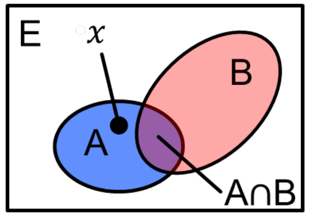
Crash course – Verification of Finite Automata
CTL model-checking

Xiaoxi He

Reminders – Big picture

Objective	Verify properties over DES models Formal method ⇒ Absolute guarantee!
Problem	Combinatorial explosion → Huge amount of states, computationally intractable
Solution	Work with sets of states → Symbolic Model-Checking → (O)BDDs

Reminders – First exercise session



Sets

$\Leftrightarrow$

$\psi_E = 1$
$\psi_A = f$
$\psi_B = g$
$\psi_{A\cap B} = f \cdot g$

Boolean functions/
Characteristic functions

- A
- $s \in A$

$\xleftrightarrow{\sigma(\cdot)}$

- ψ_A
- $\psi_A(\sigma(s)) = 1$
  $\sigma(s) \vDash \psi_A$
  or just $s \vDash \psi_A$

Reads "s satisfies ψ_A"

Example:
$\sigma(s) = x_1\overline{x_0} = (1,0)$ and $\psi_A = x_1 + x_0$
$\rightarrow s \vDash \psi_A$?

Equivalence between sets and Boolean equations

$f: x_1 + \overline{x_1}\, x_2 + \overline{x_2}\,\overline{x_3}$

Fall $x_1 = 0$

$f_{|x_1=0}: x_2 + \overline{x_2}\,\overline{x_3}$

Fall $x_2 = 0$

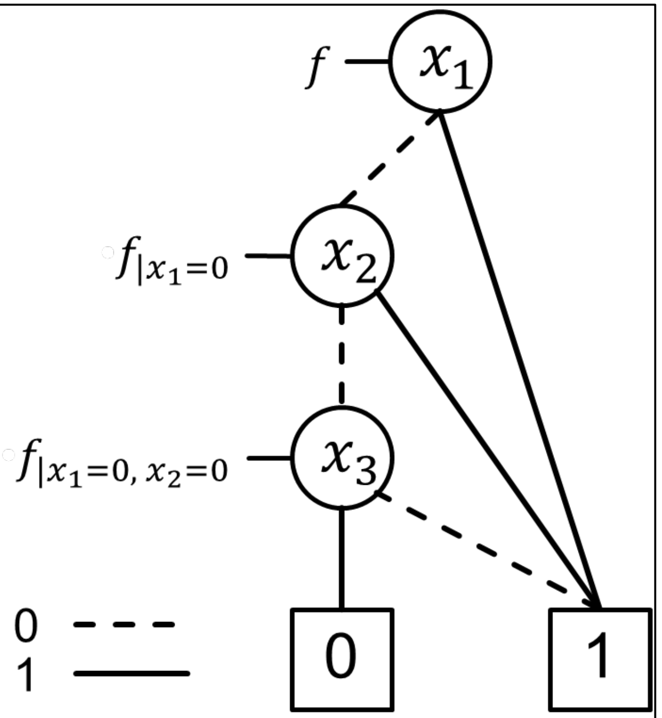
$f_{|x_1=0,\, x_2=0}: \overline{x_3}$

Fall $x_2 = 1$

$f_{|x_1=0,\, x_2=1}: 1$

Fall $x_1 = 1$

$f_{|x_1=1}: 1$



BDD representation of Boolean functions

Today’s menu

1. Reachability of states
2. Comparison of automata
3. Formulation and verification of CTL properties

Can be formulated as reachability problems

Reachability of states

Fairly simple

1. Start from the initial set of states,
2. Compute all states you can transition to in one hop (one transition),
   $\rightarrow$ The successor states,
3. Join the two sets,
4. Iterate from 2. until you reach a fix point.
5. Done !

Is this guarantee to terminate?

Reachability of states

Fairly simple

1. Start from the initial set of states,
2. Compute all states you can transition to in one hop (one transition),
   → The successor states,
3. Join the two sets,
4. Iterate from 2. until you reach a fix point.
5. Done !

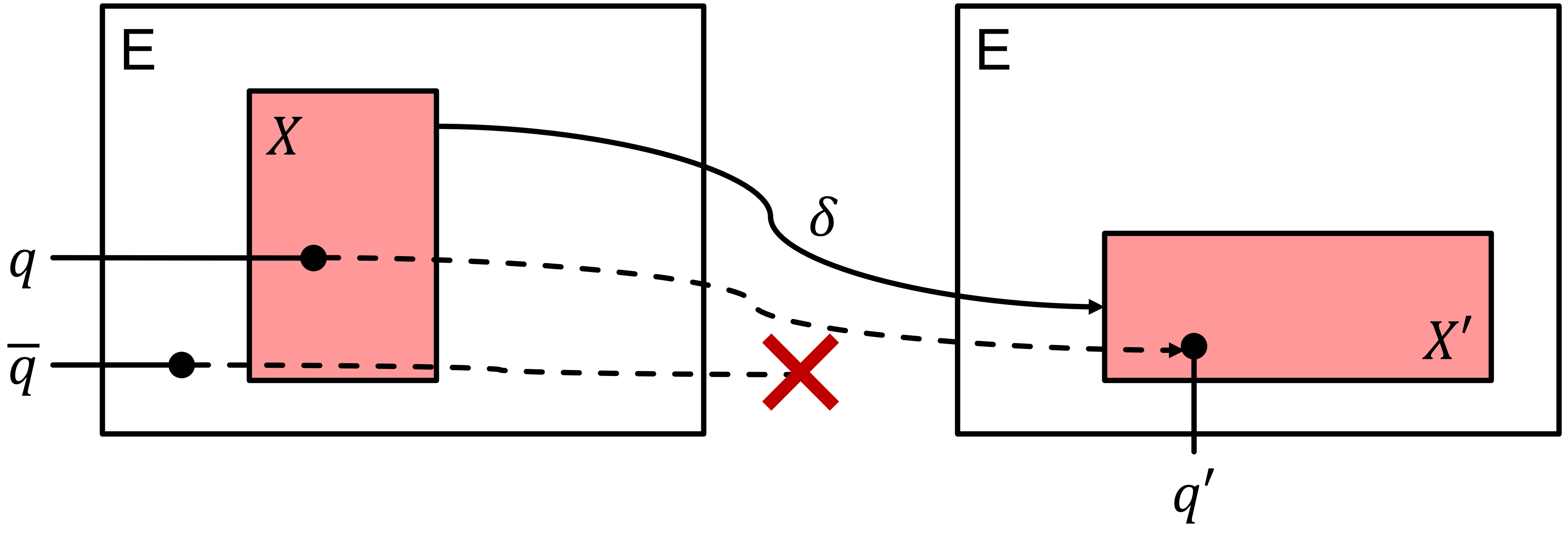
Is this guarantee to terminate?

→ Only if you have a finite model!!

How can we formalize this problem?

Formalization of reachable states

$$\delta : X \subseteq E \longrightarrow X' \subseteq E$$
$$q \;\longmapsto\; q'$$



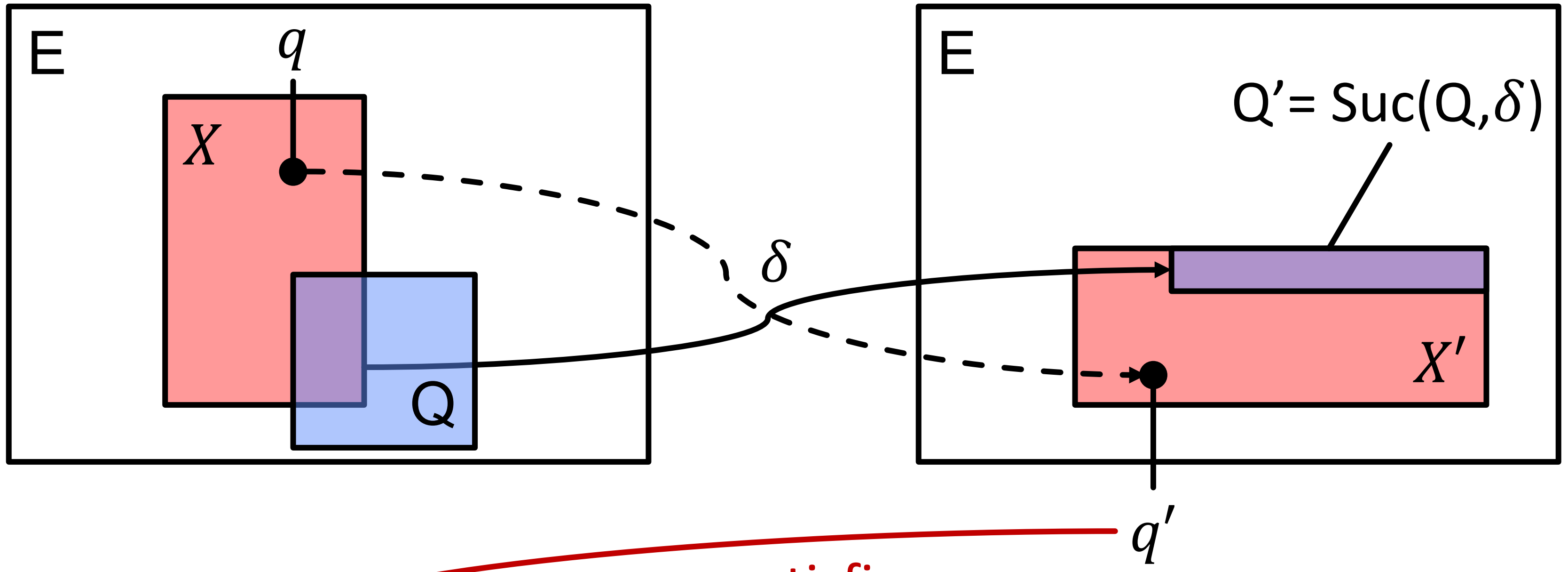
$$q \in X \Leftrightarrow \exists\, q' \in X', \left|\begin{array}{l} \delta(q,q')\ is\ defined \\ \psi_\delta(q,q') = 1 \end{array}\right.$$

$$\overline{q} \notin X \Leftrightarrow \left|\begin{array}{l} \nexists\, q' \in X', \delta(\overline{q},q')\ is\ defined \\ \forall\, q' \in X\,, \psi_\delta(\overline{q},q') = 0 \end{array}\right.$$

Formalization of reachable states

$$\delta : X \subseteq E \longrightarrow X' \subseteq E$$
$$q \;\longmapsto\; q'$$

What is Q'?



satisfies

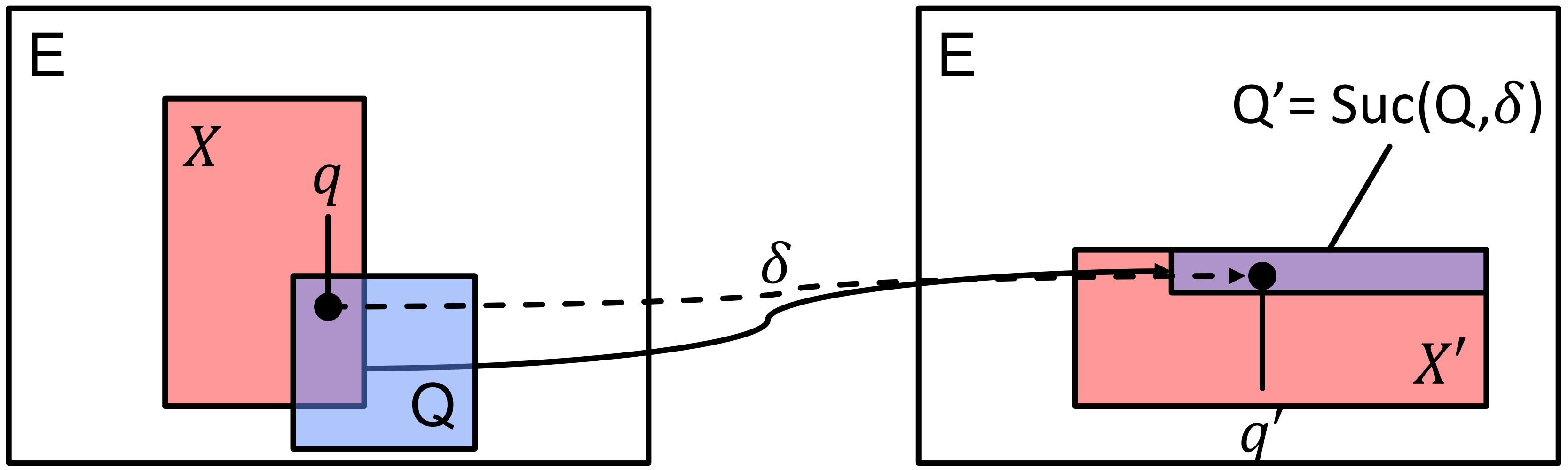
$$q' \in Q' \quad \Rightarrow \quad q' \in X' \quad \Rightarrow \quad \exists q \in X, \psi_\delta(q,q') = 1$$

Not sufficient !

