Computational Temporal Logic CTL

- a branching-time logic; it models time as a tree-like structure
- formulas can be used to reason about <u>many paths at once</u>

- **atoms** (such as p, q, r. . .) for facts like:
    - ‘printer Q5 is busy,’
    - ‘process 3259 is suspended,’

- **syntax**:

$$\phi \ ::= \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$$
$$\mid \boxed{\mathrm{AX}\ \phi \mid \mathrm{EX}\ \phi \mid \mathrm{AG}\ \phi \mid \mathrm{EG}\ \phi \mid \mathrm{AF}\ \phi \mid \mathrm{EF}\ \phi} \mid \boxed{\mathrm{A}[\phi\ \mathrm{U}\ \phi] \mid \mathrm{E}[\phi\ \mathrm{U}\ \phi]}$$

- **temporal connectives**:

AX p	along <u>A</u>ll paths, p is true in the ne<u>X</u>t state
EX p	there <u>E</u>xists one path along which p is true in the ne<u>X</u>t state
AG p	along <u>A</u>ll paths, p is true <u>G</u>lobally in the future
EG p	there <u>E</u>xists one path along which p is true <u>G</u>lobally in the future
AF p	along <u>A</u>ll paths, p is true <u>F</u>inally, sometime in the future
EF p	there <u>E</u>xists one path along which p is true <u>F</u>inally, sometime in the future
A[p U q]	along <u>A</u>ll paths, p is true <u>U</u>ntil q is true
E[p U q]	there <u>E</u>xists one path along which p is true <u>U</u>ntil q is true

Computational Temporal Logic

binding priorities:
¬, AX, EX, AG, EG, and AF, EF bind most tightly,
next come $\wedge$ and $\vee$,
and then $\to$, AU and EU

Example WFFs

$\mathrm{AG}\,(q \to \mathrm{EG}\,r)$ not the same as $\mathrm{AG}\,q \to \mathrm{EG}\,r$

$\mathrm{EF}\,\mathrm{E}[r\ \mathrm{U}\ q]$

$\mathrm{A}[p\ \mathrm{U}\ \mathrm{EF}\,r]$

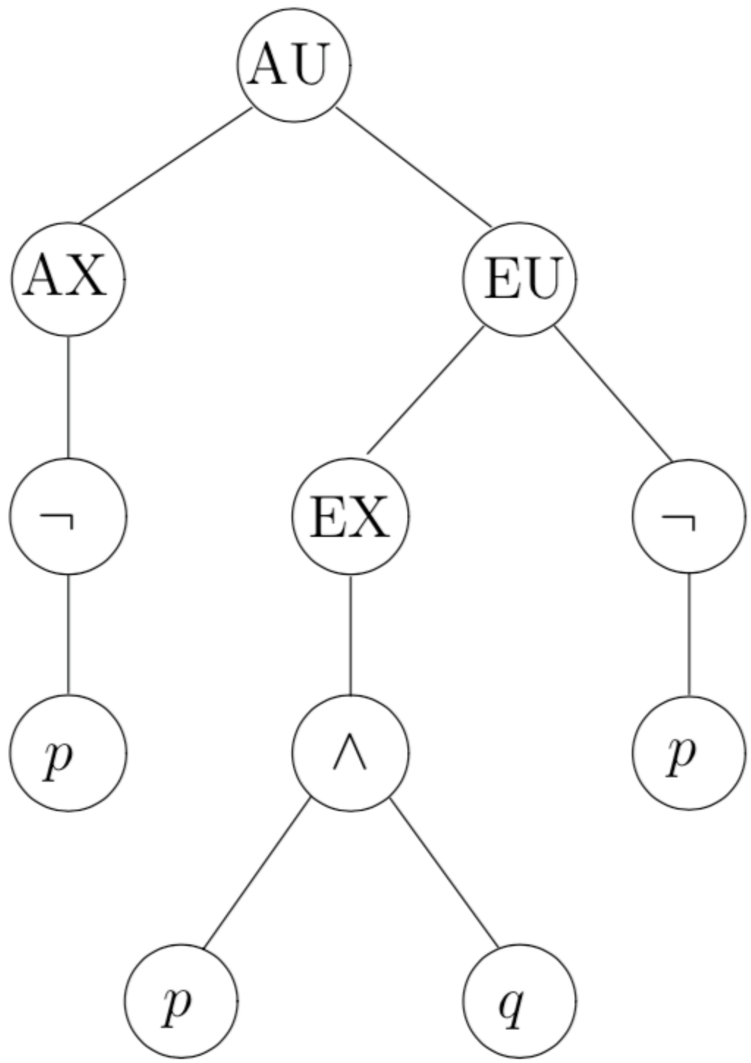
$\mathrm{EF}\,\mathrm{EG}\,p \to \mathrm{AF}\,r$ not the same as $\mathrm{EF}\,(\mathrm{EG}\,p \to \mathrm{AF}\,r)$ or $\mathrm{EF}\,\mathrm{EG}\,(p \to \mathrm{AF}\,r)$

$\mathrm{A}[p\ \mathrm{U}\ \mathrm{A}[q\ \mathrm{U}\ r]]$

Not WFFs

$\mathrm{EF}\,\mathrm{G}r$	since G can occur only when paired with an A or an E
$\mathrm{A}\neg\mathrm{G}\neg p$	since G can occur only when paired with an A or an E
$\mathrm{F}[r\ \mathrm{U}\ q]$	since U can occur only when paired with an A or an E
$\mathrm{EF}\,(r\ \mathrm{U}\ q)$	since U can occur only when paired with an A or an E

The **parse tree** for $\mathbf{A}[\ \mathrm{AX}\ \neg p\ \ \mathbf{U}\ \ \mathrm{E}[\mathrm{EX}\ (p \wedge q)\ \mathrm{U}\ \neg p]\ ]$



A **subformula** of a CTL formula ϕ is any formula ψ whose parse tree is a subtree of ϕ's parse tree.

CTL Semantics

CTL formulas are interpreted over models called transition systems.

Let $\mathcal{M} = (S, \rightarrow, L)$ be such a model, $s \in S$ and ϕ a CTL formula.

The definition of whether $\mathcal{M}, s \vDash \phi$ holds is recursive on the structure of ϕ, and can be roughly understood as follows:

- The idea of temporal logic is that a formula is not statically true or false in a model, as it is in propositional and predicate logic.
- Instead, the models of temporal logic contain several states and a formula can be true in some states and false in others.
- Thus, the static notion of truth is replaced by a dynamic one, in which the formulas may change their truth values as the system evolves from state to state.

CTL Semantics

The systems we analyze and verify with CTL are modeled as **transition systems**.

Definition 3.15

A transition system $\mathcal{M} = (S, \rightarrow, L)$ is:

1. a set of **states** S,
2. a **transition relation** $\rightarrow$, (a binary relation on S)
3. a **labeling function** $L\colon S \rightarrow \mathcal{P}(atoms)$

such that every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$

L(s) contains all atoms which are true in state s.

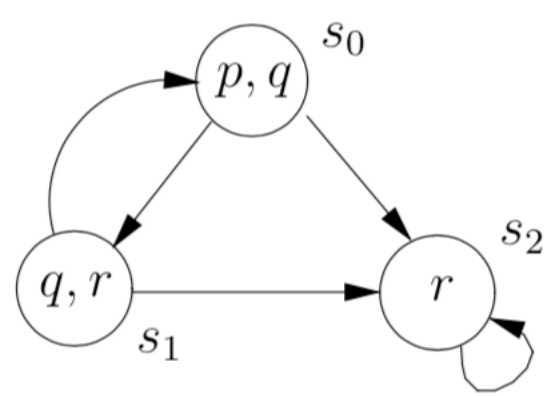
the power set of atoms, for example, the power set of {p,q} is {∅,{p},{q},{p,q}}

Example

S: $\{s_0, s_1, s_2\}$

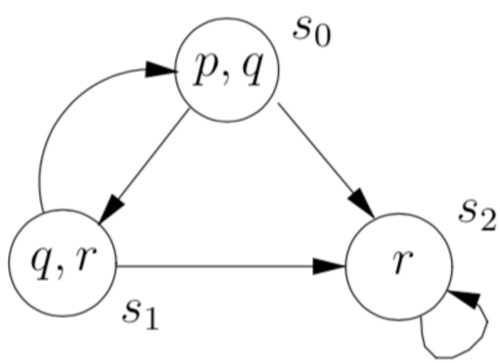
R: $s_0 \rightarrow s_1$, $s_0 \rightarrow s_2$, $s_1 \rightarrow s_0$, $s_1 \rightarrow s_2$ and $s_2 \rightarrow s_2$
$R(s_0, s_1), R(s_0, s_2), R(s_1, s_0), R(s_1, s_2), R(s_2, s_2)$

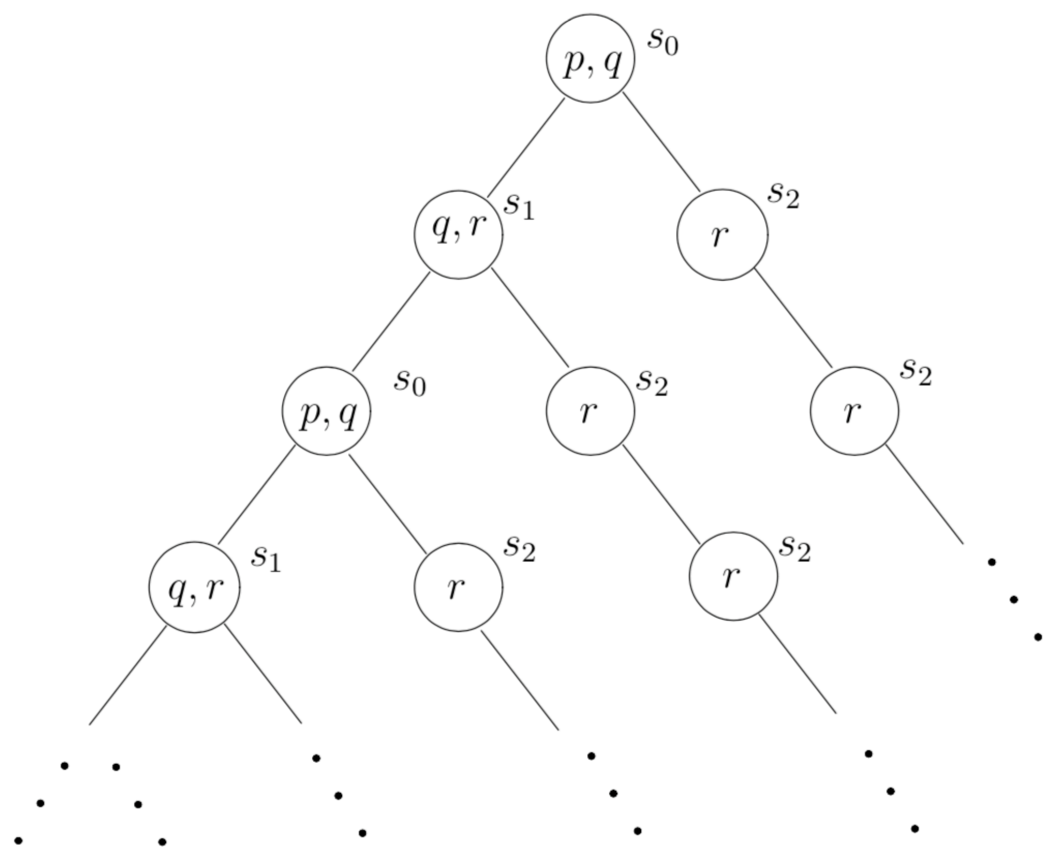
L: $L(s_0) = \{p, q\}$, $L(s_1) = \{q, r\}$ and $L(s_2) = \{r\}$



CTL Semantics

It is useful to visualize <u>all possible execution paths from a given state s</u> by unwinding the transition system to obtain an infinite computation tree.





