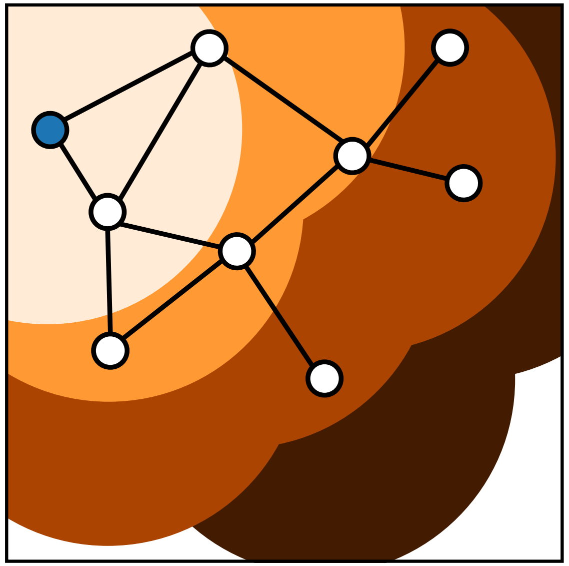
Discrete Event Systems
Verification of Finite Automata (Part 2)



Romain Jacob

www.romainjacob.net

ETH Zurich (D-ITET)

December 2, 2021

Most materials from Lothar Thiele

Thank you for your feedback!

- Slightly too fast
- Reachability was covered too quickly
- More examples would be nice
- More interaction would be nice

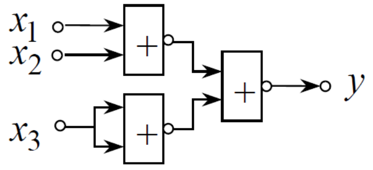
▶ Will hopefully improve already today ☺

Last week in
Discrete Event Systems

Verification Scenarios

Example

$$y = (x_1 + x_2) \cdot x_3$$



Comparison of specification and implementation

reference system → data structure
system under test → data structure
→ comparison

Proving properties

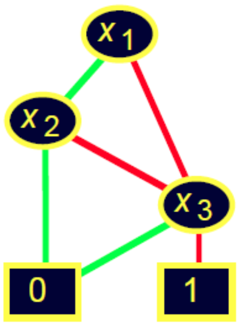
“The device can always be switched off.”

property
system under test → data structure
→ fixed-point calculation

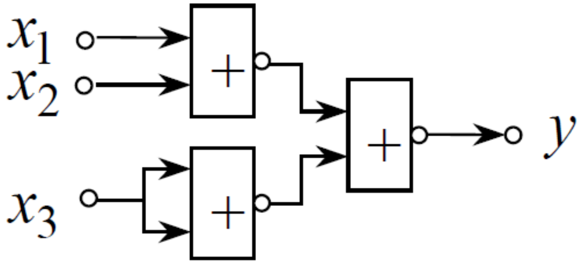
Comparison using BDDs

- Boolean (combinatorial) circuits: Compare specification and implementation, or compare two implementations.
- Method:
  - Representation of the two systems in ROBDDs, e.g., by applying the APPLY operator repeatedly.
  - Compare the structures of the ROBDDs.
- Example:

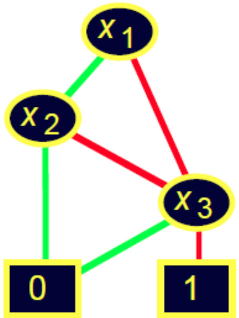
$$y = (x_1 + x_2) \cdot x_3 \xrightarrow{\text{APPLY}}$$



compare



$$\longrightarrow y = \overline{\overline{x_1 + x_2} + \overline{x_3 + x_3}} \xrightarrow{\text{APPLY}}$$



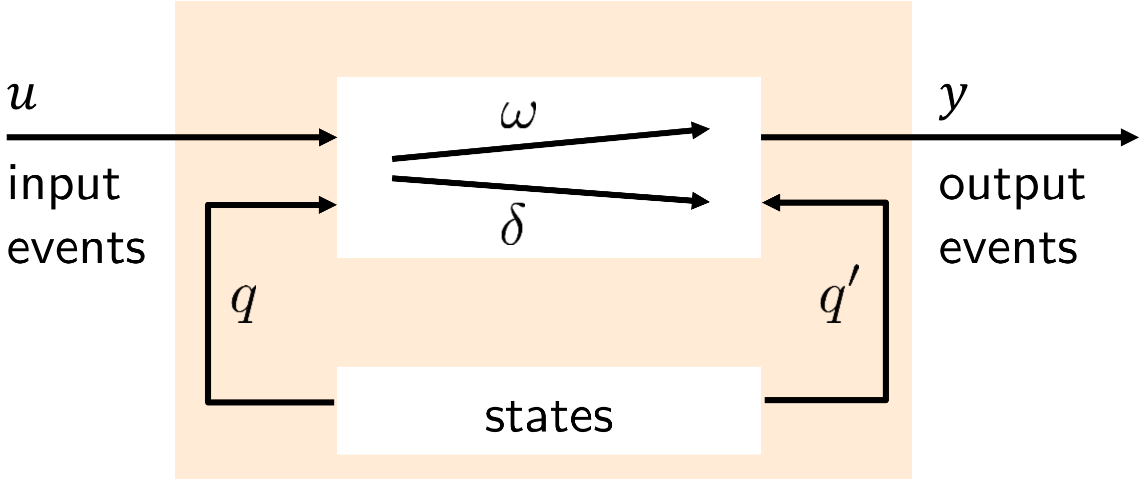
Sets and Relations using Boolean Expressions

- Representation of a relation $R \subseteq A \times B$
  - Binary encoding $\sigma(a)$, $\sigma(b)$ of all elements $a \in A$, $b \in B$
  - Representation of R

$$(a,b) \in R \;\Leftrightarrow\; \psi_R(\sigma(a),\sigma(b))$$

characteristic function of the relation R

- Example finite automaton:



finite automaton

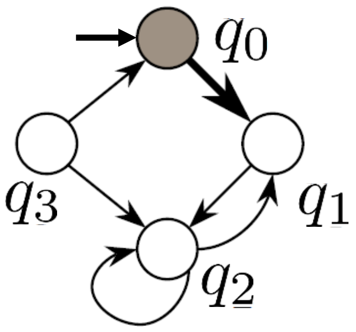
$$\psi_\delta(u,q,q') = 1$$
$$\psi_\omega(u,q,y) = 1$$

we remove the binary encoding for convenience in our notation; but u, q, q' are actually represented as binary vectors

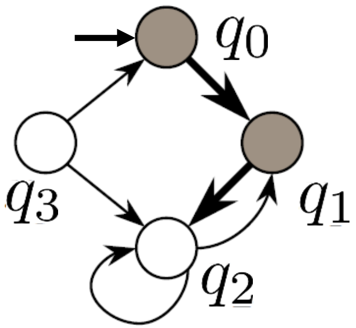
Reachability of States – State Diagram

Question — Is a state $q \in Q$ reachable by a sequence of state transitions?

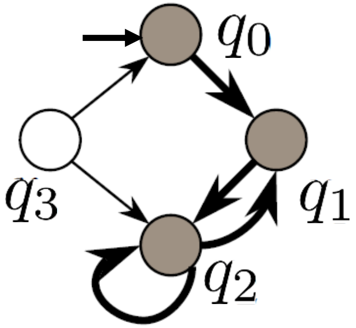
$Q_0 = \{q_0\}$



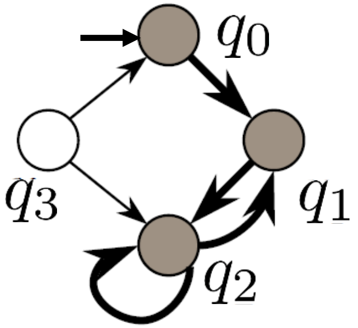
$Q_1 = Q_0 \cup \{q_1\}$



$Q_2 = Q_1 \cup \{q_1, q_2\}$



$Q_3 = Q_2 \cup \{q_1, q_2\}$



Problem — Drawing state diagrams is not feasible in general.

Reachability of States – Boolean Expressions

Fixed-point computation

- Start with the initial state
- Determine the set of states that can be reached in one
- Take the union and iterate until a fixed-point is reached

$$Q_0 = \{q_0\}$$

$$\boxed{Q_{i+1}} = Q_i \cup Suc(Q_i, \delta)$$

until $Q_{i+1} = Q_i$

$$\psi_{Q_{i+1}}(q') = \psi_{Q_i}(q') + (\, \exists q : \psi_{Q_i}(q) \cdot \psi_\delta(q, q'))$$

Test by comparing the ROBDDs of $Q_{i+1} = Q_i$

Q_R: set of reachable states

$$\boxed{Q_R} = Q_0 \cup_{i \geq 0} Suc(Q_i, \delta)$$

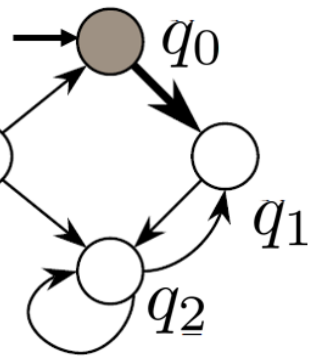
Finite union if model is finite

$$\psi_{Q_R}(q') = \psi_{Q_0}(q') \sum_{i \geq 0} (\exists q : \; \psi_{Q_i}(q) \cdot \psi_\delta(q, q'))$$

Reachability of States – Example

State encoding $(x_1, x_0) = \sigma(q)$

$\boldsymbol{\sigma(q)}$	x_1	x_0
q_0	0	0
q_1	0	1
q_2	1	0
q_3	1	1



Transition relation encoding $\psi_\delta(q, q')$

entries where $\psi_\delta(q, q') = 1$ only

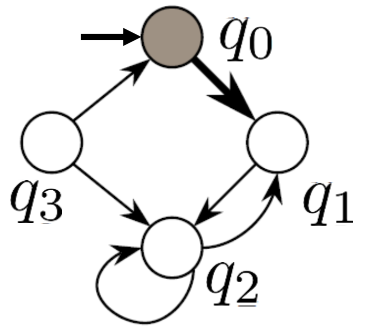
x_1	x_0	x_1'	x_0'
0	0	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	1	0
1	1	0	0

e.g.
$q_0 \rightarrow q_1$

$q_2 \rightarrow q_2$

As a Boolean function

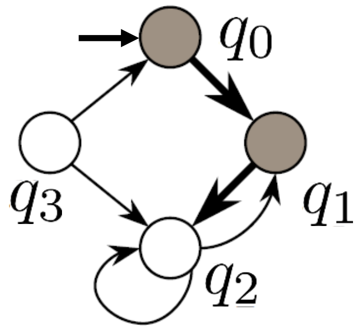
$\psi_\delta(q, q') = \overline{x_0}' \cdot (x_0 \cdot (x_1 + x_1') + x_1 \cdot x_1') + \overline{x_0} \cdot x_0' \cdot \overline{x_1}'$



$$\psi_{Q_{i+1}}(q') = \psi_{Q_i}(q') + (\,\exists q : \psi_{Q_i}(q) \cdot \psi_\delta(q, q'))$$

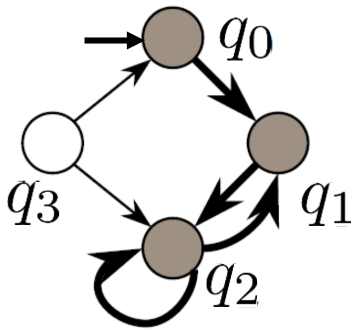
$$Q_0 = \{q_0\}$$

$$\psi_{Q_0}(q') = \overline{x'_1} \cdot \overline{x'_0}$$



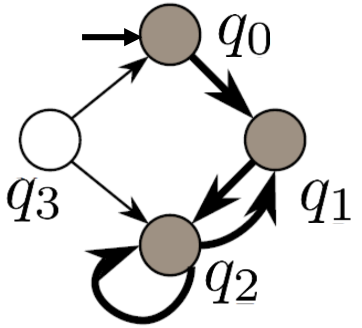
$$Q_1 = Q_0 \cup \{q_1\}$$

$$\psi_{Q_1}(q') = \overline{x'_1} \cdot \overline{x'_0} + \overline{x'_1} \cdot x'_0 = \overline{x'_1}$$



$$Q_2 = Q_1 \cup \{q_1, q_2\}$$

$$\psi_{Q_2}(q') = \overline{x'_1} + (x'_1 \cdot \overline{x'_0} + \overline{x'_1} \cdot x'_0) = \overline{x'_1} + \overline{x'_0}$$



$$Q_3 = Q_2 \cup \{q_1, q_2\}$$

$$\psi_{Q_3}(q') = (\overline{x'_1} + \overline{x'_0}) + (\overline{x'_1} + \overline{x'_0}) = \overline{x'_1} + \overline{x'_0}$$

States

$\sigma(q)$	x_1	x_0
q_0	0	0
q_1	0	1
q_2	1	0
q_3	1	1

Transitions

x_1	x_0	x_1'	x_0'
0	0	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	1	0
1	1	0	0

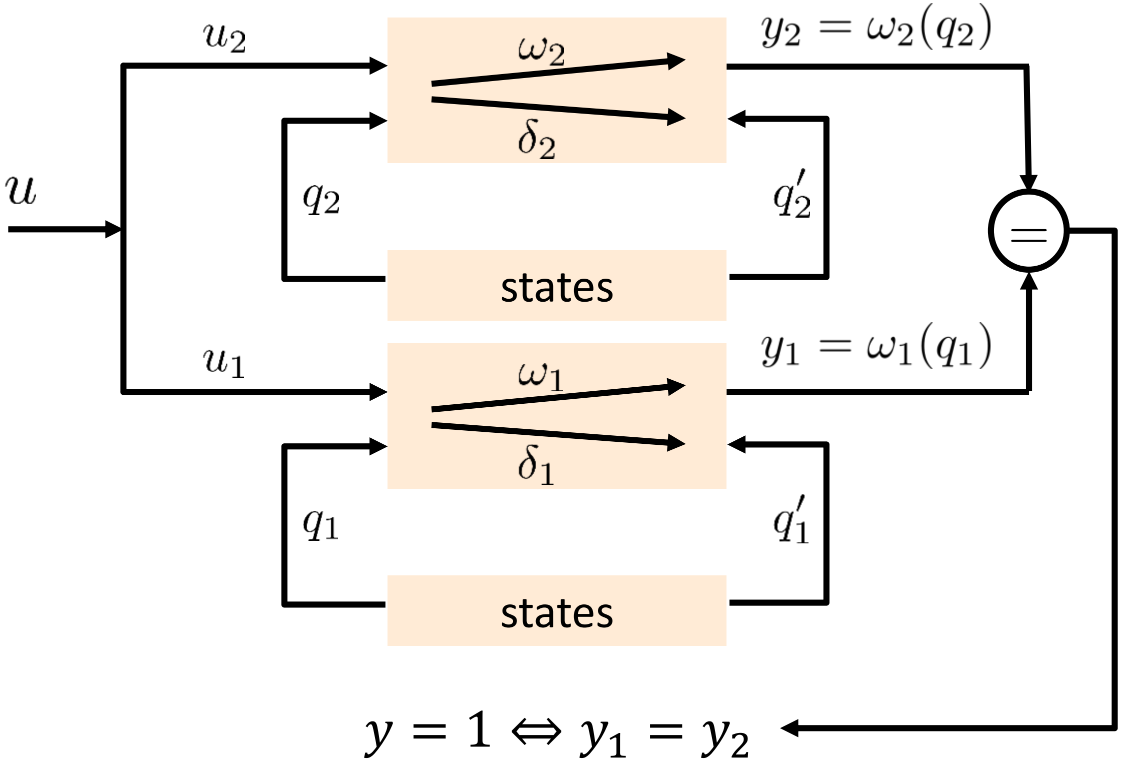
Comparison of Finite Automata

For simplicity, we only consider Moore automata, i.e., the output depends on the current state only. The output function is $\omega : Q \to \Sigma$ and $y = \omega(q)$.

Strategy

1. Compute the set of jointly reachable states.

2. Compare the output values of the two finite automata.



This week in
Discrete Event Systems

	Efficient state representation	- Set of states as Boolean function - Binary Decision Diagram representation
	Computing reachability	- Leverage efficient state representation - Explore successor sets of states
Today	Proving properties	- Temporal logic (CTL) - Encoding as reachability problem

Temporal logics

- Verify properties of a finite automaton, for example
  - Can we always reset the automaton?
  - Is every request followed by an acknowledgement?
  - Are both outputs always equivalent?

There exists other logics (e.g. LTL, CTL*)

- Specification of the query in a formula of temporal logic.
- We use a simple form called Computation Tree Logic (CTL).

- Let us start with a minimal set of operators.
  - Any atomic proposition is a CTL formula.
  - CTL formula are constructed by composition of other CTL formula.

Formula	Examples
Atomic proposition	The printer is busy. The light is on.
Boolean logic	$\phi_1 + \phi_2$; $\neg\phi_1$
CTL logic	EX ϕ_1

Formulation of CTL properties

Based on atomic propositions (ϕ) and quantifiers

$A\phi$	$\rightarrow$ «All ϕ»,	ϕ holds on all paths	Quantifiers over paths
$E\phi$	$\rightarrow$ «Exists ϕ»,	ϕ holds on at least one path	
$X\phi$	$\rightarrow$ «NeXt ϕ»,	ϕ holds on the next state	Path-specific quantifiers
$F\phi$	$\rightarrow$ «Finally ϕ»,	ϕ holds at some state along the path	
$G\phi$	$\rightarrow$ «Globally ϕ»,	ϕ holds on all states along the path	
$\phi_1 U \phi_2$	$\rightarrow$ «ϕ_1Until ϕ_2»,	ϕ_1 holds until ϕ_2 holds	
		implies that ϕ_2 has to hold eventually	

Formulation of CTL properties

CTL quantifiers works in pairs

$\{A,E\}\ \{X,F,G,U\}\phi$

You need one of each!

Can be more than one pair

AG ϕ_1 where ϕ_1 = EF ϕ_2 $\equiv$ AG EF ϕ_2

E,G,X,U are sufficient to define the whole logic.