We also need that q belongs to Q : $q \in Q$ or equivalently $\psi_Q(q) = 1$

Formalization of reachable states

$$\delta : X \subseteq E \longrightarrow X' \subseteq E$$
$$q \;\longmapsto\; q'$$



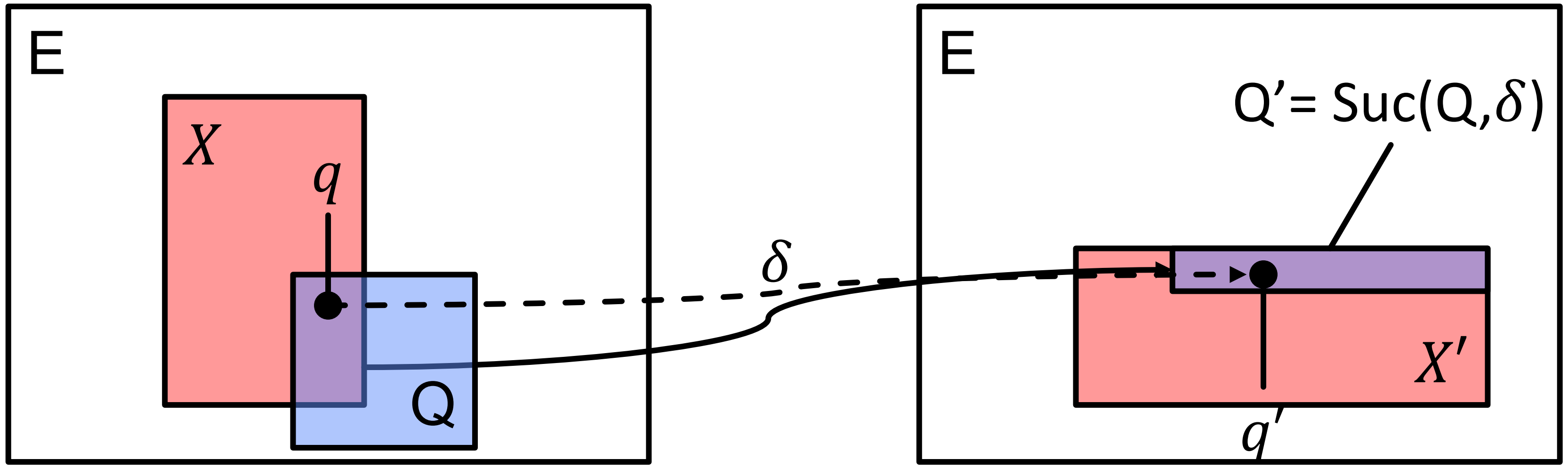
What is Q'?

$$q' \in Q' \;\Leftrightarrow \exists q \in X,\; \psi_Q(q) = 1 \text{ and } \boxed{\psi_\delta(q, q') = 1}$$
$$\Leftrightarrow \exists q \in X,\; \psi_Q(q) \cdot \psi_\delta(q, q') = 1$$

$$Q' = Suc(Q, \delta) = \{q' \mid \exists q \in X, \psi_Q(q) \cdot \psi_\delta(q, q') = 1\}$$

Formalization of reachable states

$$\delta : X \subseteq E \longrightarrow X' \subseteq E$$
$$q \;\mapsto\; q'$$



$$Q' \;=\; Suc(Q,\delta) = \{q' \mid \exists q \in X, \psi_Q(q) \cdot \psi_\delta(q,q') = 1\}$$
$$\Leftrightarrow \;\; \psi_{Q'} = \psi_Q \cdot \psi_\delta$$

Q_R: set of reachable states

$$Q_R \;= Q_0 \cup_{i \geq 0} Suc(Q_i,\delta)$$
$$\Leftrightarrow \;\; \psi_{Q_R} = \psi_{Q_0} + \textstyle\sum_{i \geq 0} \psi_{Q_i} \cdot \psi_\delta$$

Again, finite union if finite model

Comparison of automata

Two automata are equivalent $\Leftrightarrow$ Same <u>input</u> produces same <u>output</u>

$\rightarrow$ Don't compare states!

- Computation of the joint transition function,

$$\psi_\delta(q_1, q_2, q_1', q_2') = (\exists u \; : \; \psi_{\omega_1}(u, q_1, q_1') \cdot \psi_{\omega_2}(u, q_2, q_2'))$$

➢ Get rid of the input

- Computation of the reachable states (method according to previous slides),

$$\psi_Q(q_1, q_2)$$

➢ Compute Q_R

- Computation of the reachable output values,

$$\psi_Y(y_1, y_2) = (\exists q_1, q_2 \; : \; \psi_Q(q_1, q_2) \cdot \psi_{\omega_1}(q_1, y_1) \cdot \psi_{\omega_2}(q_2, y_2))$$

➢ Deduce reachable outputs

- The automata are not equivalent if the following term is true,

$$\exists y_1, y_2 \; : \; \psi_Y(y_1, y_2) \cdot (y_1 \neq y_2)$$

➢ Test for equivalence

Formulation of CTL properties

Based on atomic propositions (ϕ) and quantifiers

Aϕ	→ «**A**ll ϕ»,	ϕ holds on all paths	Quantifiers over paths
Eϕ	→ «**E**xists ϕ»,	ϕ holds on at least one path	
Xϕ	→ «Ne**X**t ϕ»,	ϕ holds on the next state	Path-specific quantifiers
Fϕ	→ «**F**inally ϕ»,	ϕ holds at some state along the path	
Gϕ	→ «**G**lobally ϕ»,	ϕ holds on all states along the path	
ϕ_1Uϕ_2	→ «ϕ_1**U**ntil ϕ_2»,	ϕ_1 holds until ϕ_2 holds	

Formulation of CTL properties

Proper CTL formula: {A,E} {X,F,G,U}ϕ

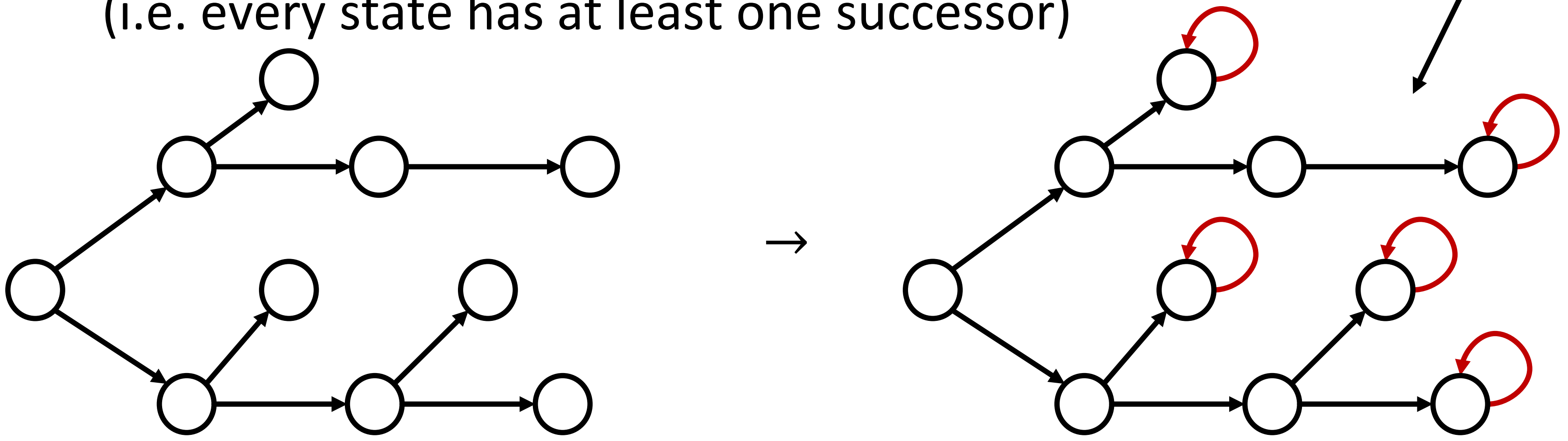
→ Quantifiers **go by pairs**, you need one of each.

Simple "means" that we get rid of leaf nodes…
→ They transition to themselves

Missing Hypothesis

Interpretation on CTL formula

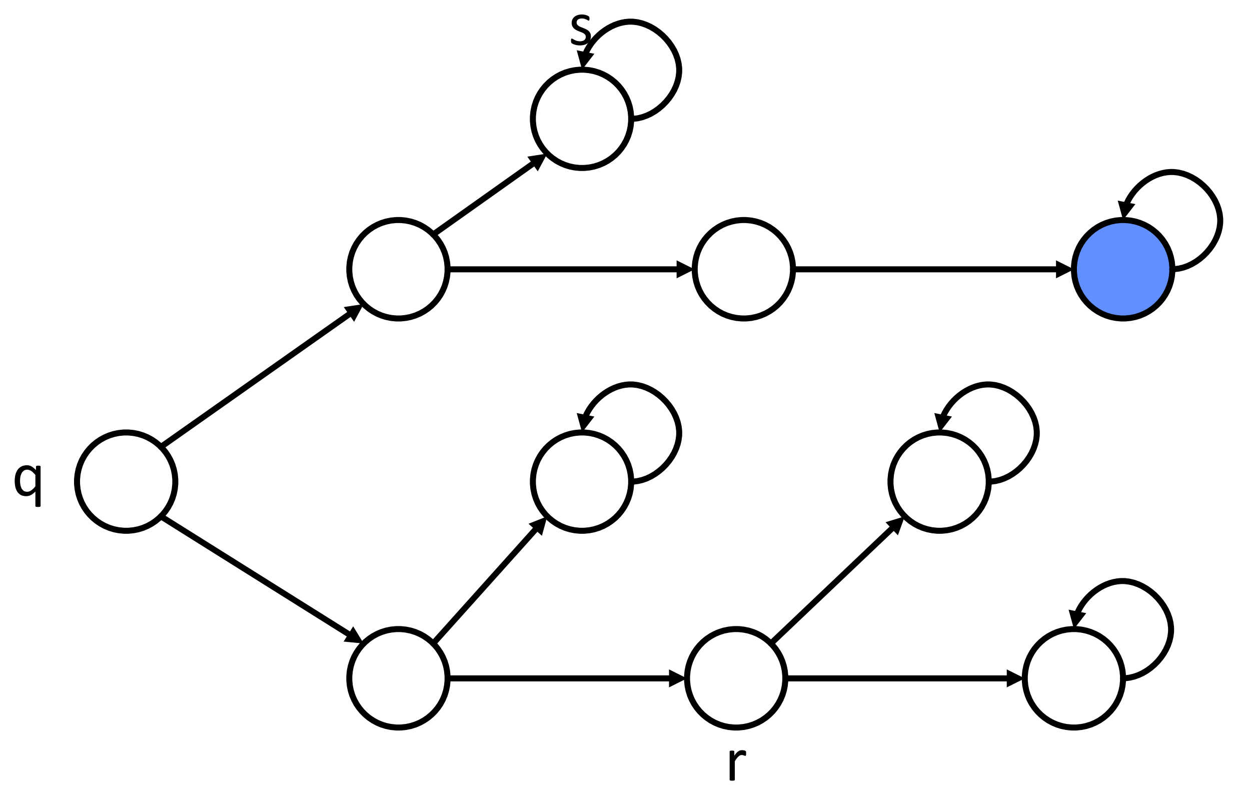
→ Transition functions are **fully defined**
(i.e. every state has at least one successor)



Automaton of interest **Automaton to work with**

Formulation of CTL properties

EF ϕ : "There exists a path along which at some state ϕ holds."