Definition 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:

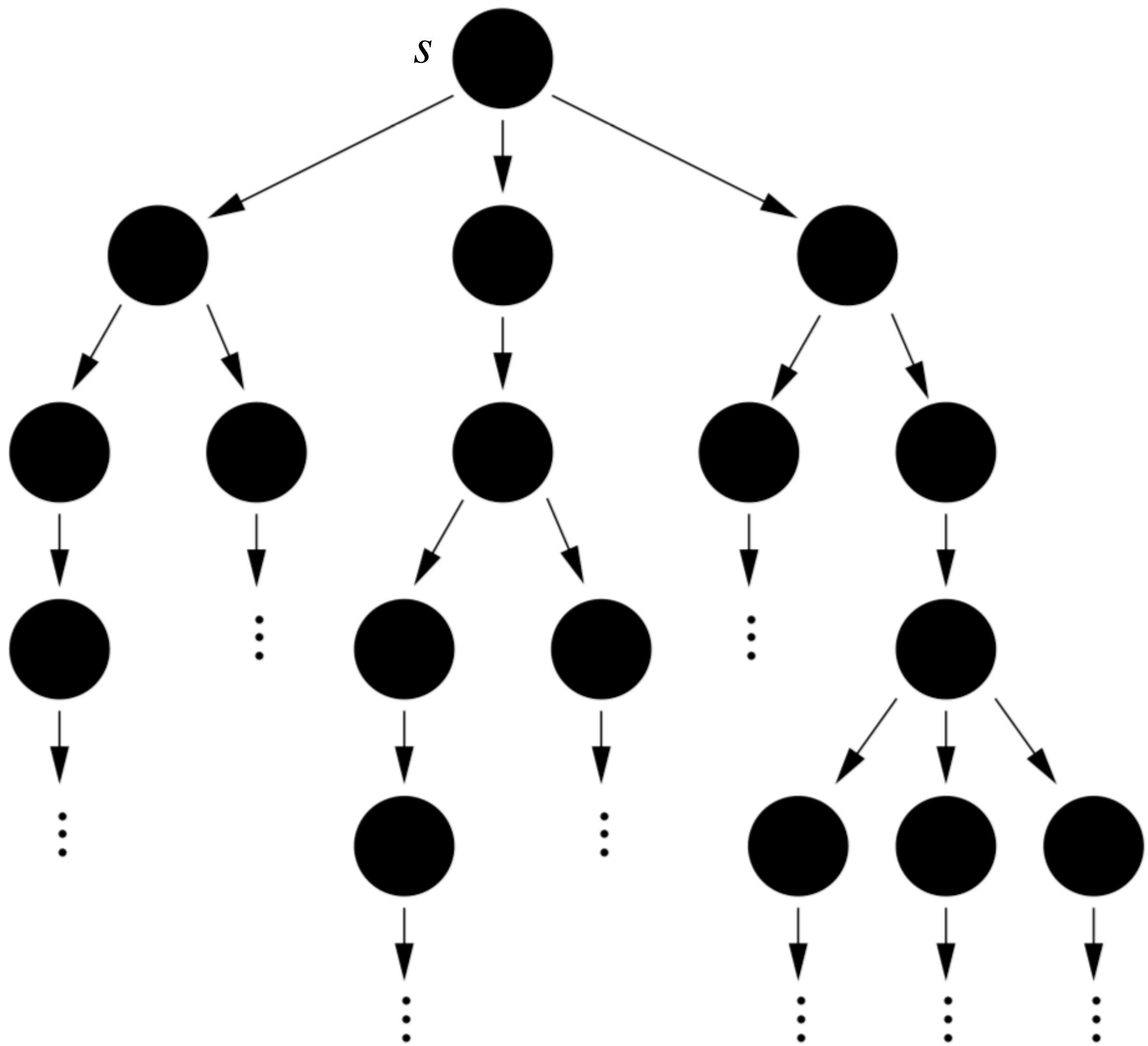
1. $\mathcal{M}, s \vDash \top$ and $\mathcal{M}, s \nvDash \bot$
2. $\mathcal{M}, s \vDash p$ iff $p \in L(s)$
3. $\mathcal{M}, s \vDash \neg\phi$ iff $\mathcal{M}, s \nvDash \phi$
4. $\mathcal{M}, s \vDash \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s \vDash \phi_1$ and $\mathcal{M}, s \vDash \phi_2$
5. $\mathcal{M}, s \vDash \phi_1 \vee \phi_2$ iff $\mathcal{M}, s \vDash \phi_1$ or $\mathcal{M}, s \vDash \phi_2$
6. $\mathcal{M}, s \vDash \phi_1 \rightarrow \phi_2$ iff $\mathcal{M}, s \nvDash \phi_1$ or $\mathcal{M}, s \vDash \phi_2$.
7. $\mathcal{M}, s \vDash \mathrm{AX}\,\phi$ iff for all s_1 such that $s \rightarrow s_1$ we have $\mathcal{M}, s_1 \vDash \phi$. Thus, AX says: 'in every next state.'
8. $\mathcal{M}, s \vDash \mathrm{EX}\,\phi$ iff for some s_1 such that $s \rightarrow s_1$ we have $\mathcal{M}, s_1 \vDash \phi$. Thus, EX says: 'in some next state.' E is dual to A – in exactly the same way that $\exists$ is dual to $\forall$ in predicate logic.
9. $\mathcal{M}, s \vDash \mathrm{AG}\,\phi$ holds 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 $\mathcal{M}, s_i \vDash \phi$. Mnemonically: for All computation paths beginning in s the property ϕ holds Globally. Note that 'along the path' includes the path's initial state s.
10. $\mathcal{M}, s \vDash \mathrm{EG}\,\phi$ holds iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where s_1 equals s, and for all s_i along the path, we have $\mathcal{M}, s_i \vDash \phi$. Mnemonically: there Exists a path beginning in s such that ϕ holds Globally along the path.

11. $\mathcal{M}, s \vDash \mathrm{AF}\,\phi$ holds iff for all paths $s_1 \to s_2 \to \ldots$, where s_1 equals s, there is some s_i such that $\mathcal{M}, s_i \vDash \phi$. Mnemonically: for All computation paths beginning in s there will be some Future state where ϕ holds.
12. $\mathcal{M}, s \vDash \mathrm{EF}\,\phi$ holds iff there is a path $s_1 \to s_2 \to s_3 \to \ldots$, where s_1 equals s, and for some s_i along the path, we have $\mathcal{M}, s_i \vDash \phi$. Mnemonically: there Exists a computation path beginning in s such that ϕ holds in some Future state;
13. $\mathcal{M}, s \vDash \mathrm{A}[\phi_1 \mathrm{~U~} \phi_2]$ holds iff for all paths $s_1 \to s_2 \to s_3 \to \ldots$, where s_1 equals s, that path satisfies $\phi_1 \mathrm{~U~} \phi_2$, i.e., there is some s_i along the path, such that $\mathcal{M}, s_i \vDash \phi_2$, and, for each $j < i$, we have $\mathcal{M}, s_j \vDash \phi_1$. Mnemonically: All computation paths beginning in s satisfy that ϕ_1 Until ϕ_2 holds on it.
14. $\mathcal{M}, s \vDash \mathrm{E}[\phi_1 \mathrm{~U~} \phi_2]$ holds iff there is a path $s_1 \to s_2 \to s_3 \to \ldots$, where s_1 equals s, and that path satisfies $\phi_1 \mathrm{~U~} \phi_2$ as specified in 13. Mnemonically: there Exists a computation path beginning in s such that ϕ_1 Until ϕ_2 holds on it.