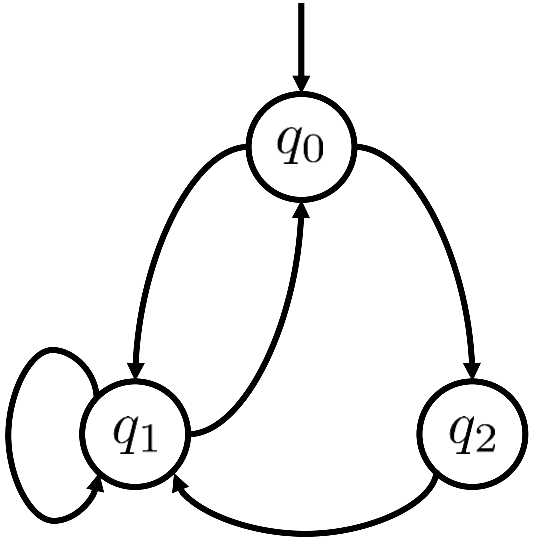
A and F are convenient, but not necessary

$$\begin{aligned}
\mathrm{AF}\phi &\equiv \neg\mathrm{EG}(\neg\phi) \\
\mathrm{AG}\phi &\equiv \neg\mathrm{EF}(\neg\phi) \\
\mathrm{AX}\phi &\equiv \neg\mathrm{EX}(\neg\phi) \\
\mathrm{EF}\phi &\equiv \mathrm{true}\,\mathrm{EU}\phi
\end{aligned}$$

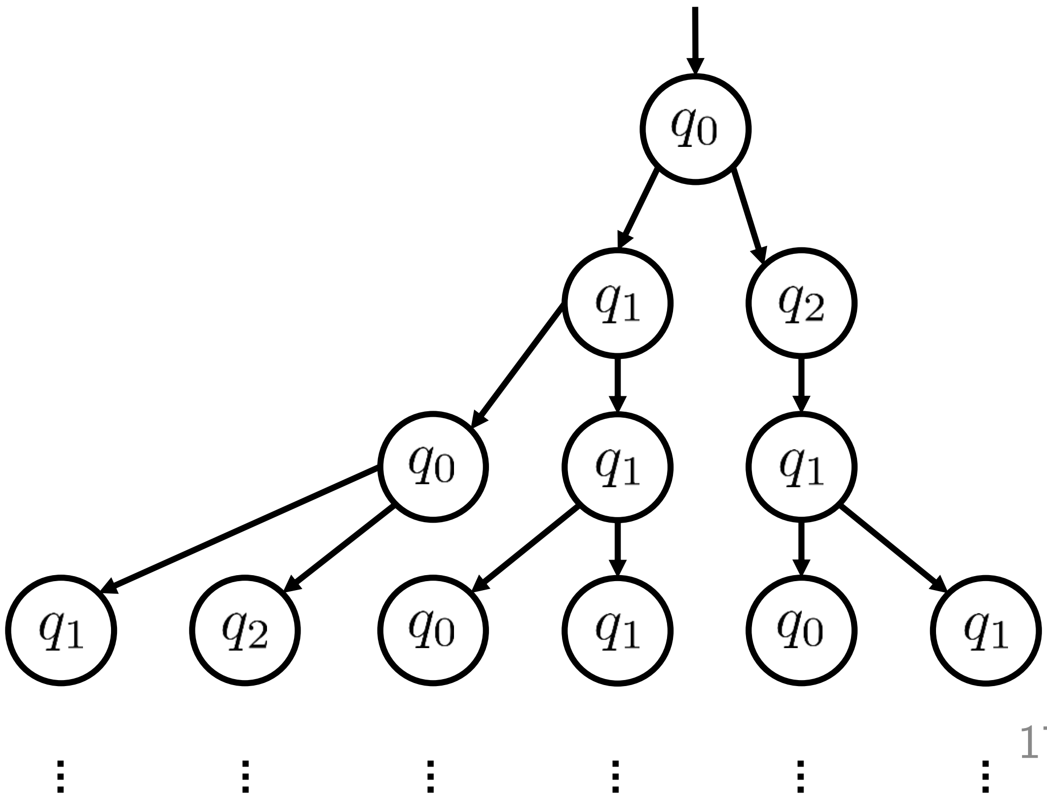
No need to know that one ▸ $\phi_1\mathrm{AU}\phi_2 \equiv \neg[(\neg\phi_1)\mathrm{EU}\neg(\phi_1+\phi_2)] + \mathrm{EG}(\neg\phi_2)$

CTL works on computation trees

Automaton



Computation tree



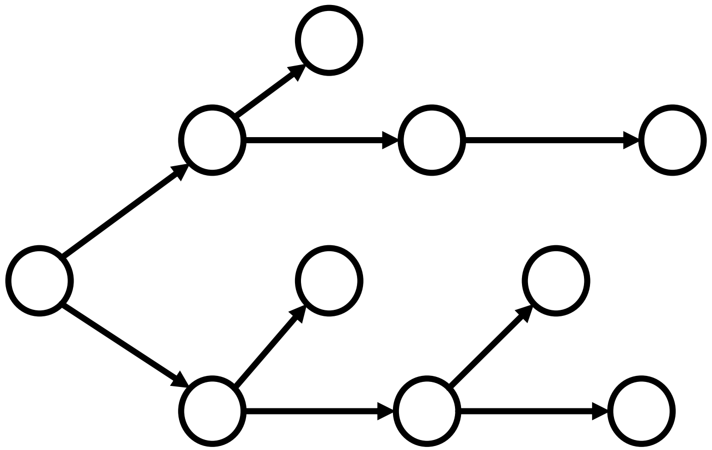
CTL works on computation trees

M satisfies $\phi \quad \Leftrightarrow \quad q_0 \vDash \phi$

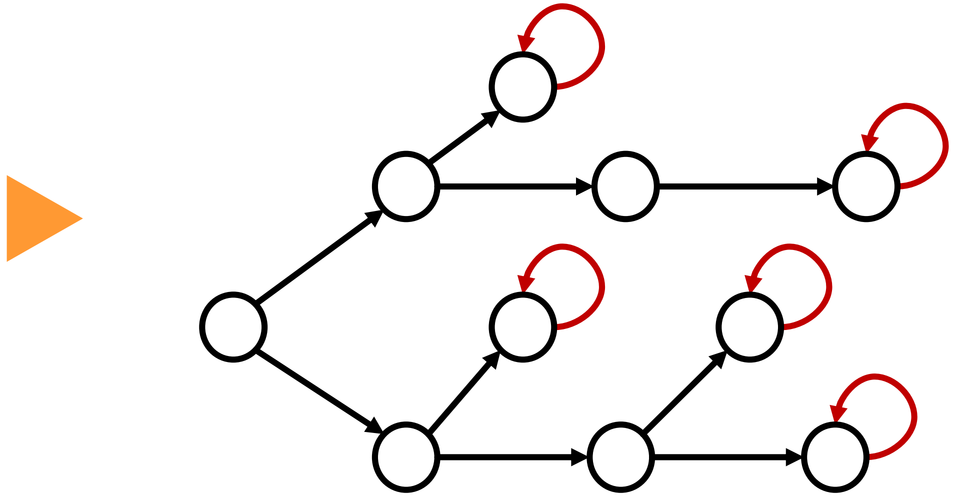
where q_0 is the initial state of M

Required fully-defined transition functions

Each state has at least one successor (can be itself)



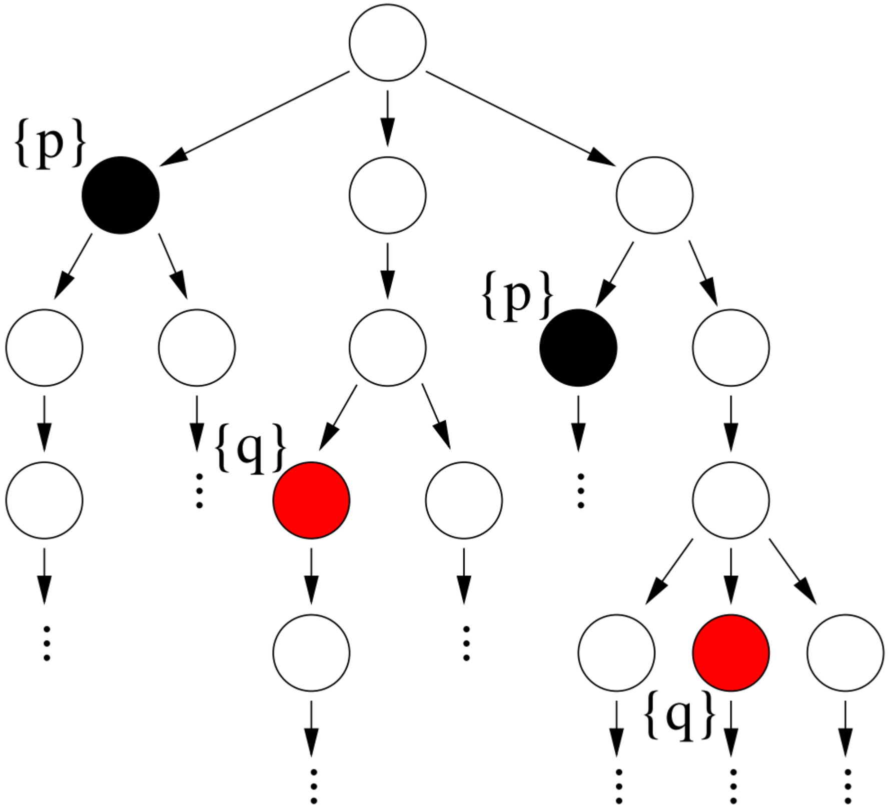
Automaton of interest



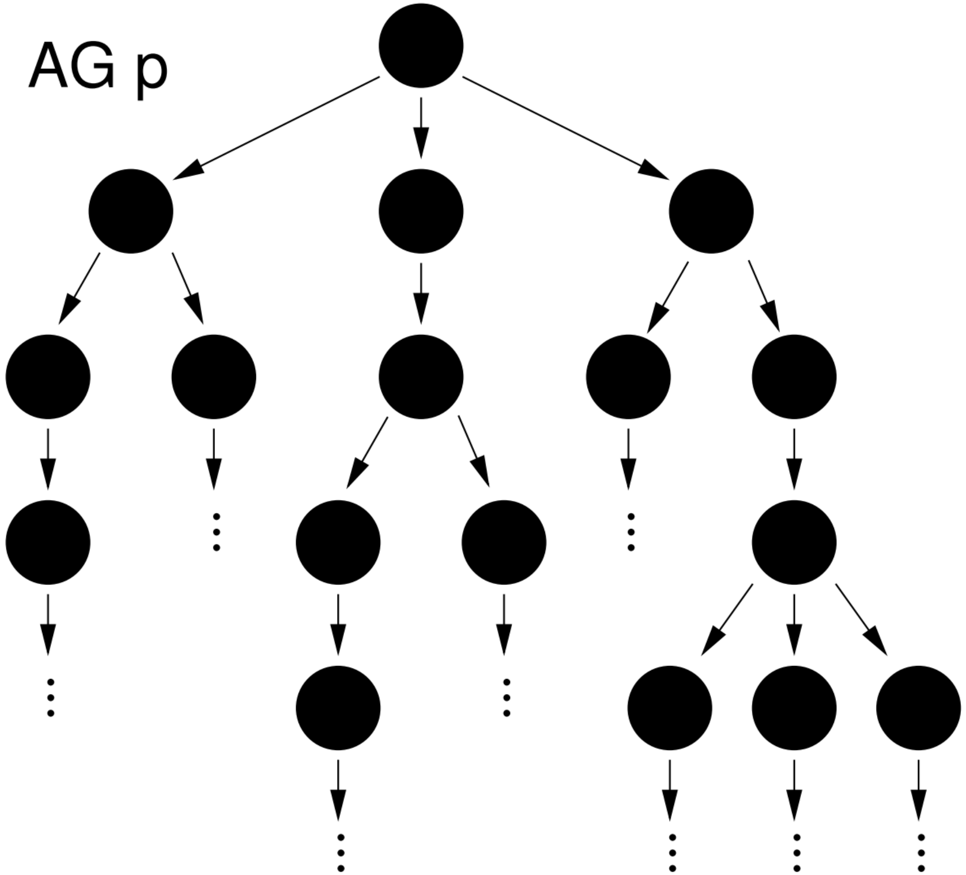
Automaton to work with

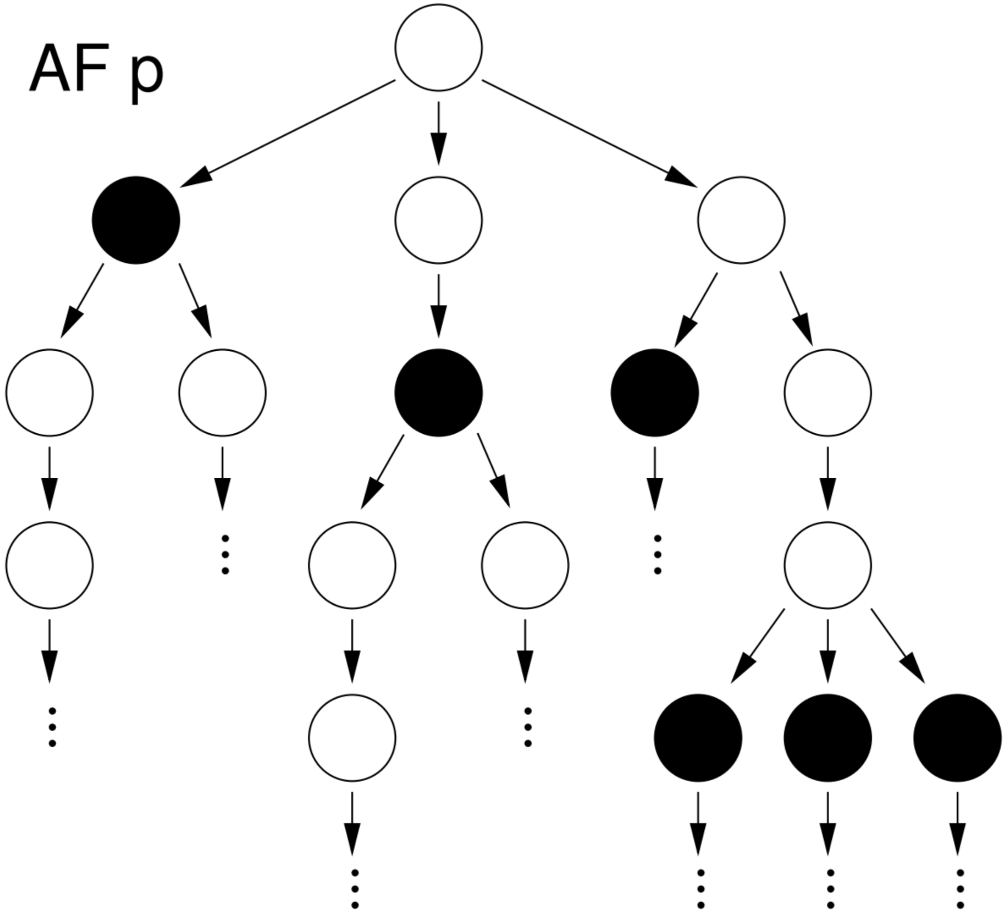
Visualizing CTL formula

- We use this computation tree
  as a running example.
- We suppose that the black and red states
  satisfy atomic properties p and q, respectively.
- The topmost state is the initial state;
  in the examples, it always satisfies the given formula.