● $\models \phi$

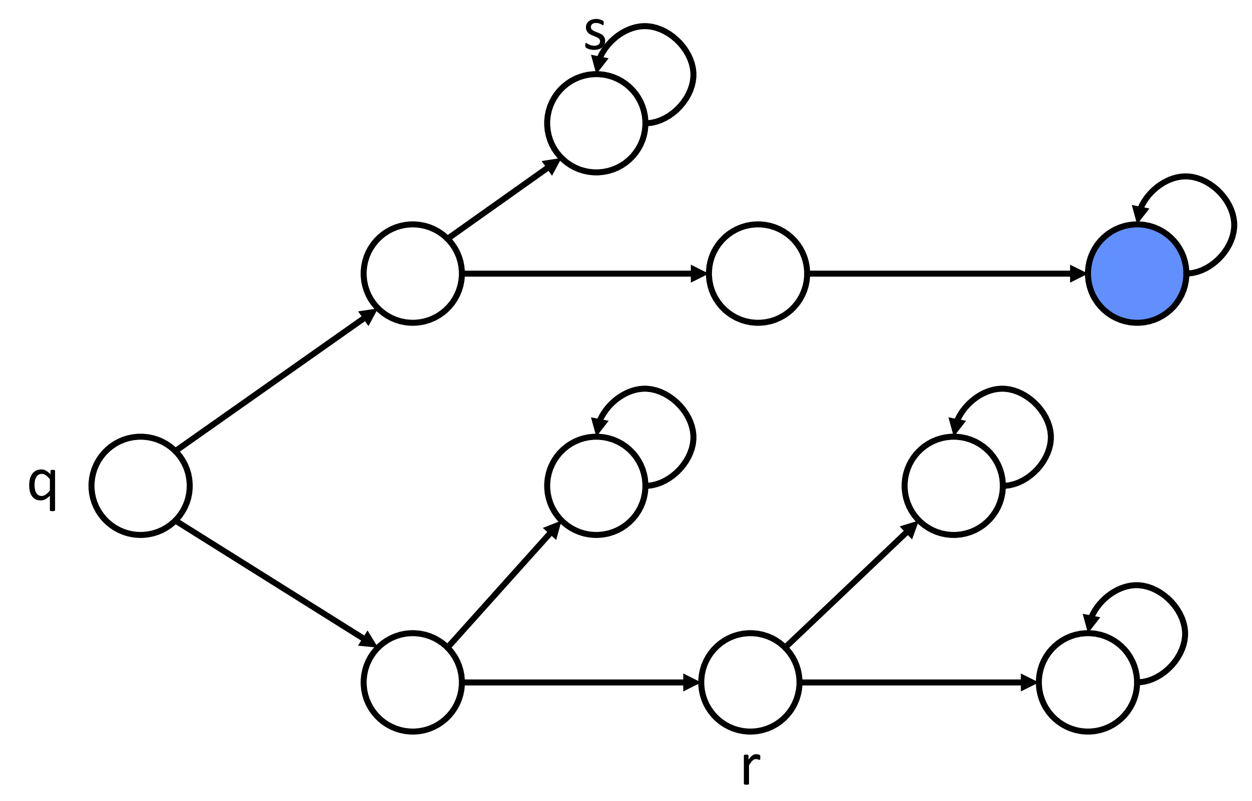
$q \models EF\ \phi$

$r \models ?$

$s \models ?$

Formulation of CTL properties

EF ϕ : "There exists a path along which at some state ϕ holds."



● $\vDash \phi$

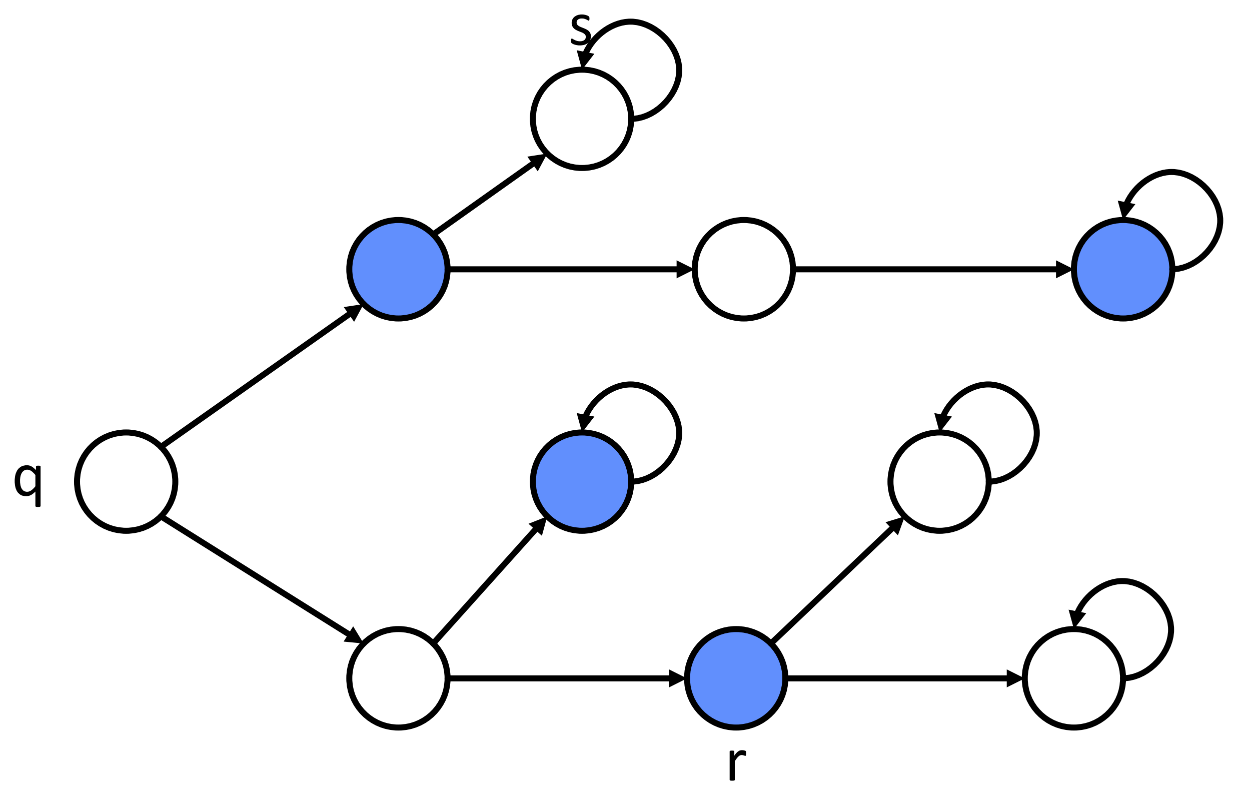
q $\vDash$ EF ϕ

r $\nvDash$ EF ϕ

s $\nvDash$ EF ϕ

Formulation of CTL properties

AF ϕ : "On all paths, at some state ϕ holds ."



● $\models \phi$

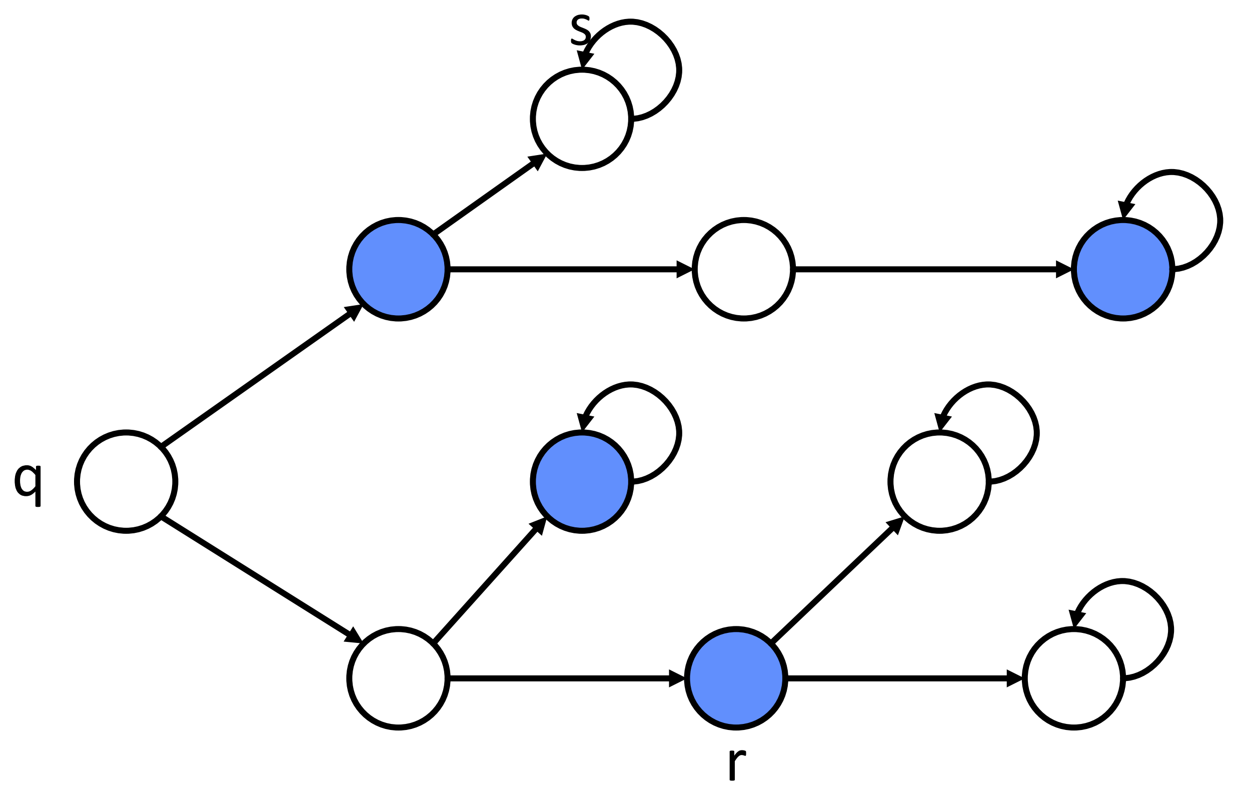
$q \models \mathrm{AF}\ \phi$

$r \models ?$

$s \models ?$

Formulation of CTL properties

AF ϕ : "On all paths, at some state ϕ holds ."



● $\models \phi$

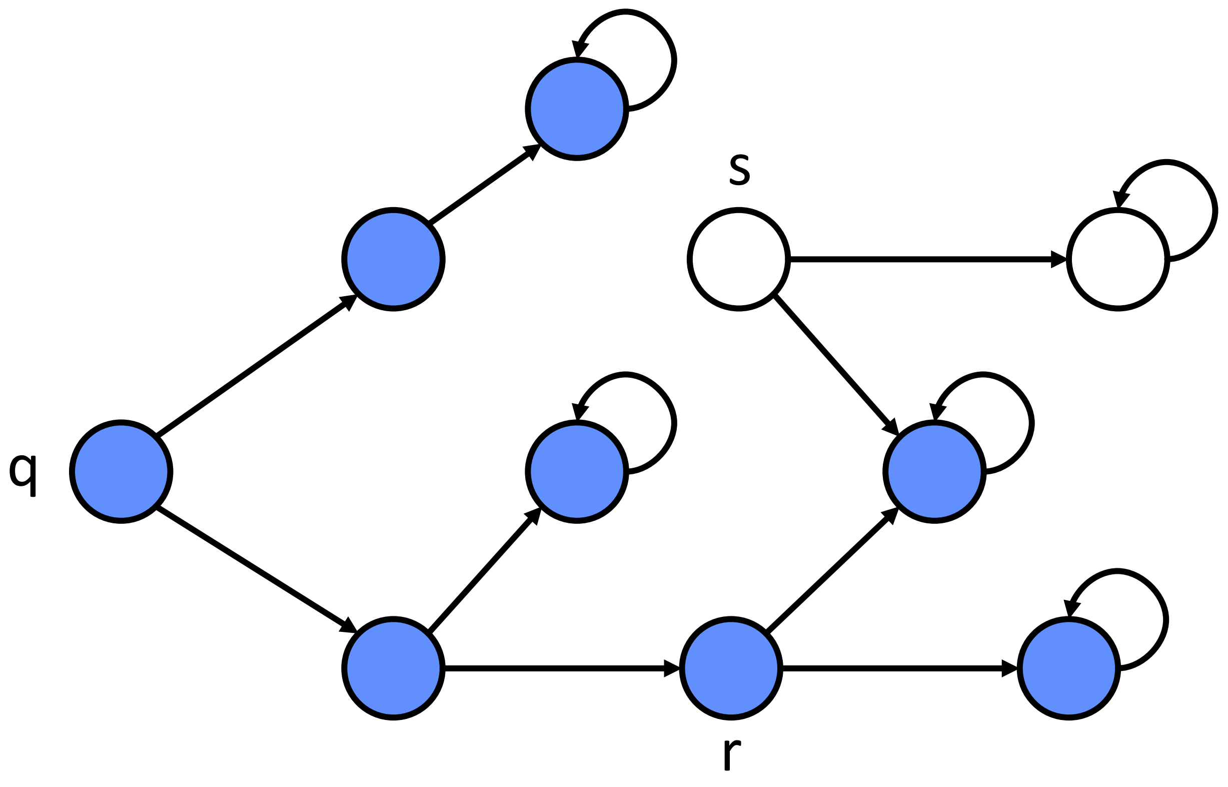
q $\models$ AF ϕ

r $\models$ AF ϕ

s $\not\models$ AF ϕ

Formulation of CTL properties

AG ϕ : "On all paths, for all states ϕ holds."