If p is true everywhere there is a filled circle

then $\mathcal{M}, s \vDash \phi$

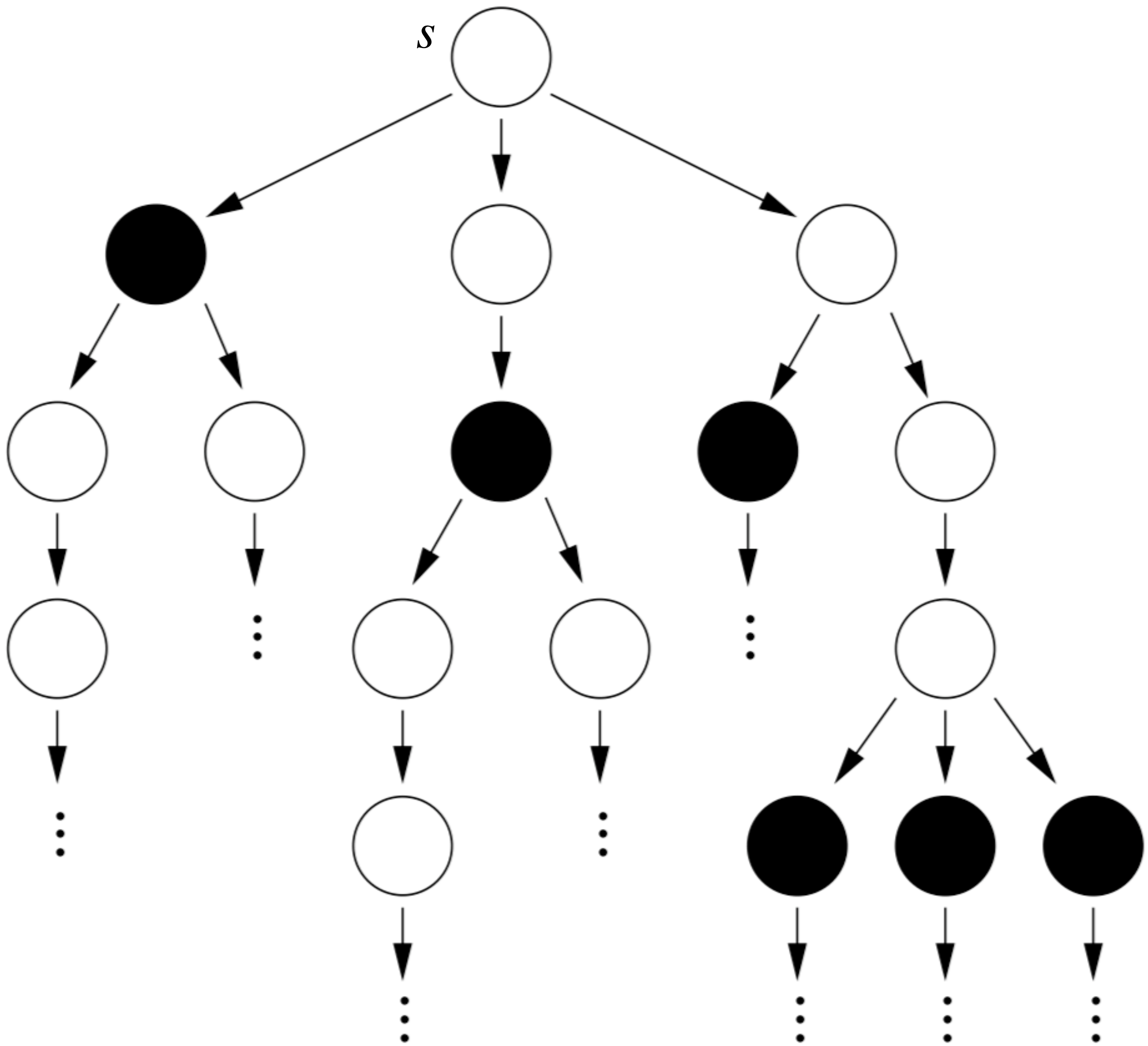
if ϕ is AG p



If p is true everywhere there is a filled circle

then $\mathcal{M},s \vDash \phi$

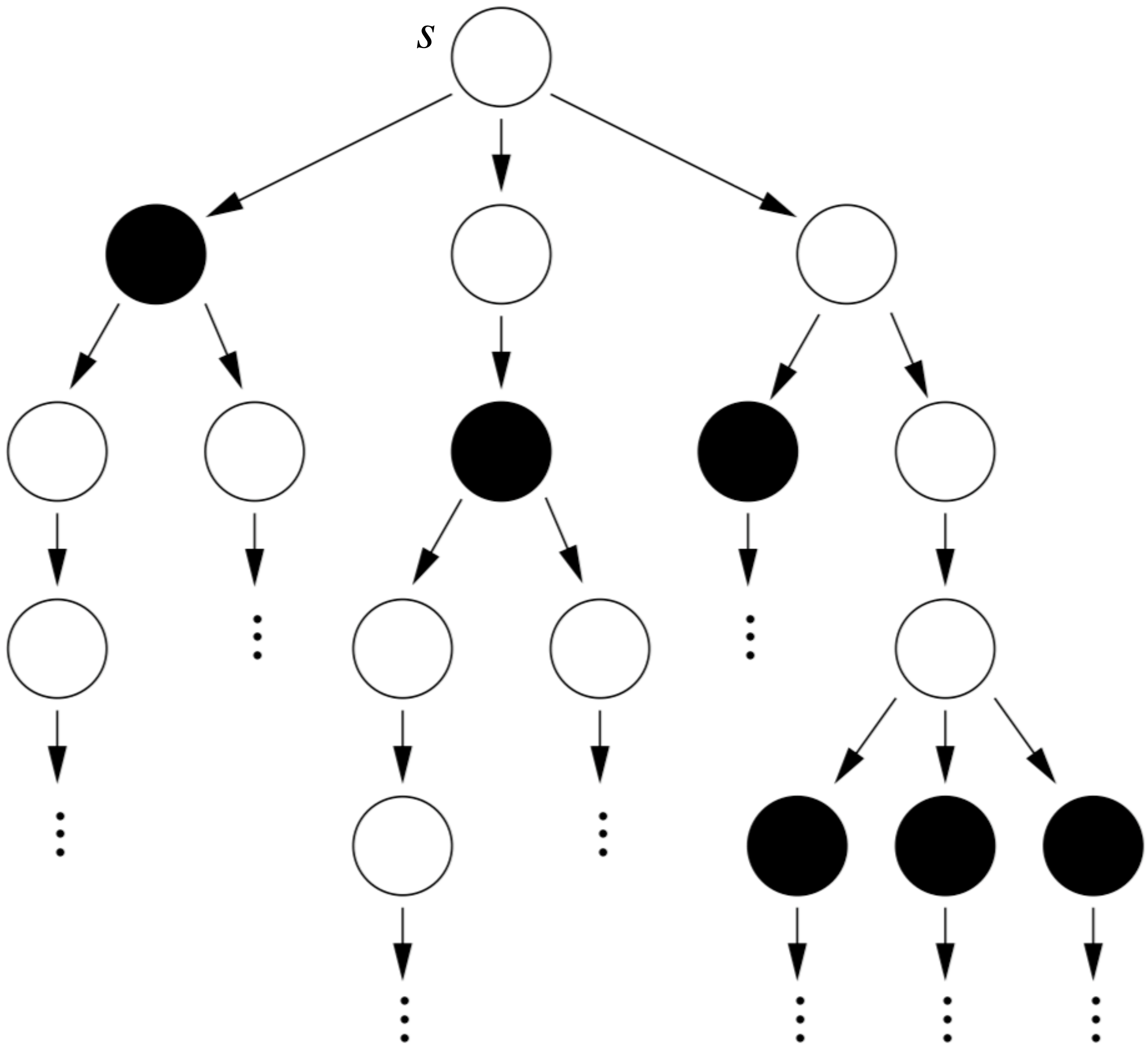
if ϕ is...



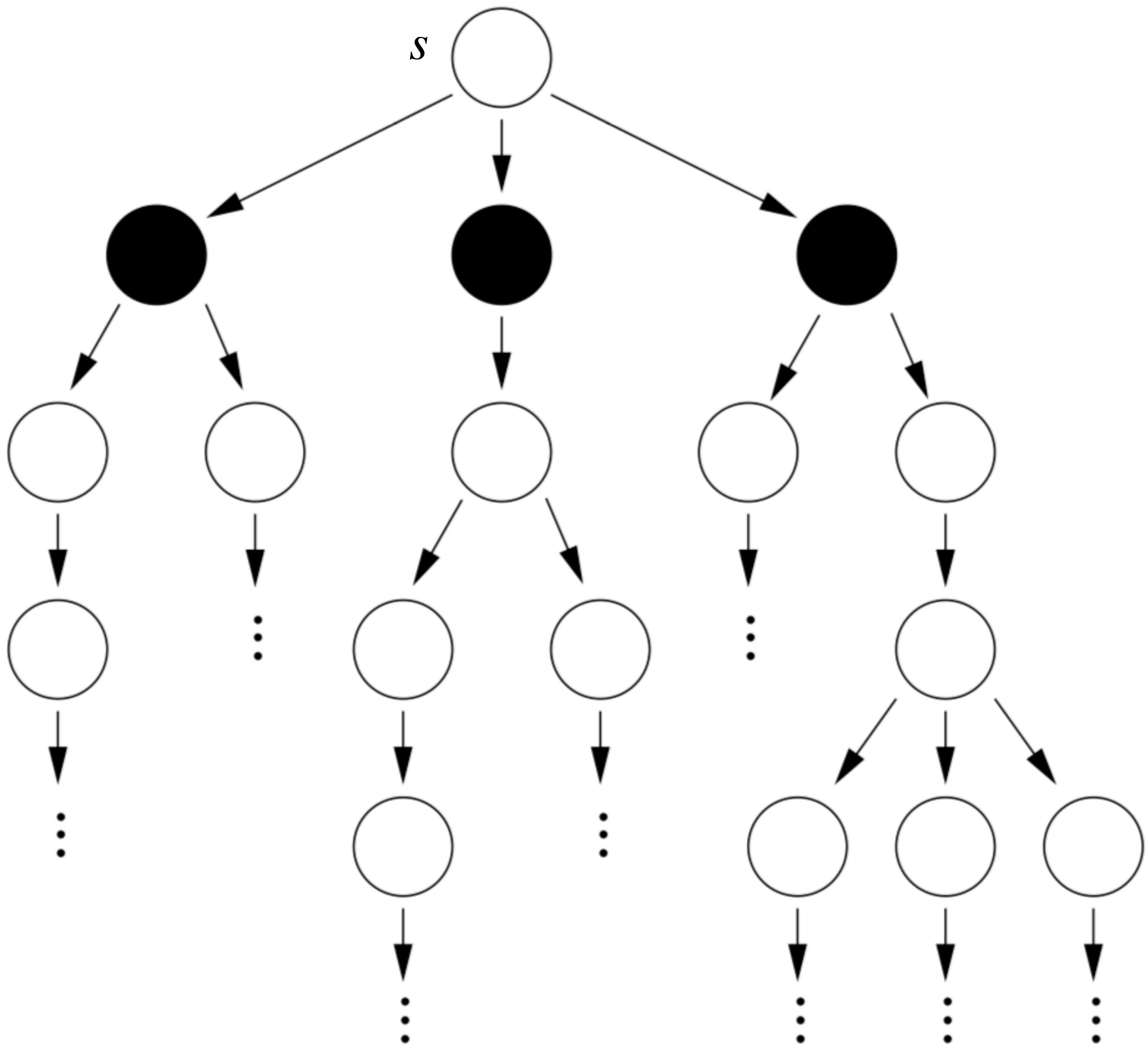
If p is true everywhere there is a filled circle