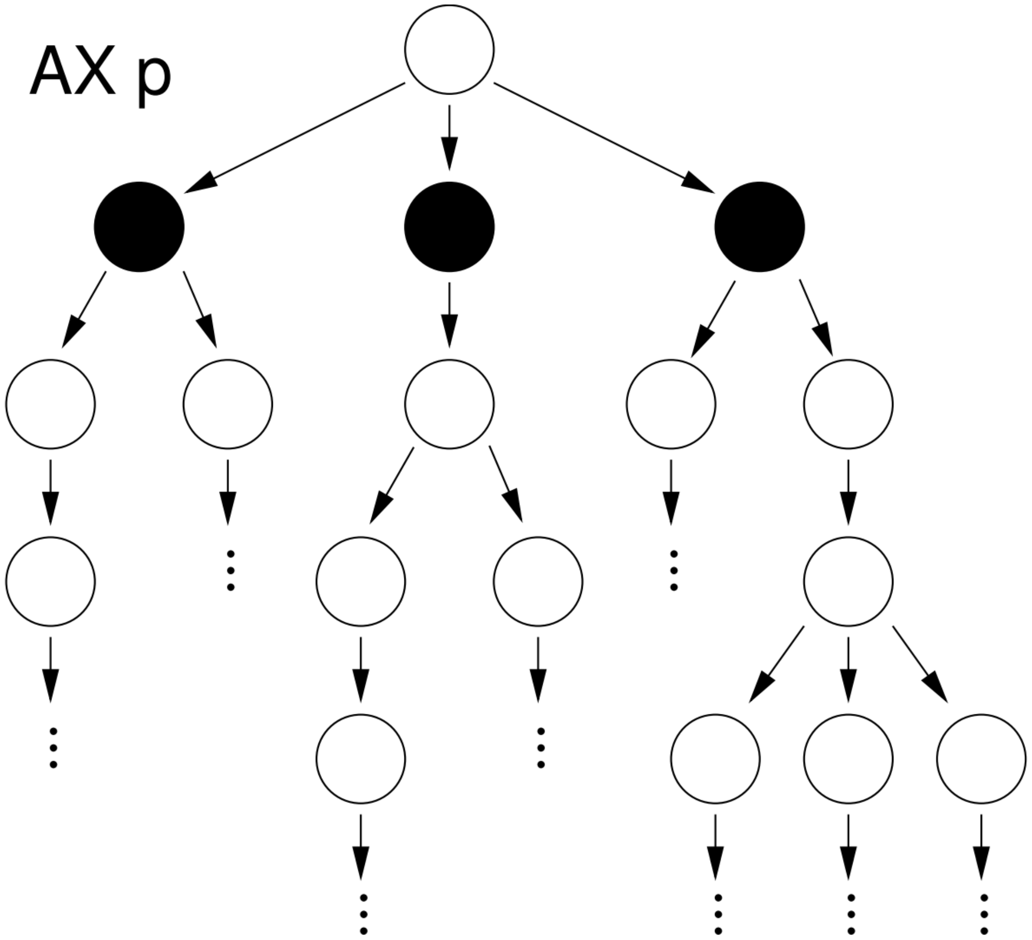


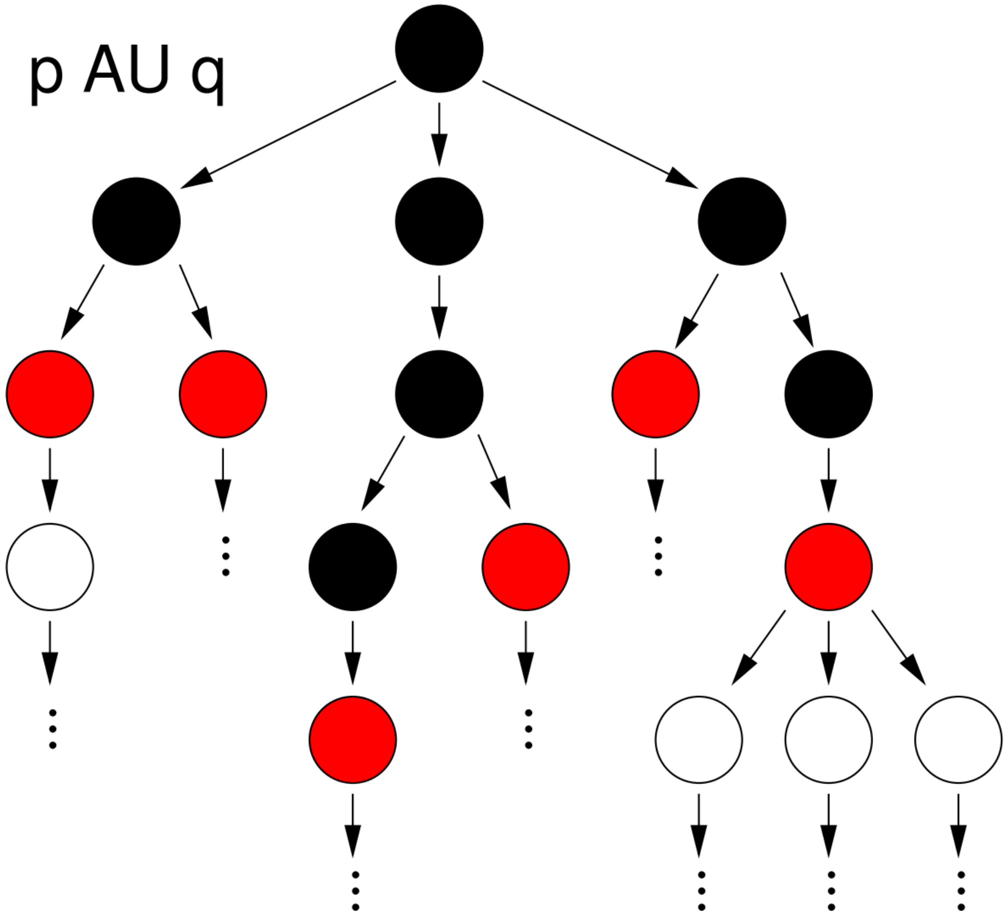
Visualizing CTL formula





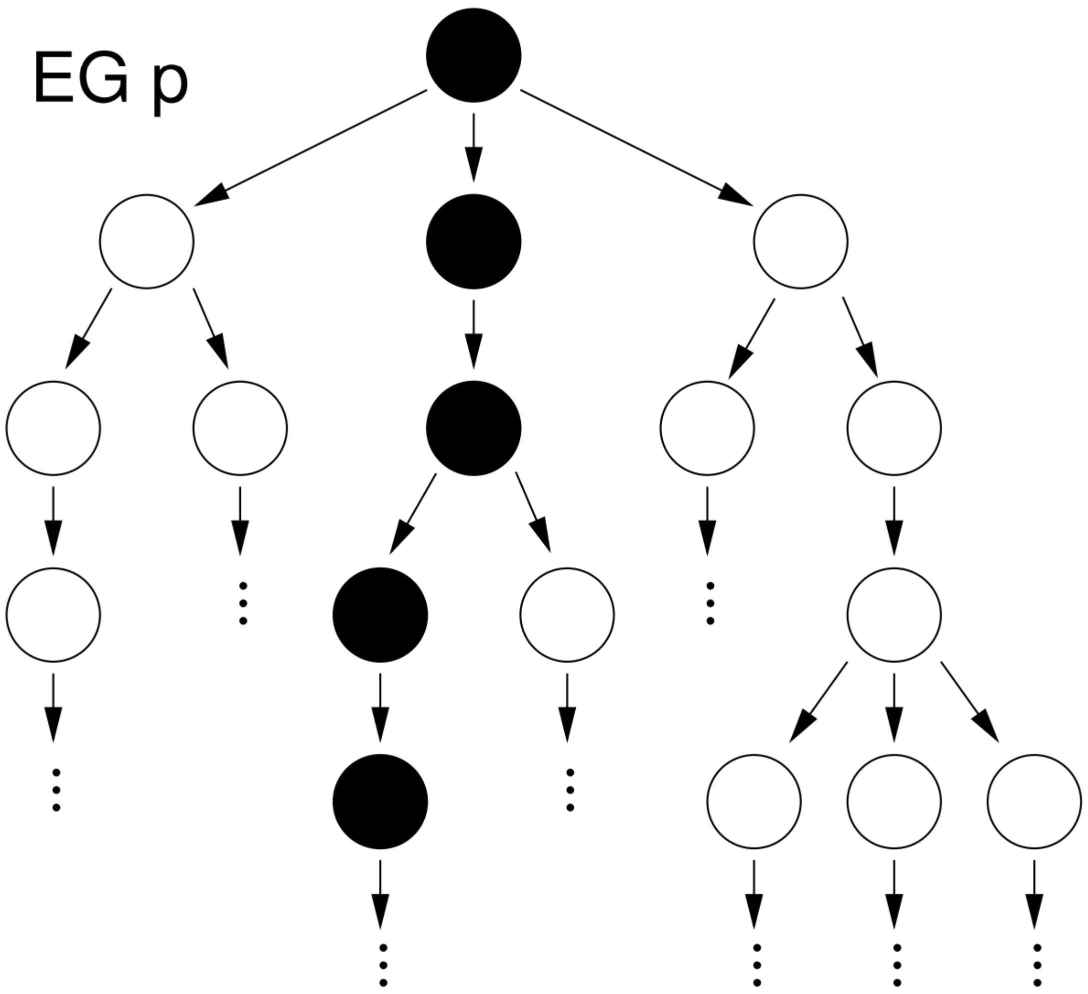
Visualizing CTL formula



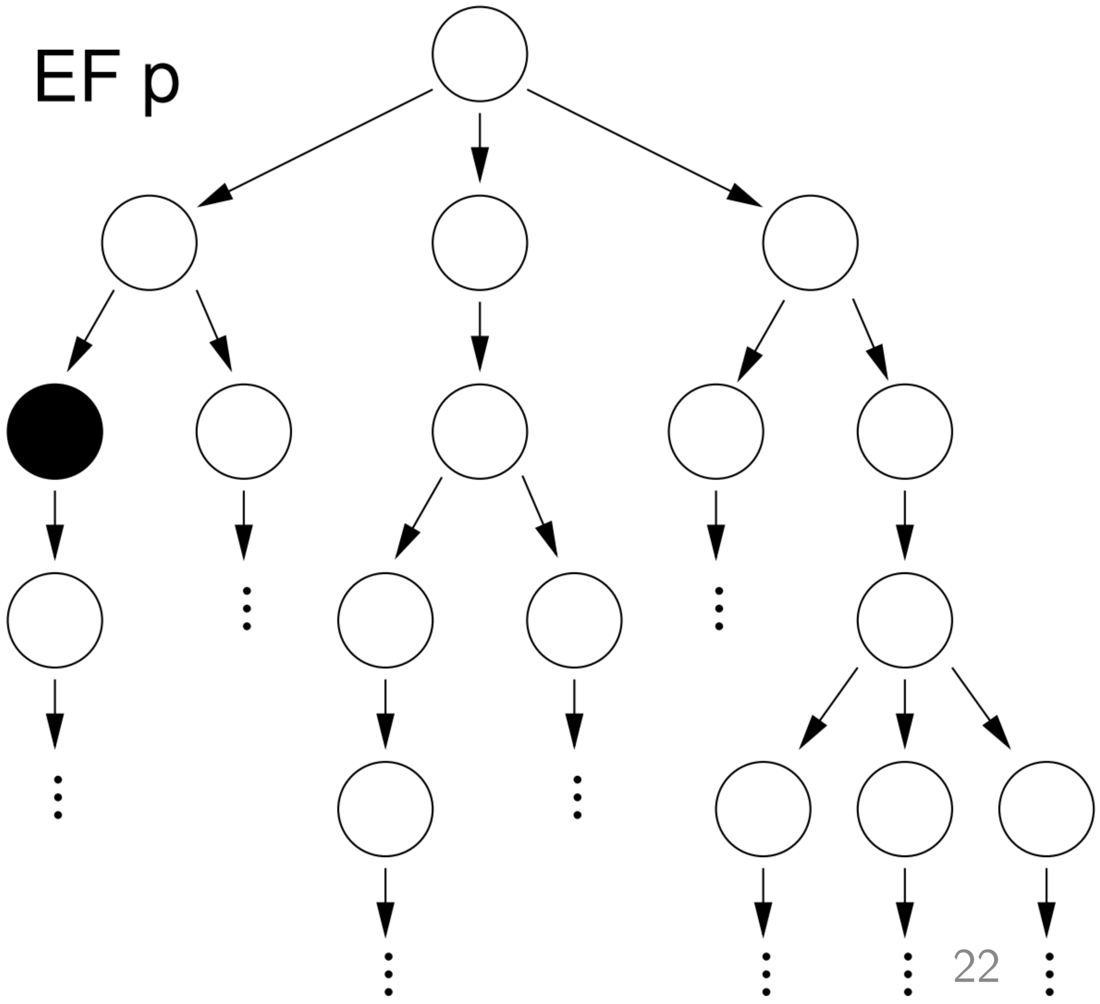


Visualizing CTL formula

EG p

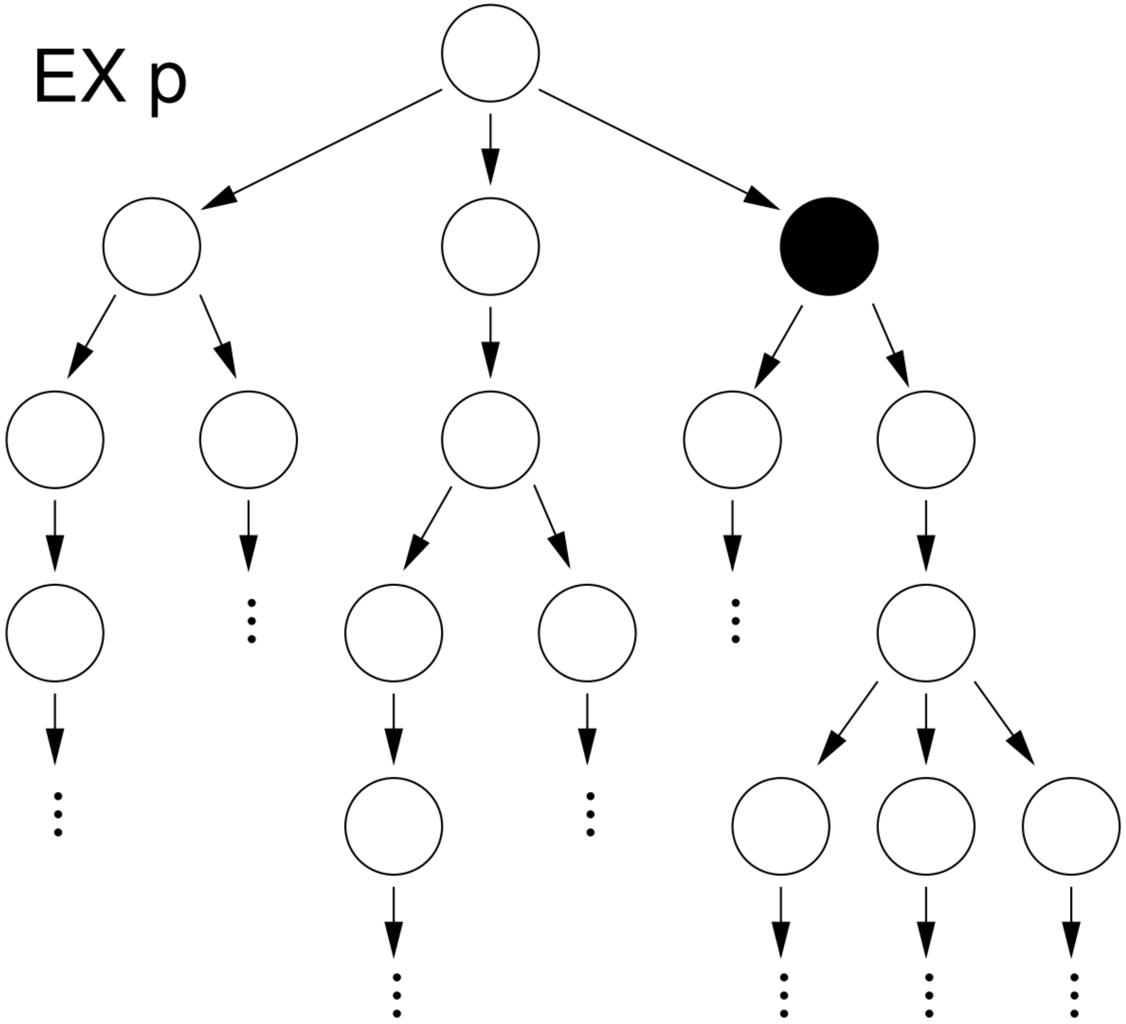


EF p

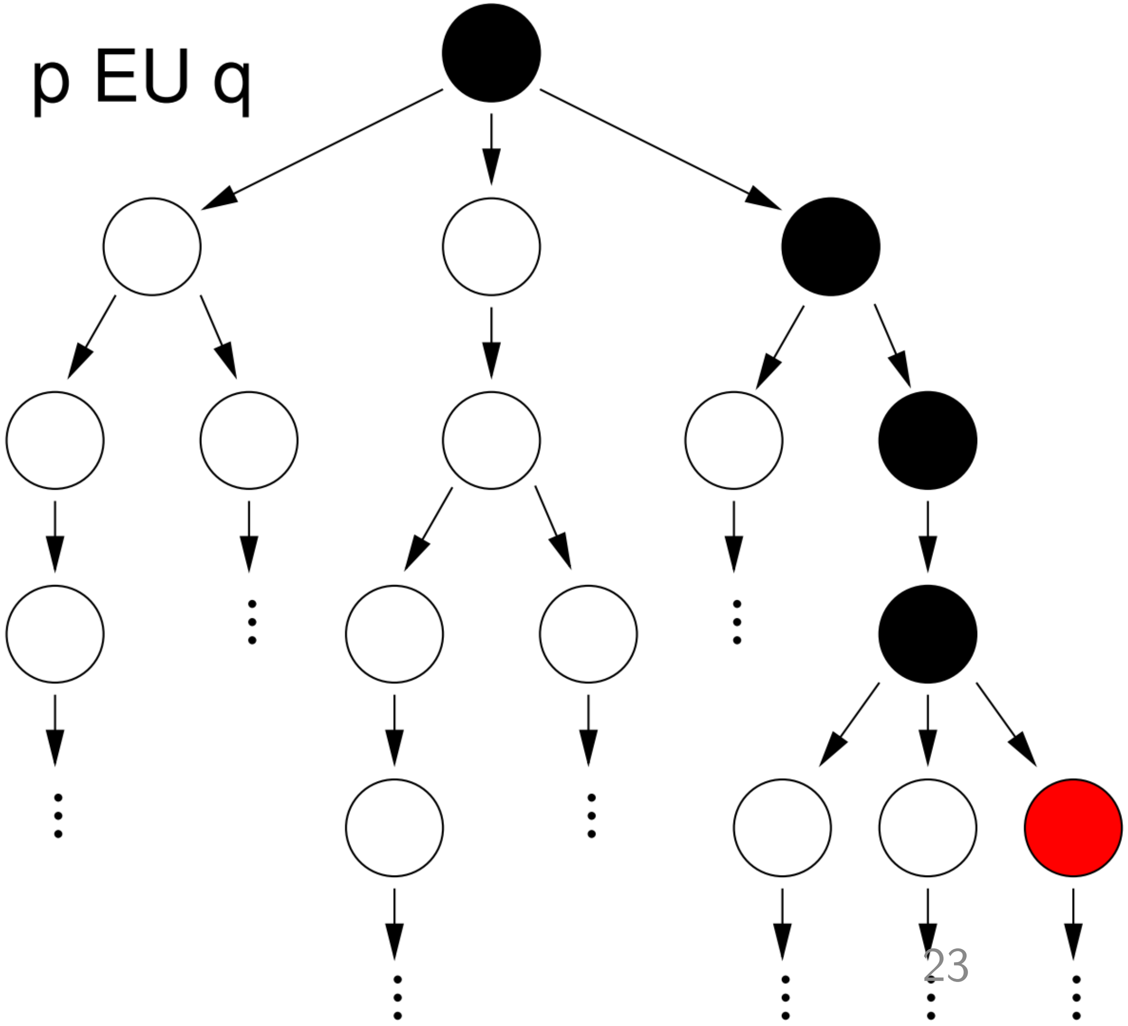


Visualizing CTL formula

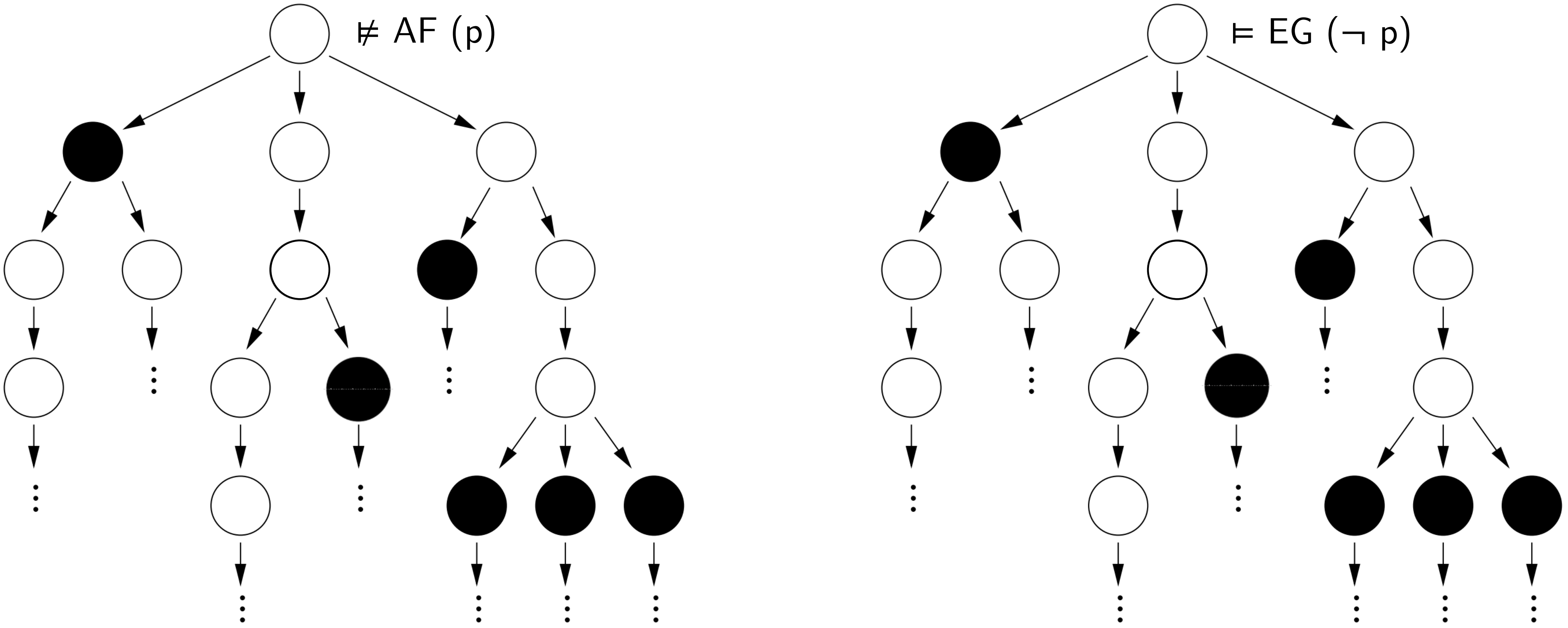
EX p



p EU q



Intuition for "AF p = ¬ EG (¬ p)"



Interpreting CTL formula

Encoding	Proposition
p	I like chocolate
q	It's warm outside

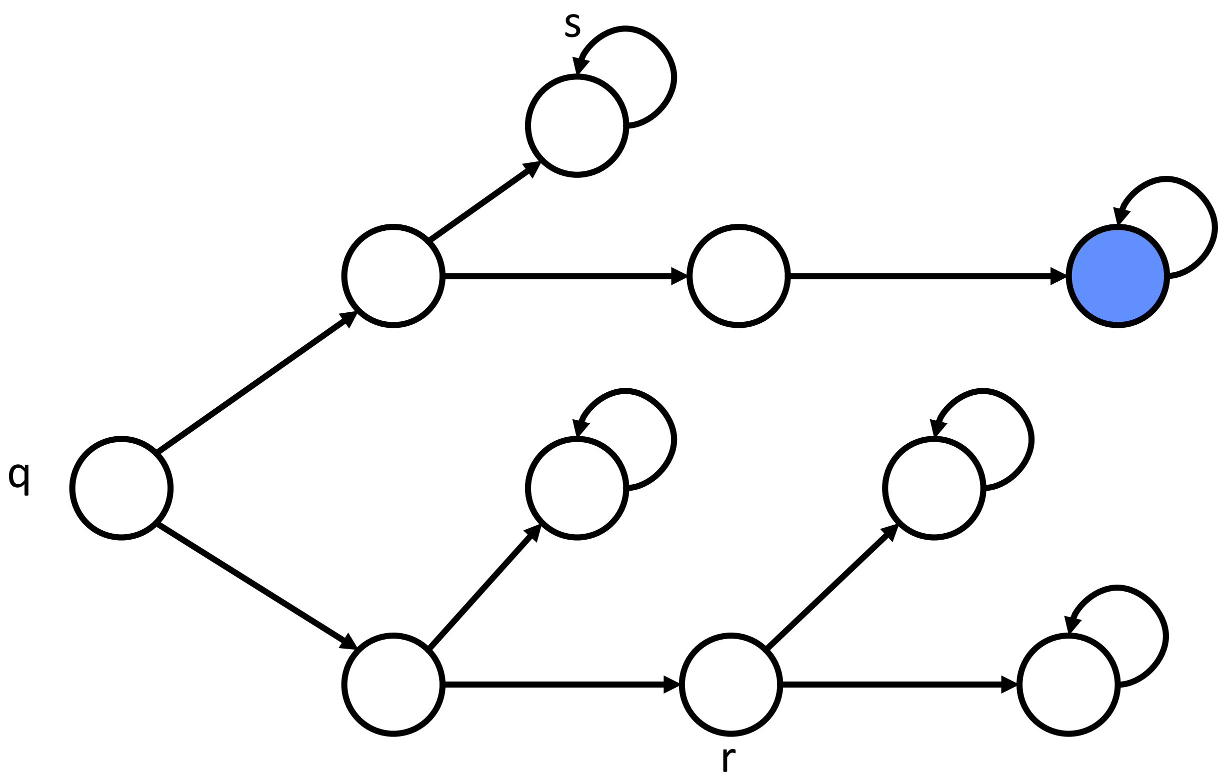
- AG p
- EF p
- AF EG p
- EG AF p
- p AU q

Interpreting CTL formula

Encoding	Proposition
p	I like chocolate
q	It's warm outside

- AG p — I will like chocolate from now on, no matter what happens.
- EF p — It's possible I may like chocolate someday, at least for one day.
- AF EG p — There will be always sometime in the future (AF) that I may suddenly start liking chocolate for the rest of time (EG).
- EG AF p — This is a critical time in my life. Depending on what happens (E), it's possible that for the rest of time (G), there will always be some time in the future (AF) when I will like chocolate. However, if the wrong thing happens next, then all bets are off and there's no guarantee about whether I will ever like chocolate.
- p AU q — No matter what happens, I will like chocolate from now on. But when it gets warm outside, I don't know whether I still like it. And it will get warm outside someday.

