CS 181U Applied Logic

Lecture 9

Computation Tree Logic

*v*SMV peculiarities

The `process` keyword will be deprecated.
Ignore the warning.

*v*SMV peculiarities

The `process` keyword will be deprecated.
Ignore the warning.

Comments indicated with two dashes: -

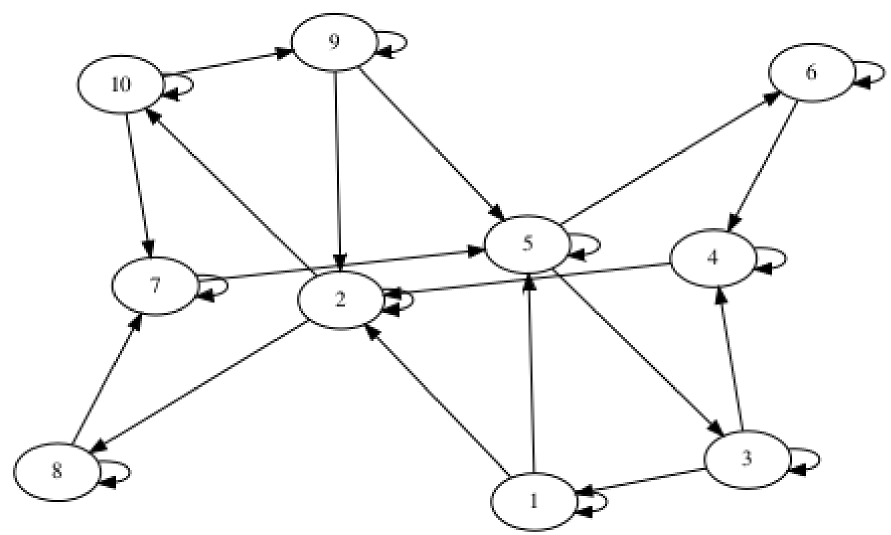
*v*SMV peculiarities

The `process` keyword will be deprecated. Ignore the warning.

Comments indicated with two dashes: –

MODULE `main`, and processes `P0` and `P1` are three separate threads. `P0` and `P1` are subthreads of `main`.

```
MODULE proc(id, ...)
    ...
MODULE main
    ...
p0 = proc(0, ...)
p1 = proc(1, ...)
    ...
```



Use `FAIRNESS running` in `proc` specification.

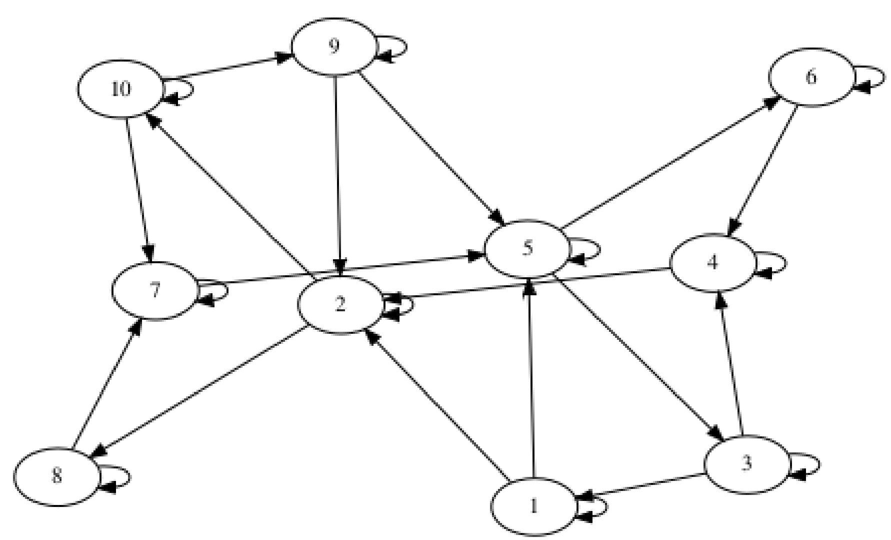
νSMV peculiarities

The `process` keyword will be deprecated.
Ignore the warning.

Comments indicated with two dashes: –

MODULE `main`, and processes `P0` and `P1` are three separate threads. `P0` and `P1` are subthreads of `main`.

```
MODULE proc(id, ...)
    ...
MODULE main
    ...
p0 = proc(0, ...)
p1 = proc(1, ...)
    ...
```



Use `FAIRNESS running` in `proc` specification.

Interesting Quote

If what is exactly stated can be done by a machine, the residue of the uniquely human becomes coextensive with the linguistic qualities that interfere with precise specification—ambiguity, metaphoric play, multiple encoding, and allusive exchanges between one symbol system and another. The uniqueness of human behavior thus becomes assimilated to the ineffability of language, and the common ground that humans and machines share is identified with the univocality of an instrumental language that has banished ambiguity from its lexicon.

-N. Katherine Hayles

How we Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics

Reminder

Linear Temporal Logic (LTL)

We will assign symbols for expressing temporal system requirements like *always* (G), *eventually* (F), *next* (X), *until* (U), and a few more. We will give a formal and unambiguous semantics to these symbols.

Transition Systems

We will learn a formal system of specifying transition systems (which we often depict as a transition diagram).

Concurrency Concepts

Safety, liveness, mutual exclusion, ...

Verification Software

Symbolic Model Verifier (NuSMV)

Reminder

Linear Temporal Logic (LTL)

We will assign symbols for expressing temporal system requirements like *always* (G), *eventually* (F), *next* (X), *until* (U), and a few more. We will give a formal and unambiguous semantics to these symbols.

Transition Systems

We will learn a formal system of specifying transition systems (which we often depict as a transition diagram).

Concurrency Concepts

Safety, liveness, mutual exclusion, ...

Verification Software

Symbolic Model Verifier (NuSMV)

We did all this.

Next

Computation Tree Logic (CTL)

We will learn a different way to write temporal properties of systems.

Verification Software + CTL

Symbolic Model Verifier (NuSMV) with CTL

Next

Computation Tree Logic (CTL)

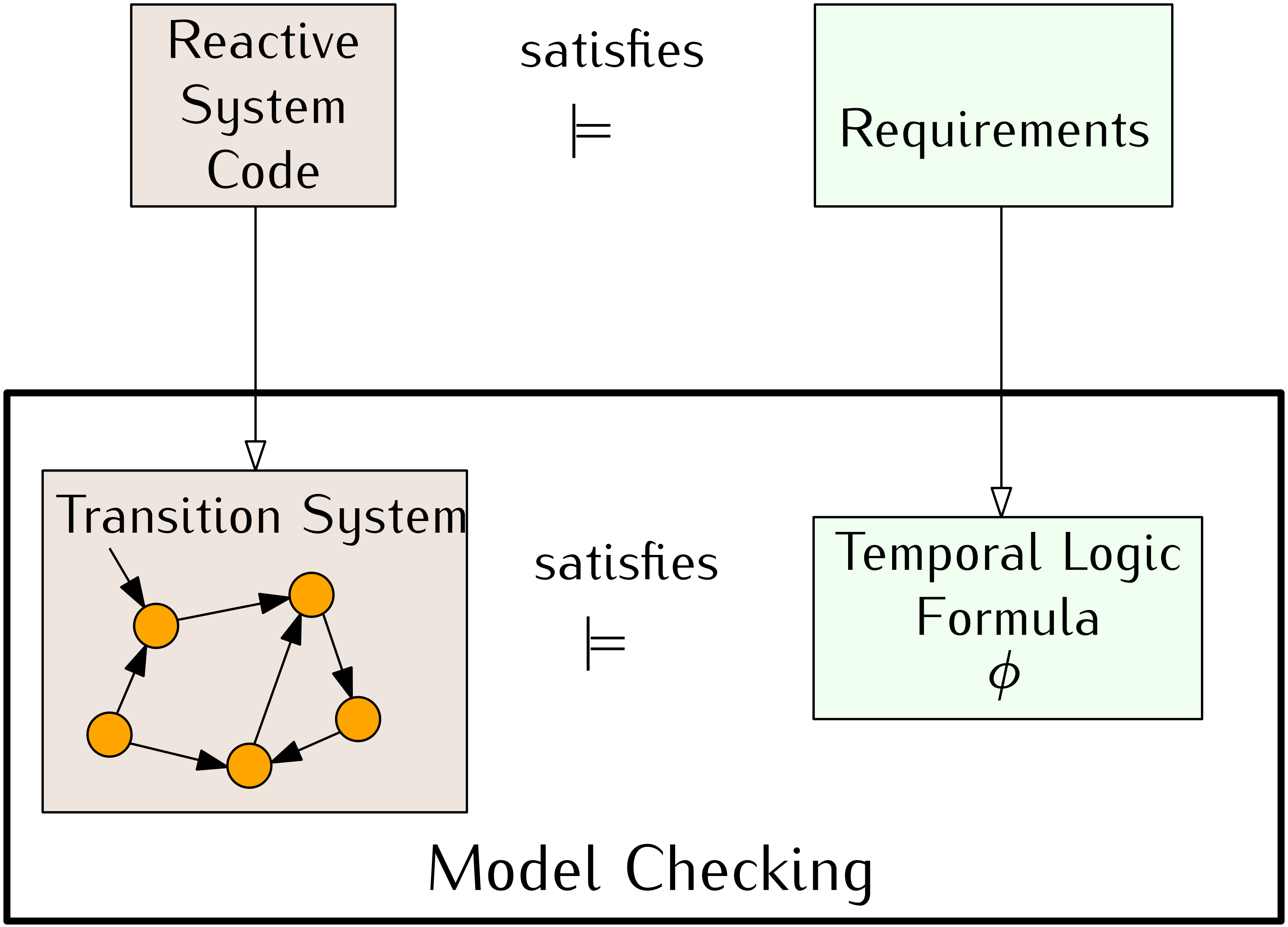
We will learn a different way to write temporal properties of systems.