● $\vDash \phi$

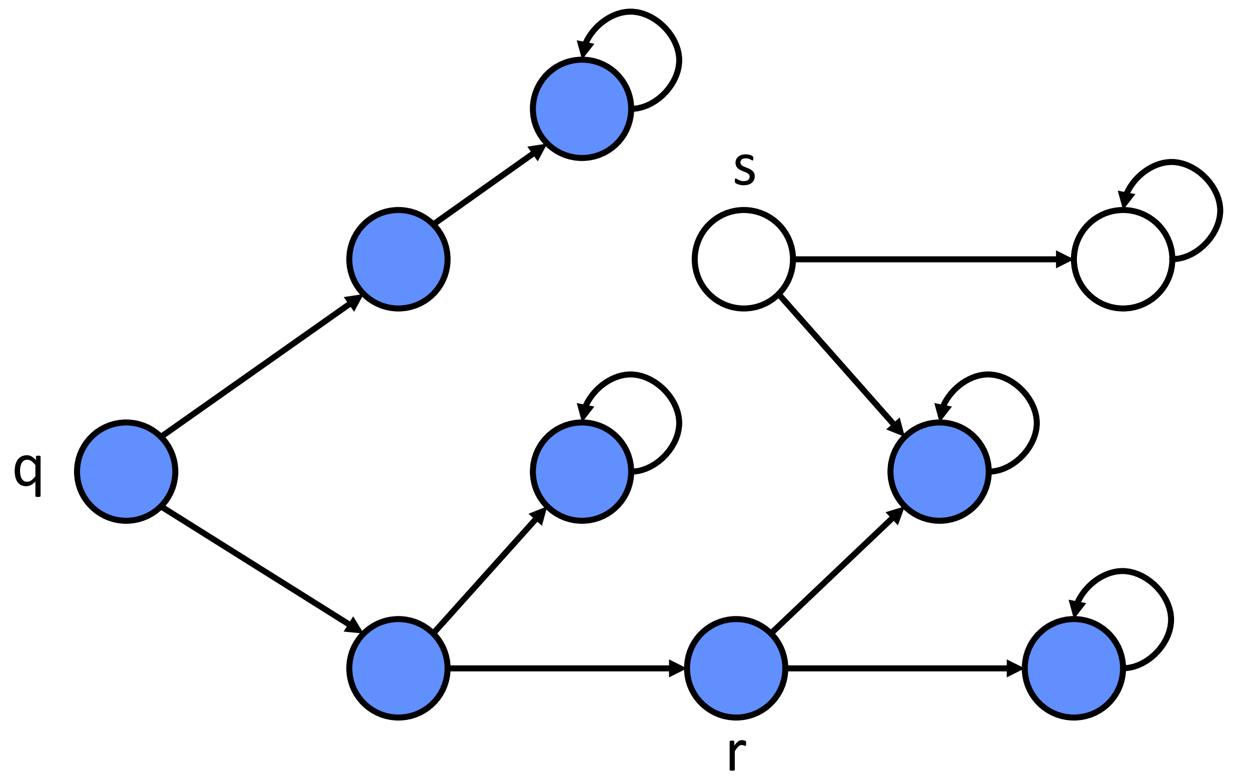
$q \vDash AG\ \phi$

$r \vDash ?$

$s \vDash ?$

Formulation of CTL properties

AG ϕ : "On all paths, for all states ϕ holds."



● $\models \phi$

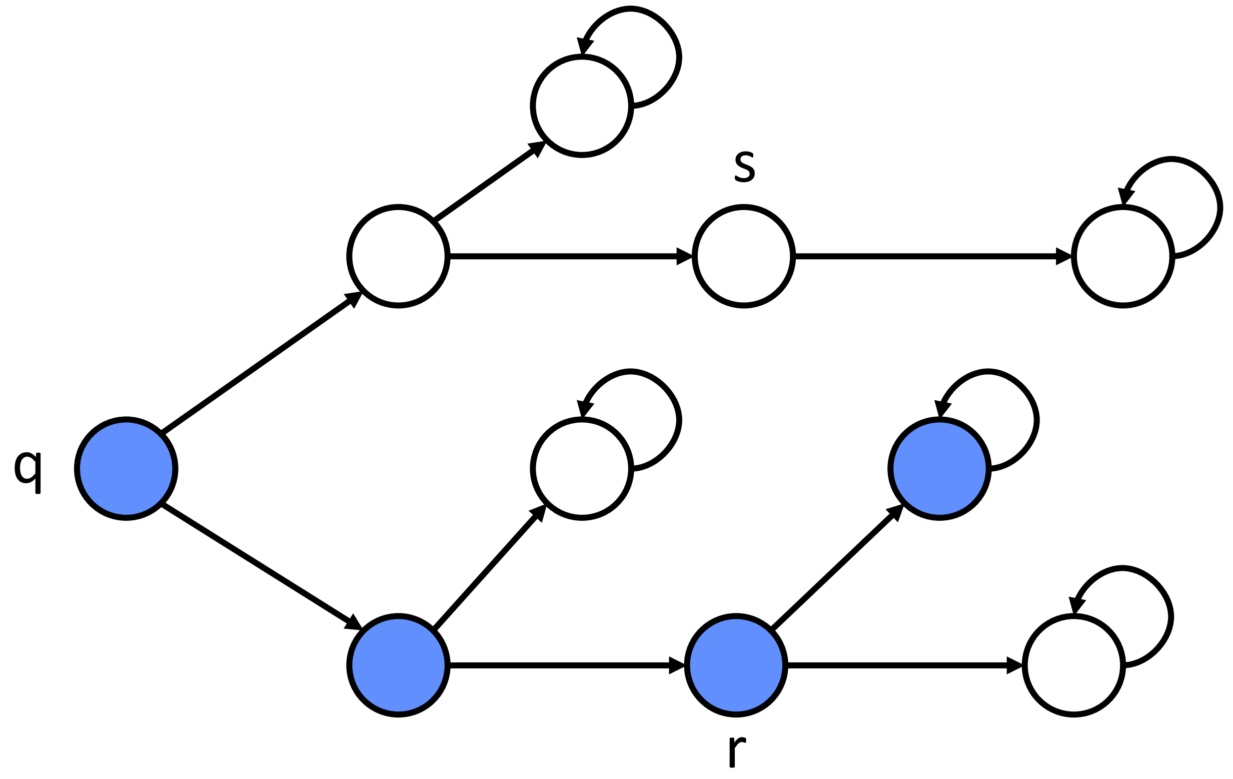
$q \models AG\ \phi$

$r \models AG\ \phi$

$s \not\models AG\ \phi$

Formulation of CTL properties

EG ϕ : "There exists a path along which for all states ϕ holds ."



● $\vDash \phi$

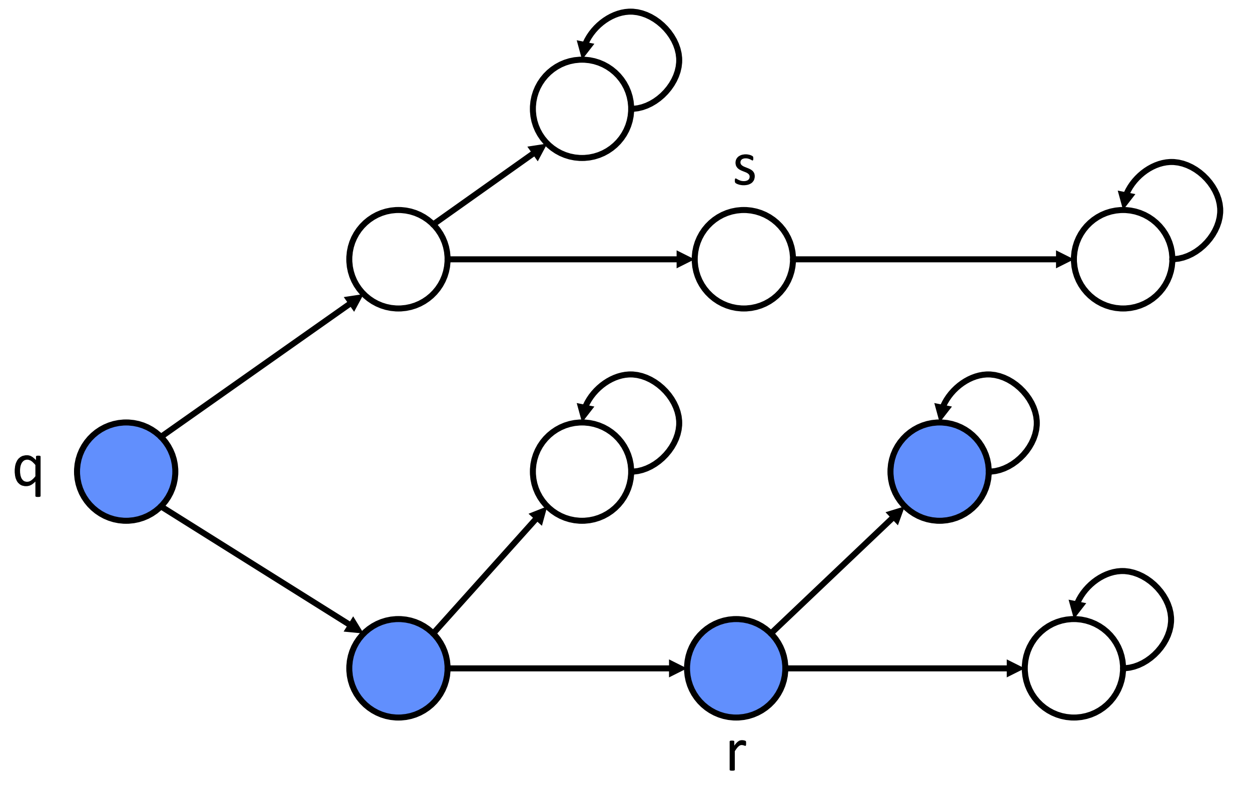
q $\vDash$ EG ϕ

r $\vDash$?

s $\vDash$?

Formulation of CTL properties

EG ϕ : "There exists a path along which for all states ϕ holds ."