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



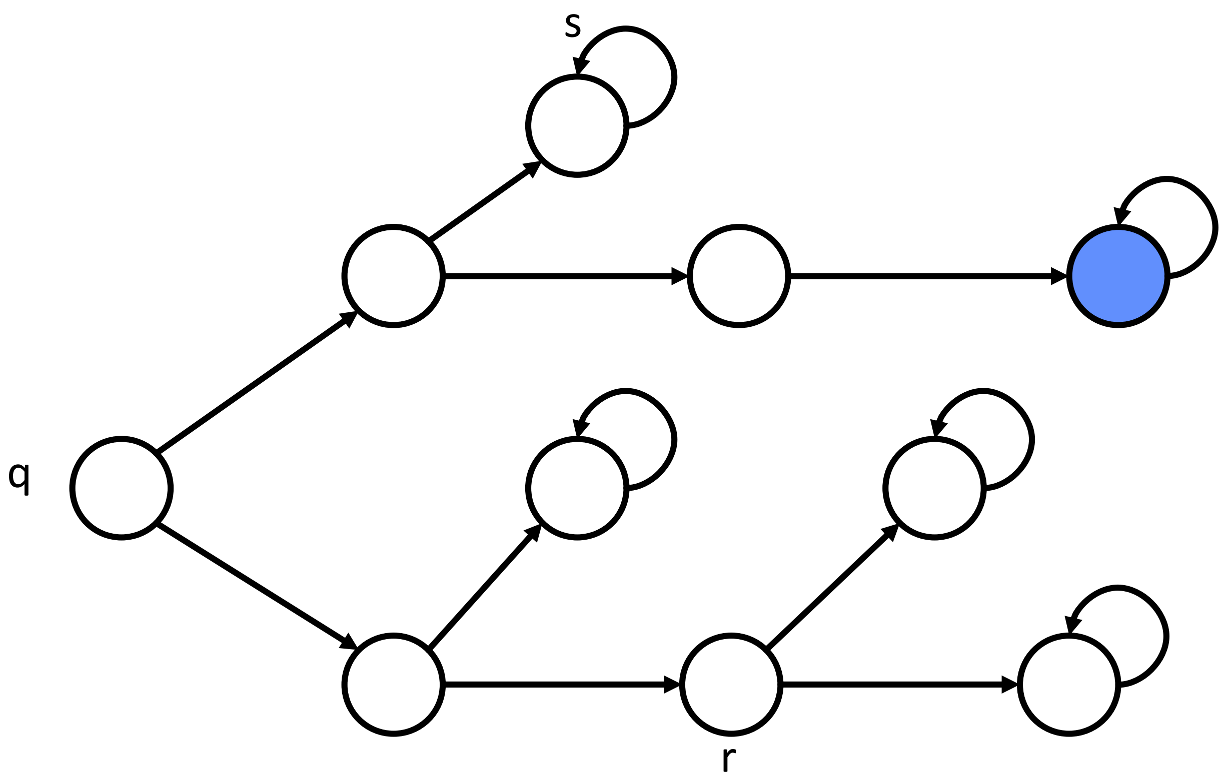
$\models \phi$

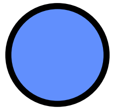
q $\models$ EF ϕ

r $\models$?

s $\models$?

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



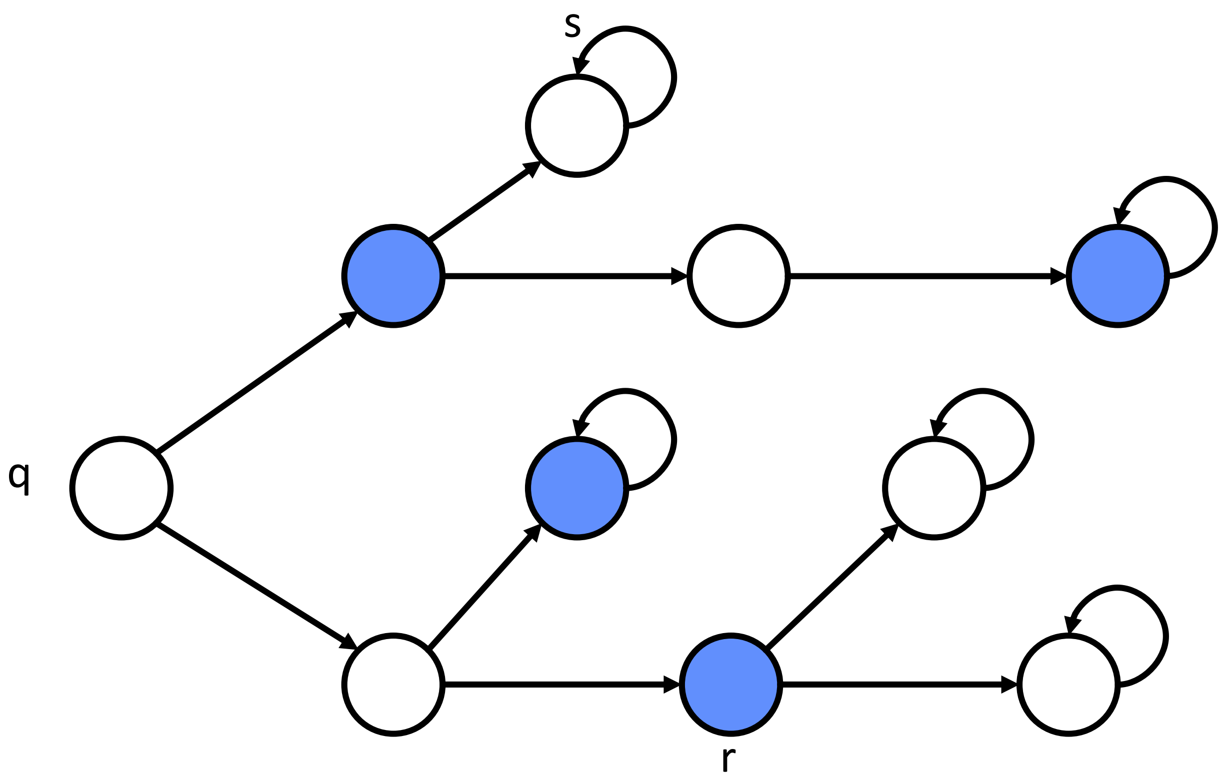
 $\models \phi$

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

$r \not\models \mathrm{EF}\ \phi$

$s \not\models \mathrm{EF}\ \phi$

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



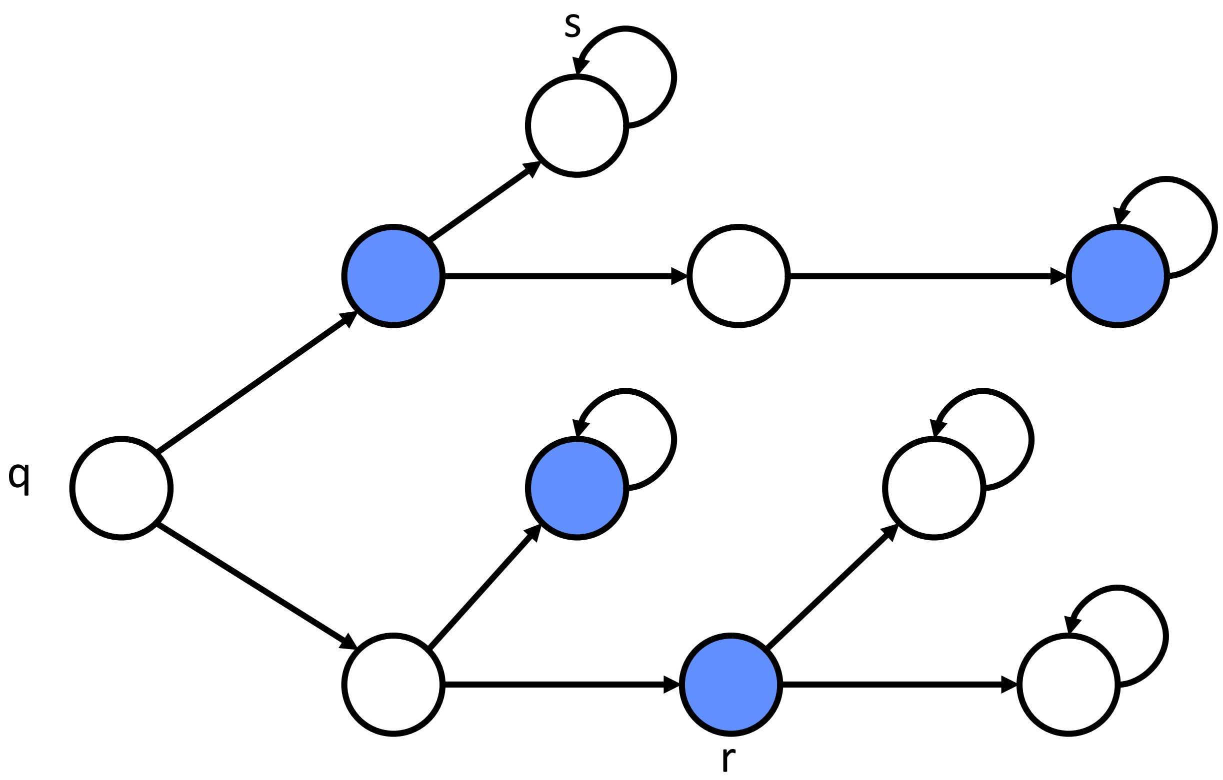
$\models \phi$

q $\models$ AF ϕ

r $\models$?

s $\models$?