Today

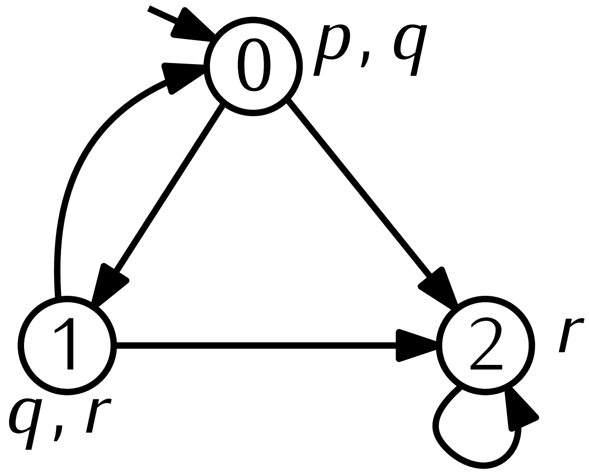
Verification Software + CTL

Symbolic Model Verifier (NuSMV) with CTL

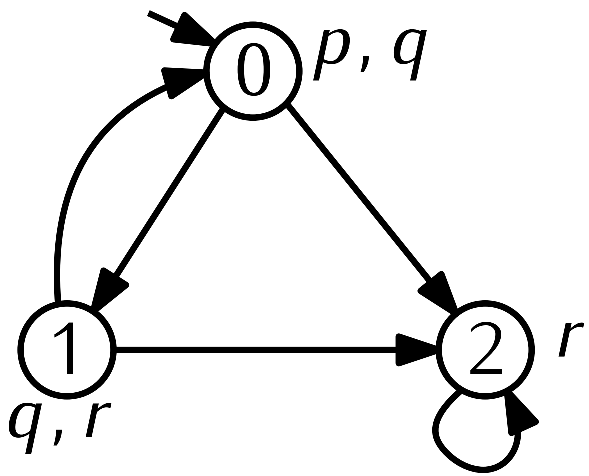
Remember the big picture



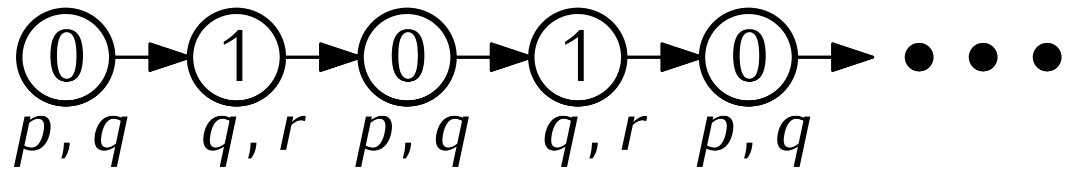
Linear vs Branching Time Logic



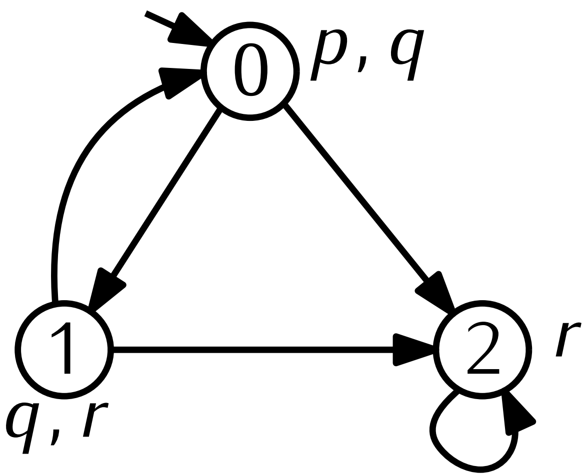
Linear vs Branching Time Logic



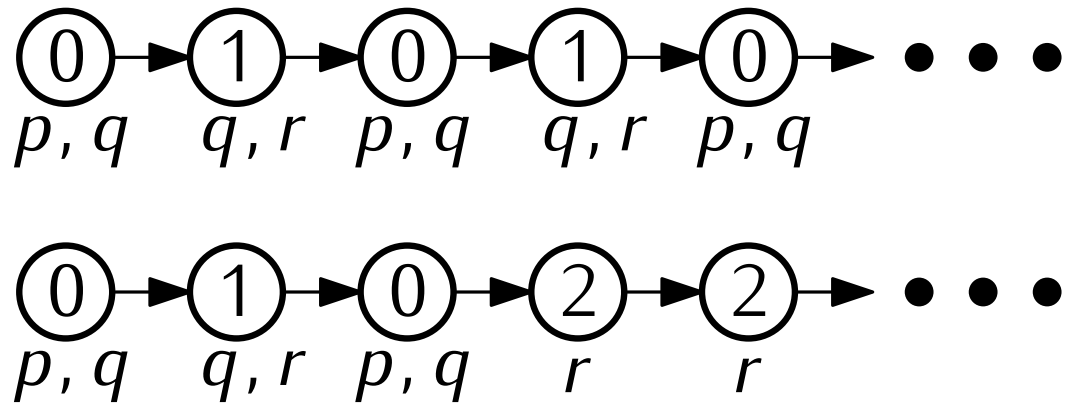
Some paths of $\mathcal{M}$



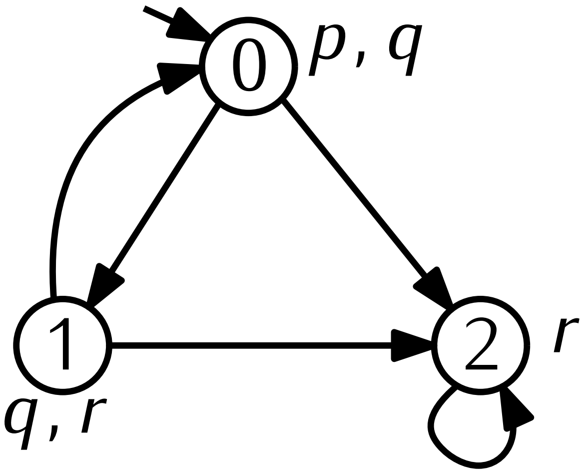
Linear vs Branching Time Logic



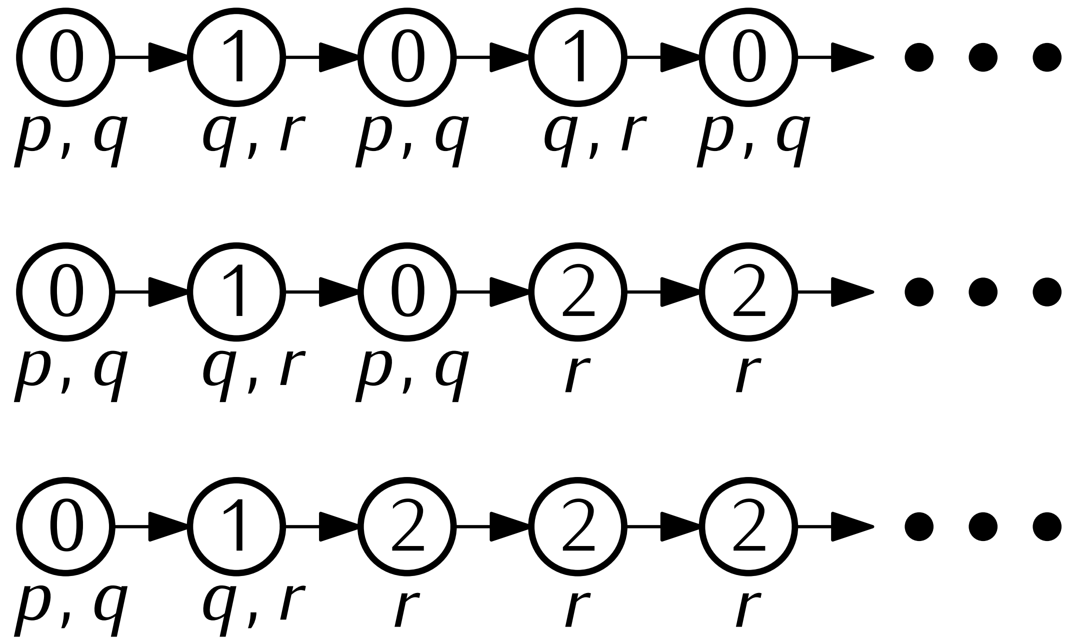
Some paths of $\mathcal{M}$



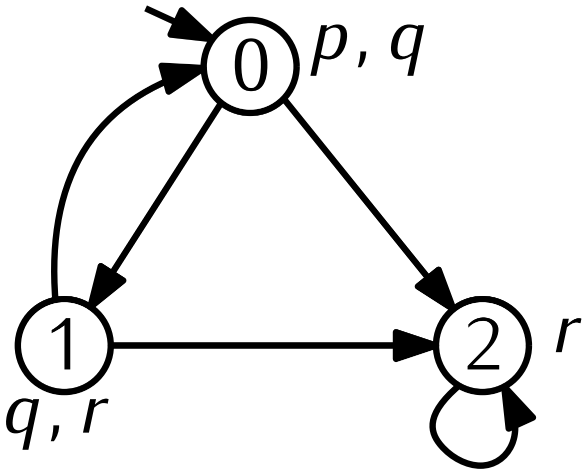
Linear vs Branching Time Logic



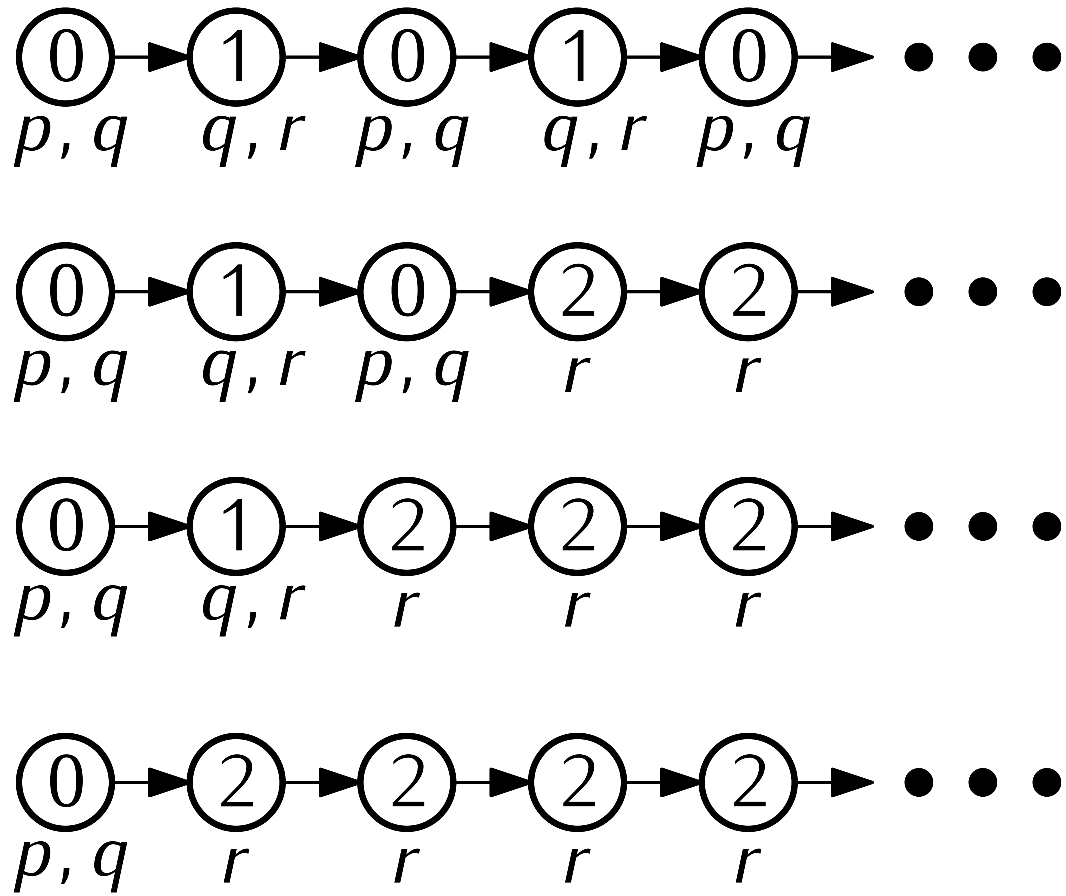
Some paths of $\mathcal{M}$



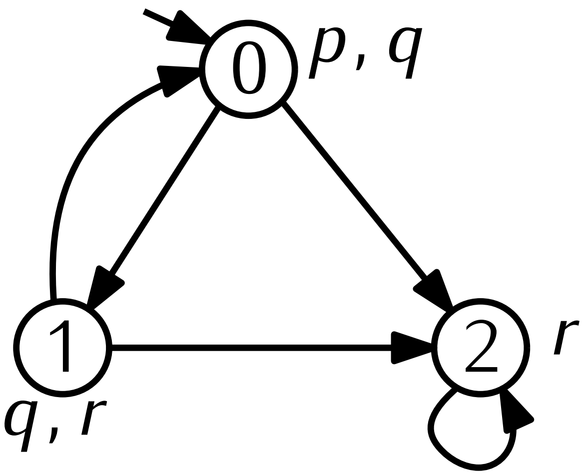
Linear vs Branching Time Logic



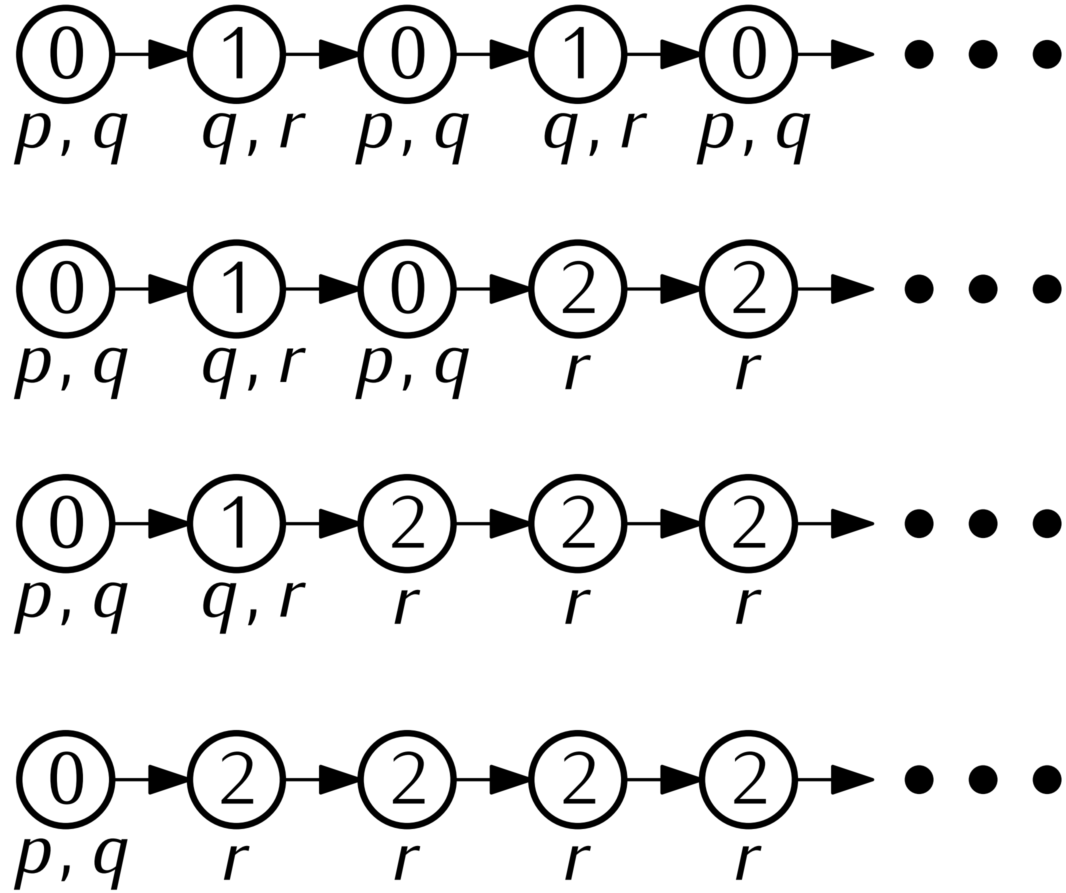
Some paths of $\mathcal{M}$



Linear vs Branching Time Logic

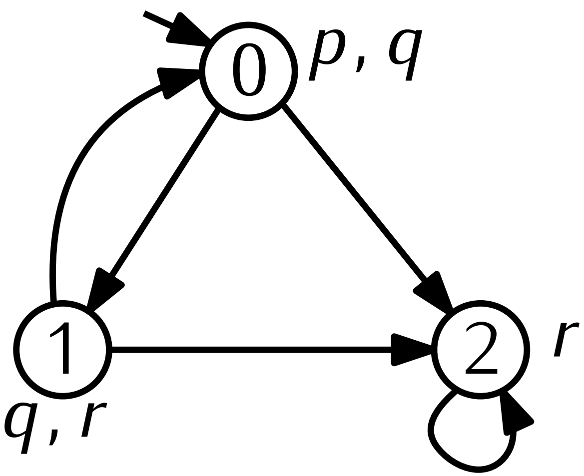


Some paths of $\mathcal{M}$

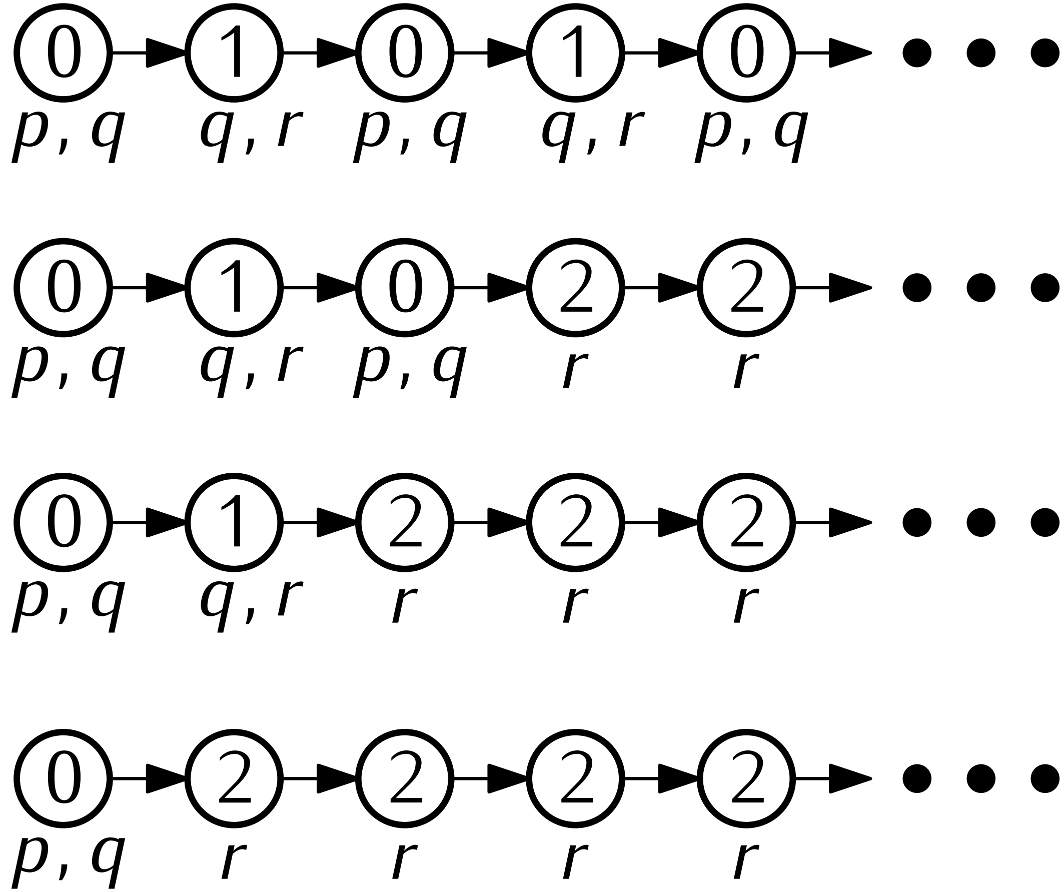


LTL Model
Checking

Linear vs Branching Time Logic



Some paths of $\mathcal{M}$

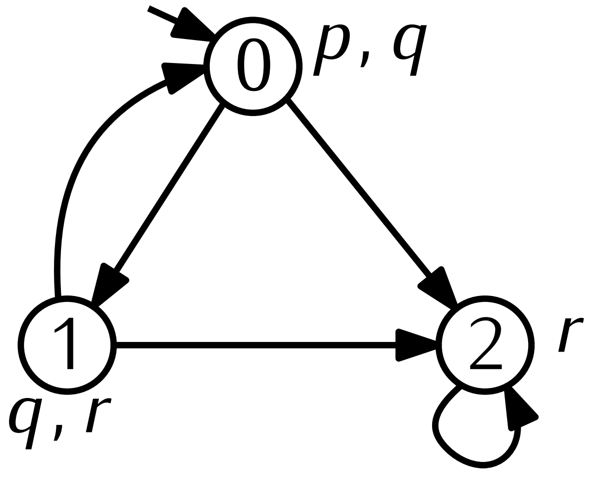


LTL Model Checking

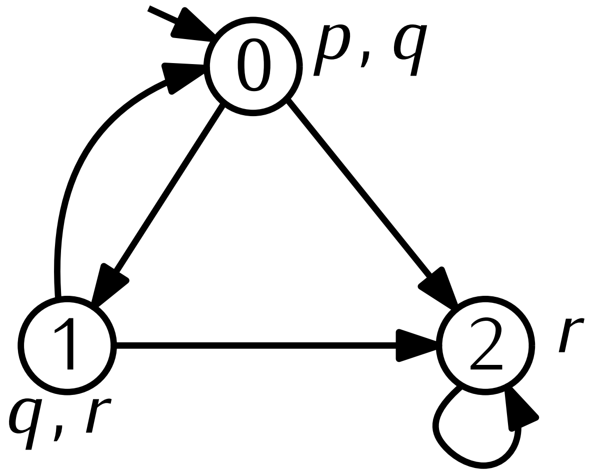
$$\mathcal{M} \models \phi \Leftrightarrow \forall \pi \; [\pi \models \phi]$$