● $\vDash \phi$

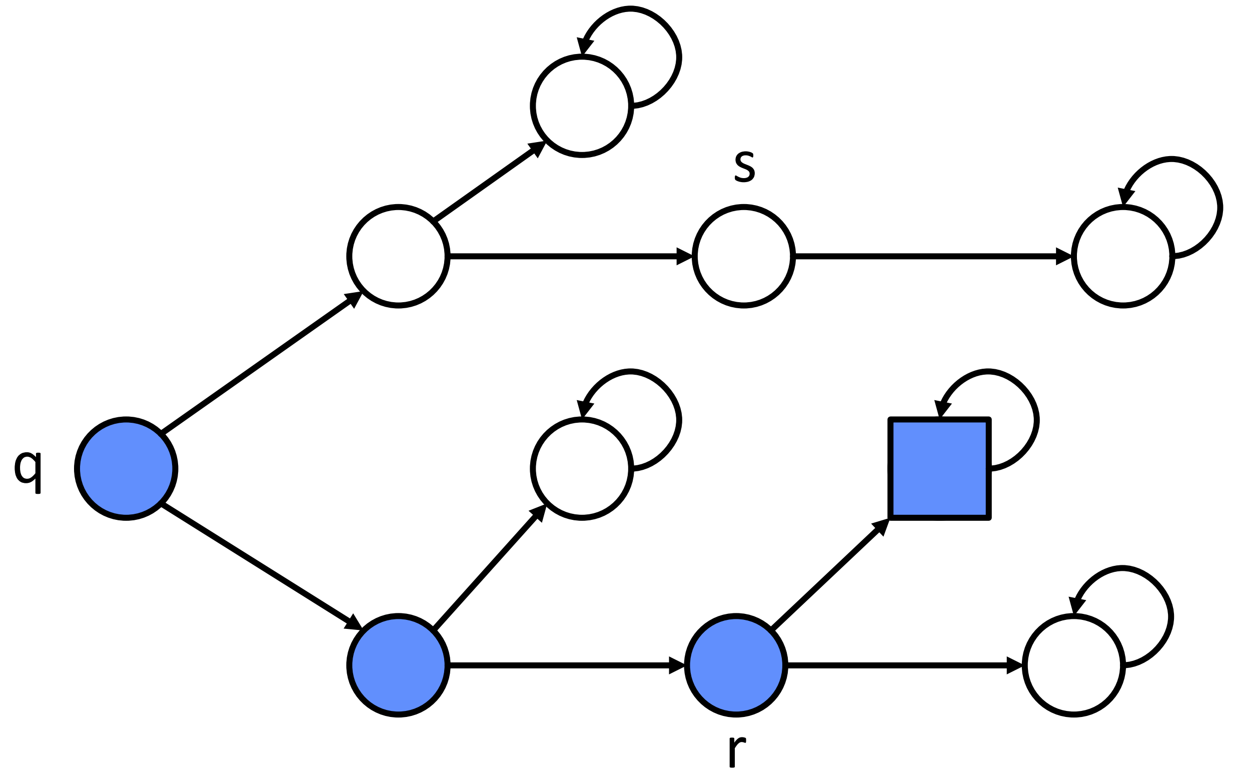
$q \vDash EG\ \phi$

$r \vDash EG\ \phi$

$s \nvDash EG\ \phi$

Formulation of CTL properties

$\phi E U \Psi$: "There exists a path along which ϕ holds until Ψ holds."



■ $\models \Psi$

● $\models \phi$

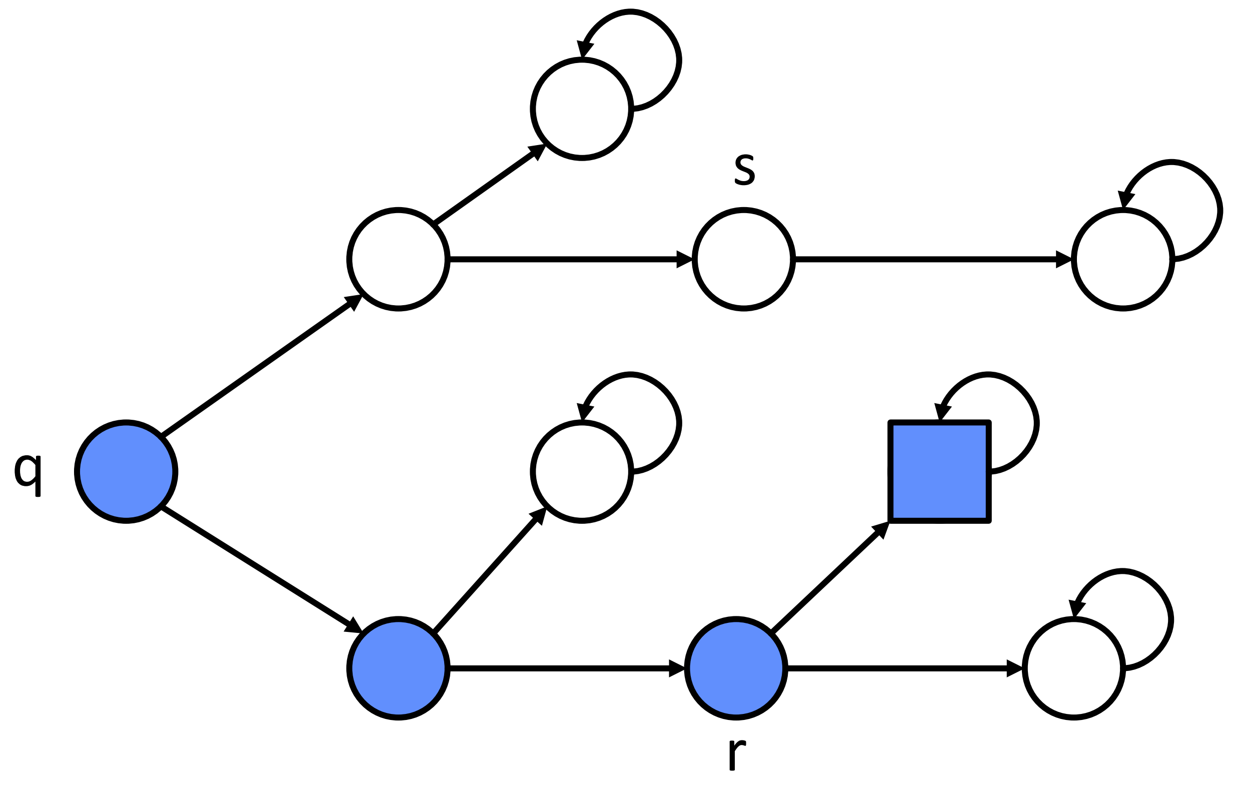
$q \models \phi E U \Psi$

$r \models ?$

$s \models ?$

Formulation of CTL properties

$\phi EU\Psi$: "There exists a path along which ϕ holds until Ψ holds."



■ $\vDash \Psi$

● $\vDash \phi$

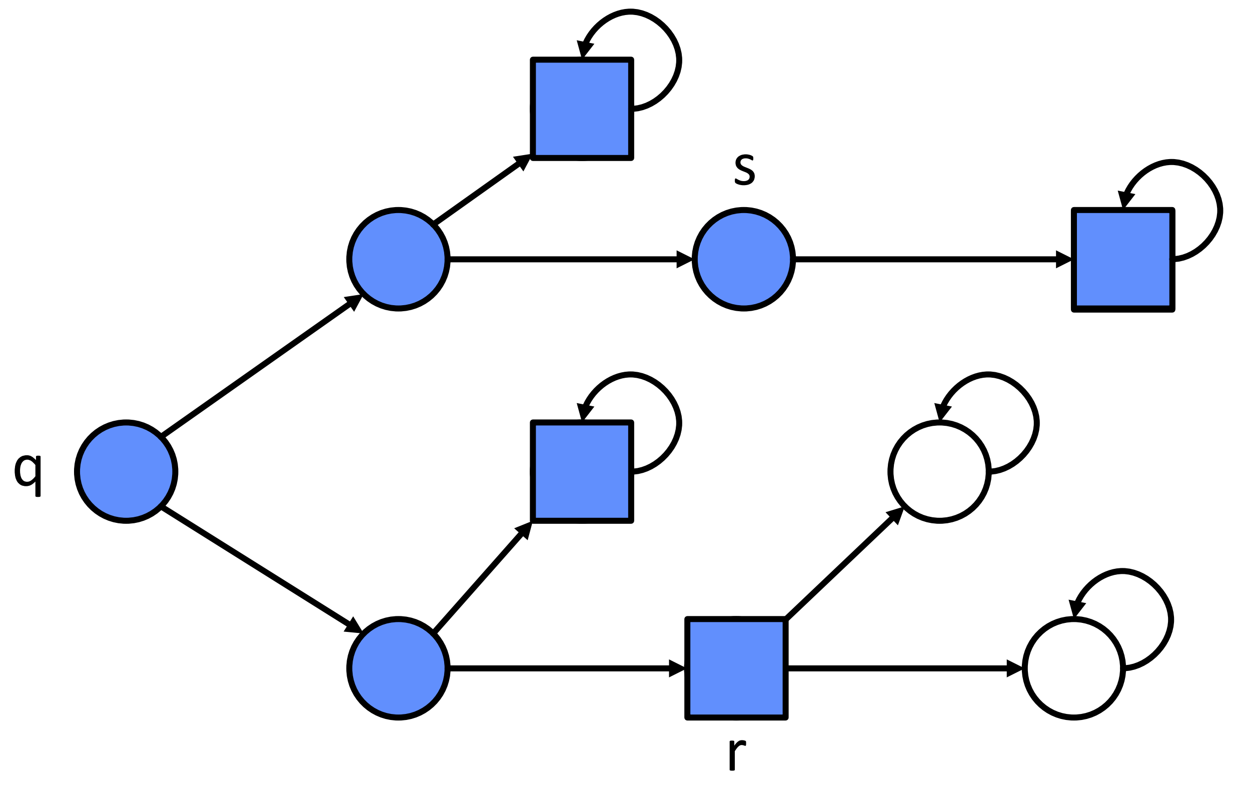
$q \vDash \phi EU\Psi$

$r \vDash \phi EU\Psi$

$s \nvDash \phi EU\Psi$

Formulation of CTL properties

$\phi AU\Psi$: "On all paths, ϕ holds until Ψ holds."



■ $\vDash \Psi$

● $\vDash \phi$

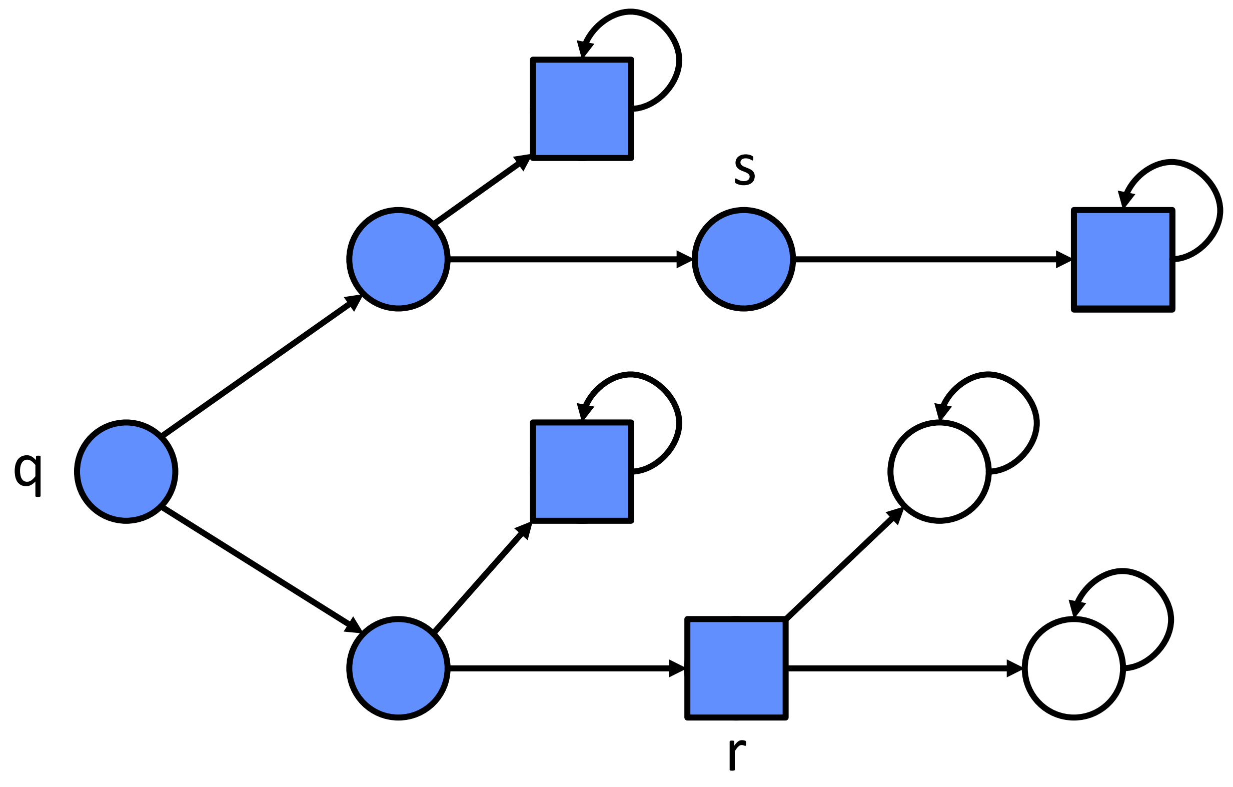
$q \vDash \phi AU\Psi$

$r \vDash ?$

$s \vDash ?$

Formulation of CTL properties

$\phi AU\Psi$: "On all paths, ϕ holds until Ψ holds."