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



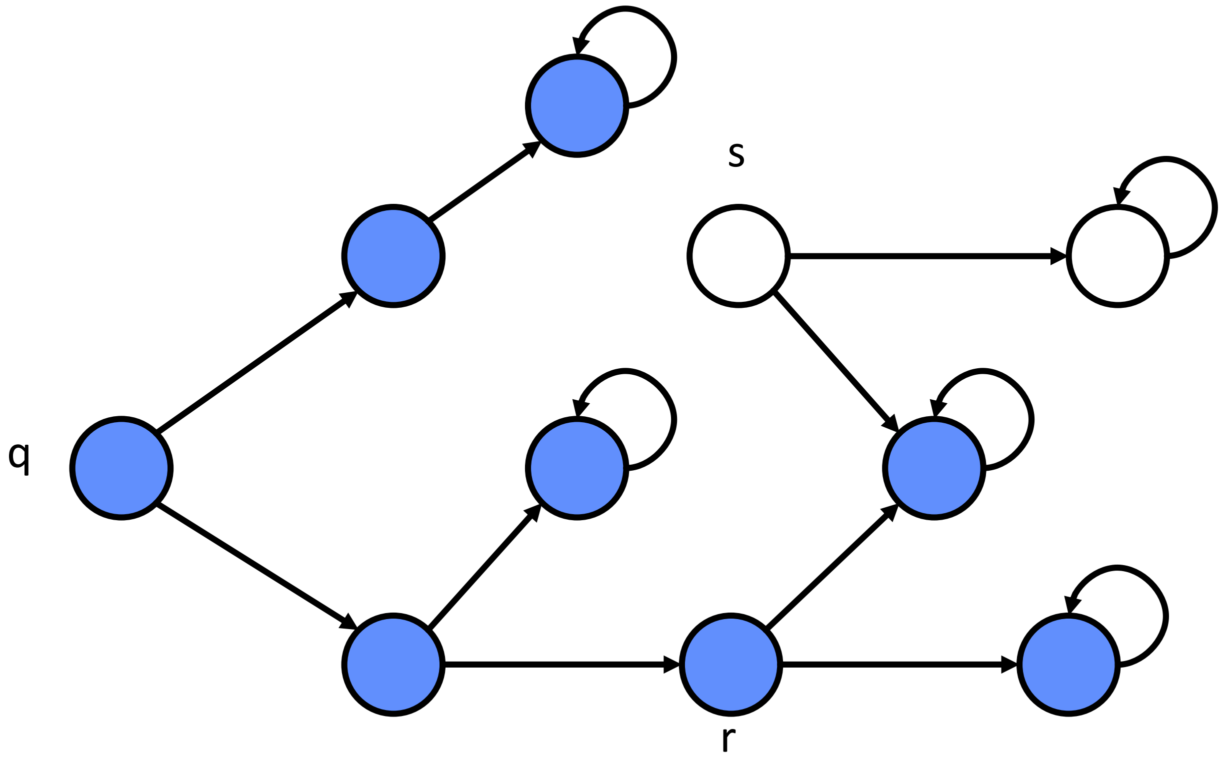
$\bullet \models \phi$

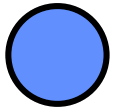
$q \models AF\ \phi$

$r \models AF\ \phi$

$s \not\models AF\ \phi$

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



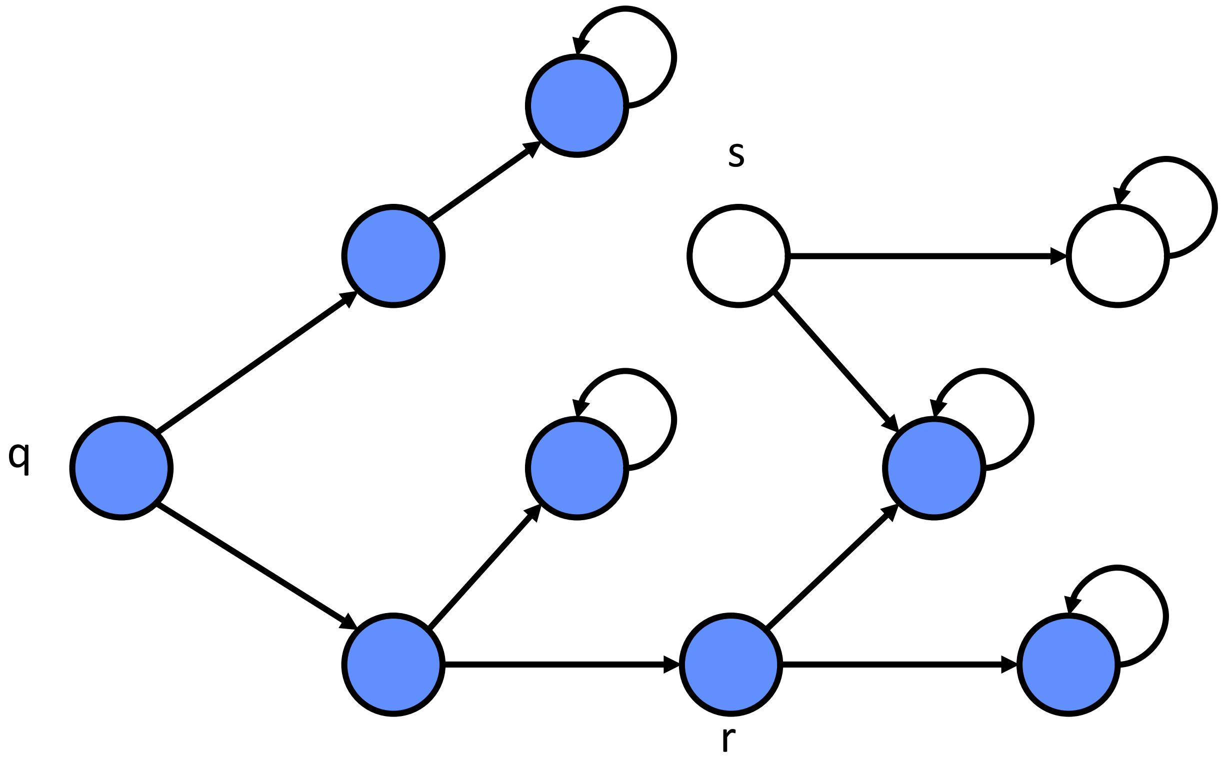
 $\models \phi$

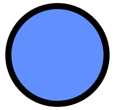
$q \models \text{AG } \phi$

$r \models ?$

$s \models ?$

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





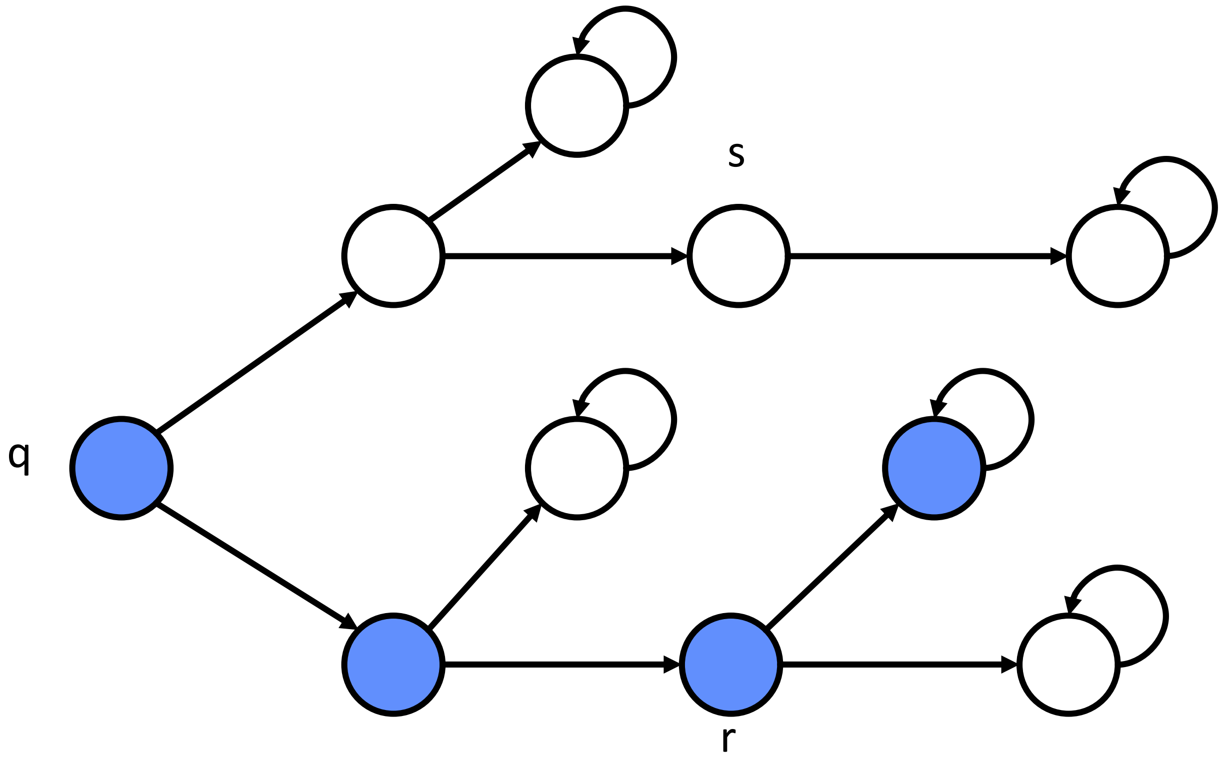
$\vDash \phi$

$q \vDash \text{AG}\ \phi$

$r \vDash \text{AG}\ \phi$

$s \nvDash \text{AG}\ \phi$

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



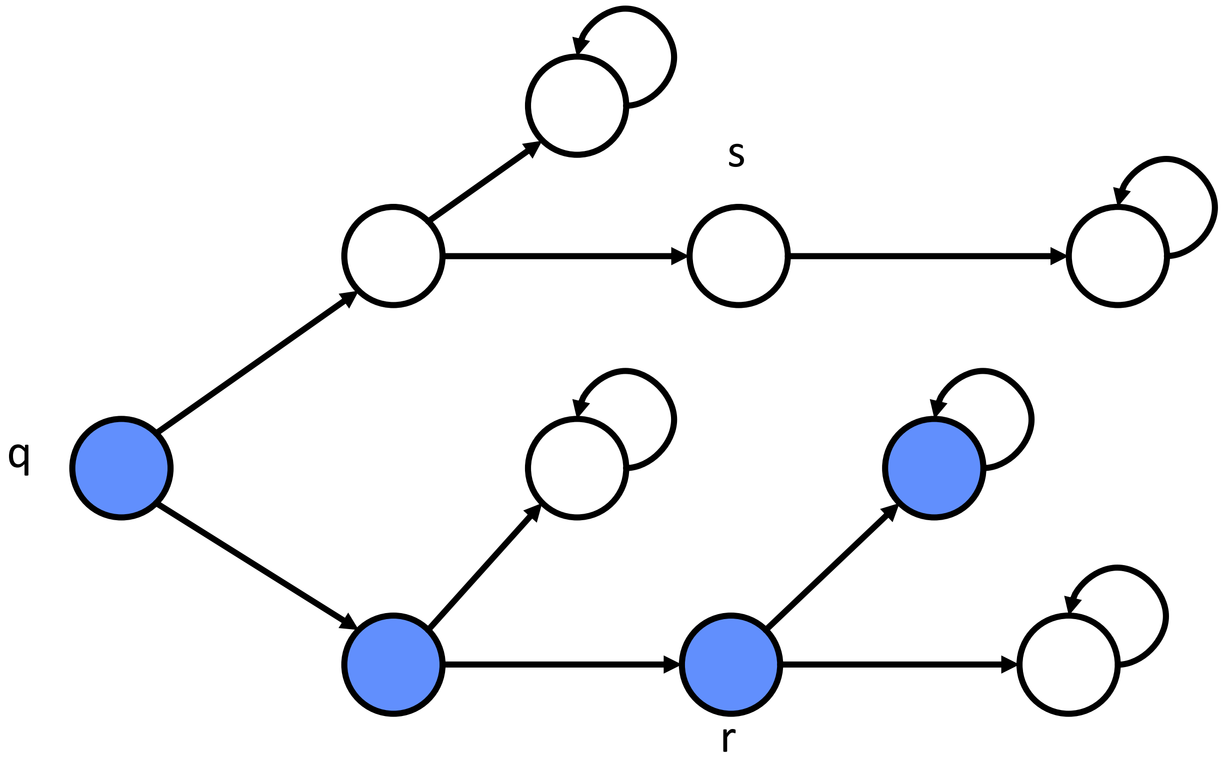
$\models \phi$

q $\models$ EG ϕ

r $\models$?

s $\models$?

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



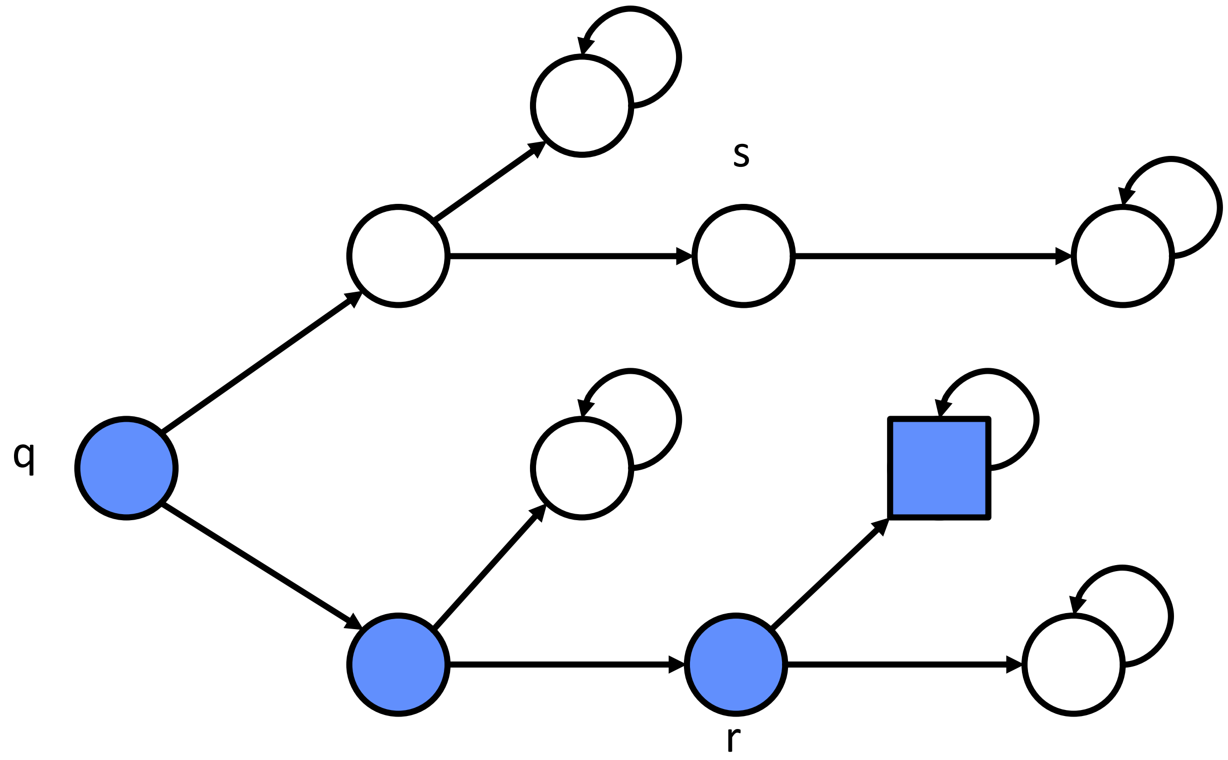
$\models \phi$

q $\models$ EG ϕ

r $\models$ EG ϕ

s $\not\models$ EG ϕ

ϕEUΨ : "There exists a path along which ϕ holds until Ψ holds."



$\blacksquare \models \Psi$

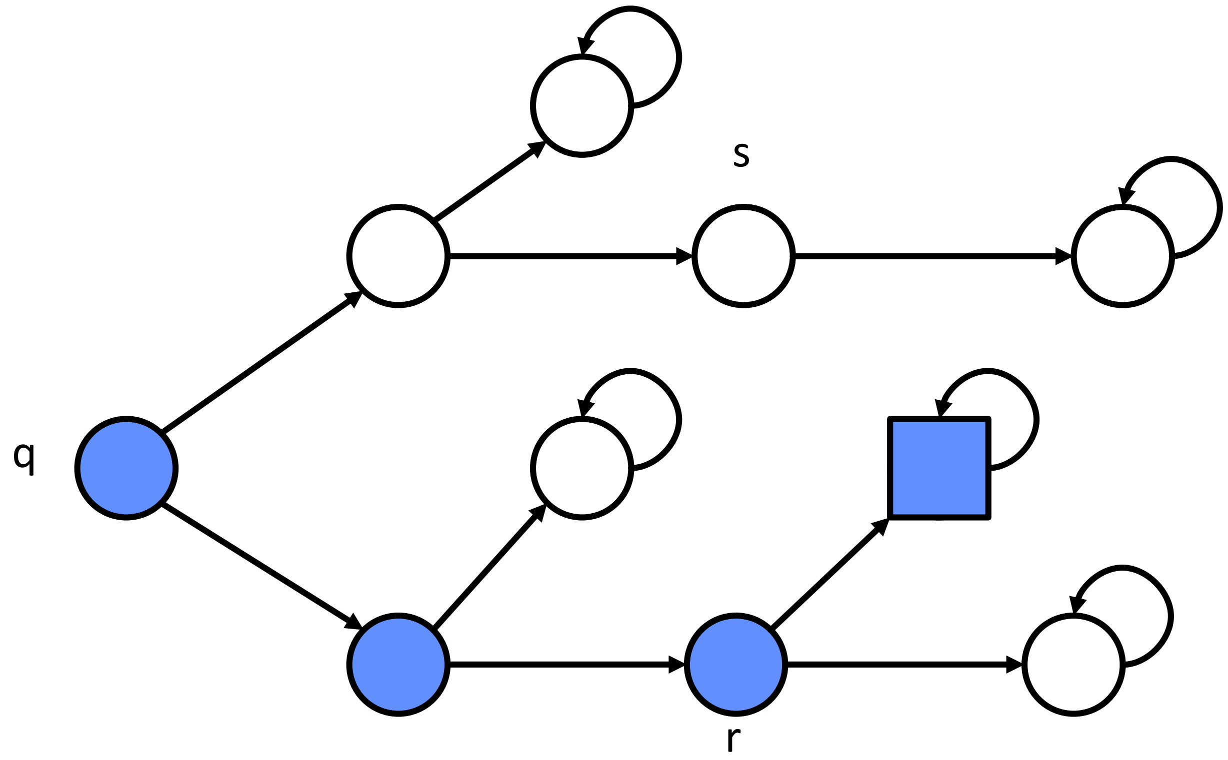
$\bullet \models \phi$

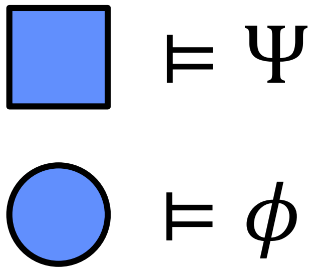
$q \models \phi \mathrm{EU} \Psi$

$r \models ?$

$s \models ?$

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



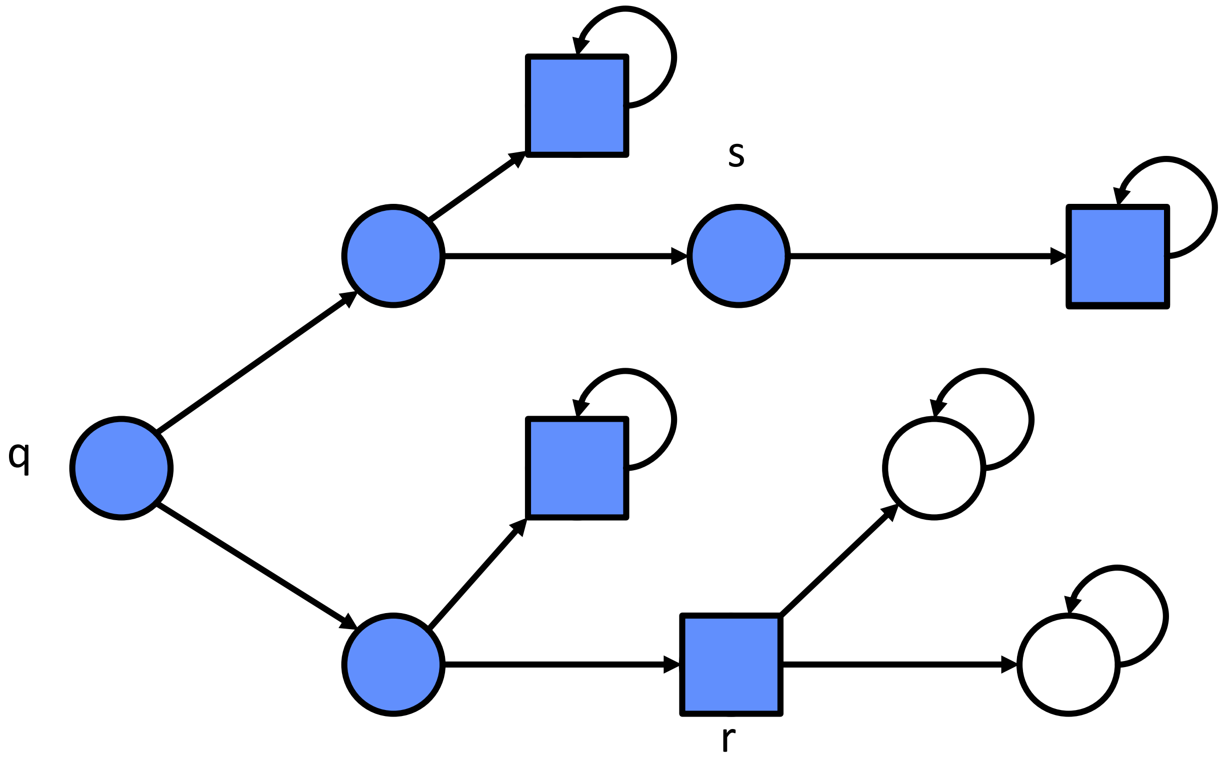


$q \vDash \phi \mathsf{EU} \Psi$

$r \vDash \phi \mathsf{EU} \Psi$

$s \nvDash \phi \mathsf{EU} \Psi$

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



$\vDash \Psi$

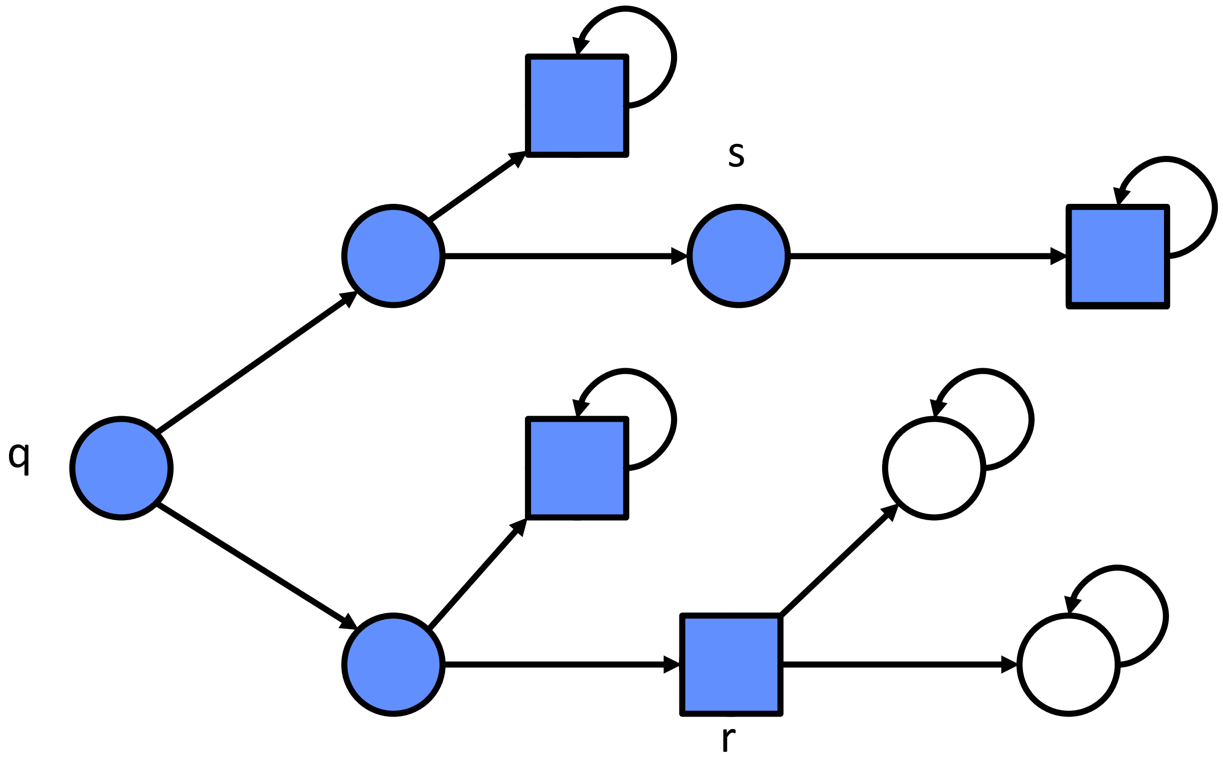
$\vDash \phi$

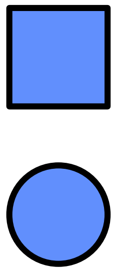
$q \vDash \phi \mathrm{AU} \Psi$

$r \vDash ?$

$s \vDash ?$

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





$\models \Psi$

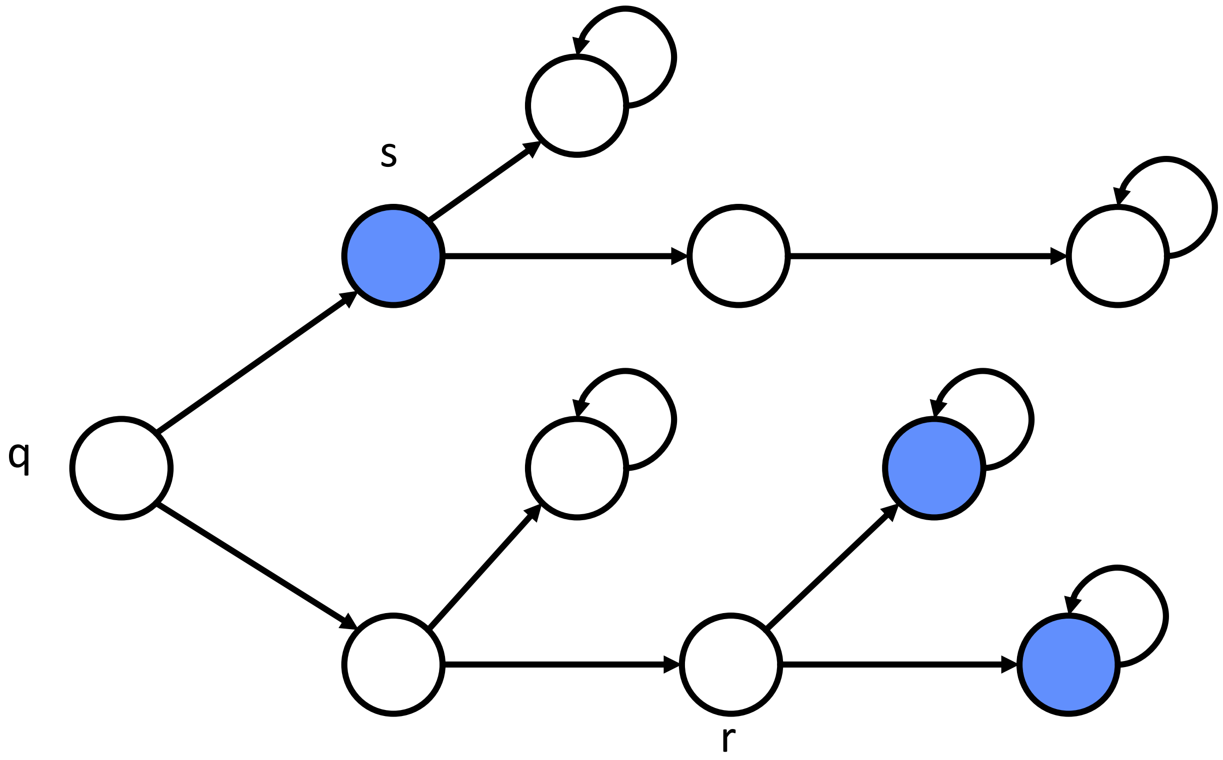
$\models \phi$

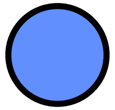
q $\models \phi \mathrm{AU} \Psi$

r $\models \phi \mathrm{AU} \Psi$

s $\models \phi \mathrm{AU} \Psi$

EXϕ : "There exists a path along which the next state satisfies ϕ."