LTL formula

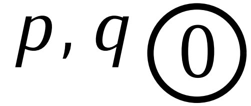
Linear vs Branching Time Logic



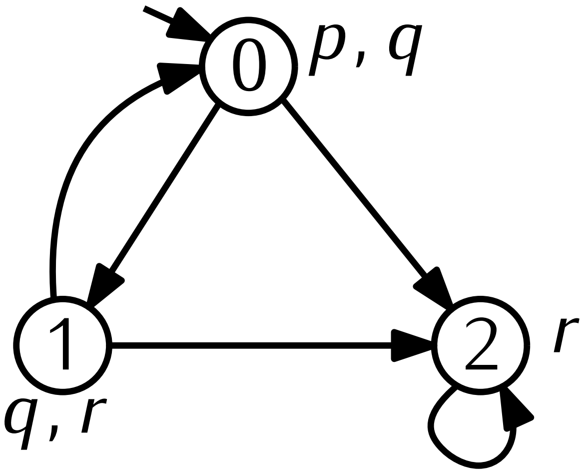
Linear vs Branching Time Logic



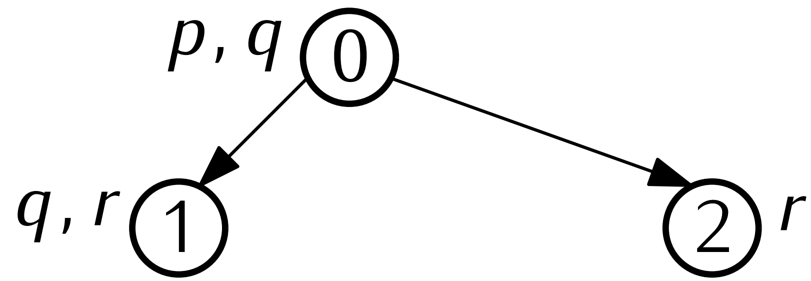
Computation tree for $\mathcal{M}$



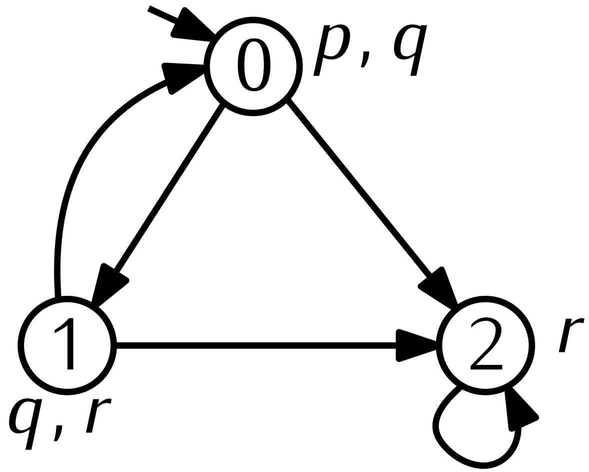
Linear vs Branching Time Logic



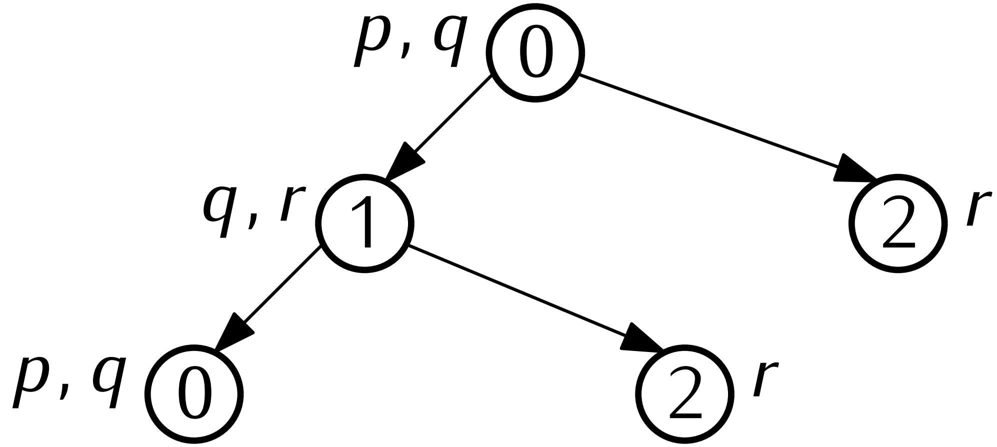
Computation tree for $\mathcal{M}$



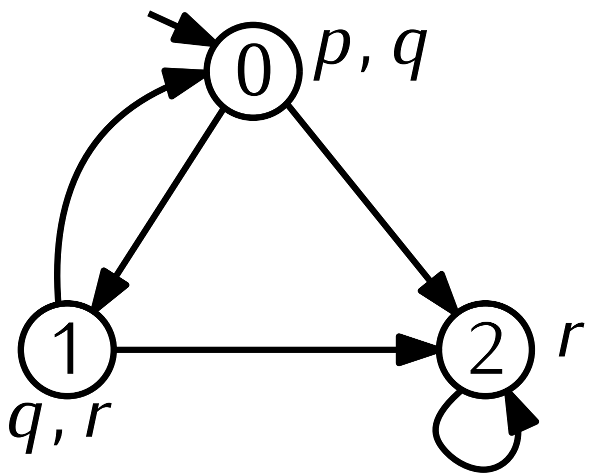
Linear vs Branching Time Logic



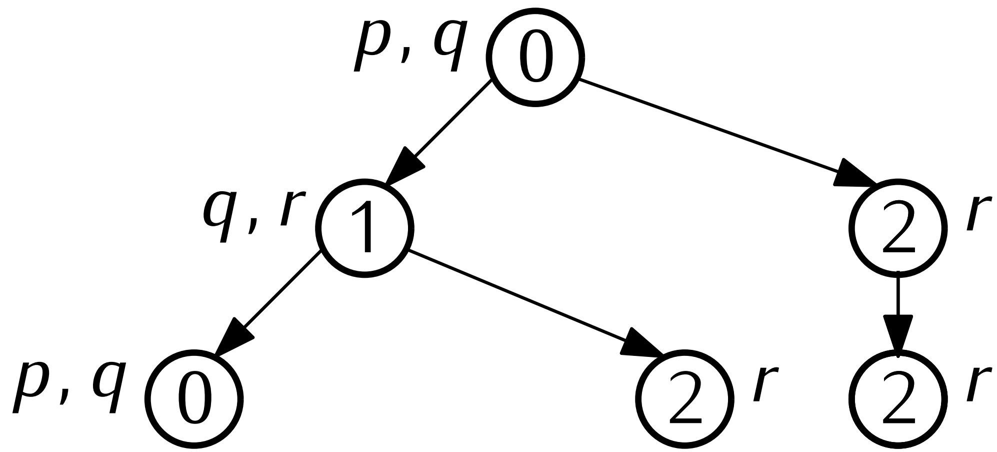
Computation tree for $\mathcal{M}$



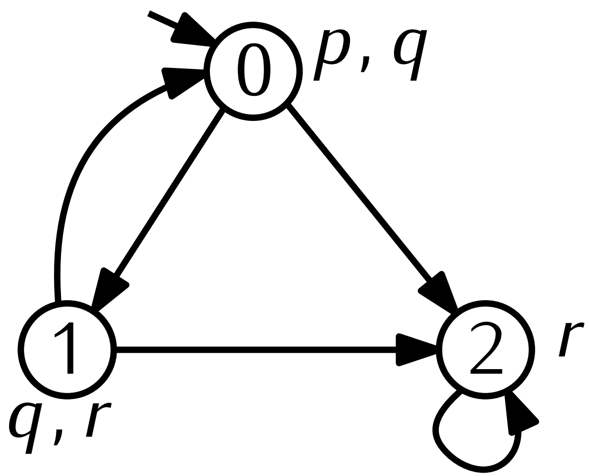
Linear vs Branching Time Logic



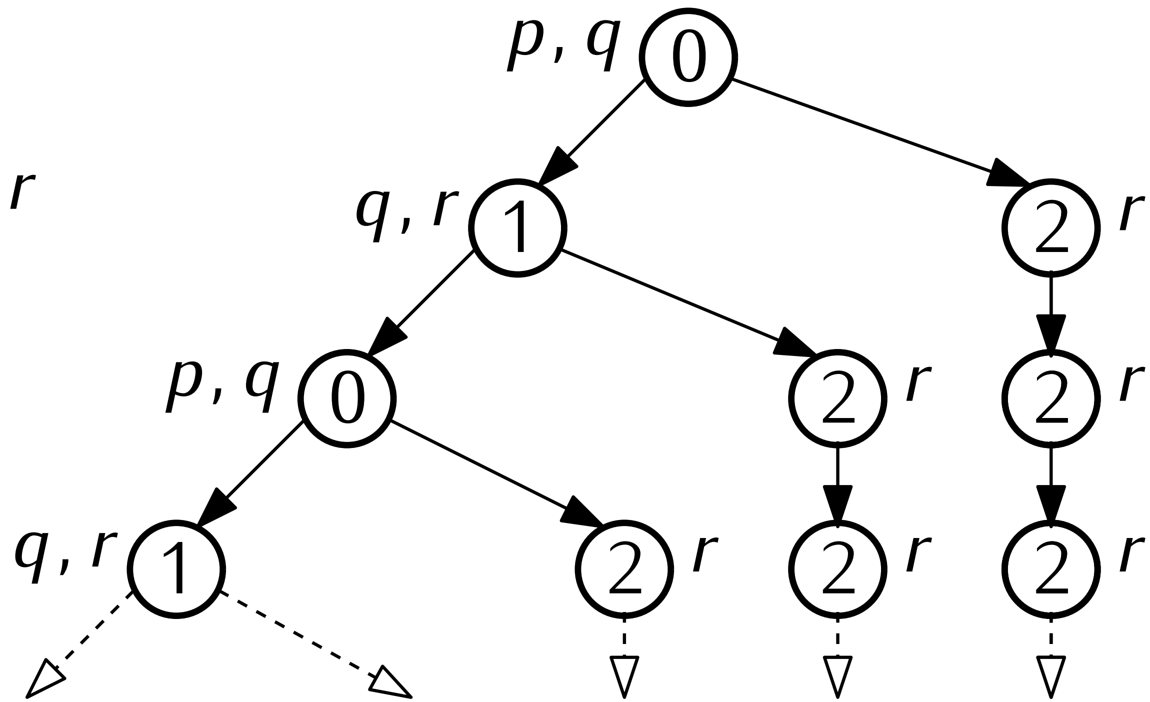
Computation tree for $\mathcal{M}$



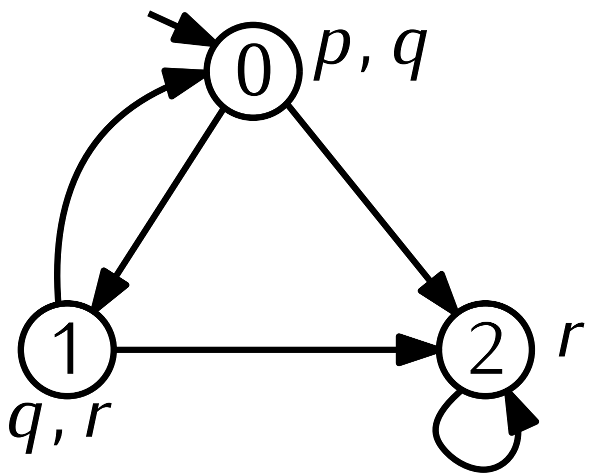
Linear vs Branching Time Logic



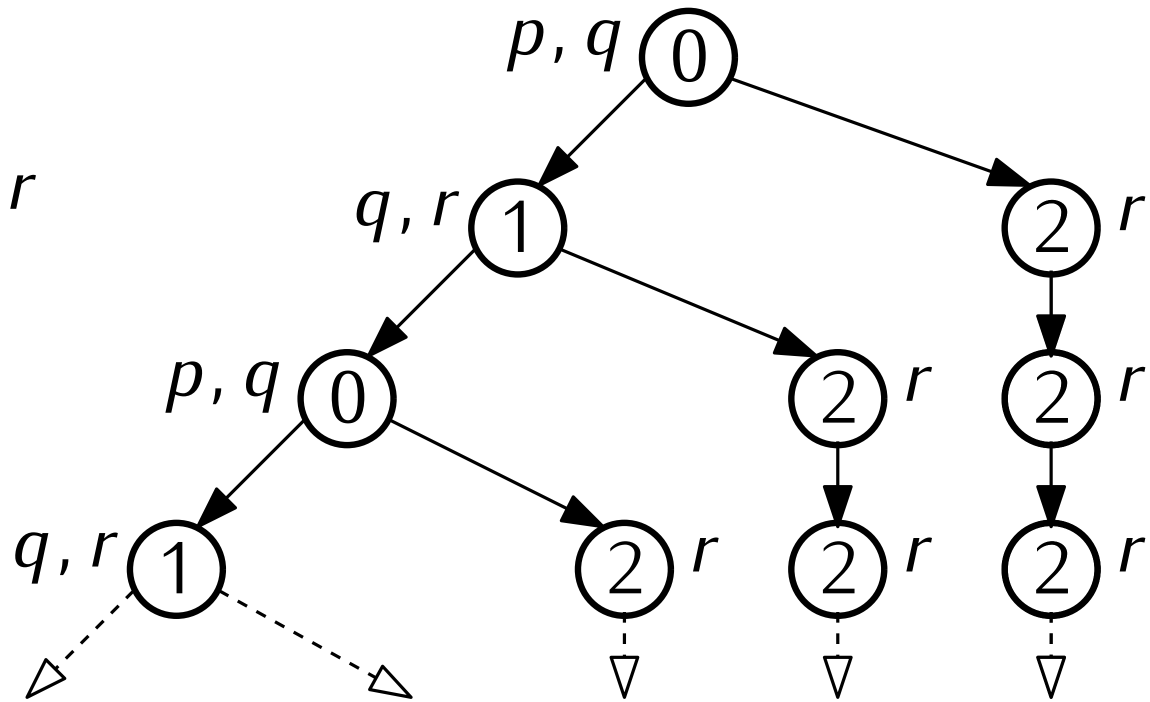
Computation tree for $\mathcal{M}$



Linear vs Branching Time Logic

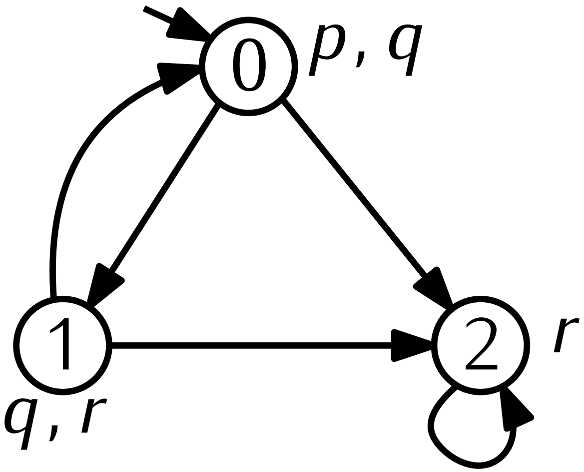


Computation tree for $\mathcal{M}$

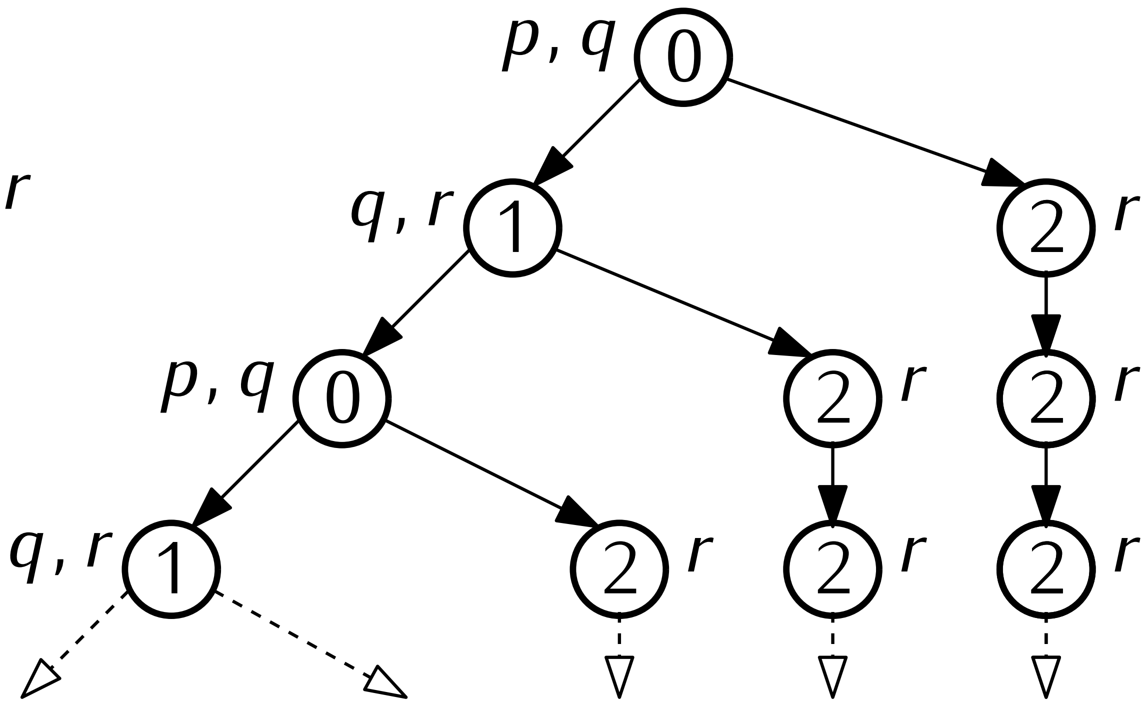


Computation Tree Logic (CTL) expresses properties of “alternative timelines”.

Linear vs Branching Time Logic



Computation tree for $\mathcal{M}$



Computation Tree Logic (CTL) expresses properties of "alternative timelines".

$$\mathcal{M} \models \phi \Leftrightarrow ??????????$$

CTL formula

Computation Tree Logic Syntax

Suppose α and β are LTL formulas.
Suppose p_i is a propositional atom.
Then the following are all LTL formulas.

$$\top \quad \bot \quad p_i$$

$$\neg\alpha \quad \alpha \vee \beta \qquad \alpha \wedge \beta \quad \alpha \to \beta$$

Computation Tree Logic Syntax

Suppose α and β are LTL formulas.
Suppose p_i is a propositional atom.
Then the following are all LTL formulas.

$$\top \quad \bot \quad p_i$$

$$\neg\alpha \quad \alpha \vee \beta \quad \alpha \wedge \beta \quad \alpha \rightarrow \beta$$

We use the same temporal operators: G, F, X, U

We attach path quantifiers, A (inevitably) or E (possibly), to each temporal operator.

$$AG \quad AF \quad AX \quad AU$$

$$EG \quad EF \quad EX \quad EU$$

Computation Tree Logic Semantics

Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, s, of $\mathcal{M}$ satisfies a CTL formula ϕ, written $s \models \phi$, according to the following recursive informal definition:

Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, s, of $\mathcal{M}$ satisfies a CTL formula ϕ, written $s \models \phi$, according to the following recursive informal definition:

- Base case: If ϕ is atomic then $s \models \phi$ iff $\phi \in L(s)$.

Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, s, of $\mathcal{M}$ satisfies a CTL formula ϕ, written $s \models \phi$, according to the following recursive informal definition:

- Base case: If ϕ is atomic then $s \models \phi$ iff $\phi \in L(s)$.
- If $\phi = \alpha \wedge \beta$, then $s \models \phi$ iff $s \models \alpha$ and $s \models \beta$. Similar rules apply if ϕ is one of $\neg, \vee, \rightarrow$.

Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, s, of $\mathcal{M}$ satisfies a CTL formula ϕ, written $s \models \phi$, according to the following recursive informal definition:

- Base case: If ϕ is atomic then $s \models \phi$ iff $\phi \in L(s)$.
- If $\phi = \alpha \wedge \beta$, then $s \models \phi$ iff $s \models \alpha$ and $s \models \beta$. Similar rules apply if ϕ is one of $\neg, \vee, \rightarrow$.
- If ϕ is an A-operator, then $s \models \phi$ iff **all paths** starting at s satisfy the 'LTL formula' made by removing the A symbol.

Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, s, of $\mathcal{M}$ satisfies a CTL formula ϕ, written $s \models \phi$, according to the following recursive informal definition:

- Base case: If ϕ is atomic then $s \models \phi$ iff $\phi \in L(s)$.
- If $\phi = \alpha \wedge \beta$, then $s \models \phi$ iff $s \models \alpha$ and $s \models \beta$. Similar rules apply if ϕ is one of $\neg, \vee, \rightarrow$.
- If ϕ is an A-operator, then $s \models \phi$ iff **all paths** starting at s satisfy the 'LTL formula' made by removing the A symbol.
- If ϕ is an E-operator, then $s \models \phi$ iff **there exists a path** starting at s satisfy the 'LTL formula' made by removing the E symbol.

Computation Tree Logic Semantics

We just defined what it means for a state to satisfy a CTL property, $s \models \phi$.

Computation Tree Logic Semantics

We just defined what it means for a state to satisfy a CTL property, $s \models \phi$.

Now we want to define what it means for a transition system $\mathcal{M}$ to satisfy a CTL propery.

Computation Tree Logic Semantics

We just defined what it means for a state to satisfy a CTL property, $s \models \phi$.

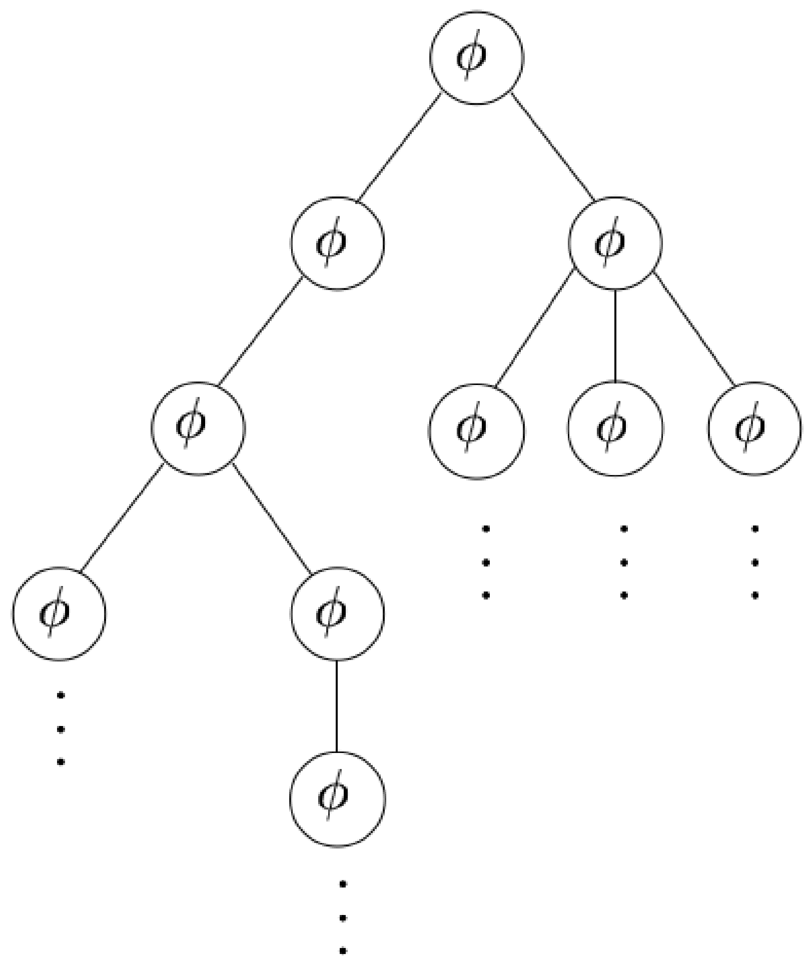
Now we want to define what it means for a transition system $\mathcal{M}$ to satisfy a CTL propery.