■ $\models \Psi$

● $\models \phi$

$q \models \phi AU\Psi$

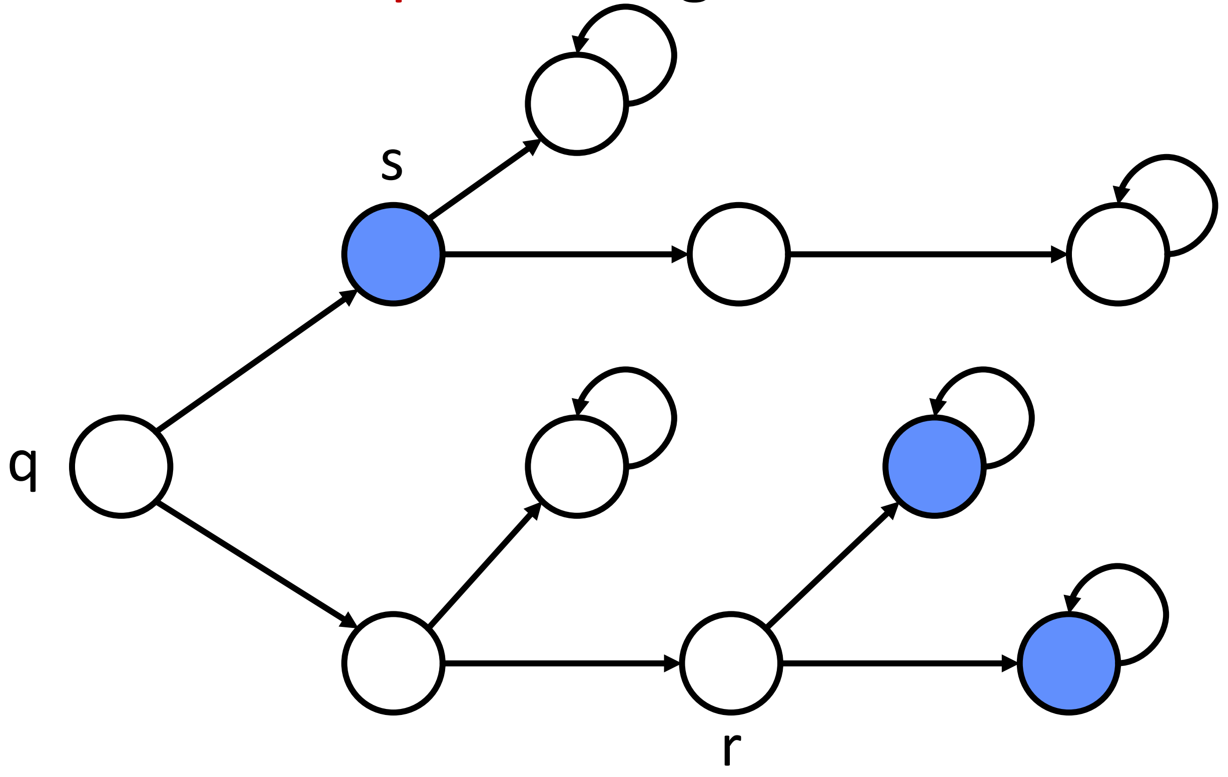
$r \models \phi AU\Psi$

$s \models \phi AU\Psi$

Formulation of CTL properties

$AX\phi$: "On all paths, the next state satisfies ϕ."

$EX\phi$: "There exists a path along which the next state satisfies ϕ."



● $\vDash \phi$

$q \vDash EX\phi$

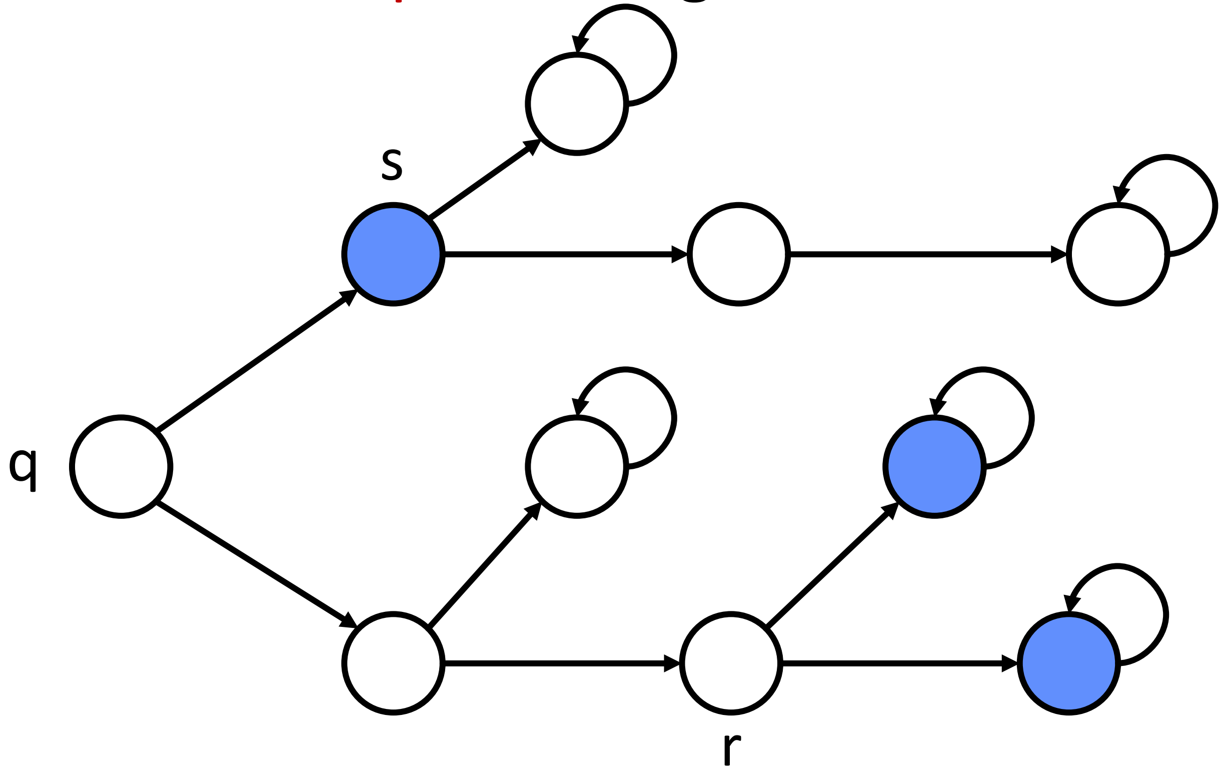
$r \vDash ?$

$s \vDash ?$

Formulation of CTL properties

$AX\phi$: "On all paths, the next state satisfies ϕ."

$EX\phi$: "There exists a path along which the next state satisfies ϕ."



● $\models \phi$

$q \models EX\phi$

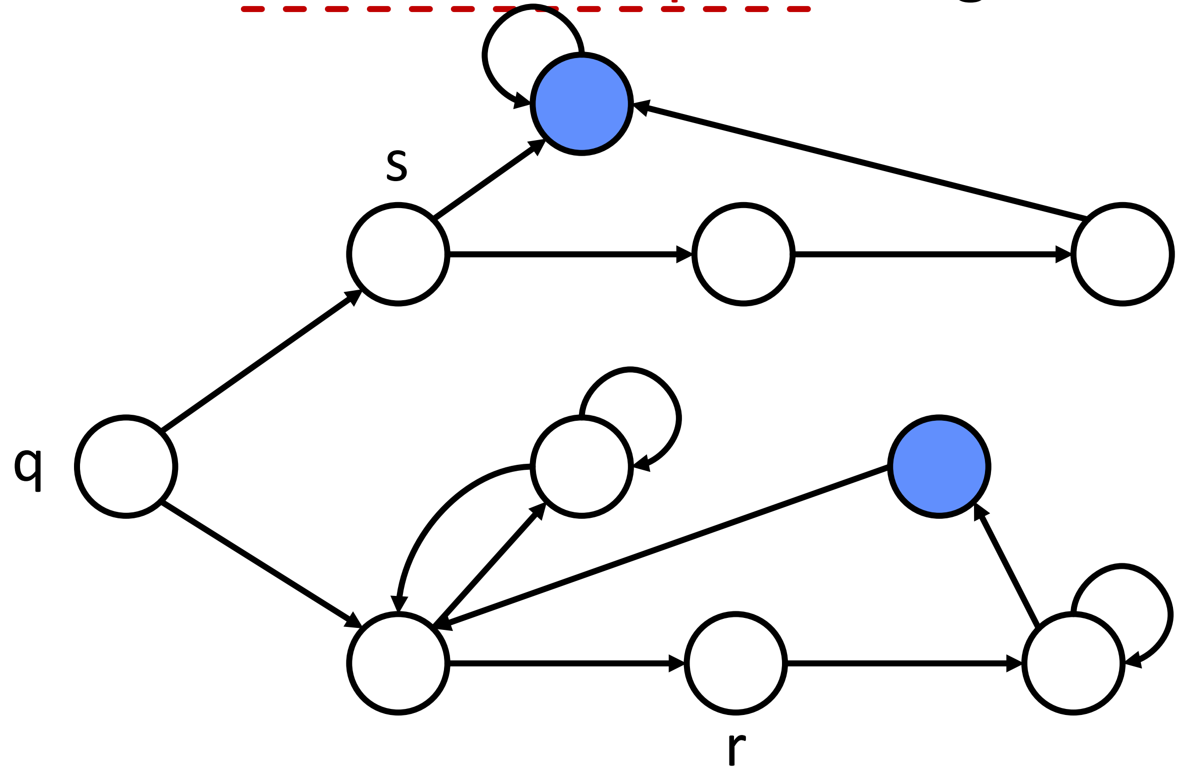
$r \models EX\phi$

$s \not\models EX\phi$

Formulation of CTL properties

AG EF ϕ : "On all paths and for all states,

there exists a path along which at some state ϕ holds."