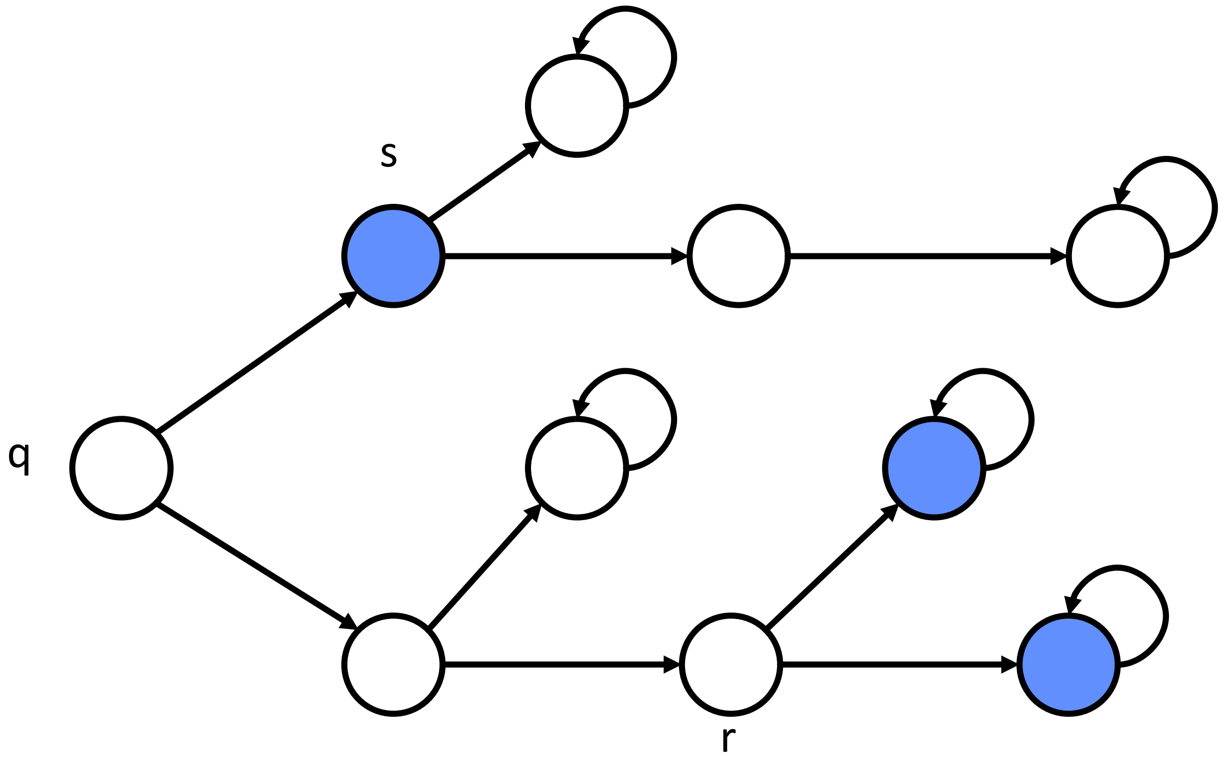
 $\models \phi$

q $\models$ EXϕ

r $\models$?

s $\models$?

EXϕ : "There exists a path along which the next state satisfies ϕ."



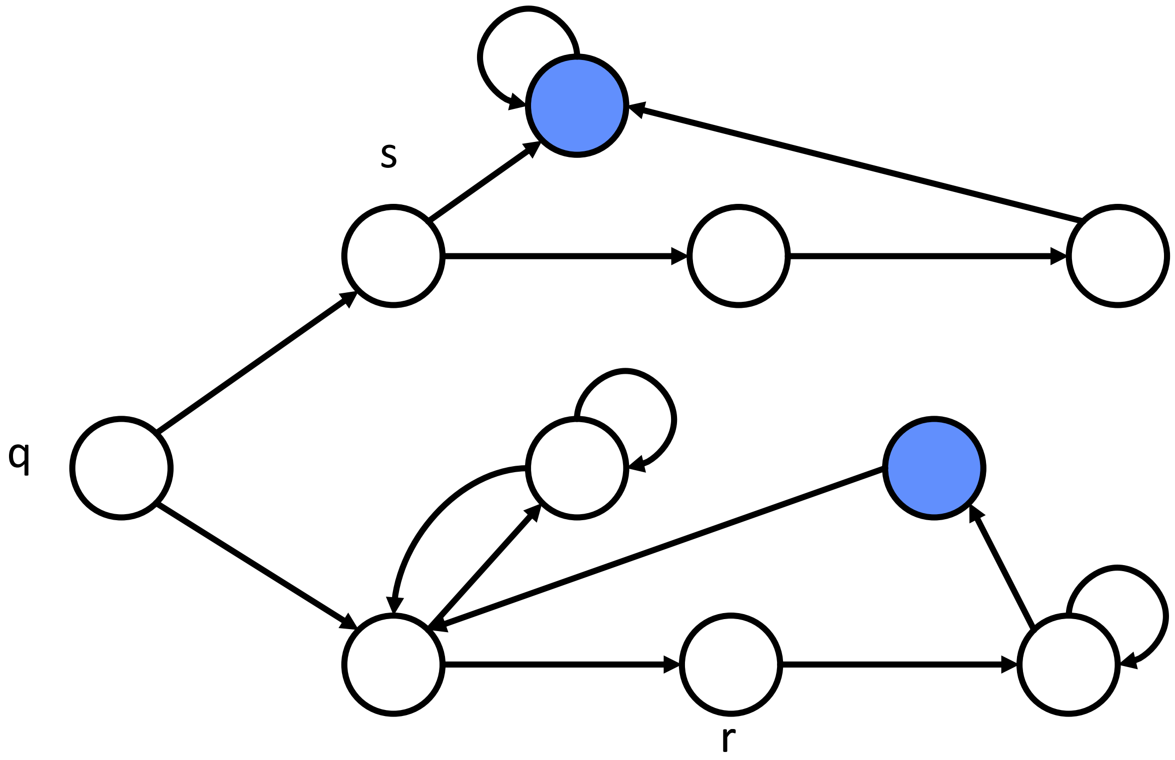
$\models \phi$

q $\models$ EXϕ

r $\models$ EXϕ

s $\not\models$ EXϕ

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



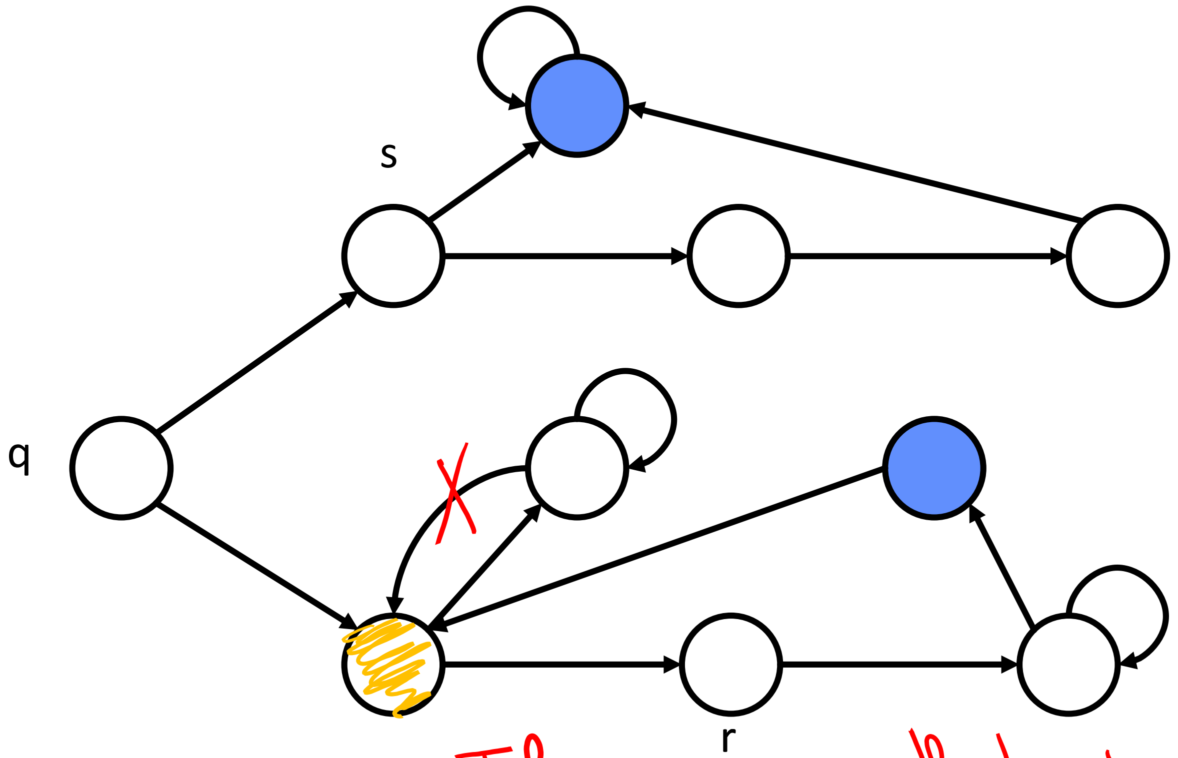
$\models \phi$

q $\models$ AG EF ϕ

r $\models$?

s $\models$?

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



$\bullet \vDash \phi$

$q \vDash \text{AG EF } \phi$

$r \vDash \text{AG EF } \phi$

$s \vDash \text{AG EF } \phi$

If we remove that edge, then

$r \nvDash \text{AG EF } \phi$

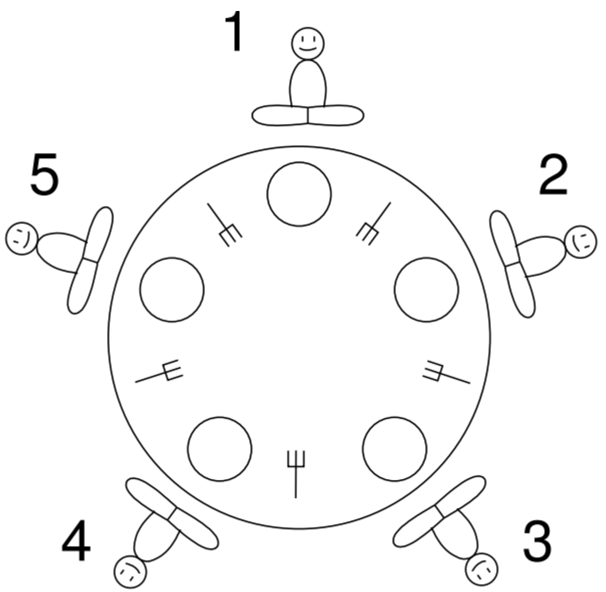
Because even if we can reach ● from r, once we reach ● it is no longer possible

Specifying using CTL formula

Famous problem

Dining Philosophers

- Five philosophers are sitting around a table, taking turns at thinking and eating.
- Each needs two forks to eat.
- They put down forks.
  only once they have eaten.
- There are only five forks.

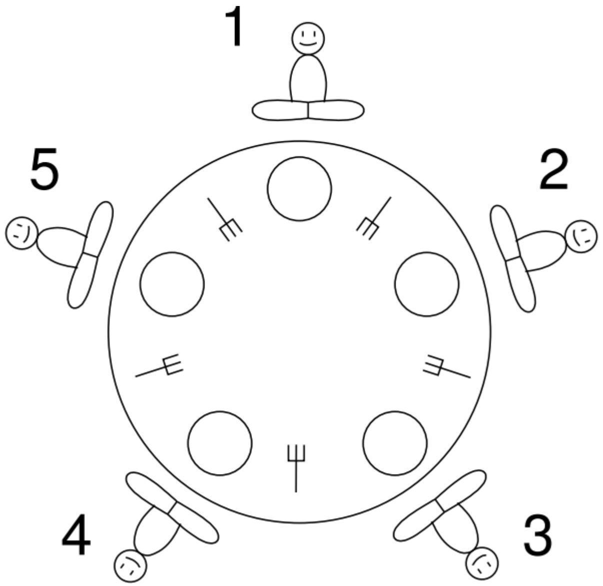


Atomic proposition

e_i : Philosopher i is currently eating.

Specifying using CTL formula

- "Philosophers 1 and 4 will never eat at the same time."
- "Every philosopher will get infinitely many turns to eat."
- "Philosopher 2 will be the first to eat."



Specifying using CTL formula

- "Philosophers 1 and 4 will never eat at the same time."

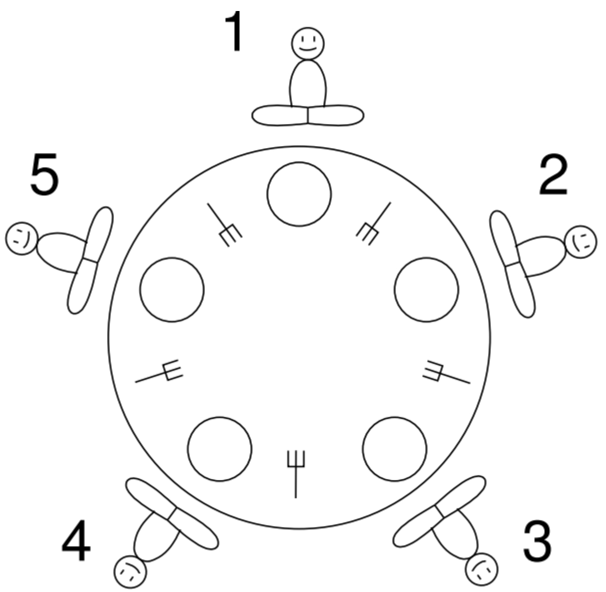
$$\mathrm{AG}\neg(e_1 \cdot e_4)$$

- "Every philosopher will get infinitely many turns to eat."

$$\mathrm{AG}(\mathrm{AF}e_1 \cdot \mathrm{AF}e_2 \cdot \mathrm{AF}e_3 \cdot \mathrm{AF}e_4 \cdot \mathrm{AF}e_5)$$

- "Philosopher 2 will be the first to eat."

$$\neg(e_1 + e_3 + e_4 + e_5)\,\mathrm{AU}\,e_2$$



Computing CTL formula

- In order to compute CTL formula, we first define $[\![\phi]\!]$ as the set of all initial states of the finite automaton for which CTL formula ϕ is true. Then we can say that a finite automaton with initial state q_0 satisfies ϕ iff

$$q_0 \in [\![\phi]\!]$$

- Now, we can use our "trick": computing with sets of states!
  - $\psi_{[\![\phi]\!]}(q)$ is true if the state q is in the set $[\![\phi]\!]$, i.e., it is a state for which the CTL formula is true.
  - Therefore, we can also say

$$q_0 \in [\![\phi]\!] \equiv \psi_{[\![\phi]\!]}(q_0)$$

characteristic function of the set $[\![\phi]\!]$

- When we compute the CTL-formulas, we start from the innermost terms.
- Remember: We suppose that every state has at least one successor state (could be itself).