$$\mathcal{M} \models \phi \Leftrightarrow \forall s \in I \;\; s \models \phi$$

CTL Model Checking

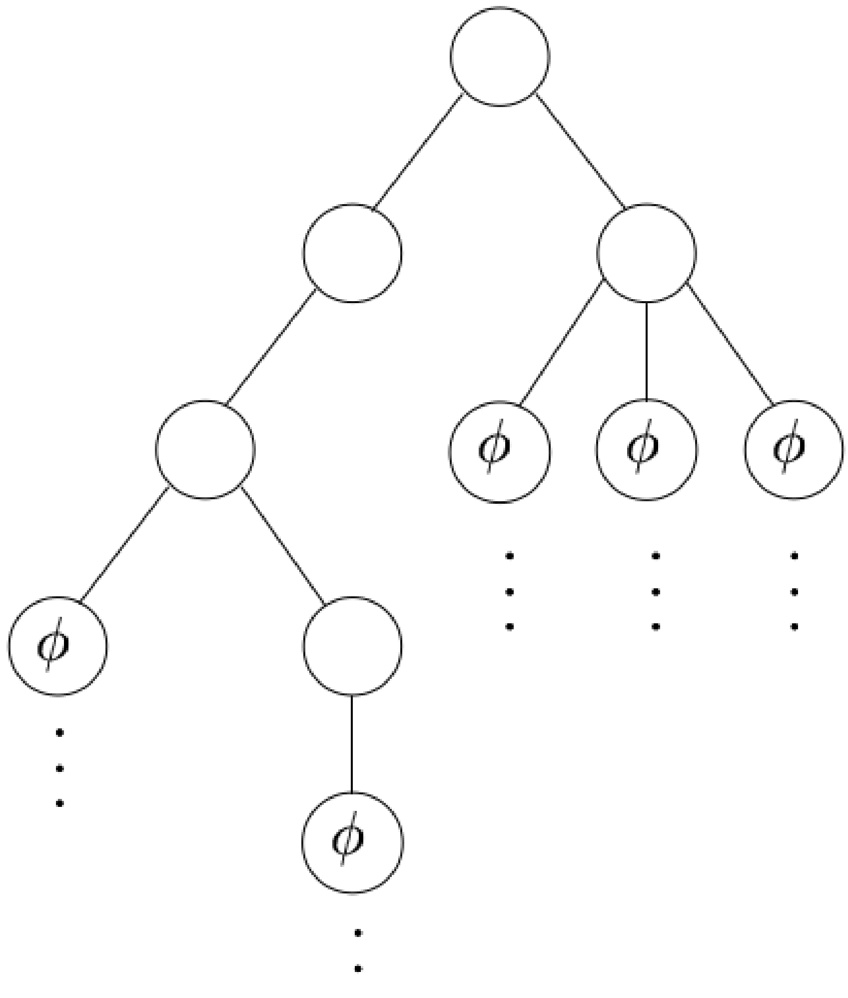
Computation Tree Logic Semantics

The idea of $AG\phi$: inevitably always ϕ



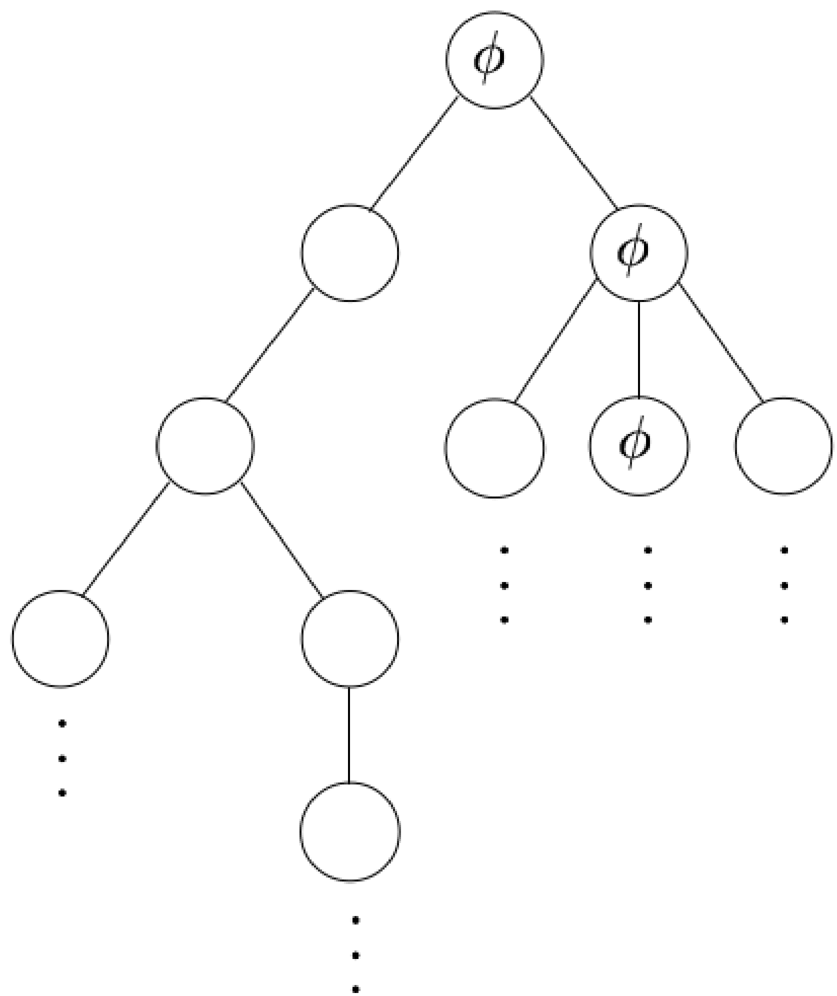
Computation Tree Logic Semantics

The idea of $AF\phi$: inevitably eventually ϕ



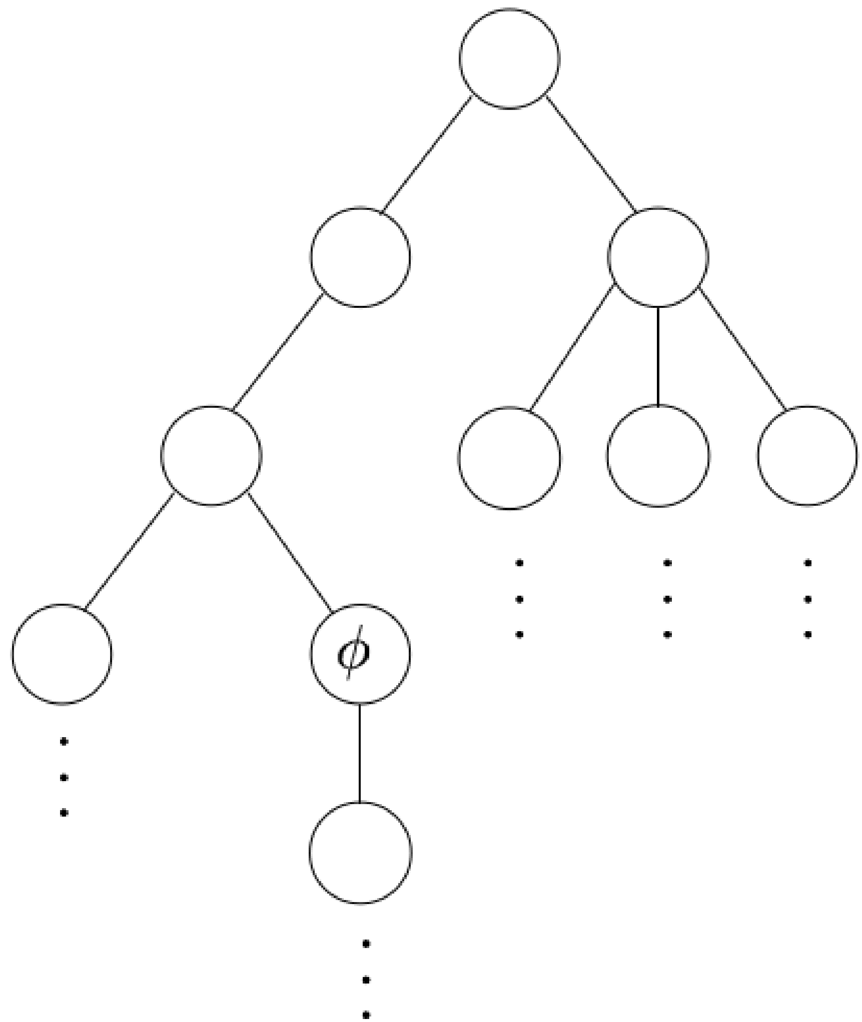
Computation Tree Logic Semantics

The idea of $EG\phi$: possibly always ϕ



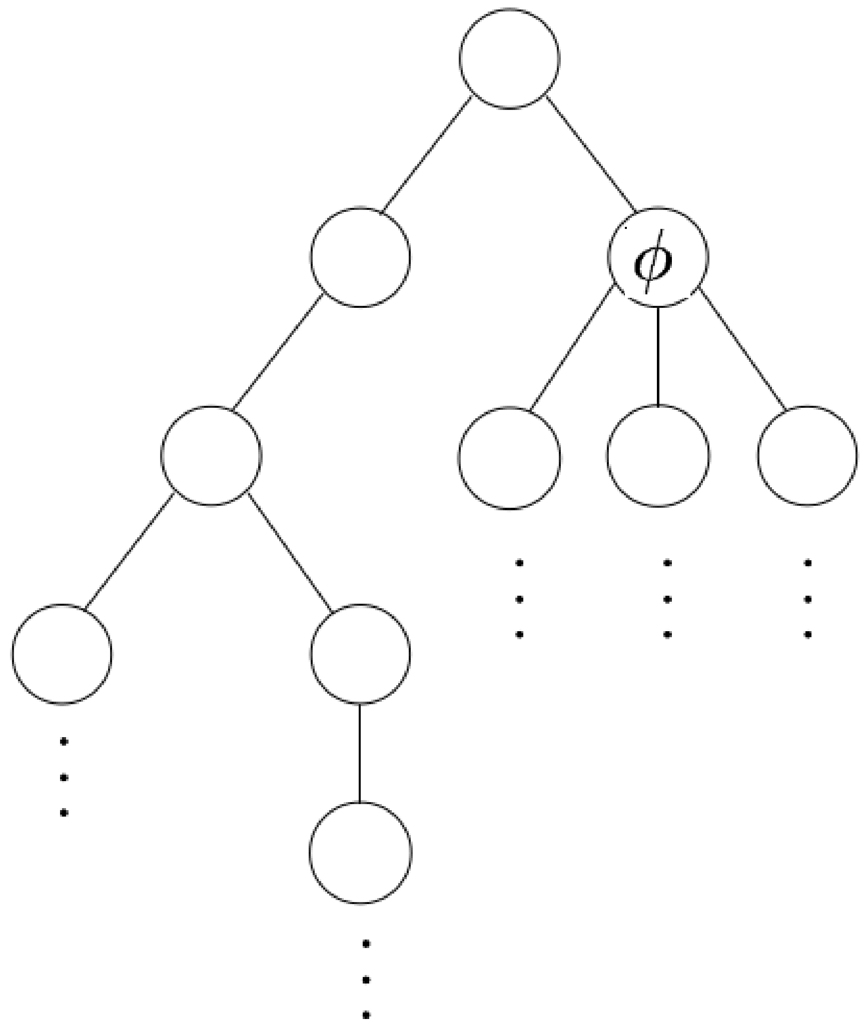
Computation Tree Logic Semantics

The idea of $EF\phi$: possibly eventually ϕ



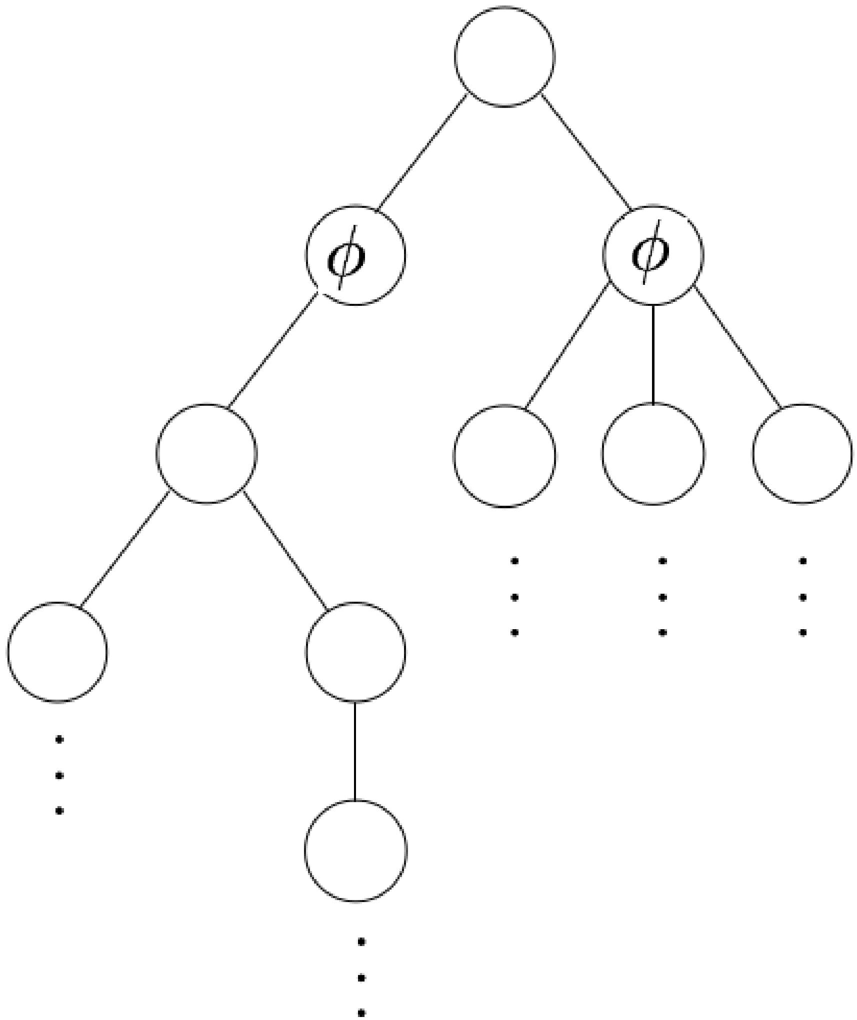
Computation Tree Logic Semantics

The idea of $EX\phi$: possibly next ϕ