then $\mathcal{M},s \vDash \phi$

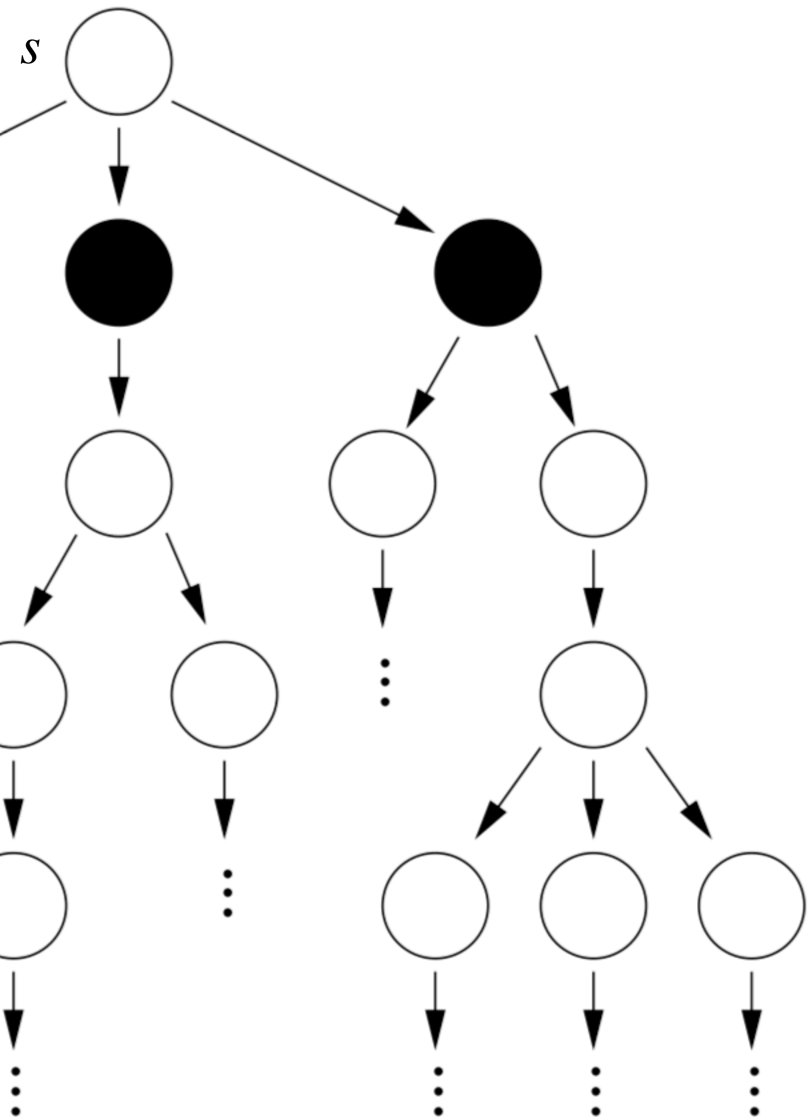
if ϕ is AF p



If p is true everywhere there is a filled circle
then $\mathcal{M},s \vDash \phi$
if ϕ is...



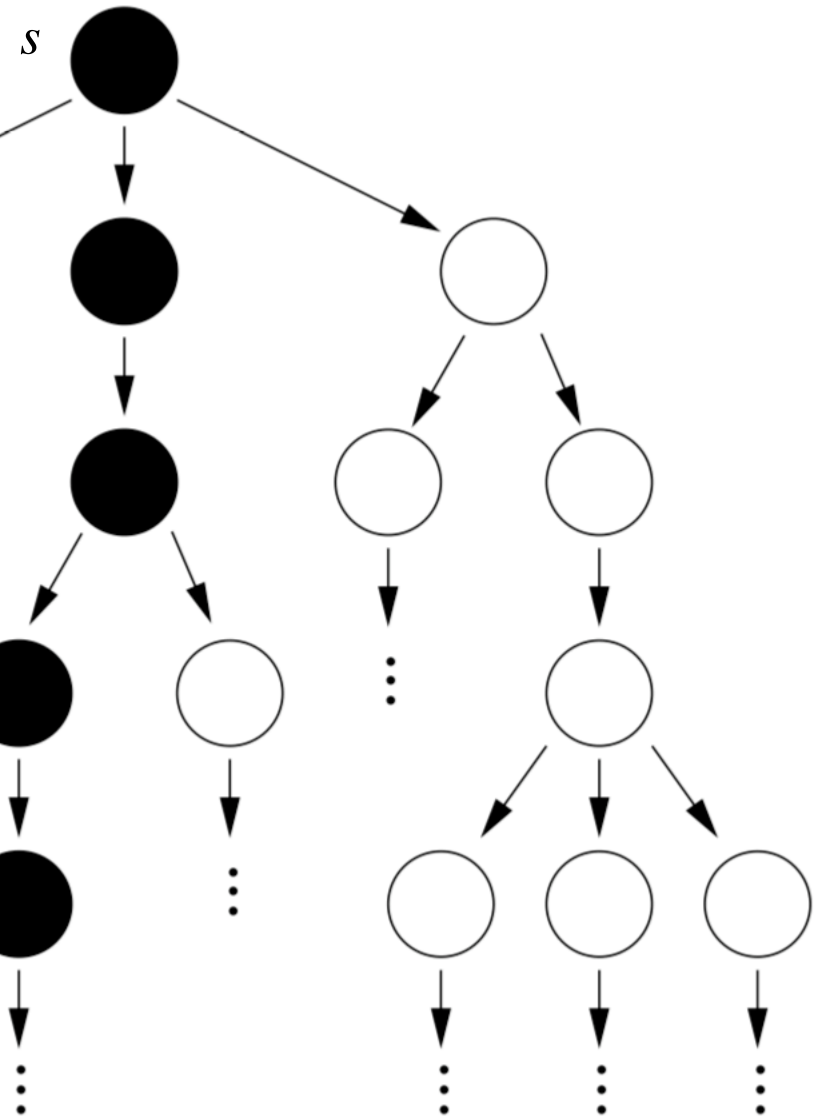
If p is true everywhere there is a filled circle
then $\mathcal{M},s \models \phi$
if ϕ is $\mathrm{AX}\ p$



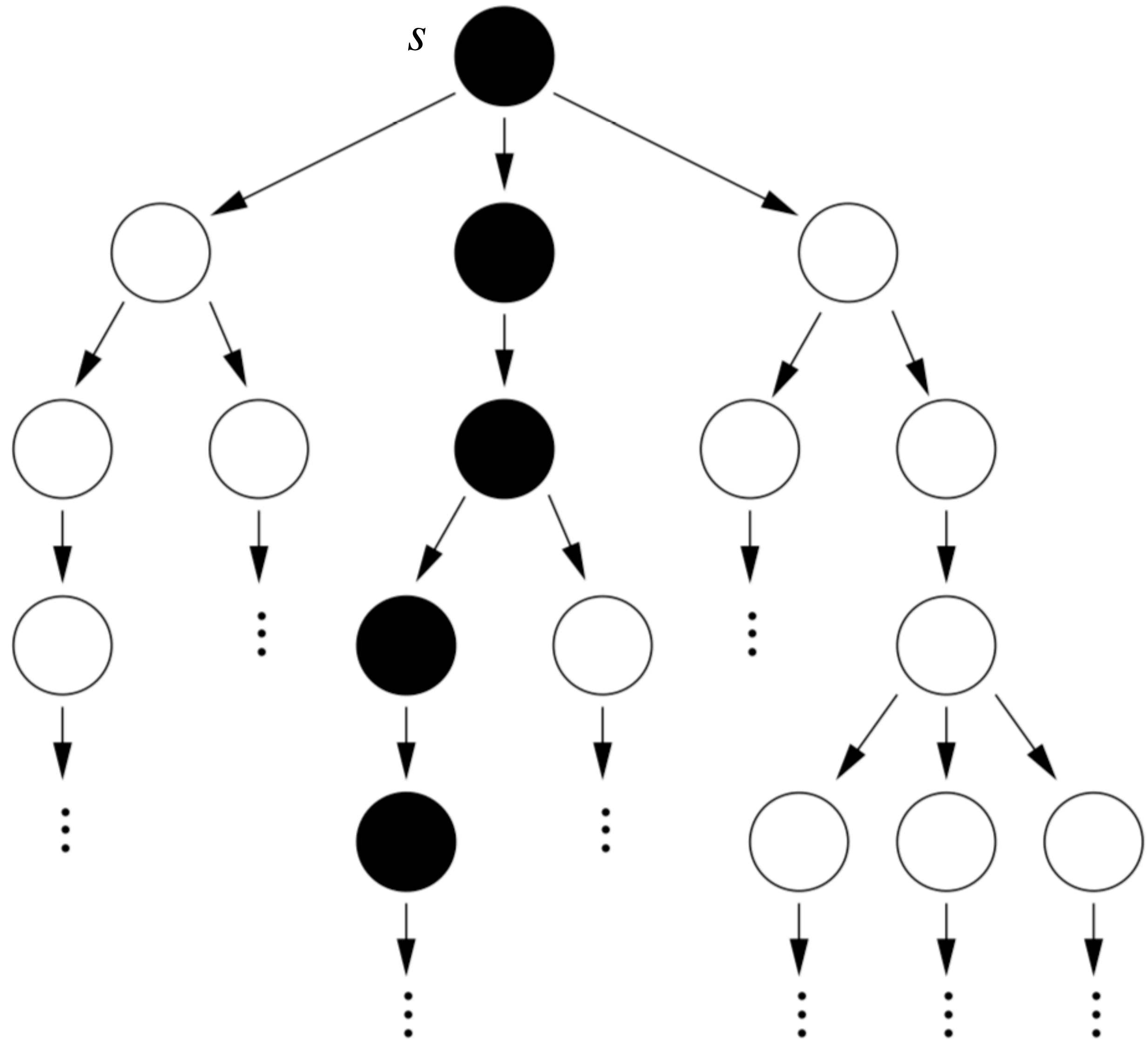
If p is true everywhere there is a filled circle

