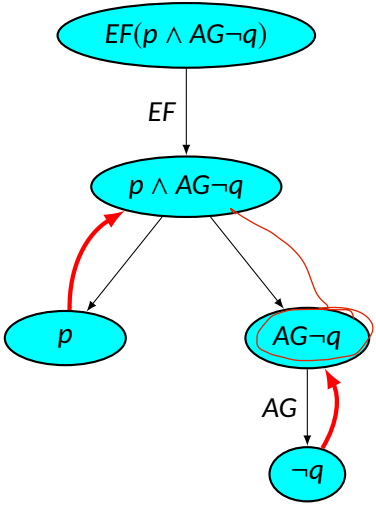
Nested CTL query



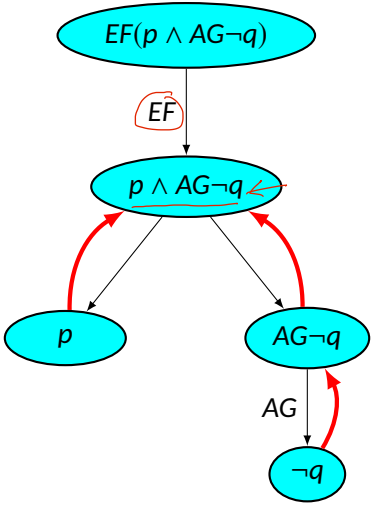
Nested CTL query



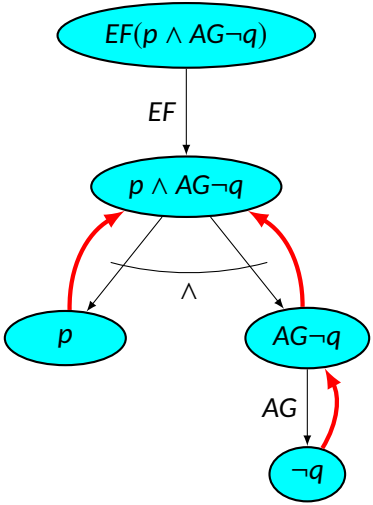
Nested CTL query



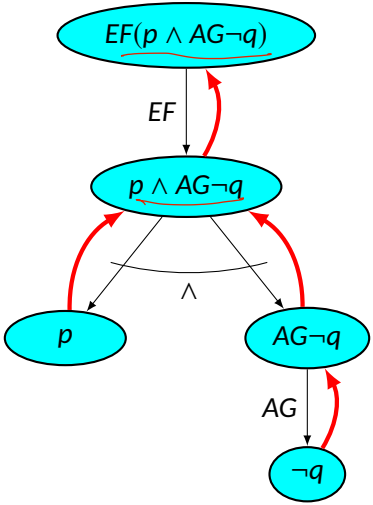
Nested CTL query



Nested CTL query



Nested CTL query



Real Time properties

- Real time systems
  - Predictable response time are necessary for correct operation
  - Safety critical systems like controller for aircraft, industrial machinery are a few examples

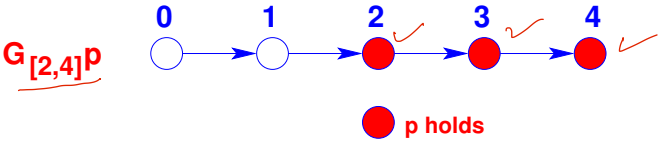
- It is difficult to express complex timing properties
  - Simple: “event p will happen in future”
    - Fp
  - Complex: “event p will happen within at most n time units”
    - $p \vee (Xp) \vee (XXp) \vee \ldots ([XX \ldots \text{n times}]p)$

Bounded Temporal Operators

- **Specify real-time constraints**
  - **Over bounded traces**

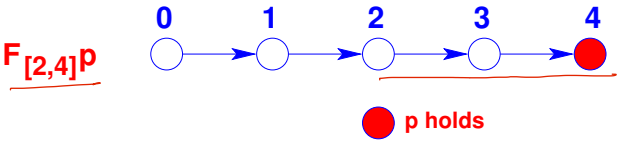
- **Various bounded temporal operators**
  - $G_{[m,n]}p$ – p **always holds between mth and nth time step**
  - $F_{[m,n]}p$ – p **eventually holds between mth and nth time step**
  - $X_{[m]}p$ – p **holds at the mth time step**
  - $p\ U_{[m,n]}\ q$ – q **eventually holds between mth and nth time step and** p **holds until that point of time**

Examples



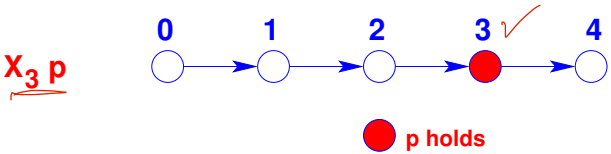
- p holds always between 2nd and 4th time step

Examples



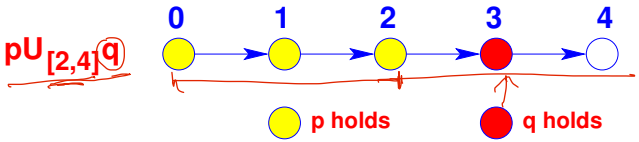
- p holds eventually between 2nd and 4th time step

Examples



- p holds in the 3rd time step

Examples



- q holds eventually between 2nd and 4th time step and p holds until q holds

Timing properties

- **Whenever request is recorded grant should take place within 4 time units**
  - $AG(posedge(req) \rightarrow AF_{[0,4]}\, posedge(gr))$
- **The arbiter will provide exactly 64 time units to high priority user in each grant**
  - $AG(posedge(hpusing) \rightarrow A(\neg negedge(hpusing)\, U_{[64,64]}\, negedge(hpusing)))$

Thank you!