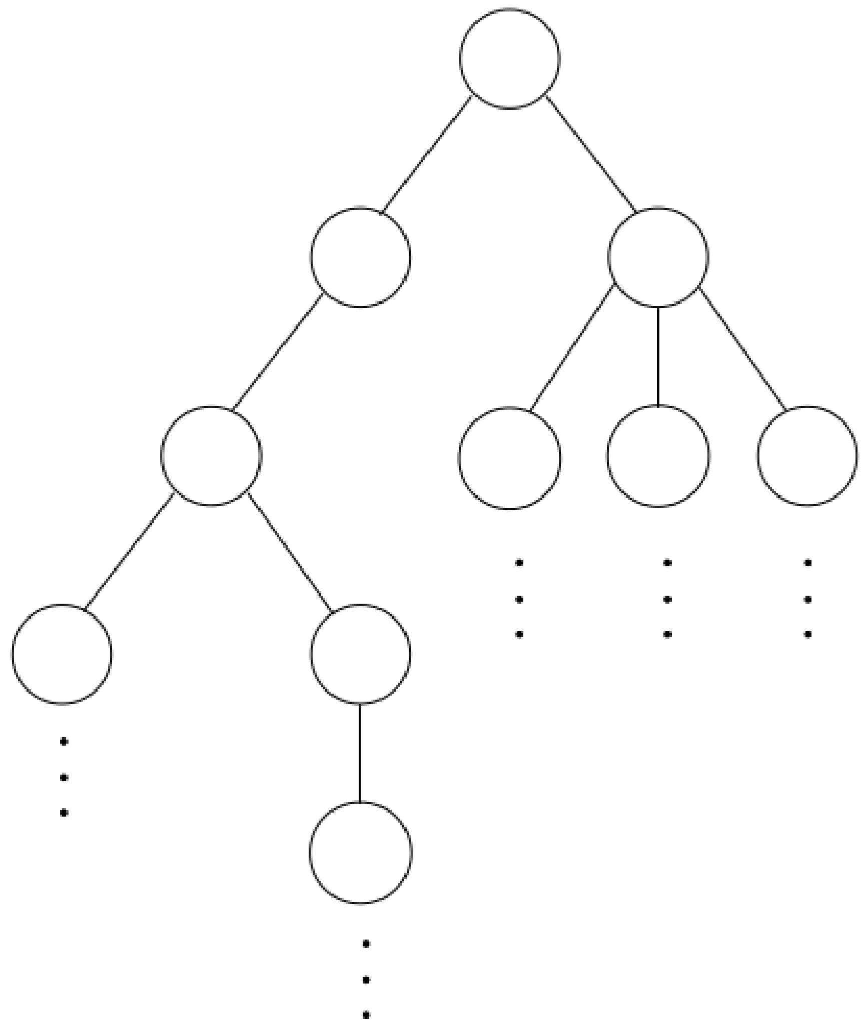
Computation Tree Logic Semantics

The idea of $AX\phi$: inevitably next ϕ



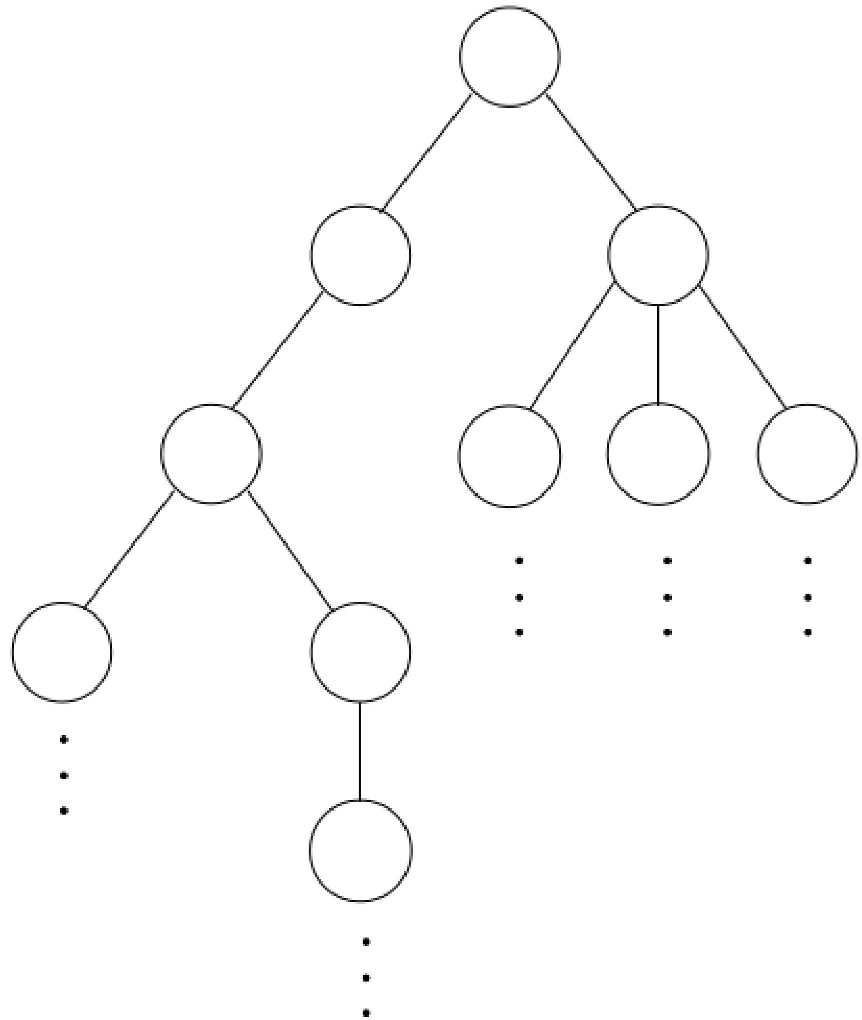
Computation Tree Logic Semantics

The idea of $\phi AU \psi$: inevitably ϕ until ψ



Computation Tree Logic Semantics

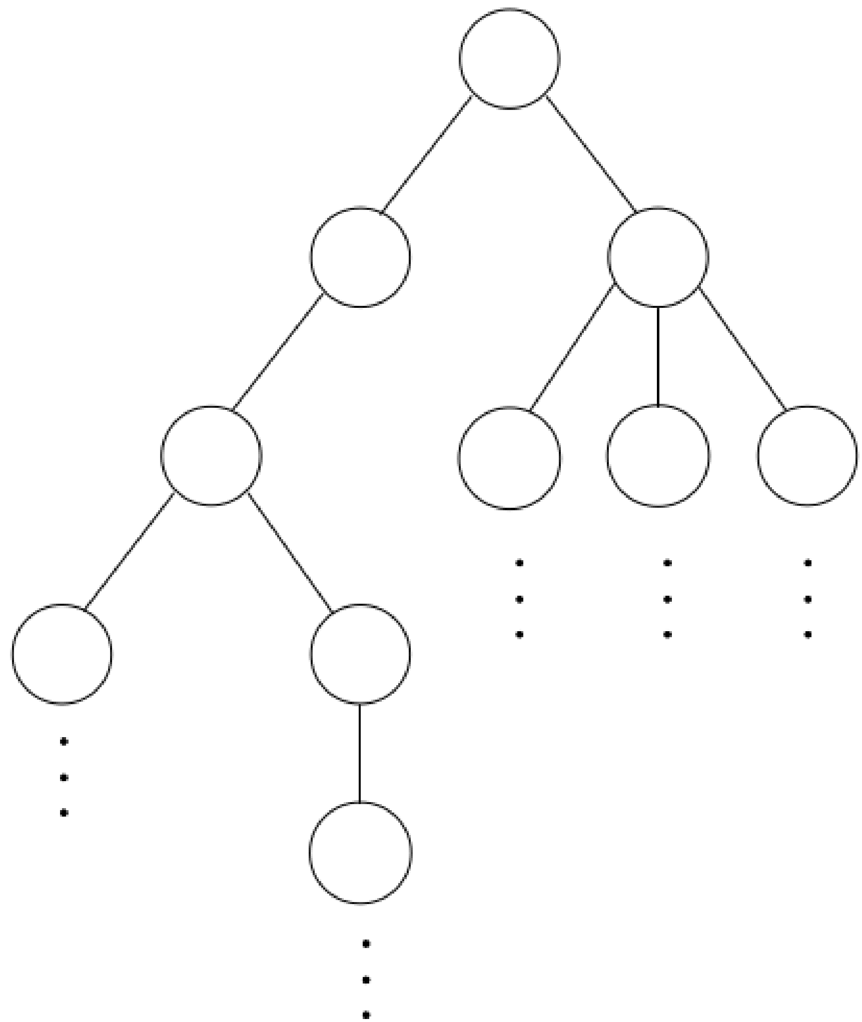
The idea of $\phi AU \psi$: inevitably ϕ until ψ



Fill in your answer in class.

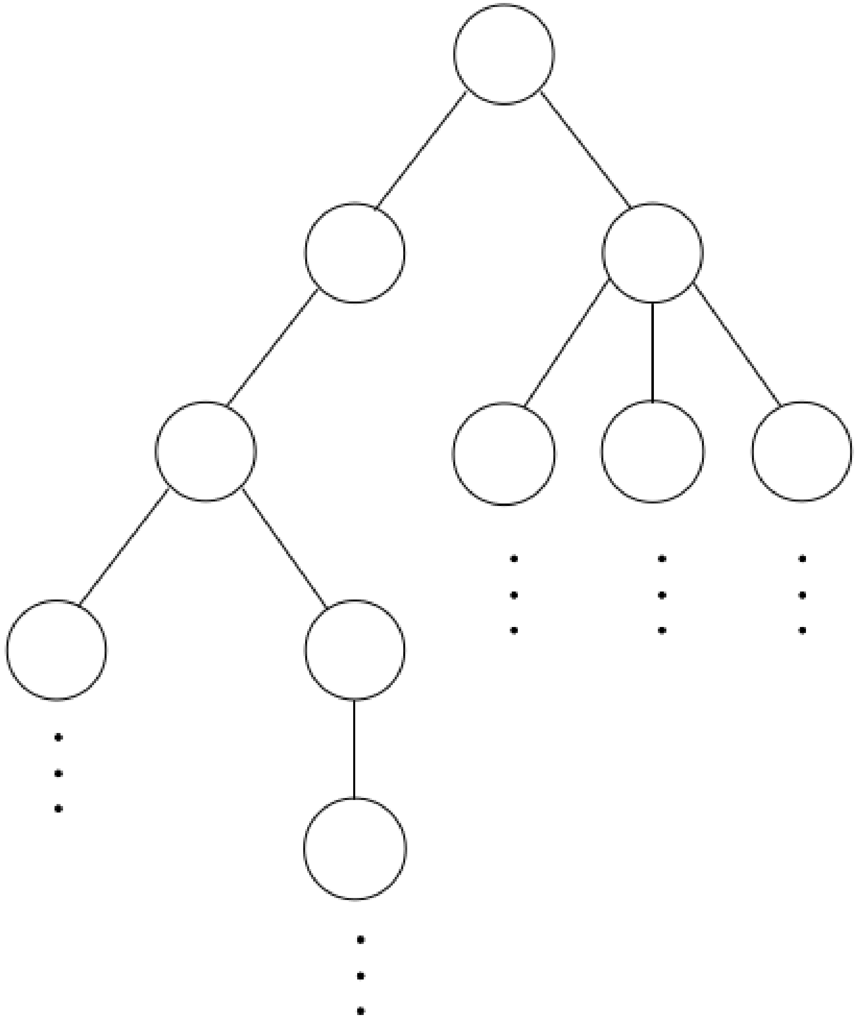
Computation Tree Logic Semantics

The idea of $\phi EU\psi$: possibly ϕ until ψ



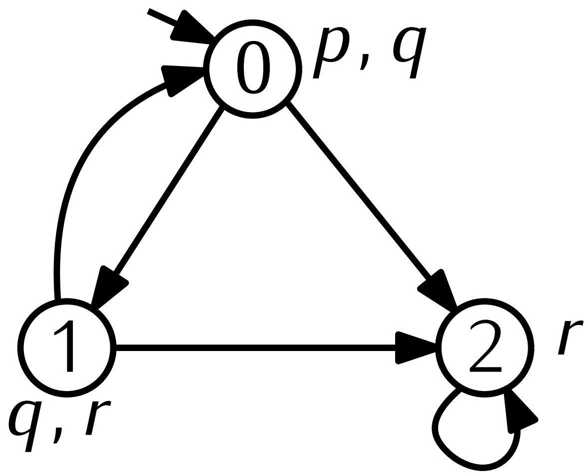
Computation Tree Logic Semantics

The idea of $\phi EU \psi$: possibly ϕ until ψ

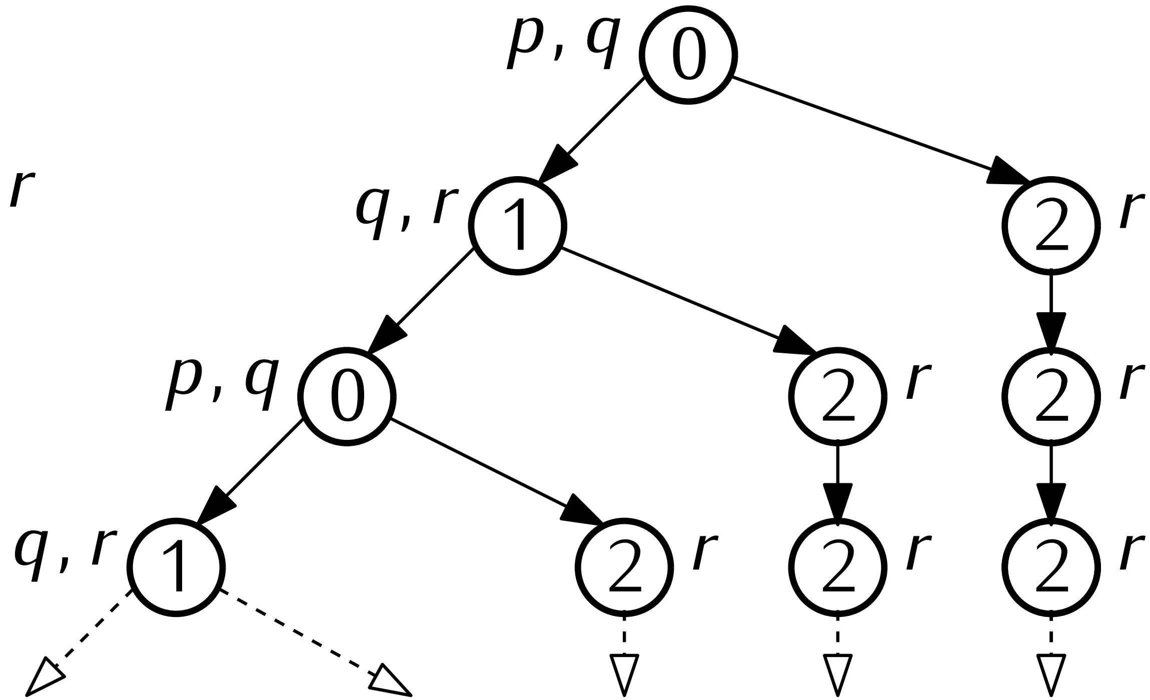


Fill in your answer in class.