● $\models \phi$

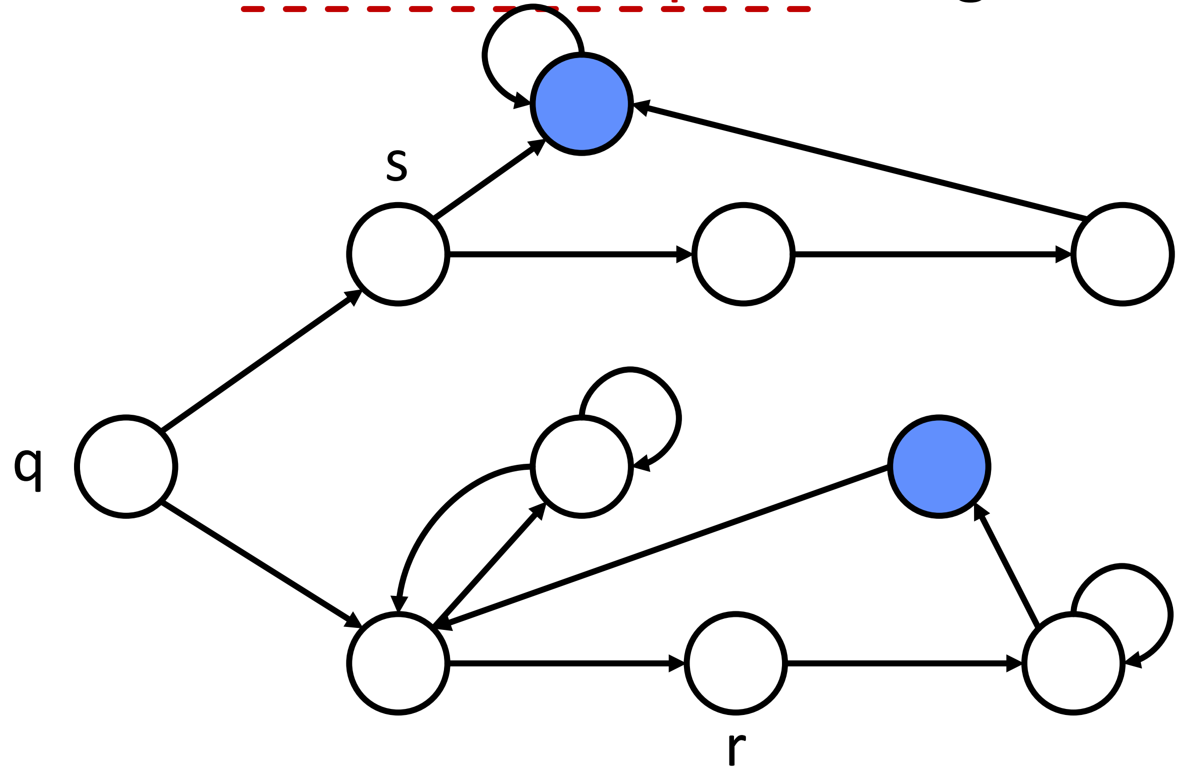
q $\models$ AG EFϕ

r $\models$?

s $\models$?

Formulation of CTL properties

AG EF ϕ : "On all paths and for all states,
there exists a path along which at some state ϕ holds."



● $\models \phi$

q $\models$ AG EFϕ

r $\models$ AG EFϕ

s $\models$ AG EFϕ

Inverting properties is sometimes useful!

$\text{AG } \phi \equiv \neg \text{ EF } \neg\phi$ $\longrightarrow$ "On all paths, for all states ϕ holds."

$\equiv$

"There exists no path along which at some state ϕ doesn't hold."

$\text{AF } \phi \equiv \neg \text{ EG } \neg\phi$ $\longrightarrow$ …

$\text{EF } \phi \equiv \neg \text{ AG } \neg\phi$

$\text{EG } \phi \equiv \neg \text{ AF } \neg\phi$

Remark There exists other temporal logics

$\rightarrow$ LTL (Linear Tree Logic)

$\rightarrow$ CTL* = {CTL,LTL}

$\rightarrow$ …

How to verify CTL properties?

Convert the property verification into a reachability problem

1. Start from states in which the property holds;
2. Compute all predecessor states for which the property still holds true; (same as for computing successor, with the inverse the transition function)
3. If initial states set is a subset, the property is satisfied by the model.

Computation specifics are described in the lecture slides.

So… what is Model-Checking exactly?

An algorithm

Input

- A DES model, **M**
  - Finite automata,
  - Petri nets,
  - Kripke machine, …
- A logic property, $\boldsymbol{\phi}$
  - CTL,
  - LTL, …

Output

- **M**$\models \boldsymbol{\phi}$?
- A trace for which the property does not hold!

Crash course – Verification of Finite Automata
CTL model-checking

Your turn to work!

Slides online on my webpage:

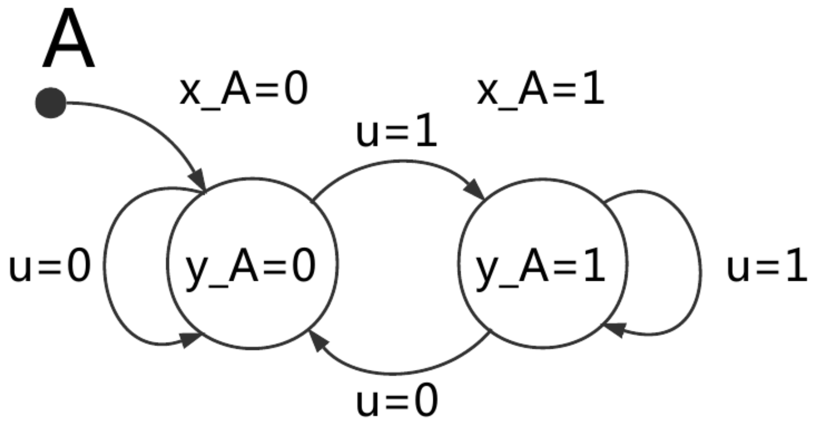
http://people.ee.ethz.ch/~jacobr/

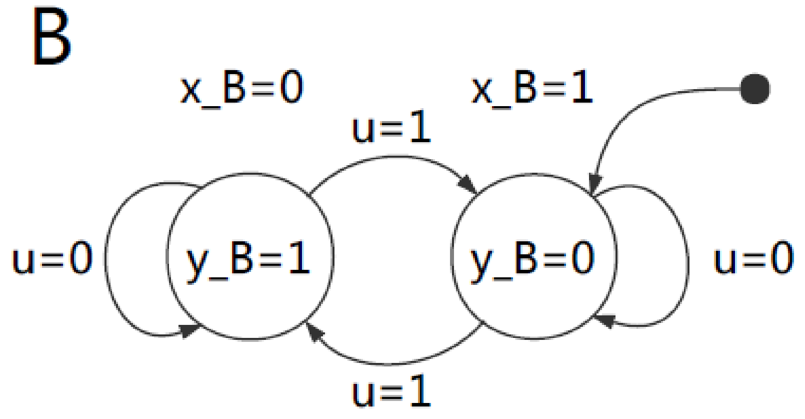
Comparison of Finite Automata

a) Express the characteristic function of the transition relation for both automaton, $\psi_r(x, x', u)$.

$$\psi_A(x_A, x'_A, u) = \overline{x_A}\,\overline{x'_A}\,\overline{u} + \overline{x_A}\,x'_A u + x_A x'_A u + x_A \overline{x'_A}\,\overline{u}$$

$$\psi_B(x_B, x'_B, u) = \overline{x_B}\,\overline{x'_B}\,\overline{u} + \overline{x_B}\,x'_B u + x_B x'_B \overline{u} + x_B \overline{x'_B} u$$





Comparison of Finite Automata

b) Express the joint transition function, ψ_f.

$$\psi_f(x_A, x'_A, x_B, x'_B) = (\exists u : \psi_A(x_A, x'_A, u) \cdot \psi_B(x_B, x'_B, u))$$

$$\psi_f(x_A, x'_A, x_B, x'_B)$$

$$= (\overline{x_A} x'_A + x_A x'_A) \cdot (\overline{x_B} x'_B + x_B \overline{x'_B}) +$$
$$(\overline{x_A}\,\overline{x'_A} + x_A \overline{x'_A}) \cdot (\overline{x_B}\,\overline{x'_B} + x_B x'_B)$$

$$= \overline{x_A} x'_A \overline{x_B} x'_B + \overline{x_A} x'_A x_B \overline{x'_B} + x_A x'_A \overline{x_B} x'_B + x_A x'_A x_B \overline{x'_B} +$$
$$\overline{x_A}\,\overline{x'_A}\,\overline{x_B}\,\overline{x'_B} + \overline{x_A}\,\overline{x'_A} x_B x'_B + x_A \overline{x'_A}\,\overline{x_B}\,\overline{x'_B} + x_A \overline{x'_A} x_B x'_B$$

Comparison of Finite Automata

c) Express the characteristic function of the reachable states, $\psi_X(x_A, x_B)$.

$$\psi_{X_0}(x_A, x_B) = \overline{x_A} x_B$$

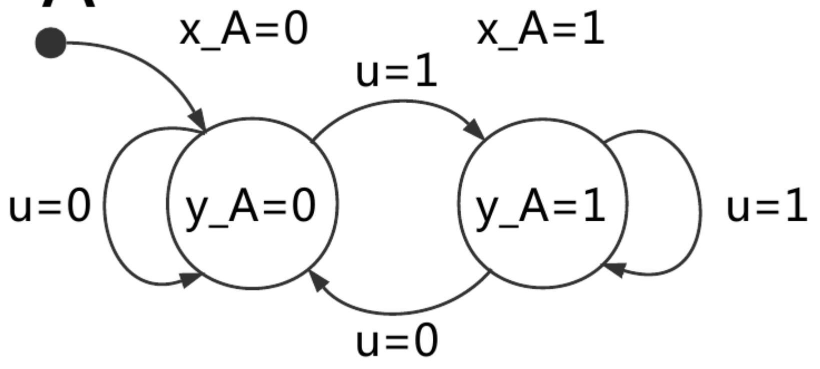
$$\psi_{X_1}(x'_A, x'_B) = \psi_{X_0}(x'_A, x'_B)$$
$$+ (\exists (x_A, x_B) : \psi_{X_0}(x_A, x_B) \cdot \psi_f(x_A, x'_A, x_B, x'_B))$$
$$= \overline{x'_A} x'_B + x'_A \overline{x'_B}$$

$$\psi_{X_2}(x'_A, x'_B) = \overline{x'_A} x'_B + x'_A \overline{x'_B} + x'_A x'_B + \overline{x'_A x'_B}$$
$$\psi_{X_3}(x'_A, x'_B) = \overline{x'_A} x'_B + x'_A \overline{x'_B} + x'_A x'_B + \overline{x'_A x'_B}$$

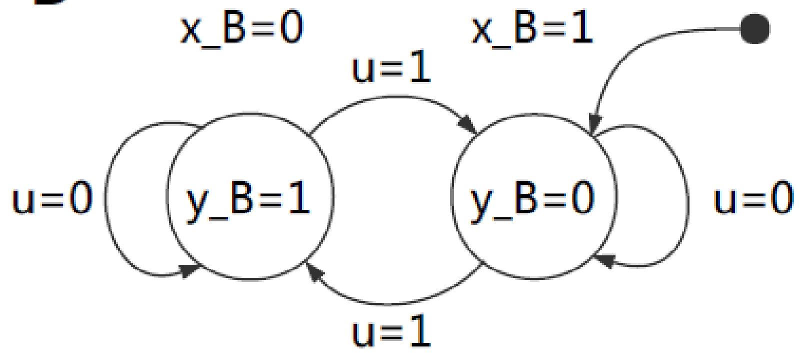
$$= \psi_{X_2} \rightarrow \text{the fix-point is reached!}$$

$$\boxed{\psi_X = \overline{x_A} x_B + x_A \overline{x_B} + x_A x_B + \overline{x_A x_B}}$$

A



B



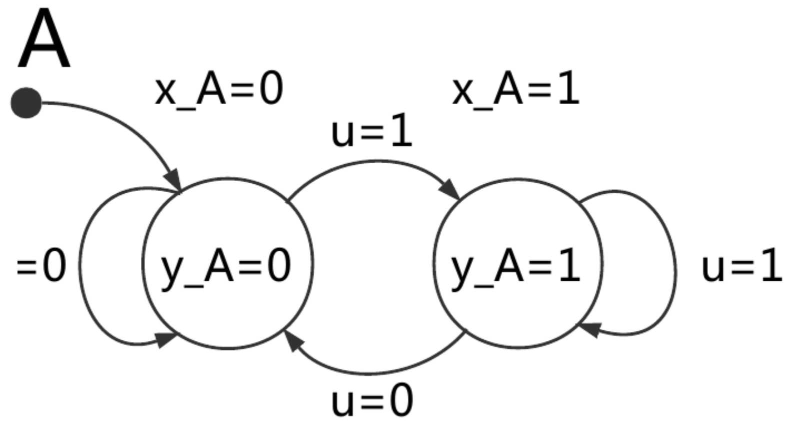
Comparison of Finite Automata

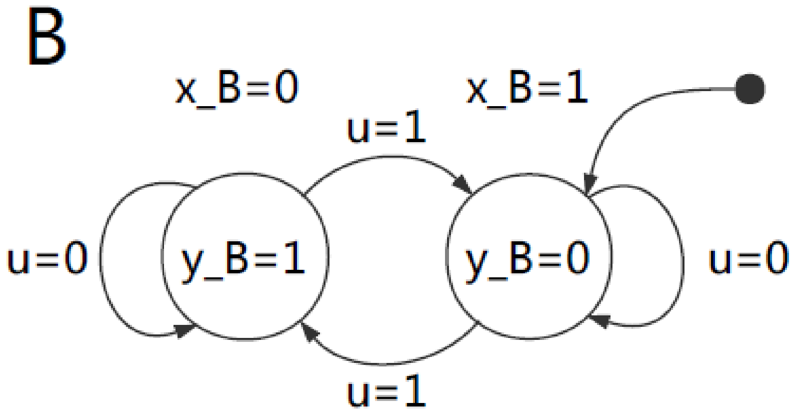
d) Express the characteristic function of the reachable output, $\psi_Y(x_A, x_B)$.