then $\mathcal{M},s \vDash \phi$

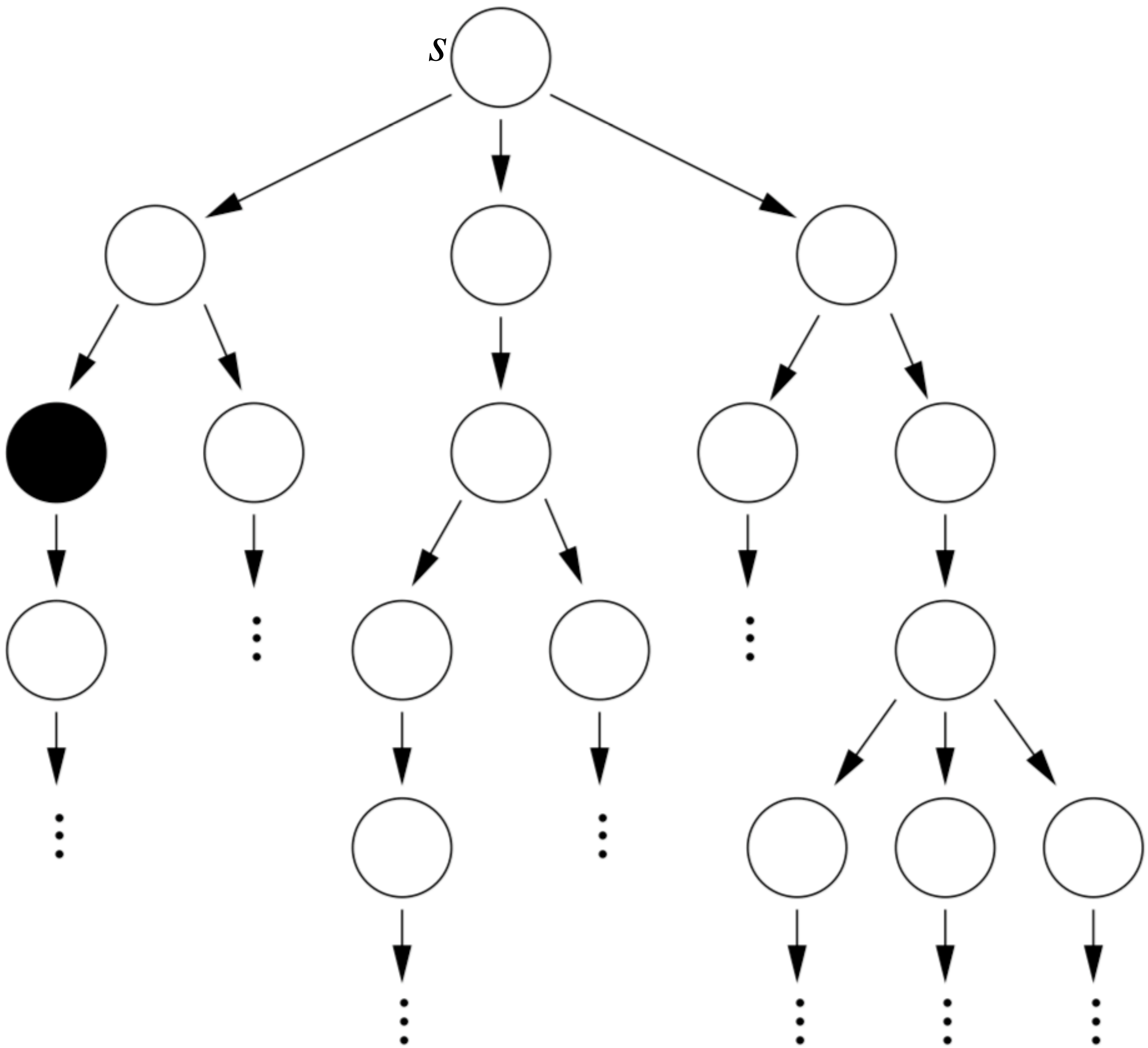
if ϕ is...



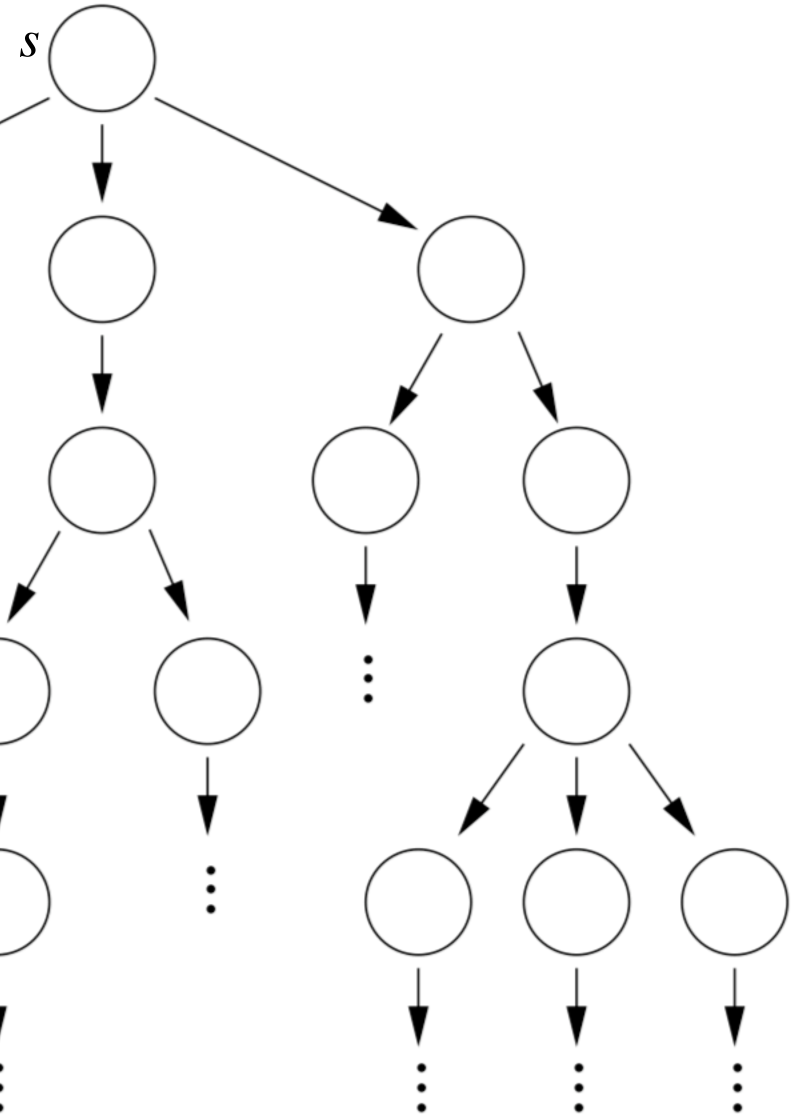
If p is true everywhere there is a filled circle
then $\mathcal{M},s \vDash \phi$
if ϕ is $\mathrm{EG}\ p$



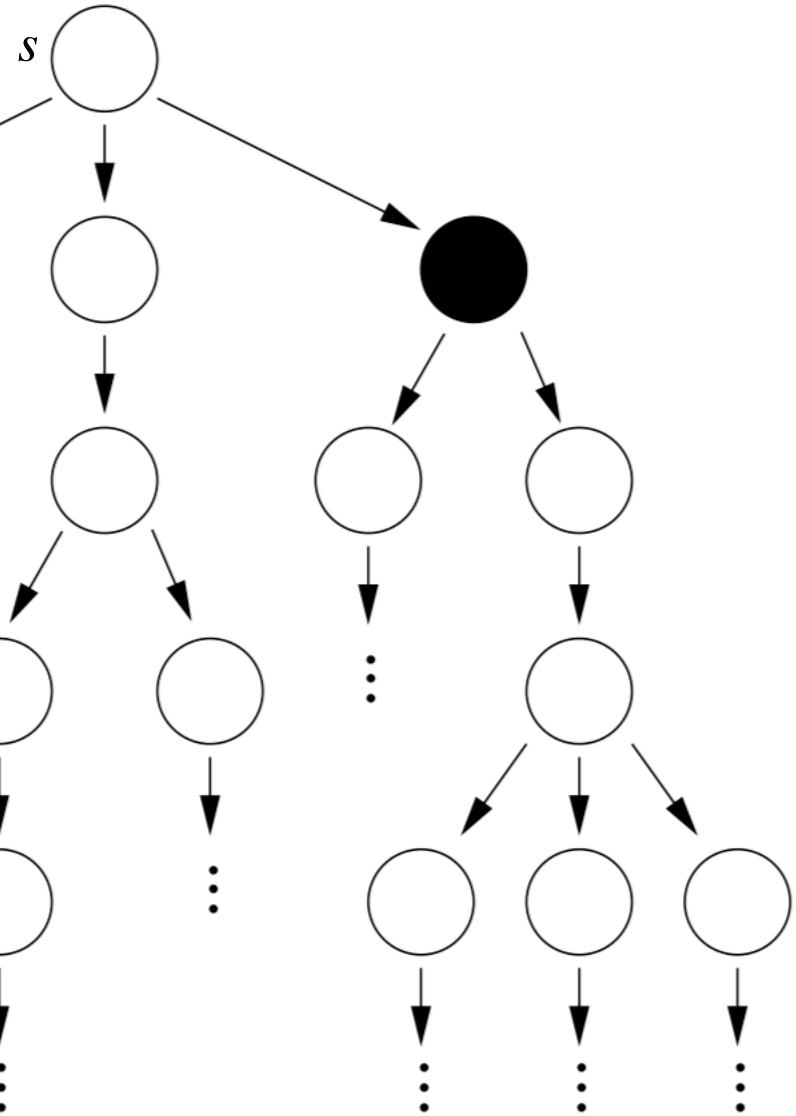
If p is true everywhere there is a filled circle
then $\mathcal{M}, s \models \phi$
if ϕ is...