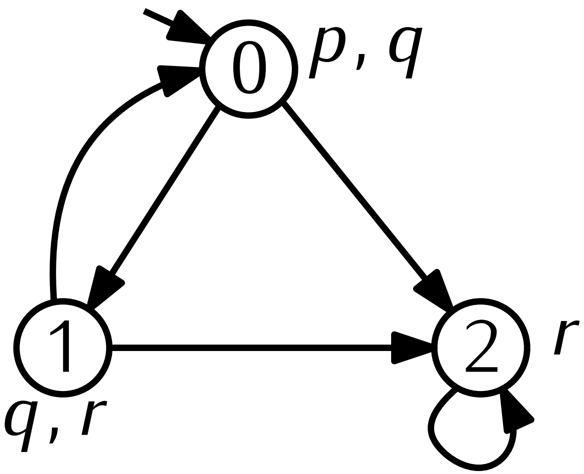
Computation Tree Logic Example Properties



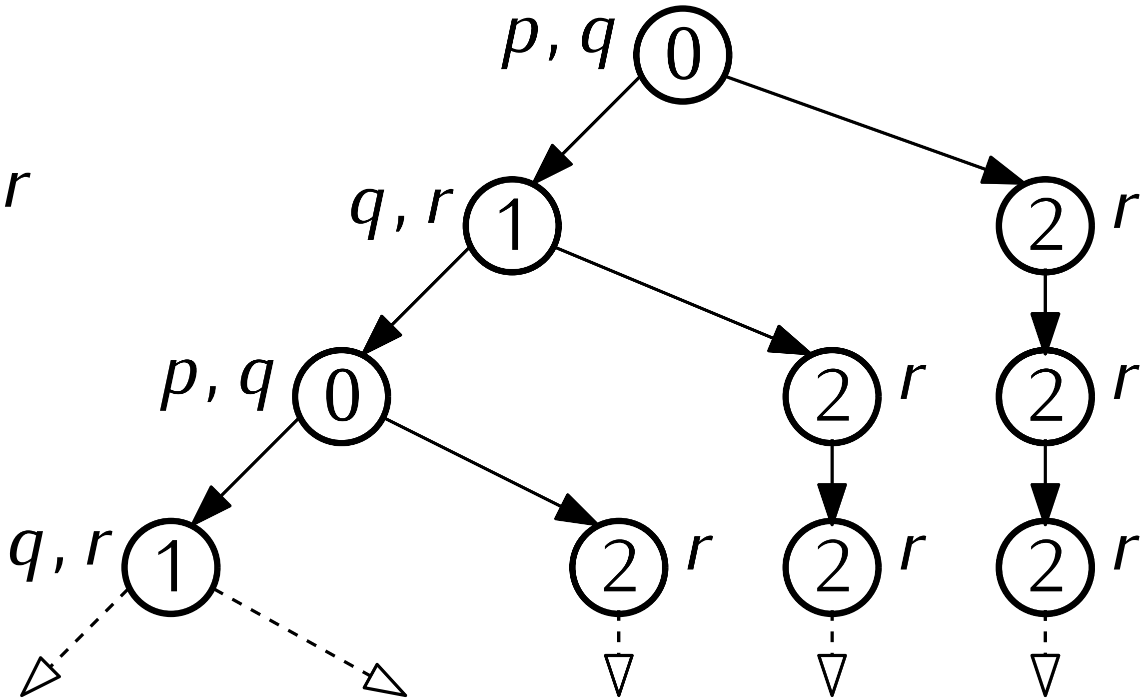
Computation tree for $\mathcal{M}$



Computation Tree Logic Example Properties

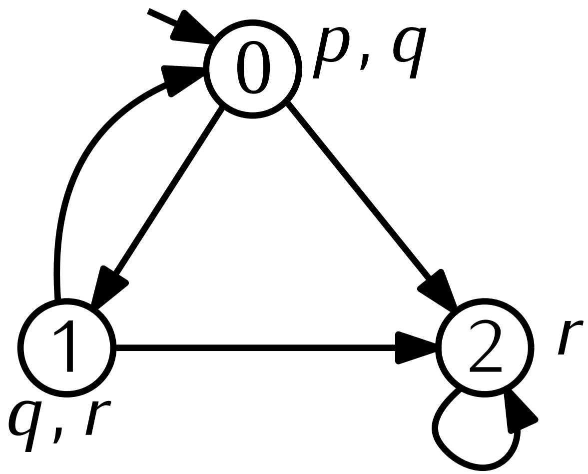


Computation tree for $\mathcal{M}$

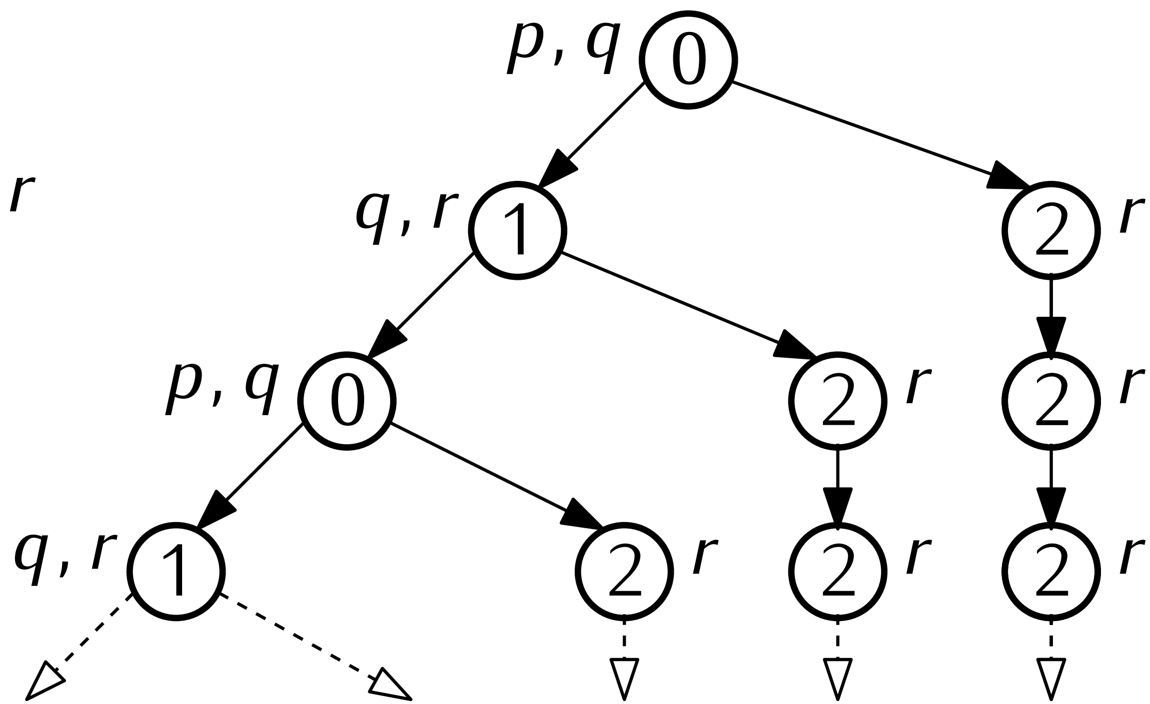


$\mathcal{M} \models p \wedge q$?

Computation Tree Logic Example Properties



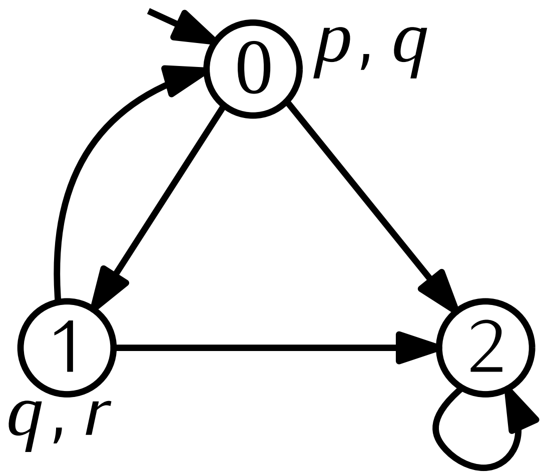
Computation tree for $\mathcal{M}$



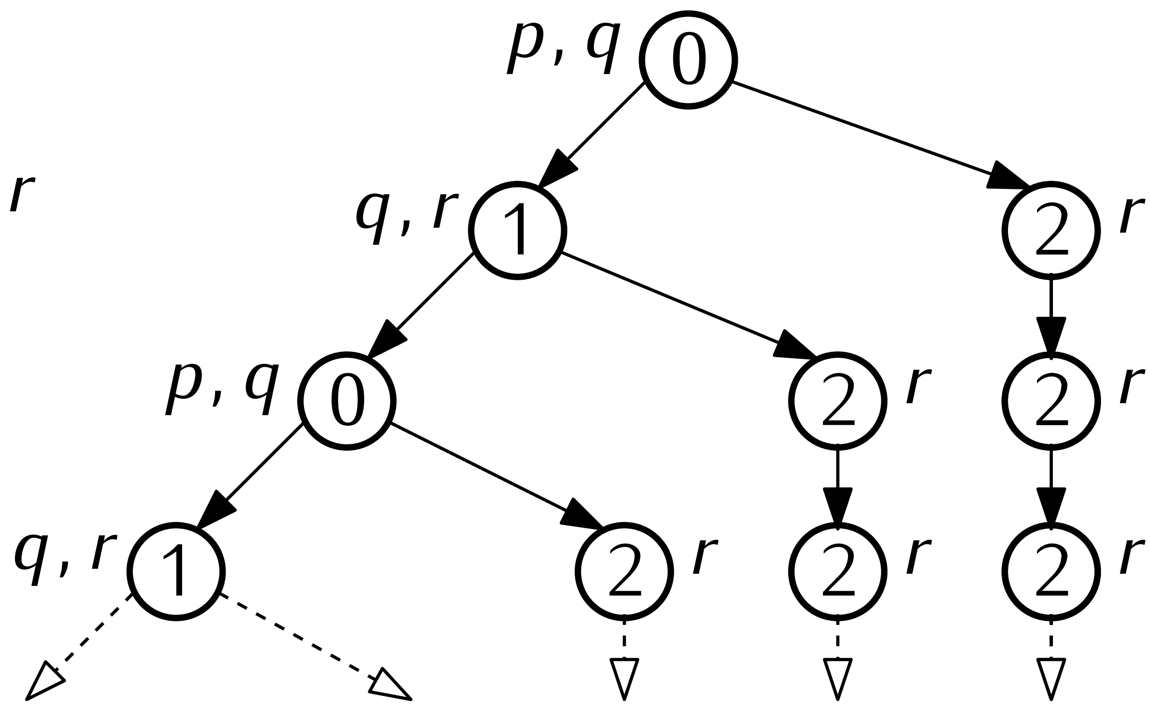
$\mathcal{M} \models p \wedge q$?

$\mathcal{M} \models \neg r$?

Computation Tree Logic Example Properties



Computation tree for $\mathcal{M}$

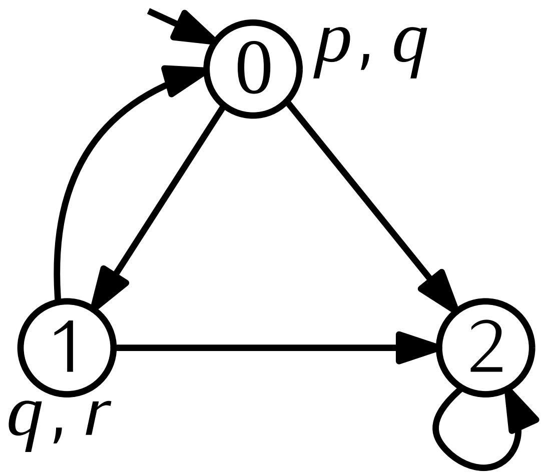


$\mathcal{M} \models p \wedge q$?

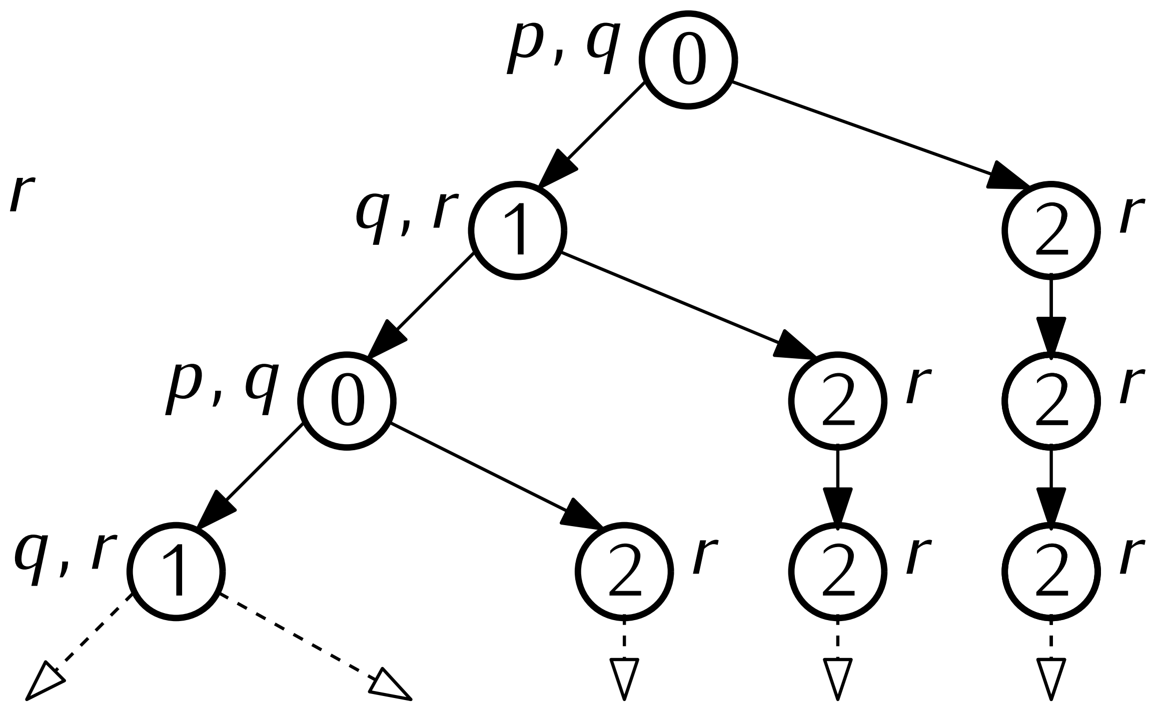
$\mathcal{M} \models \neg r$?

$\mathcal{M} \models EX(q \wedge r)$?

Computation Tree Logic Example Properties



Computation tree for $\mathcal{M}$



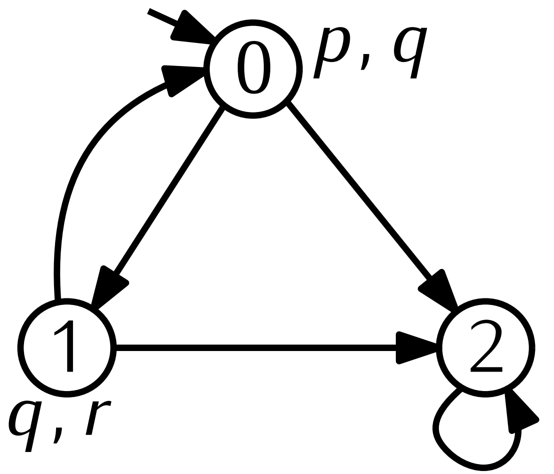
$\mathcal{M} \models p \wedge q$?

$\mathcal{M} \models \neg r$?