Computing CTL formula

- We now show how to compute some operators in CTL. All others can be determined using the equivalence relations between operators that we listed earlier.
  - EX ϕ : Let us first define the set of predecessor states of Q, i.e., the set of states that lead in one transition to a state in Q:

$$Q' = Pre(Q, \delta) = \{q' \mid \exists q : \psi_\delta(q', q) \cdot \psi_Q(q)\}$$

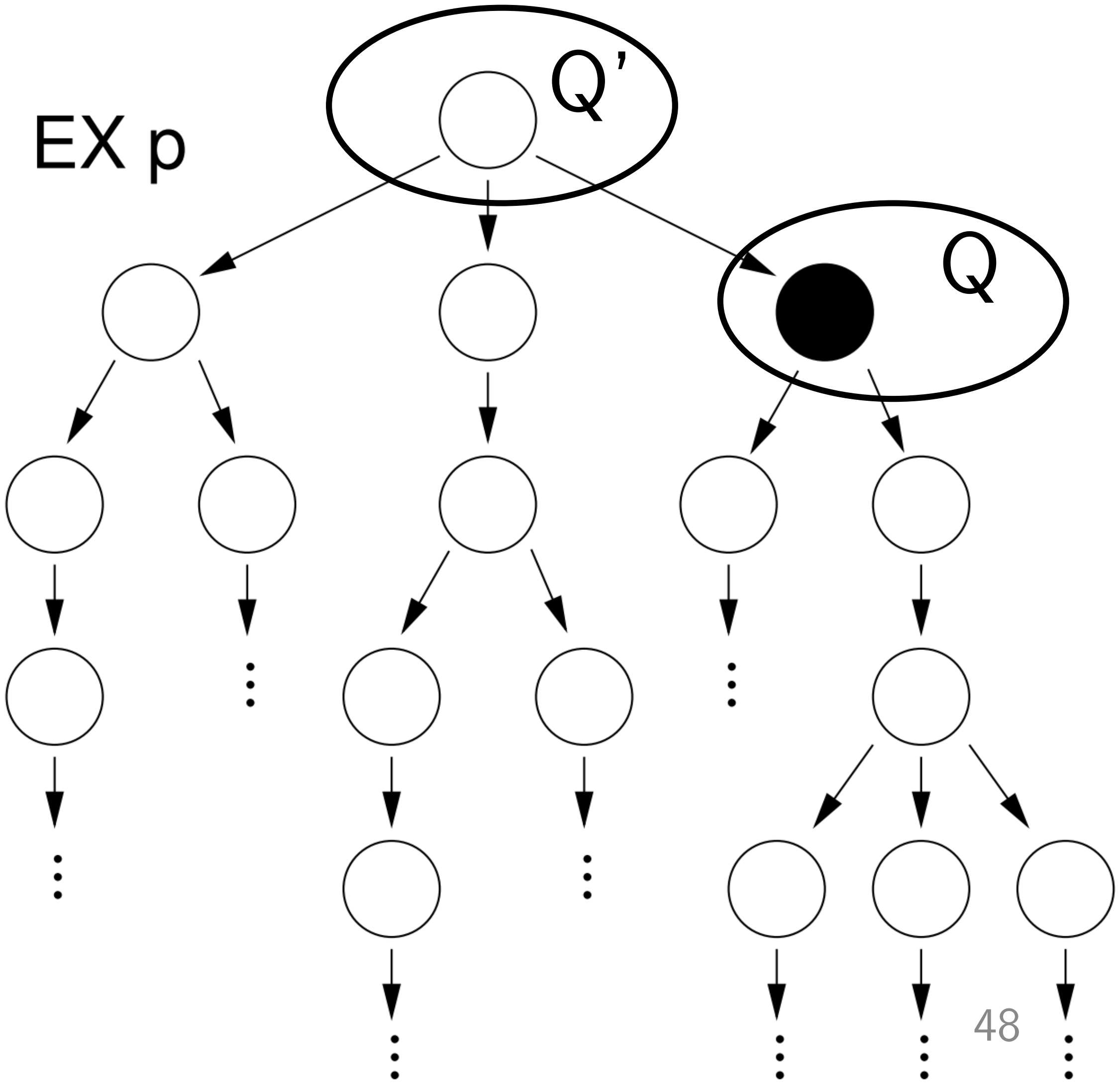
Suppose that Q is the set of initial states for which the formula ϕ is true. Then we can write

sets $\{$

characteristic functions $\{$

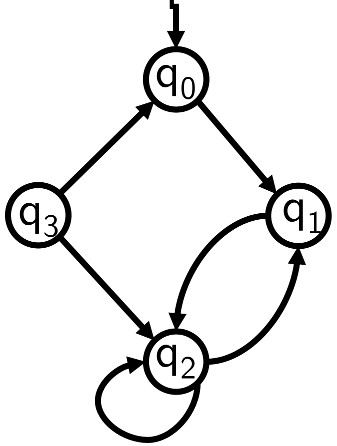
$$\begin{tikzcd}
\boxed{Q = [\![\phi]\!]} \arrow[r] \arrow[d, leftrightarrow] & \boxed{Q' = [\![\mathrm{EX}\phi]\!] = \mathrm{Pre}([\![\phi]\!], \delta)} \arrow[d, leftrightarrow] \\
\boxed{\psi_Q(q)} \arrow[r] & \boxed{\psi_{Q'}(q') = (\exists q \ : \ \psi_Q(q) \cdot \psi_\delta(q', q))}
\end{tikzcd}$$

Computing CTL formula



Computing CTL formula

- Example for EX ϕ : Compute EX q_2



$$[\![q_2]\!] = \{q_2\}$$

$$Q' = [\![EX\ q_2]\!] = Pre(\{q_2\}, \delta) = \{q_1, q_2, q_3\}$$

$$\{q' \mid \exists q : \psi_\delta(q', q) \cdot \psi_Q(q)\}$$

As $q_0 \notin [\![EX\ q_2]\!] = \{q_1, q_2, q_3\}$, the CTL formula EX q_2 is not true.

Computing CTL formula

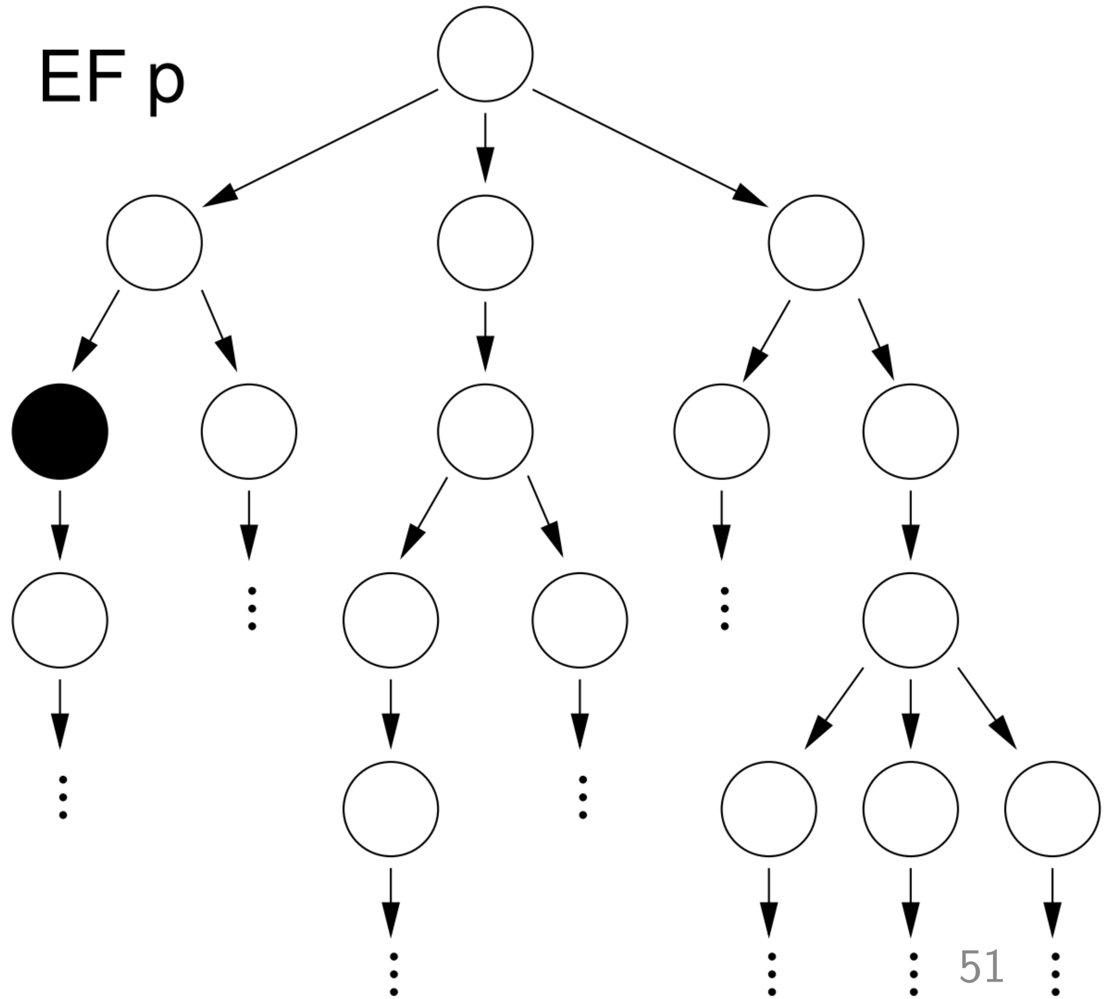
- EF ϕ: The idea here is to start with the set of initial states for which the formula ϕ is true. Then we add to this set the set of predecessor states. For the resulting set of states we do the same, …., until we reach a fixed-point. The corresponding operations can be done using BDDs (as described before).

$$Q_0 = [\![\phi]\!]$$

$$Q_i = Q_{i-1} \cup \mathrm{Pre}(Q_{i-1}, \delta) \qquad \text{for all } i > 1 \text{ until a fixed-point } Q' \text{ is reached}$$

$$[\![\mathrm{EF}\phi]\!] = Q'$$

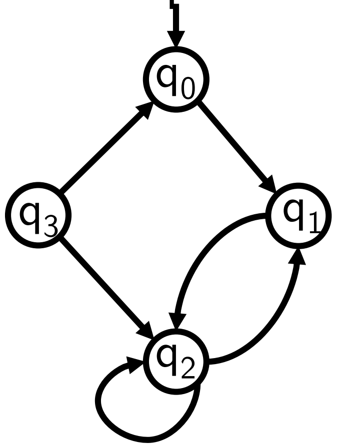
Computing CTL formula



Computing CTL formula

$$\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_1, q_2, q_3\}$$

- Example for EFϕ: Compute EF q_2



$$Q_0 = [\![q_2]\!] = \{q_2\}$$

$$Q_1 = \{q_2\} \cup \mathrm{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\}$$

$$Q_2 = \{q_1, q_2, q_3\} \cup \mathrm{Pre}(\{q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$$

$$Q_3 = \{q_0, q_1, q_2, q_3\} \cup \mathrm{Pre}(\{q_0, q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$$

$$[\![\mathrm{EF} q_2]\!] = Q_3 = \{q_0, q_1, q_2, q_3\}$$

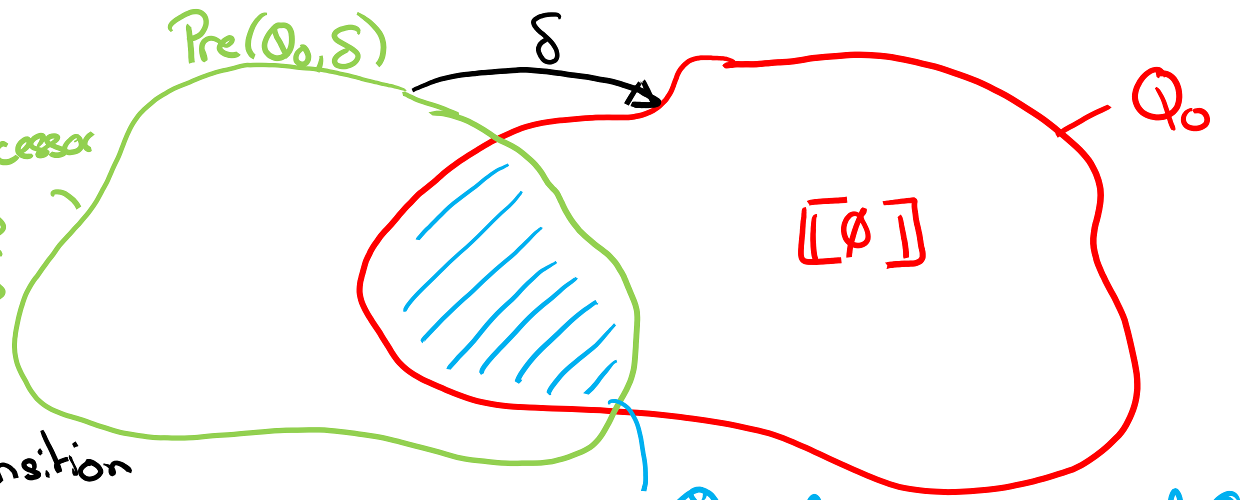
As $q_0 \in [\![\mathrm{EF} q_2]\!] = \{q_0, q_1, q_2, q_3\}$, the CTL formula EF q_2 is true.

Computing CTL formula

- EG ϕ: The idea here is to start with the set of initial states for which the formula ϕ is true. Then we cut this set with the set of predecessor states. For the resulting set of states we do the same, …., until we reach a fixed-point. The corresponding operations can be done using BDDs (as described before).

$$Q_0 = [\![\phi]\!]$$

$$Q_i = Q_{i-1} \cap \mathrm{Pre}(Q_{i-1}, \delta) \qquad \text{for all } i > 1 \text{ until a fixed-point is reached}$$



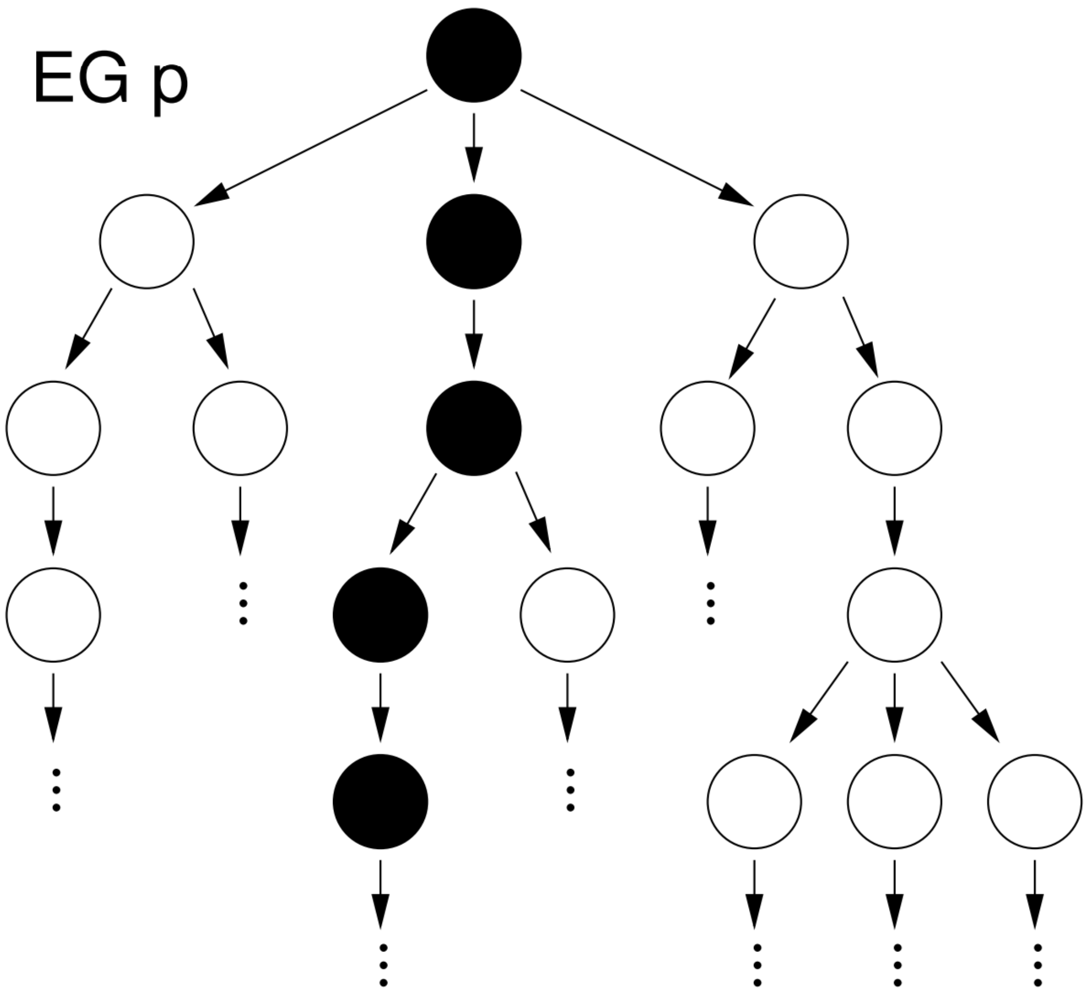
Set of precessor states of Q_0 by selection δ

Those from which I can enter Q_0 in one transition

Q_1: Set of precessor of Q_0 by δ in which ϕ holds true.

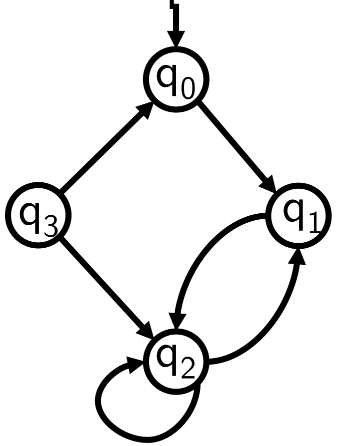
Computing CTL formula

EG p



Computing CTL formula

- Example for EG ϕ: Compute EG q_2



$$\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_1, q_2, q_3\}$$

$$Q_0 = [\![q_2]\!] = \{q_2\}$$

$$Q_1 = \{q_2\} \cap \mathrm{Pre}(\{q_2\}, \delta) = \{q_2\}$$

$$[\![\mathrm{EG} q_2]\!] = Q_2 = \{q_2\}$$

As $q_0 \notin [\![\mathrm{EG} q_2]\!] = \{q_2\}$, the CTL formula EG q_2 is not true.

Computing CTL formula

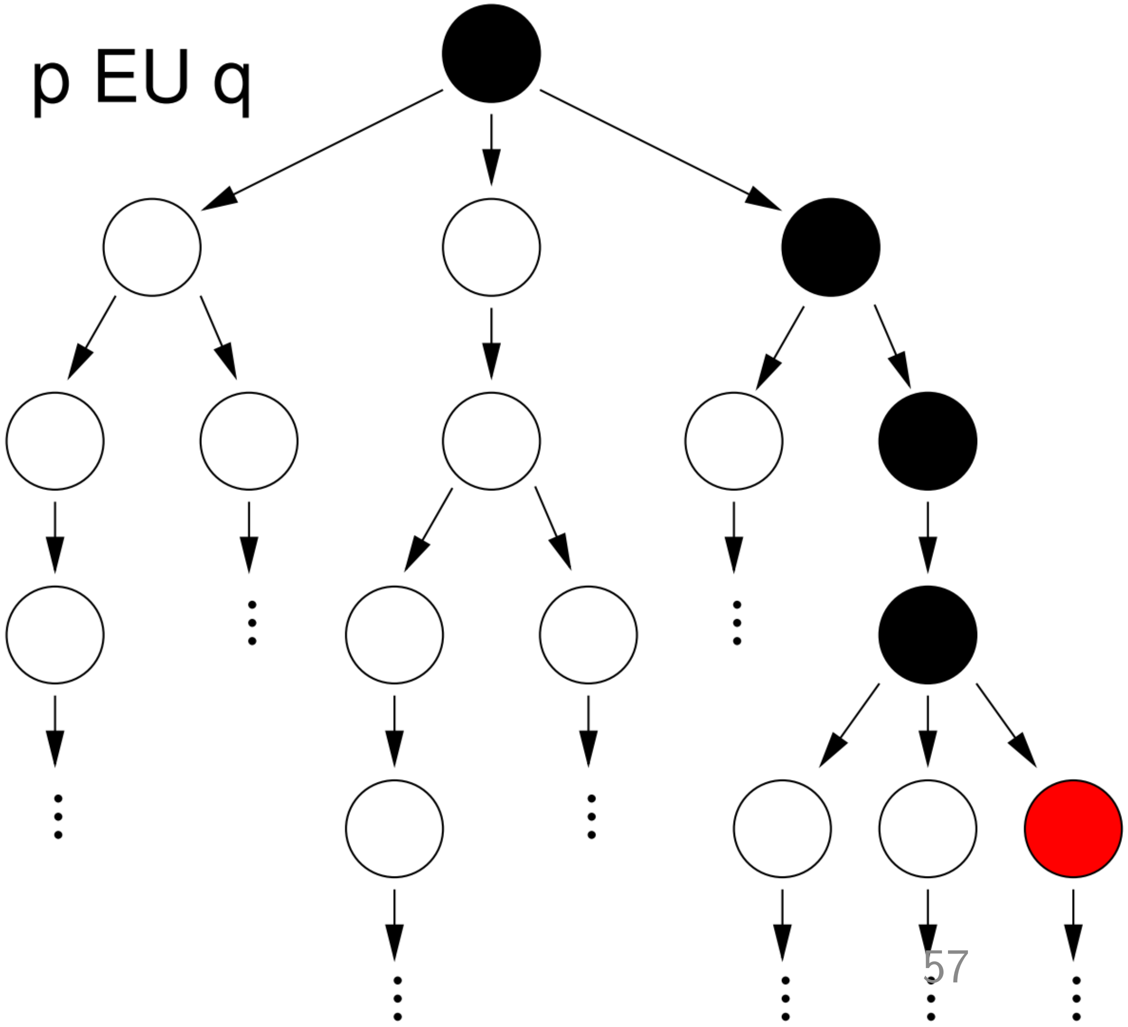
- $\phi_1 EU \phi_2$: The idea here is to start with the set of initial states for which the formula ϕ_2 is true. Then we add to this set the set of predecessor states for which the formula ϕ_1 is true. For the resulting set of states we do the same, …., until we reach a fixed-point. The corresponding operations can be done using BDDs (as described before).

$$Q_0 = [\![\phi_2]\!]$$

$$Q_i = Q_{i-1} \cup (\underline{\mathrm{Pre}(Q_{i-1}, \delta)} \cap \underline{[\![\phi_1]\!]}) \qquad \text{for all } i > 1 \text{ until a fixed-point is reached}$$

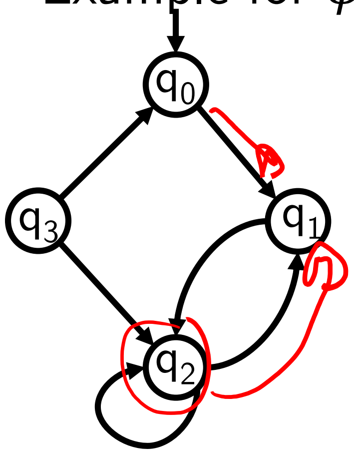
Like EF ϕ_2, the only difference is that on our path backwards, we always make sure that also ϕ_1 holds.