$$\psi_{g_A} = \overline{x_A y_A} + x_A y_A$$
$$\psi_{g_B} = \overline{x_B} y_B + x_B \overline{y_B}$$

and $\boxed{\psi_X = \overline{x_A} x_B + x_A \overline{x_B} + x_A x_B + \overline{x_A x_B}}$

$$\psi_Y(y_A, y_B)$$
$$= (\exists (x_A, x_B) : \psi_X \cdot \psi_{g_A} \cdot \psi_{g_B})$$
$$= y_A y_B + \overline{y_A y_B} + \overline{y_A} y_B + y_A \overline{y_B}$$





Comparison of Finite Automata

e) Are the automata equivalent? **Hint**: Evaluate, for example, $\psi_Y(0{,}1)$.

$$\psi_Y((y_A, y_B) = (0,1)) = 1$$

Or, in a more general way,

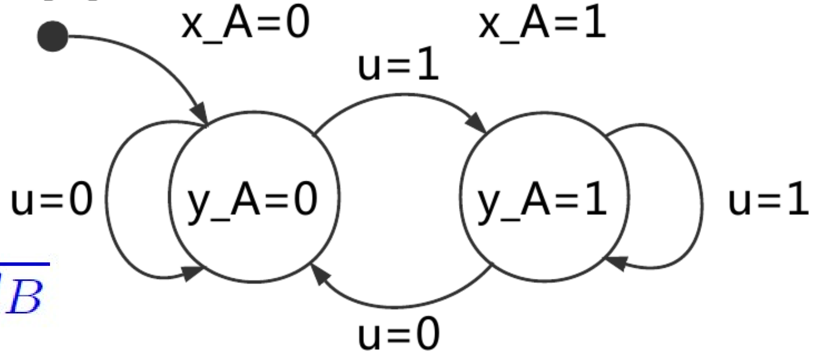
$$\psi_Y(y_A, y_B) = y_A y_B + \overline{y_A y_B} + \overline{y_A} y_B + y_A \overline{y_B}$$

and $(y_A \neq y_B) = \overline{y_A} y_B + y_A \overline{y_B}$

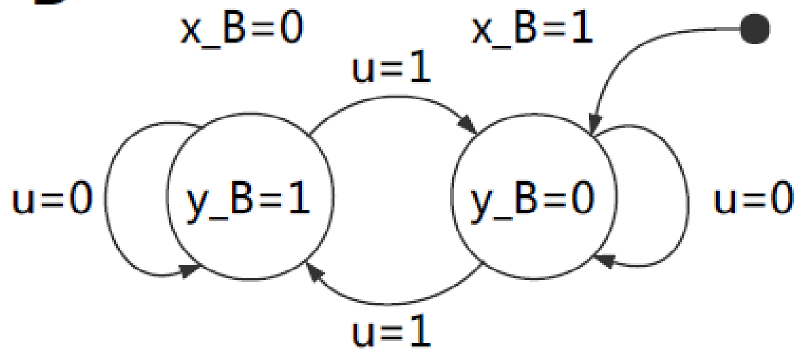
implies $\psi_Y \cdot (y_A \neq y_B) \neq 0$

→ Automata are not equivalent.

A



B



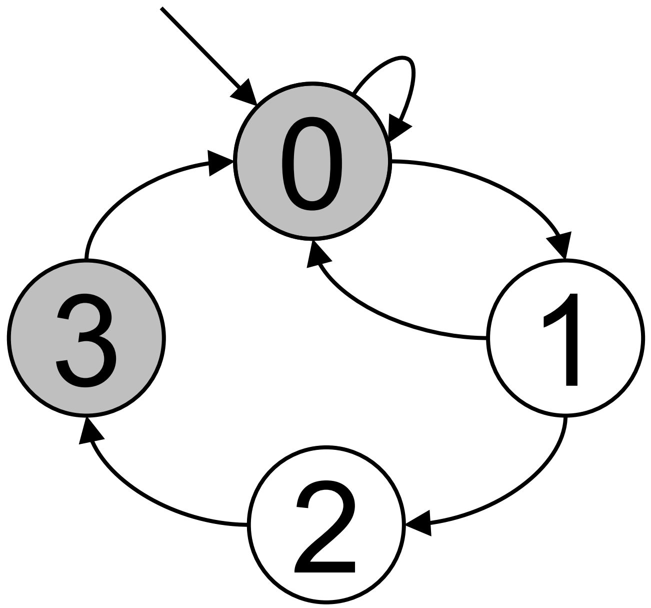
Temporal Logic

i. EF a

ii. EG a

iii. EX AX a

iv. EF (a AND EX NOT(a))



Temporal Logic

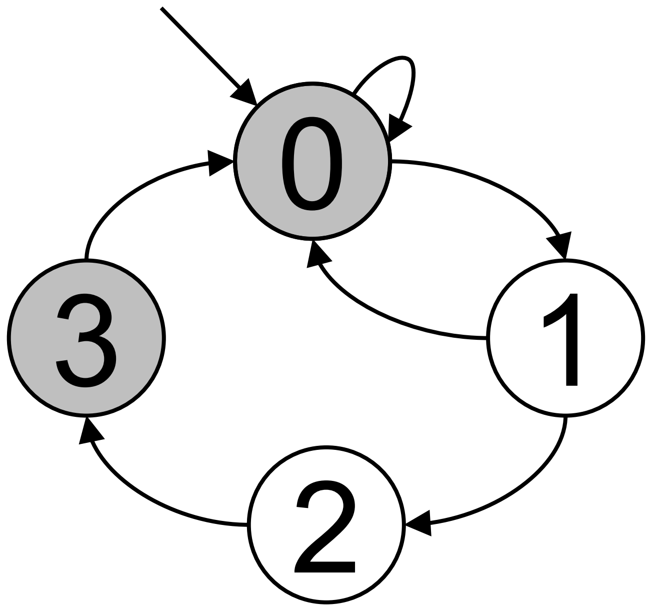
i. EF a

$$Q = \{0, 1, 2, 3\}$$

ii. EG a

iii. EX AX a

iv. EF (a AND EX NOT(a))



Temporal Logic

i. EF a

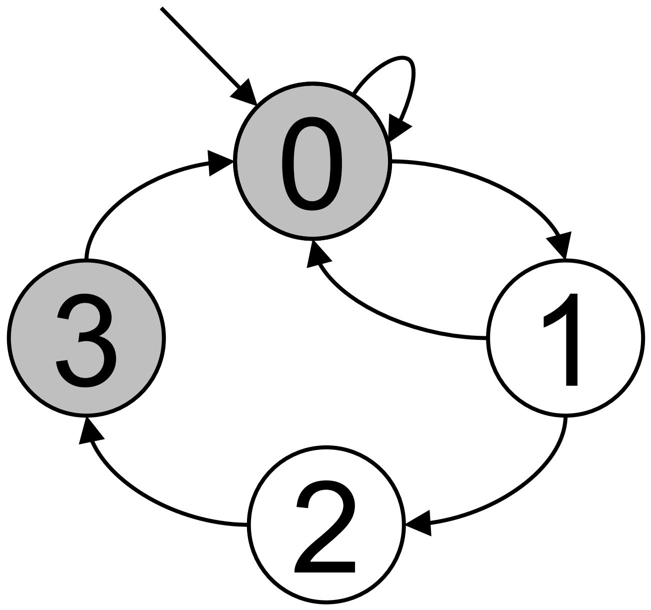
$Q = \{0, 1, 2, 3\}$

ii. EG a

$Q = \{0, 3\}$

iii. EX AX a

iv. EF (a AND EX NOT(a))



Temporal Logic

i. EF a

$Q = \{0, 1, 2, 3\}$

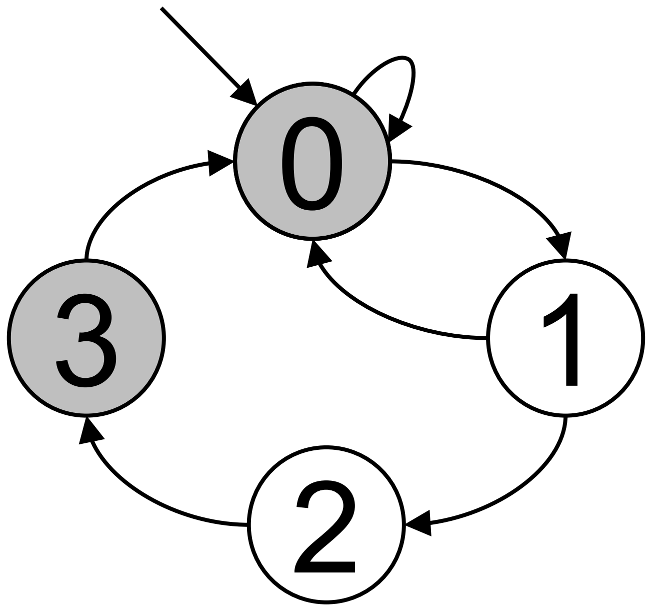
ii. EG a

$Q = \{0, 3\}$

iii. EX AX a

$Q = \{1, 2\}$

iv. EF (a AND EX NOT(a))



Temporal Logic

i. EF a

$Q = \{0, 1, 2, 3\}$

ii. EG a

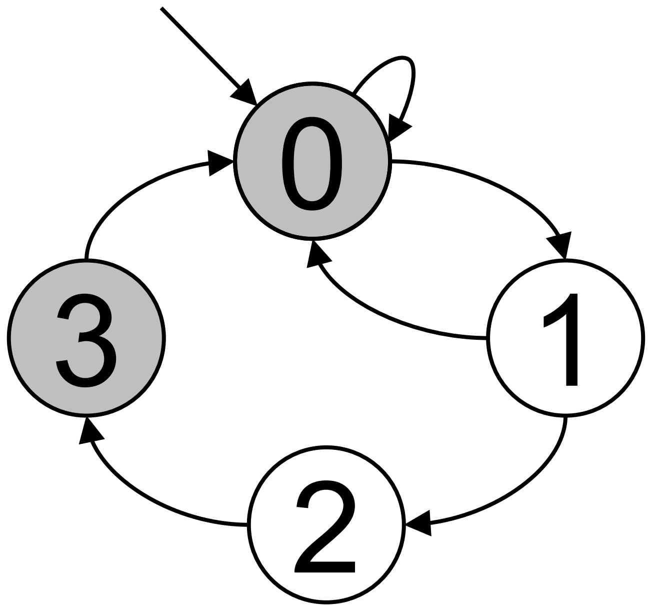
$Q = \{0, 3\}$

iii. EX AX a

$Q = \{1, 2\}$

iv. EF (a AND EX NOT(a))

$Q = \{0, 1, 2, 3\}$



Temporal Logic

Trick AF Z not(EG not(Z))

```
Require: $\psi_Z, \psi_f$
    current = NOT($\psi_Z$);                                    $\triangleright X_0$
    next = current AND $\psi_{\mathtt{PRE}(current,f)}$;        $\triangleright X_1 = X_0 \cap Pre(X_0, f)$
    while next != current do                                    $\triangleright X_i \mathrel{!}= X_{i-1}$
        current = next;
        next = current AND $\psi_{\mathtt{PRE}(current,f)}$;    $\triangleright X_i = X_{i-1} \cap Pre(X_{i-1}, f)$
    end while                                                   $\triangleright X_f \models \text{EG } \mathtt{NOT}(Z)$
    return $\psi_{\text{AF } Z}$ =NOT(current);                 $\triangleright \overline{X_f} \models \text{AF } Z = \mathtt{NOT}(\text{EG } \mathtt{NOT}(Z))$
```

$\triangleright$ Equivalence in term of sets:

Crash course – Verification of Finite Automata
CTL model-checking

See you next week!