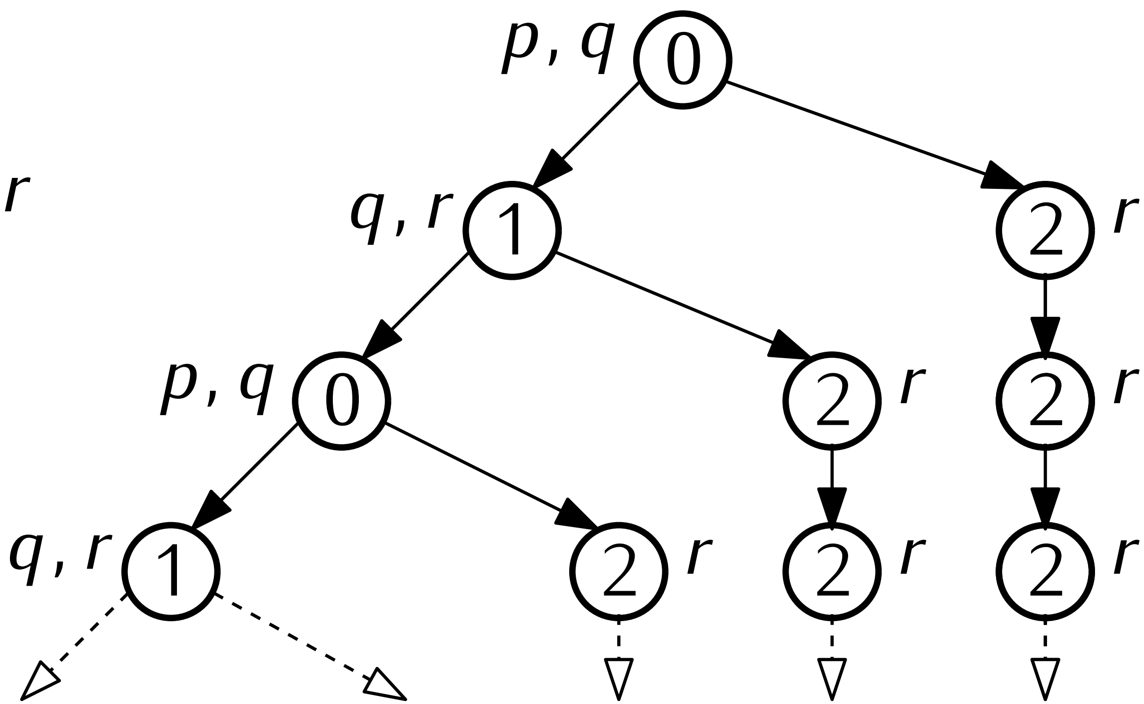
$\mathcal{M} \models EX(q \wedge r)$?

$\mathcal{M} \models AX(q \wedge r)$?

Computation Tree Logic Example Properties



Computation tree for $\mathcal{M}$



$\mathcal{M} \models p \wedge q$?

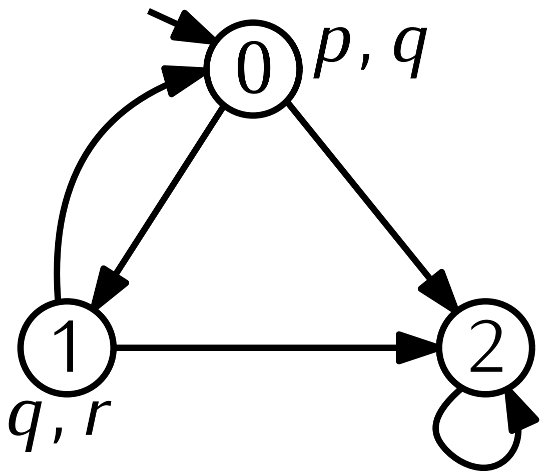
$\mathcal{M} \models \neg r$?

$\mathcal{M} \models EX(q \wedge r)$?

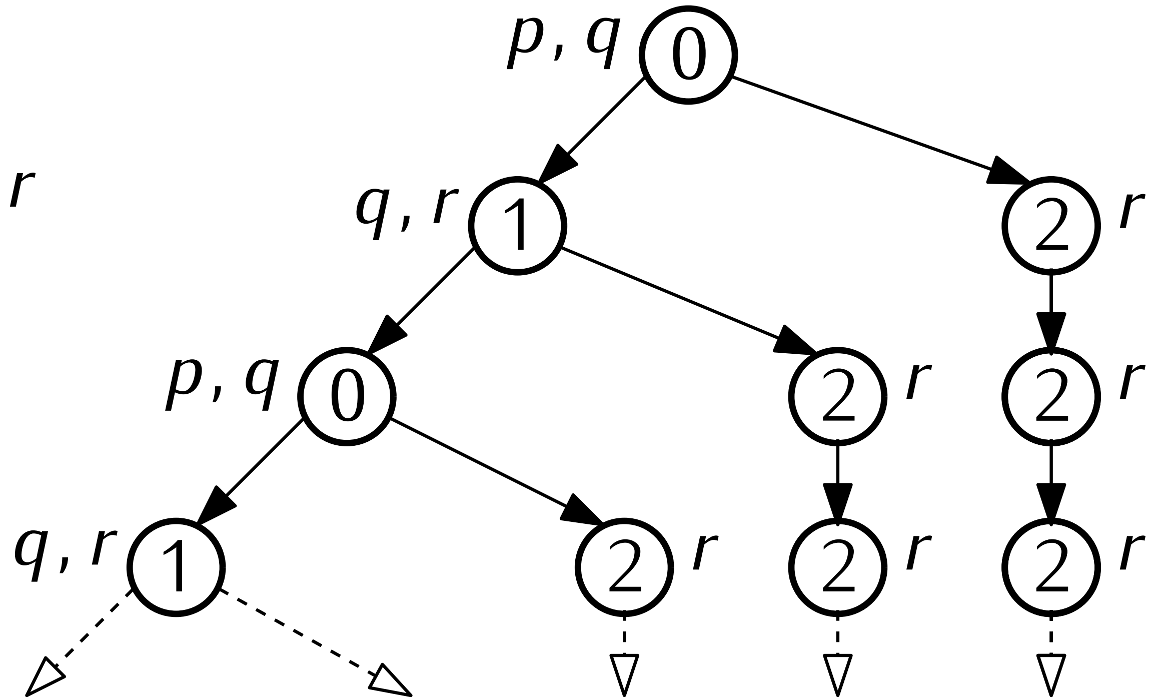
$\mathcal{M} \models AX(q \wedge r)$?

$\mathcal{M} \models \neg AX(q \wedge r)$?

Computation Tree Logic Example Properties



Computation tree for $\mathcal{M}$



$\mathcal{M} \models p \wedge q$?

$\mathcal{M} \models \neg r$?

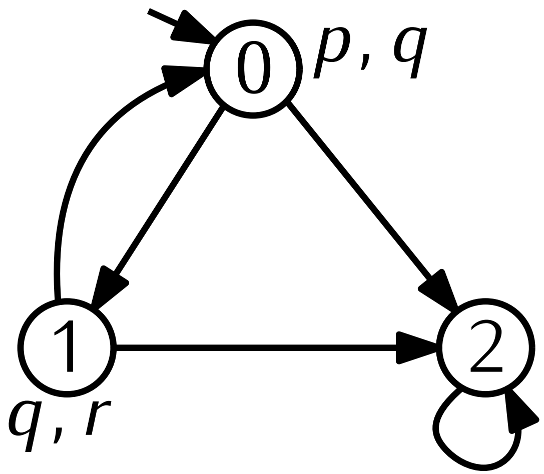
$\mathcal{M} \models EX(q \wedge r)$?

$\mathcal{M} \models AX(q \wedge r)$?

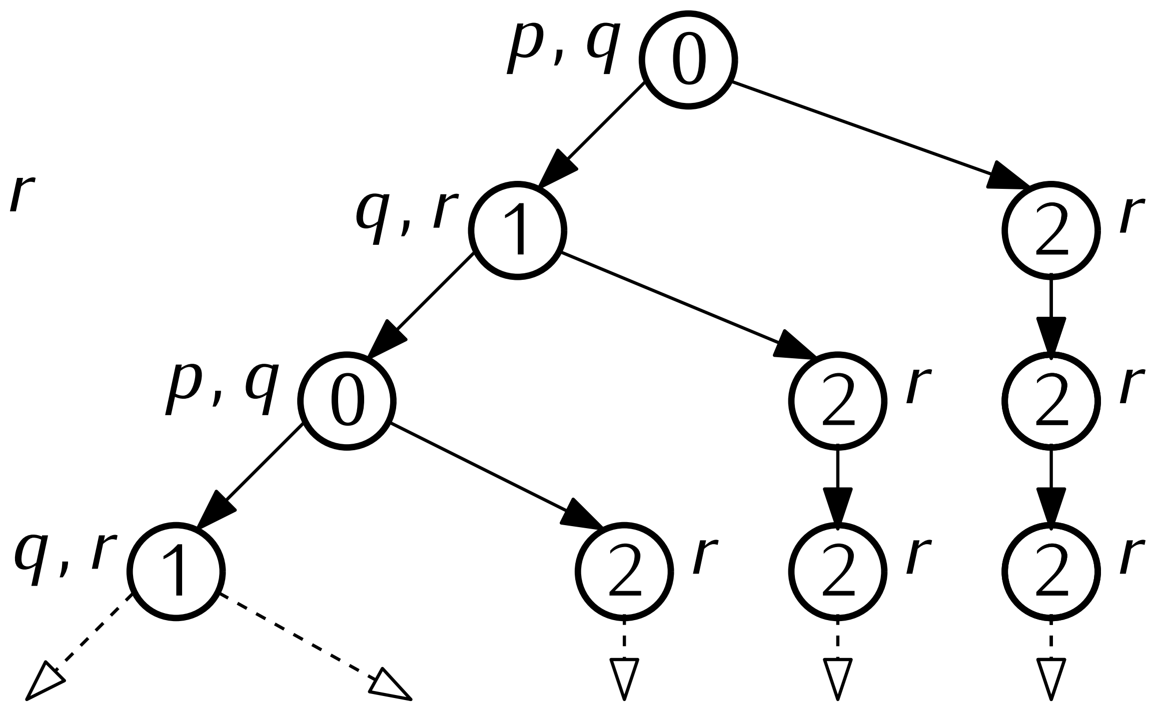
$\mathcal{M} \models \neg AX(q \wedge r)$?

$\mathcal{M} \models \neg EF(p \wedge r)$?

Computation Tree Logic Example Properties



Computation tree for $\mathcal{M}$



$\mathcal{M} \models p \wedge q$?

$\mathcal{M} \models \neg r$?

$\mathcal{M} \models EX(q \wedge r)$?

$\mathcal{M} \models AX(q \wedge r)$?

$\mathcal{M} \models \neg AX(q \wedge r)$?

$\mathcal{M} \models \neg EF(p \wedge r)$?

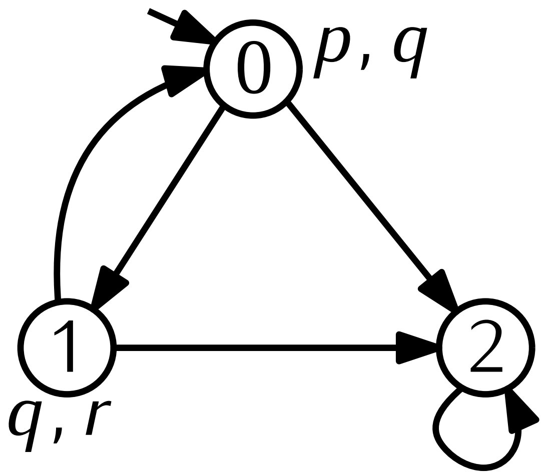
$\mathcal{M} \models EG\neg r$?

$\mathcal{M} \models AFq$?

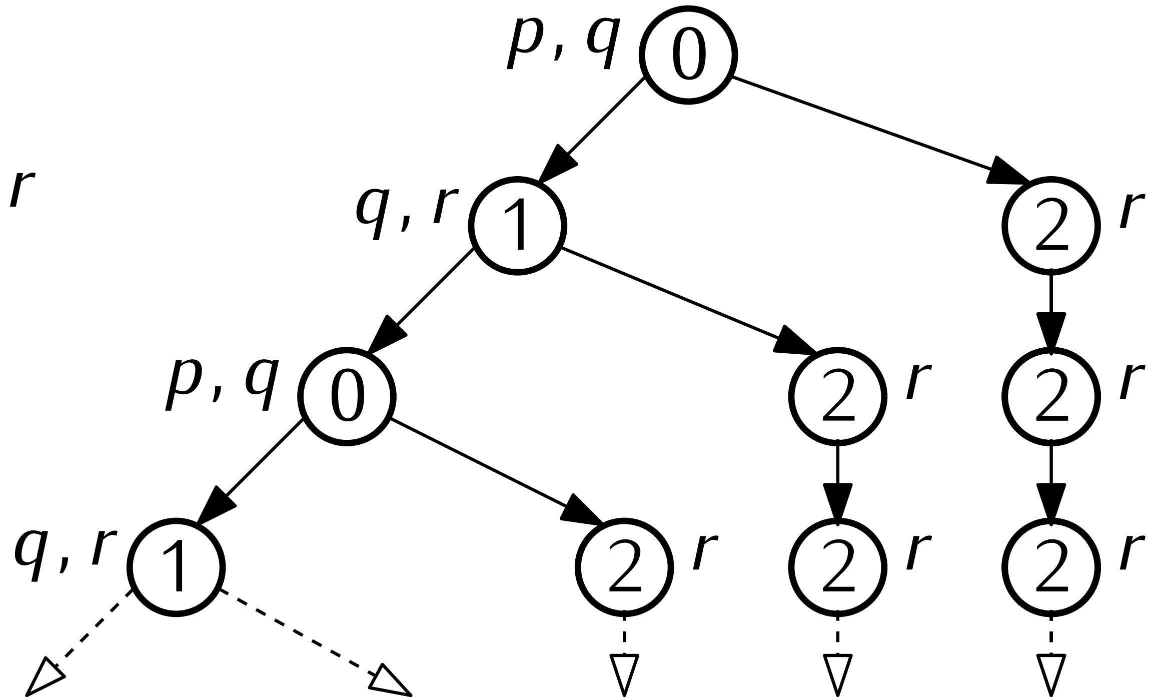
$\mathcal{M} \models p\ AU\ r$?

$\mathcal{M} \models \neg(p \wedge q)\ EU\ r$?

Computation Tree Logic Example Properties



Computation tree for $\mathcal{M}$



$\mathcal{M} \models p \wedge q$?

$\mathcal{M} \models \neg r$?

$\mathcal{M} \models EX(q \wedge r)$?

$\mathcal{M} \models AX(q \wedge r)$?

$\mathcal{M} \models \neg AX(q \wedge r)$?

$\mathcal{M} \models \neg EF(p \wedge r)$?

$\mathcal{M} \models EG \neg r$?

$\mathcal{M} \models AFq$?

$\mathcal{M} \models p \; AU \; r$?

$\mathcal{M} \models \neg(p \wedge q) \; EU \; r$?