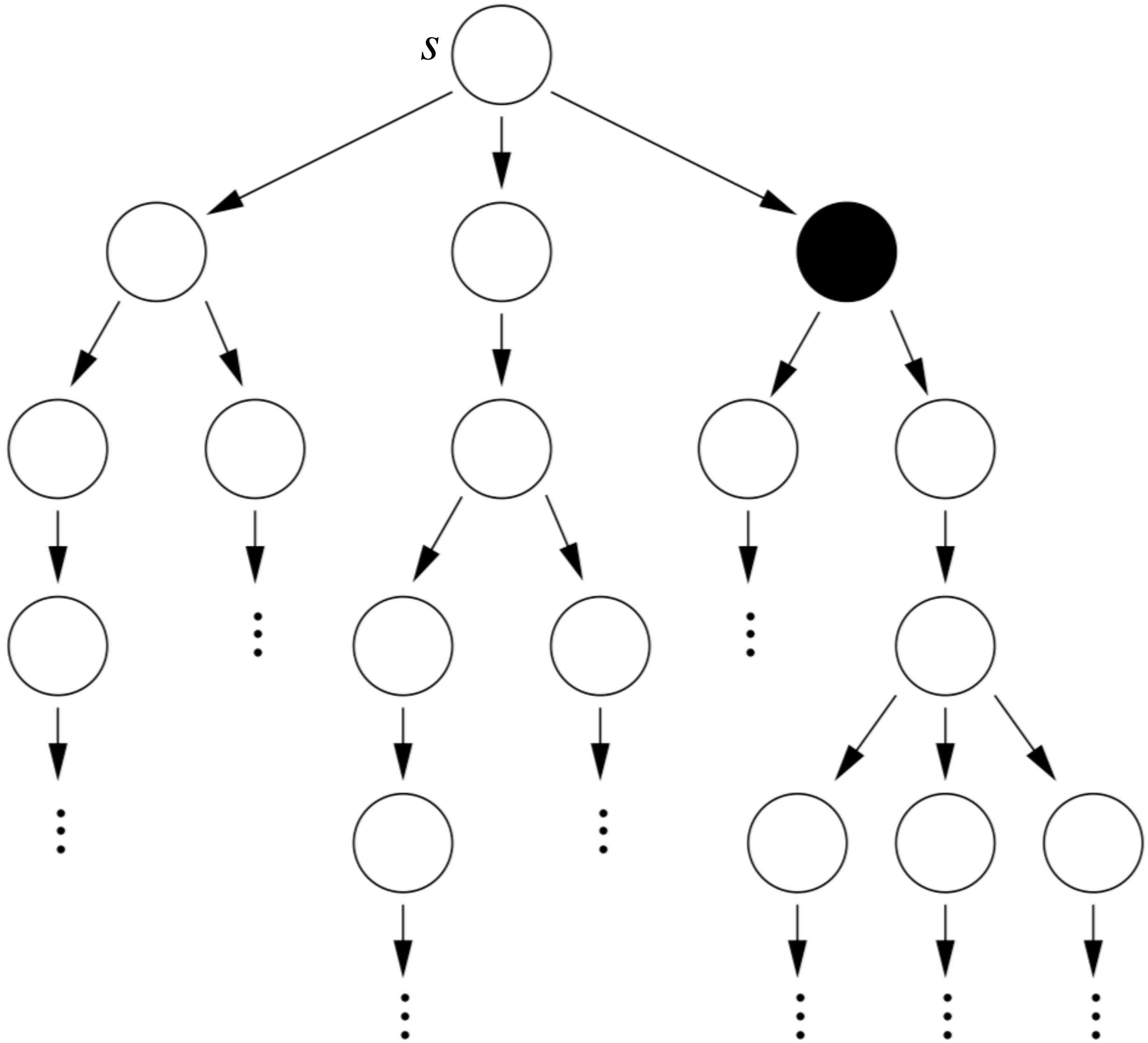
If p is true everywhere there is a filled circle
then $\mathcal{M}, s \models \phi$
if ϕ is $\mathrm{EF}\ p$



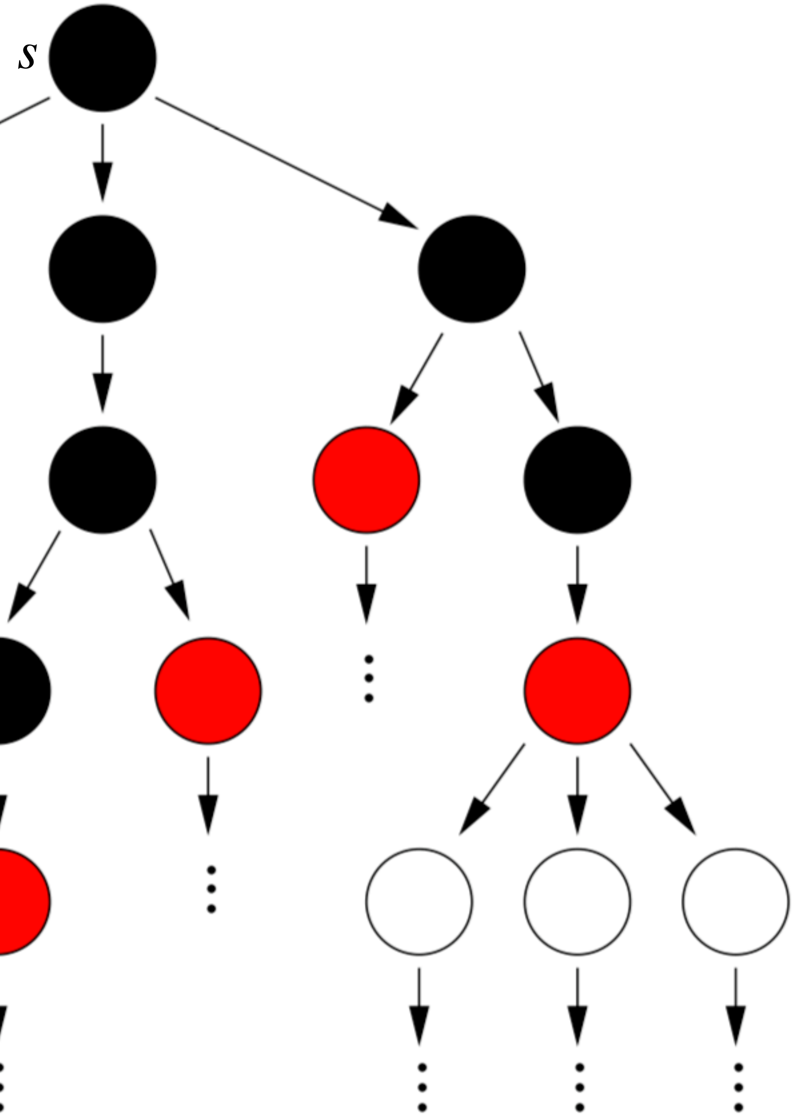
If p is true everywhere there is a filled circle
then $\mathcal{M}, s \models \phi$
if ϕ is...



If p is true everywhere there is a filled circle
then $\mathcal{M},s \vDash \phi$
if ϕ is $\mathrm{EX}\ p$



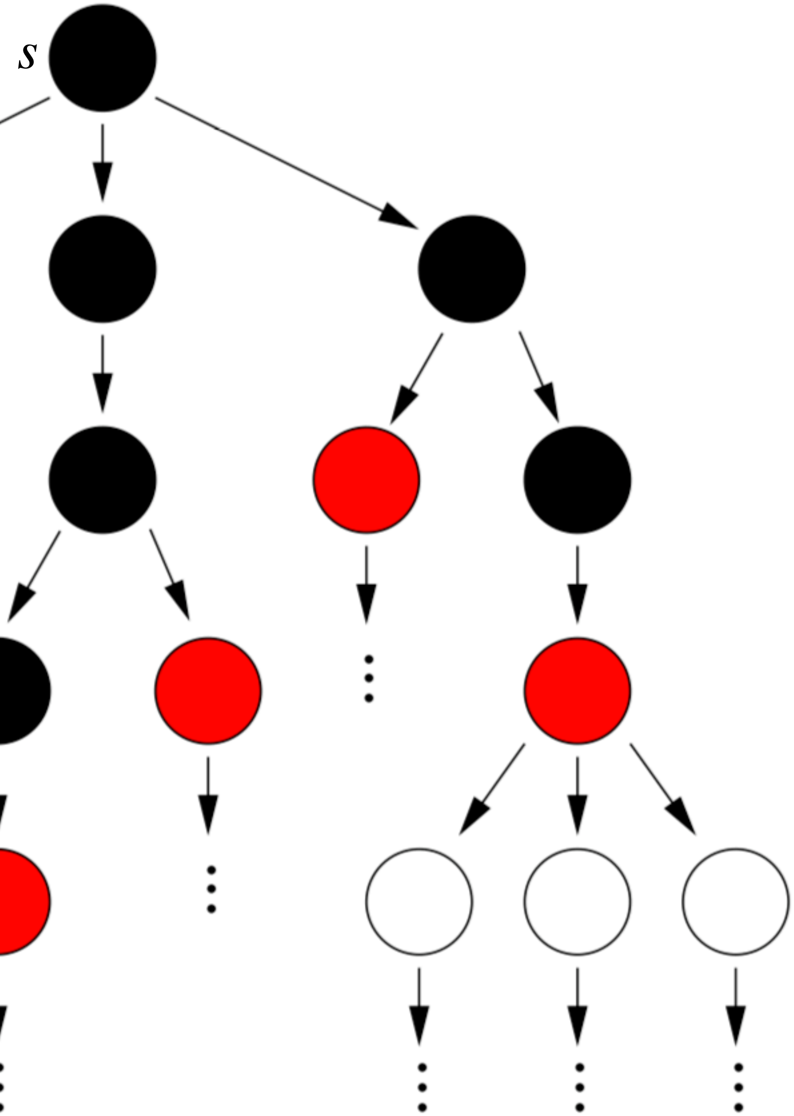
If p is true everywhere there is a black-filled circle and q is true in every red-filled circle
then $\mathcal{M}, s \vDash \phi$
if ϕ is...



If p is true everywhere there is a black-filled circle and q is true in every red-filled circle