Computing CTL formula



Computing CTL formula

- Example for $\phi_1 EU \phi_2$: Compute $q_0\ EU\ q_1$



$$\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_0, q_2\}$$

$$Q_0 = [\![q_1]\!] = \{q_1\}$$

$$Q_1 = \{q_1\} \cup (\mathrm{Pre}(\{q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\}$$

$$Q_2 = \{q_0, q_1\} \cup (\mathrm{Pre}(\{q_0, q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\}$$

$$[\![q_0 \mathrm{EU} q_1]\!] = Q_2 = \{q_0, q_1\}$$

$$\{q_0, q_2, q_3\}$$

As $q_0 \in [\![q_0 \mathrm{EU} q_1]\!] = \{q_0, q_1\}$, the CTL formula q_0 EG q_1 is true.

So… what is model-checking exactly?

Model-checking is an algorithm which takes two inputs

- a DES model $\boldsymbol{M}$
- a formula $\boldsymbol{\phi}$

Finite automato
Petri nets
Kripke machine
…

CTL, LTL, …

It explores the state space of $\boldsymbol{M}$ such as to either

- prove that $\boldsymbol{M} \vDash \boldsymbol{\phi}$, or
- return a trace where the formula does not hold in $\boldsymbol{M}$.

So... what is model-checking exactly?

Model-checking is an algorithm which takes two inputs

- a DES model $\boldsymbol{M}$
- a formula $\boldsymbol{\phi}$

Finite automato
Petri nets
Kripke machine
...

CTL, LTL, ...

It explores the state space of $\boldsymbol{M}$ such as to either

- prove that $\boldsymbol{M} \vDash \boldsymbol{\phi}$, or
- return a trace where the formula does not hold in $\boldsymbol{M}$.

a counter-example

Extremely useful!

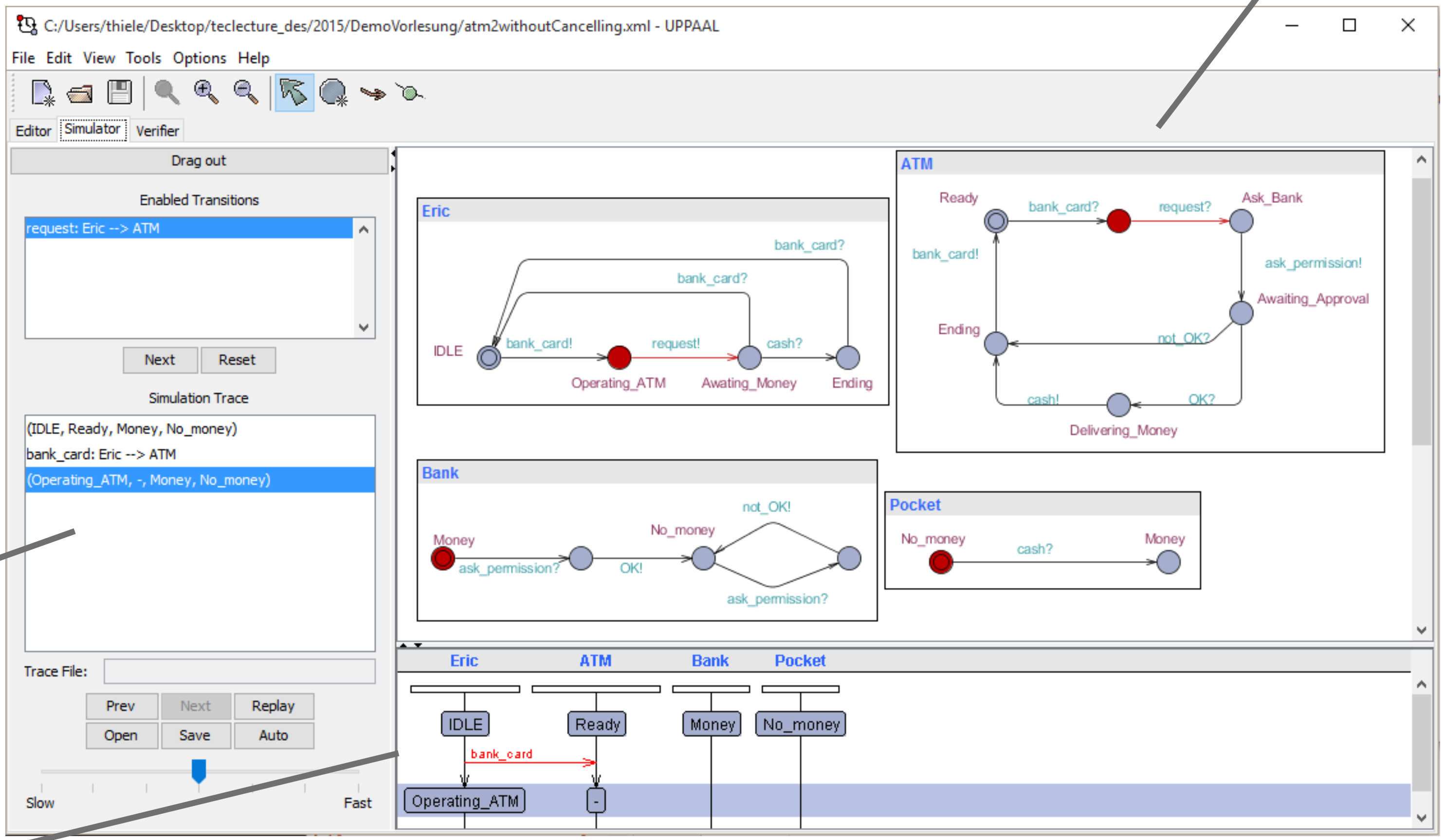
- Debugging the model
- Searching a specific execution sequence

Let's see how it works in practice…

UPPAAL model-checker

- free for academia
- (much) more general than what we show here
- can verify the timed behavior of communicating finite automata

communicating finite automata



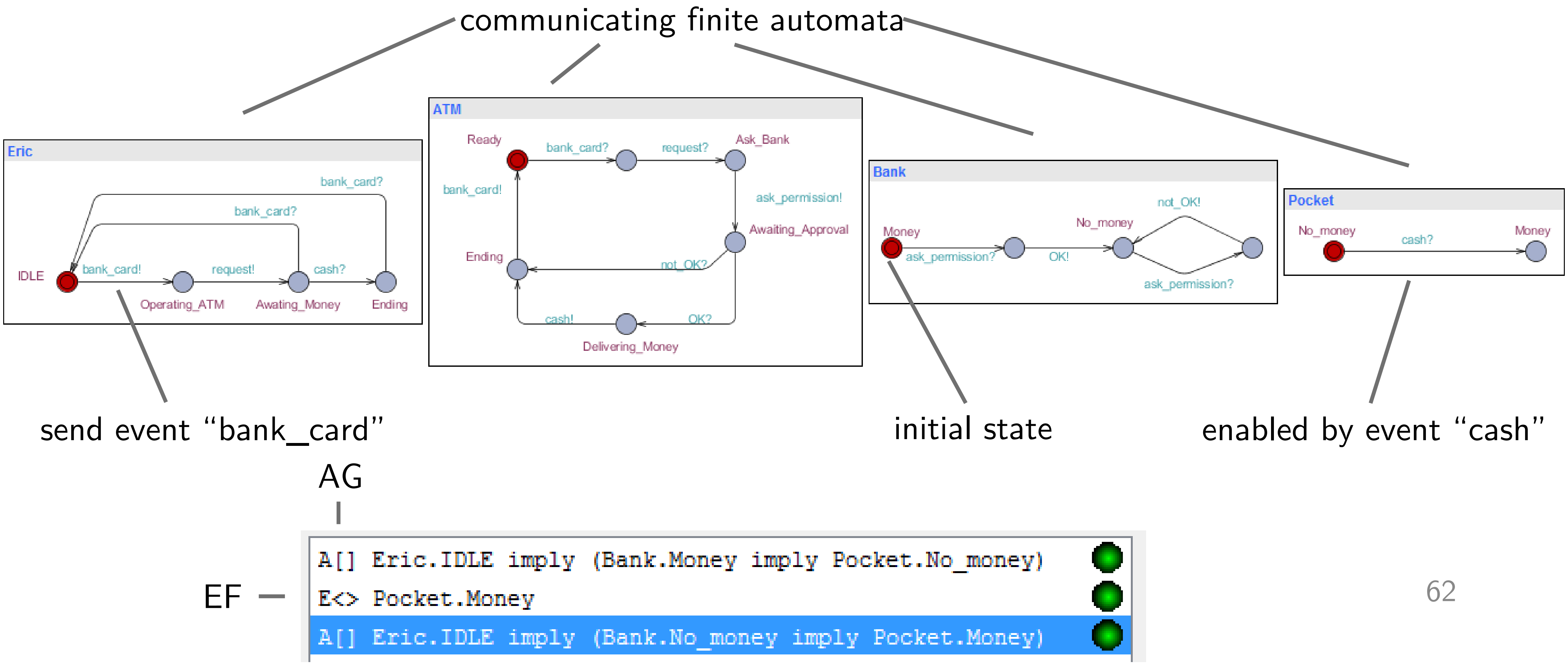
simulation trace

Example

Modeling and verification of a simple protocol for ATM-Money-Withdrawal

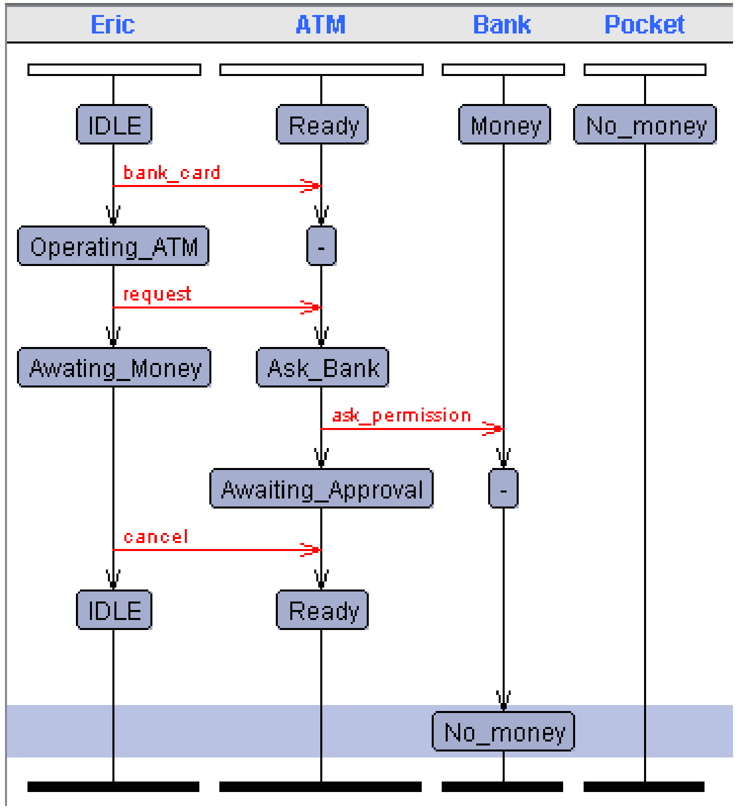
sequence diagram

Step 1. ATM without Cancel

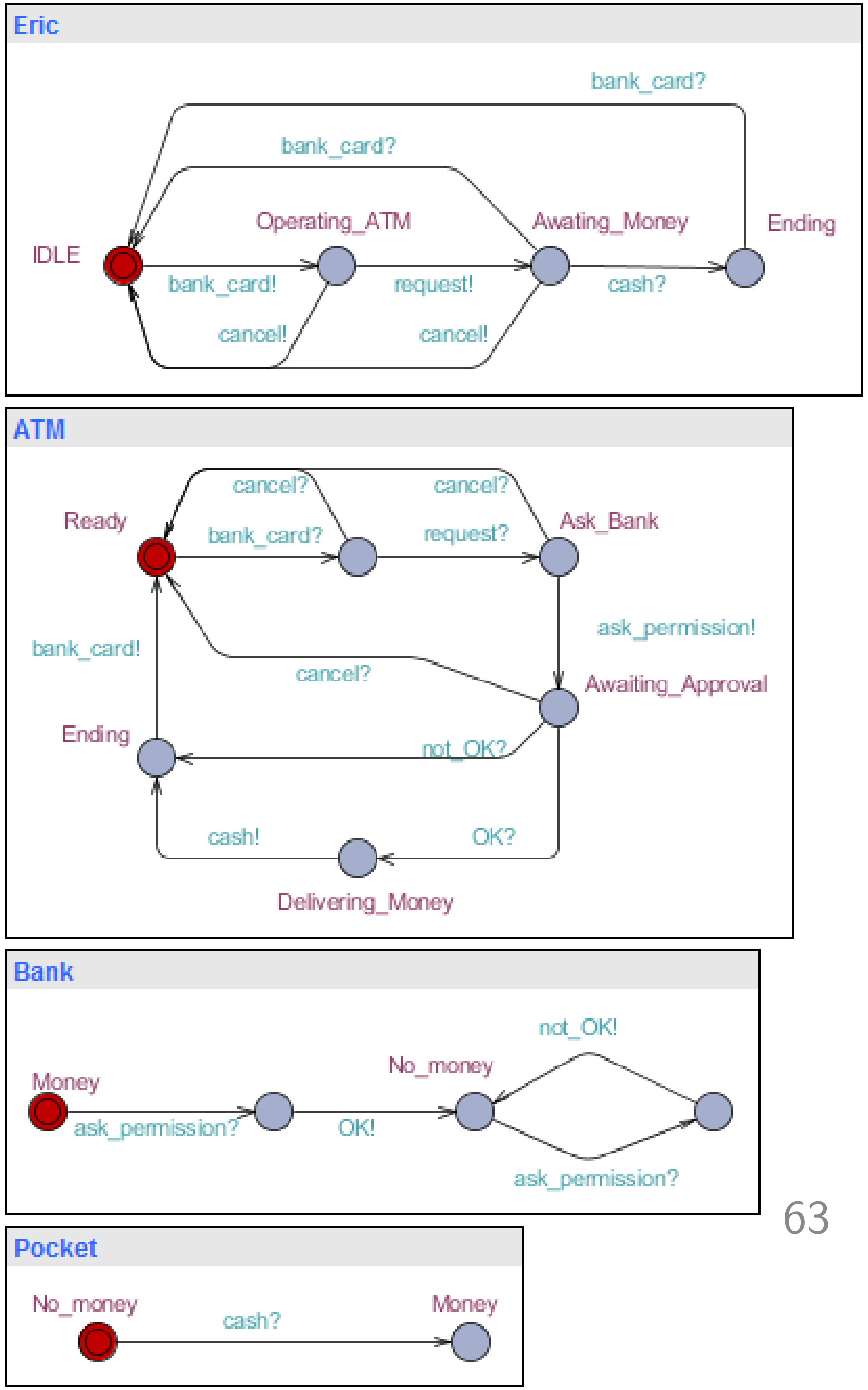


Step 2. ATM with Cancel

```
A[] Eric.IDLE imply (Bank.Money imply Pocket.No_money)
E<> Pocket.Money
A[] Eric.IDLE imply (Bank.No_money imply Pocket.Money)
```



counter example



Your turn to practice!
after the break

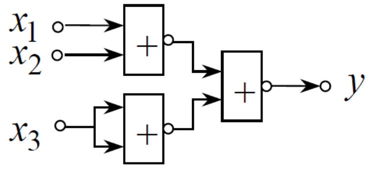
1. Familiarise yourself with CTL logic
and how to compute sets of states
satisfying a given formula

2. Convert a concrete problem into
a state reachability question
(adapted from state-of-the-art research!)

Conclusion and perspectives

Example

$y = (x_1 + x_2) \cdot x_3$



Comparison of specification and implementation

reference system → data structure
system under test → data structure
→ comparison

Proving properties

"The device can always be switched off."

property
system under test → data structure
→ fixed-point calculation

Conclusion and perspectives

Next week(s)

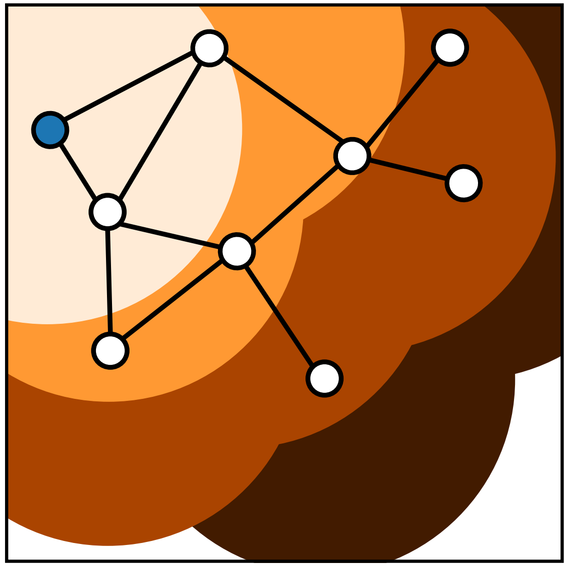
Petri Nets

- asynchronous DES model
- tailored model concurrent distributed systems
- capture an infinite state space with a finite model

a computer
a network

How they work?
How to use them for modeling systems?
How to verify them?

See you next week!
in Discrete Event Systems



Romain Jacob

www.romainjacob.net

ETH Zurich (D-ITET)

December 2, 2021

Most materials from Lothar Thiele