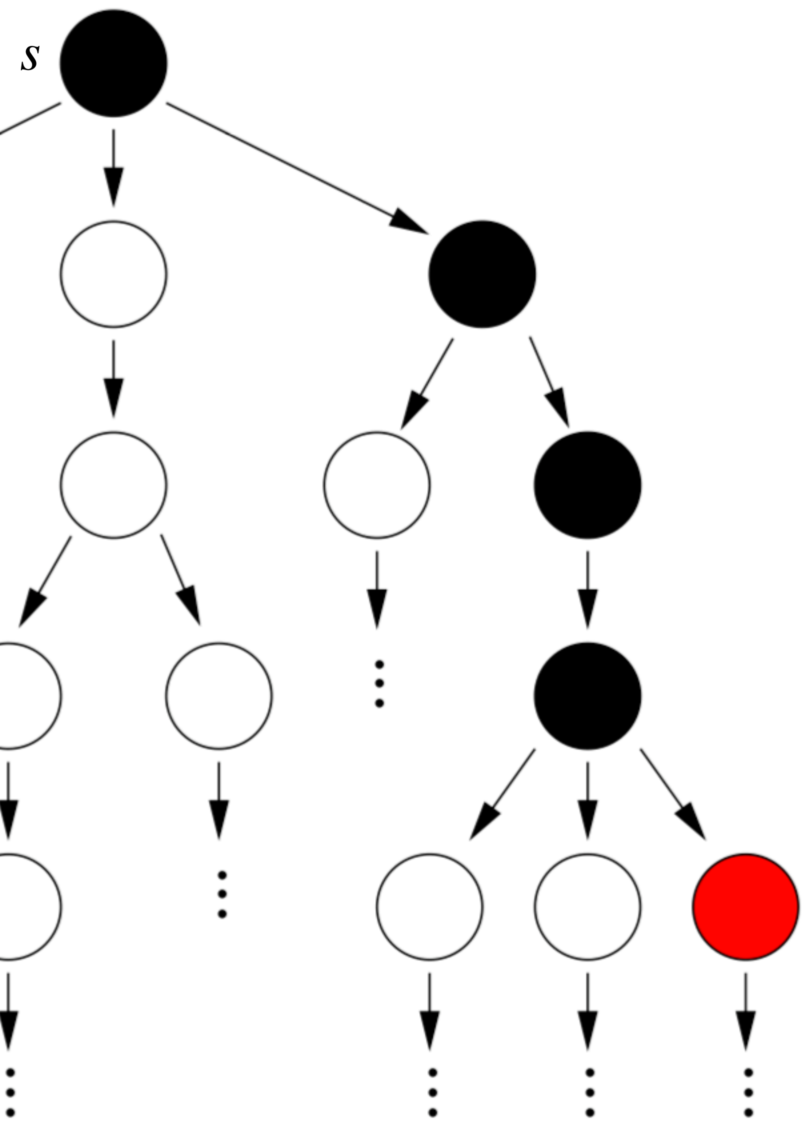
then $\mathcal{M},s \vDash \phi$

if ϕ is $\mathrm{A}[p \mathrm{~U~} q]$



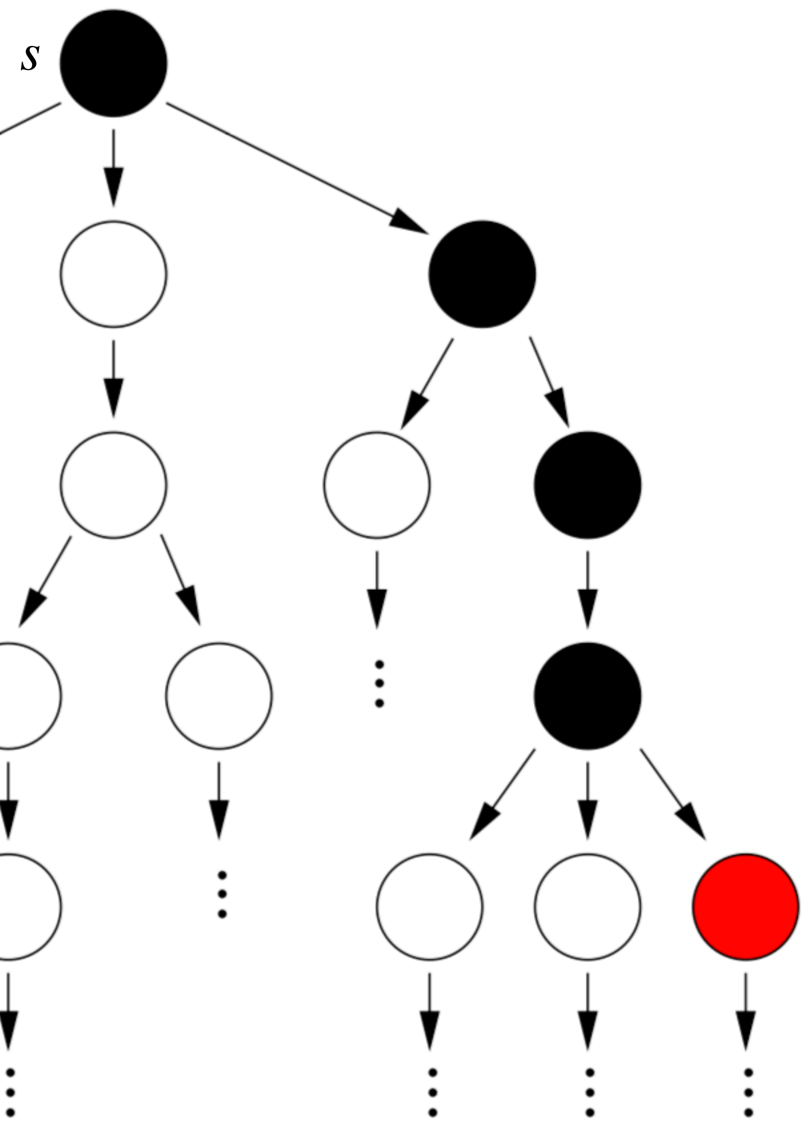
If p is true everywhere there is a black-filled circle and q is true in every red-filled circle then $\mathcal{M},s \vDash \phi$

if ϕ is...

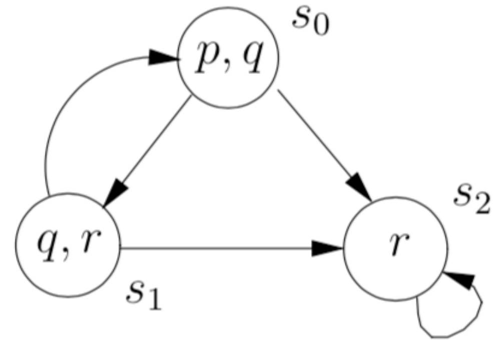


If p is true everywhere there is a black-filled circle and q is true in every red-filled circle then $\mathcal{M},s \models \phi$

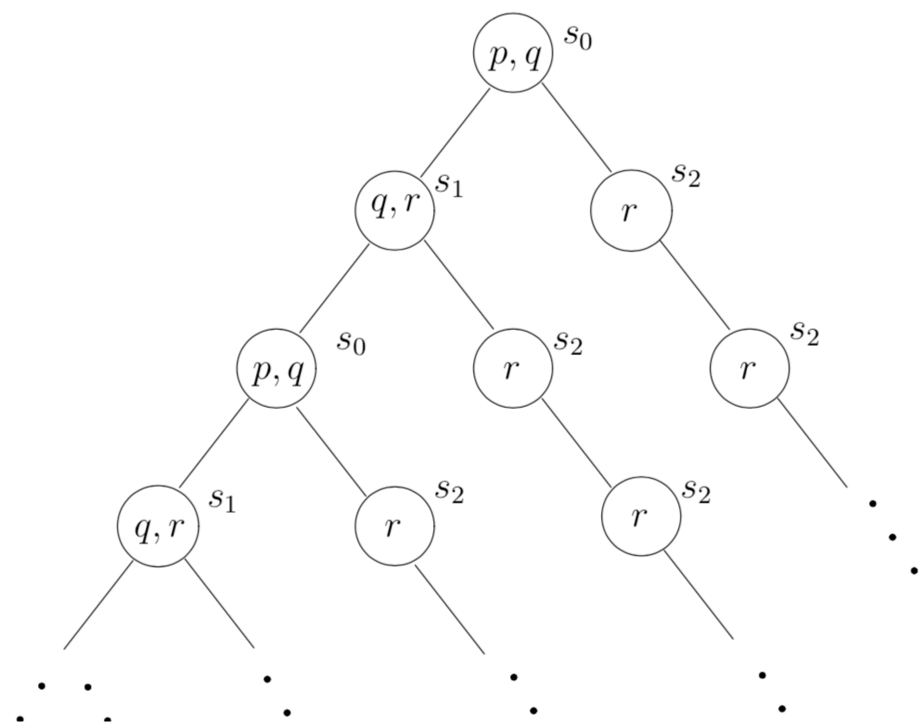
if ϕ is $\mathrm{E}[p \ \mathrm{U} \ q]$

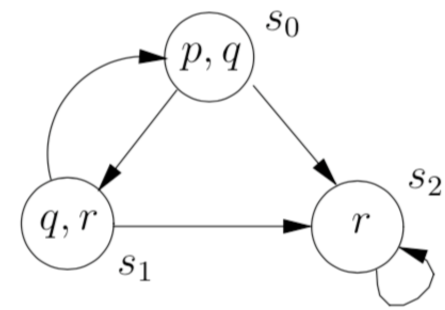


Let's look at some example checks for this system:



1. $\mathcal{M}, s_0 \models p \wedge q$ holds since the atomic symbols p and q are contained in the node of s_0.
2. $\mathcal{M}, s_0 \models \neg r$ holds since the atomic symbol r is *not* contained in node s_0.





3. $\mathcal{M}, s_0 \models \top$ holds by definition.
4. $\mathcal{M}, s_0 \models \text{EX}\ (q \wedge r)$

5. $\mathcal{M}, s_0 \models \neg\text{AX}\ (q \wedge r)$

6. $\mathcal{M}, s_0 \models \neg\text{EF}\,(p \wedge r)$

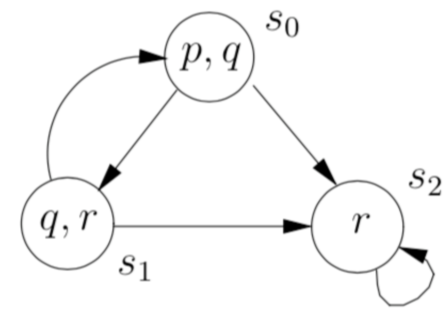
7. $\mathcal{M}, s_2 \models \text{EG}\, r$

8. $\mathcal{M}, s_0 \models \text{AF}\, r$

9. $\mathcal{M}, s_0 \models \text{E}[(p \wedge q)\ \text{U}\ \ r]$

10. $\mathcal{M}, s_0 \models \text{A}[p\ \text{U}\ r]$

11. $\mathcal{M}, s_0 \models \text{AG}\ (p \vee q \vee r \rightarrow \text{EF EG}\ r)$



3. $\mathcal{M}, s_0 \vDash \top$ holds by definition.
4. $\mathcal{M}, s_0 \vDash \mathrm{EX}\ (q \wedge r)$ holds since we have the leftmost computation path $s_0 \to s_1 \to s_0 \to s_1 \to \dots$ in Figure 3.5, whose second node s_1 contains q and r.
5. $\mathcal{M}, s_0 \vDash \neg\mathrm{AX}\ (q \wedge r)$ holds since we have the rightmost computation path $s_0 \to s_2 \to s_2 \to s_2 \to \dots$ in Figure 3.5, whose second node s_2 only contains r, but *not* q.
6. $\mathcal{M}, s_0 \vDash \neg\mathrm{EF}\,(p \wedge r)$ holds since there is no computation path beginning in s_0 such that we could reach a state where $p \wedge r$ would hold. This is so because there is simply no state whatsoever in this system where p and r hold at the same time.
7. $\mathcal{M}, s_2 \vDash \mathrm{EG}\, r$ holds since there is a computation path $s_2 \to s_2 \to s_2 \to \dots$ beginning in s_2 such that r holds in all future states of that path; this is the only computation path beginning at s_2 and so $\mathcal{M}, s_2 \vDash \mathrm{AG}\, r$ holds as well.
8. $\mathcal{M}, s_0 \vDash \mathrm{AF}\, r$ holds since, for all computation paths beginning in s_0, the system reaches a state (s_1 or s_2) such that r holds.
9. $\mathcal{M}, s_0 \vDash \mathrm{E}[(p \wedge q)\ \mathrm{U}\ r]$ holds since we have the rightmost computation path $s_0 \to s_2 \to s_2 \to s_2 \to \dots$ in Figure 3.5, whose second node s_2 ($i = 1$) satisfies r, but all previous nodes (only $j = 0$, i.e., node s_0) satisfy $p \wedge q$.
10. $\mathcal{M}, s_0 \vDash \mathrm{A}[p\ \mathrm{U}\ r]$ holds since p holds at s_0 and r holds in any possible successor state of s_0, so $p\ \mathrm{U}\ r$ is true for all computation paths beginning in s_0 (so we may choose $i = 1$ independently of the path).
11. $\mathcal{M}, s_0 \vDash \mathrm{AG}\ (p \vee q \vee r \to \mathrm{EF\ EG}\ r)$ holds since in all states reachable from s_0 and satisfying $p \vee q \vee r$ (all states in this case) the system can reach a state satisfying $\mathrm{EG}\ r$ (in this case state s_2).

Practical Patterns of Specifications

Suppose the atoms for a system use words such as `busy` and `requested`. We may require some of the following **properties** of the system:

It is impossible to get to a state where `started` holds, but `ready` does not hold:

$\mathrm{G}\neg(\mathtt{started} \wedge \neg\mathtt{ready})$

It is possible to get to a state where `started` holds, but `ready` does not hold:

$\mathrm{EF}\,(\mathtt{started} \wedge \neg\mathtt{ready})$

For any state, if a request (of a resource) occurs, then it will eventually be acknowledged:

$\mathrm{G}\,(\mathtt{requested} \rightarrow \mathrm{F}\,\mathtt{acknowledged})$

For any state, if a request (of a resource) occurs, then it will eventually be acknowledged:

$\mathrm{AG}\,(\mathrm{requested} \rightarrow \mathrm{AF}\,\mathrm{acknowledged})$

Whatever happens, a certain process will eventually be permanently deadlocked:

$\mathrm{F}\,\mathrm{G}\,\mathtt{deadlock}$

Whatever happens, a certain process will eventually be permanently deadlocked:

$\mathrm{AF}\,(\mathrm{AG}\,\mathtt{deadlock})$

Practical Patterns of Specifications

An upwards travelling elevator at the second floor does not change its direction when it still has passengers who want to go to the fifth floor:

$$\mathrm{G}\ (\texttt{floor2} \wedge \texttt{directionup} \wedge \texttt{ButtonPressed5} \rightarrow (\texttt{directionup}\ \mathrm{U}\ \texttt{floor5}))$$

An upwards travelling elevator at the second floor does not change its direction when it still has passengers who want to go to the fifth floor:

$$\mathrm{AG}\ (\texttt{floor2} \wedge \texttt{directionup} \wedge \texttt{ButtonPressed5} \rightarrow \mathrm{A}[\texttt{directionup}\ \mathrm{U}\ \texttt{floor5}])$$

The elevator can remain idle on the third floor with its doors closed:

$$\mathrm{AG}\ (\texttt{floor3} \wedge \texttt{idle} \wedge \texttt{doorclosed} \rightarrow \mathrm{EG}\ (\texttt{floor3} \wedge \texttt{idle} \wedge \texttt{doorclosed}))$$

A process can always request to enter its critical section.

$$\mathrm{AG}\ (n_1 \rightarrow \mathrm{EX}\ t_1)$$

non-critical state (n),
critical state (t)

Expressive Powers of LTL and CTL

- CTL allows <u>quantification over paths</u> so it is more expressive than LTL.
- But it does not allow one to select a range of paths with a formula, as LTL does.
- In that respect, LTL is more expressive.

For example, in LTL we can say:

‘all paths which have a *p* along them also have a *q* along them’

$$\mathrm{F}\,p \rightarrow \mathrm{F}\,q$$

We cannot write this in CTL because of the constraint that every F has an A or E.

The formula AF $p \rightarrow$ AF q says:

‘if <u>all paths have a p along them</u>, then all paths have a *q* along them’

One might write AG ($p \rightarrow$ AF q), which is closer:

‘every way of extending every path to a p eventually meets a q’

but that still does not capture the meaning of F p $\rightarrow$ F q.

CTL* is a logic which combines the expressive powers of LTL and CTL, by dropping the constraint that (X, U, F, G) is associated with a unique path quantifier (A, E).

Equivalences Between CTL Formulas

Definition 3.16

We say that two CTL formulas ϕ and ψ are semantically equivalent if <u>any state</u> in <u>any model</u> which satisfies one of them also satisfies the other; we denote this by

$$\phi \equiv \psi$$

G and F: universal and existential quantifiers over the states <u>along a specific path</u>

A and E: universal and existential quantifiers <u>on paths</u>

Not surprisingly, de Morgan rules exist:

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

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

$$\neg \mathrm{AX}\ \phi \equiv \mathrm{EX}\ \neg\phi$$

We also have the equivalences:

$$\mathrm{AF}\ \phi \equiv \mathrm{A}[\top\ \mathrm{U}\ \phi] \qquad \mathrm{EF}\ \phi \equiv \mathrm{E}[\top\ \mathrm{U}\ \phi]$$

Adequate Sets of Connectives for CTL

In propositional logic, the set {⊥, ∧, ¬} forms an adequate set of connectives, since the other connectives ∨, →, ⊤, can be written in terms of those three.

Adequate sets of connectives also exist in CTL:

Theorem 3.17
A set of temporal connectives in CTL is adequate if, and only if, it contains at least one of $\{\mathrm{AX}, \mathrm{EX}\}$, at least one of $\{\mathrm{EG}, \mathrm{AF}, \mathrm{AU}\}$ and EU.

Therefore EX, AU, and EU form an adequate set of temporal connectives.

The temporal connectives AR, ER, AW and EW are all definable in CTL:

- $\mathrm{A}[\phi \mathrel{\mathrm{R}} \psi] = \neg\mathrm{E}[\neg\phi \mathrel{\mathrm{U}} \neg\psi]$
- $\mathrm{E}[\phi \mathrel{\mathrm{R}} \psi] = \neg\mathrm{A}[\neg\phi \mathrel{\mathrm{U}} \neg\psi]$
- $\mathrm{A}[\phi \mathrel{\mathrm{W}} \psi] = \mathrm{A}[\psi \mathrel{\mathrm{R}} (\phi \vee \psi)]$, and then use the first equation above
- $\mathrm{E}[\phi \mathrel{\mathrm{W}} \psi] = \mathrm{E}[\psi \mathrel{\mathrm{R}} (\phi \vee \psi)]$, and then use the second one.

Model Checking in CTL

We use a model $\mathcal{M}$ to describe a system of interest.

We use an CTL formula ϕ to describe a property of the system.

We can check that the model satisfies the property: $\mathcal{M}, s_0 \vDash \phi$

There are two main ways to consider this problem:

1. The inputs could be the model $\mathcal{M}$, the formula ϕ, and a state s_0 as input; the output is ‘yes’ ($\mathcal{M}, s_0 \vDash \phi$ holds), or ‘no’ ($\mathcal{M}, s_0 \vDash \phi$ does not hold).

2. Or, the inputs could be just $\mathcal{M}$ and ϕ, and the output would be all states s of the model $\mathcal{M}$ which satisfy ϕ.

It is easier to design an algorithm to solve number 2.

This will also solve number 1, since we can check if s_0 is an element of the output set.

The Labeling Algorithm

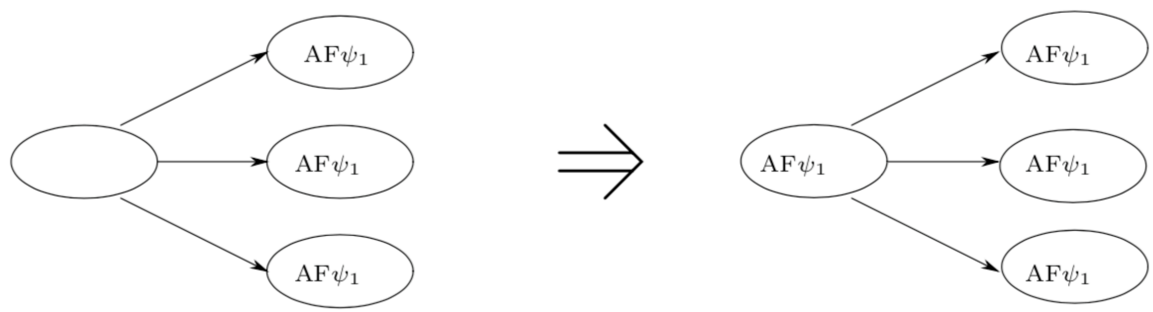
INPUT: a CTL model $\mathcal{M} = (S, \rightarrow, L)$ and a CTL formula ϕ.

OUTPUT: the set of states of $\mathcal{M}$ which satisfy ϕ.

// write ϕ in terms of AF, EU, EX, ∧, ¬ and ⊥ using the equivalences

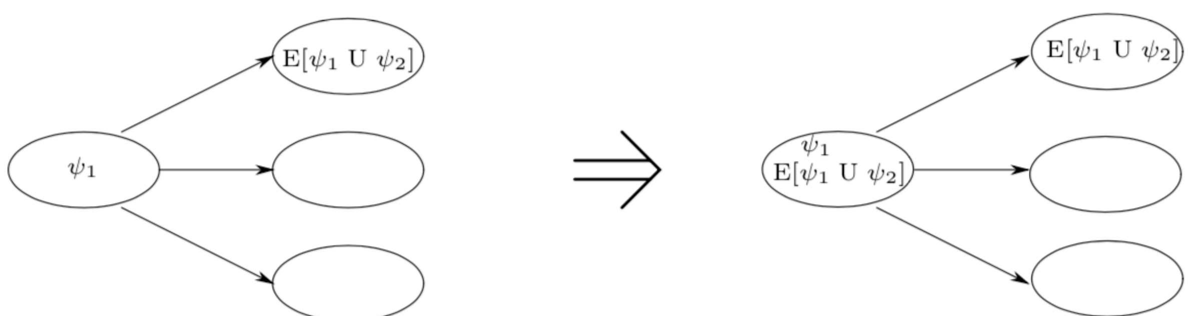
ϕ = TRANSLATE (ϕ)

// label the states of $\mathcal{M}$ with the subformulas of ϕ that are satisfied there,
// starting with the smallest subformulas and working outwards towards ϕ



If ψ is AF ψ_1

Repeat: label any state with AF ψ_1 if all successor states are labeled with AF ψ_1, until there is no change.



If ψ is E[ψ_1 U ψ_2]

Repeat: label any state with E[ψ_1 U ψ_2] if it is labelled with ψ_1 and at least one of its successors is labelled with E[ψ_1 U ψ_2], until there is no change.

If ψ is EX ψ_1

label any state with EX ψ_1 if one of its successors is labelled with ψ_1

Having performed the labeling for all the subformulas of ϕ (including ϕ itself), we output the states which are labelled ϕ.