UvA

Multi-Agent Systems

Computation-Tree Logic

Mina Young Pedersen
m.y.pedersen@uva.nl

Fall 2025

What is Computation-Tree Logic?

- Computation-Tree Logic (CTL) is a temporal logic
    - Used to reason about time in a **state transition system**
    - Which we can think of as models of a discrete (computing) system
- In CTL, time is *branching* in contrast to *linear* time
    - Its model of time is a tree-like structure where the future is not determined
    - There are different *paths* in the future, any one of which might be the “actual” path

State Transition Systems

- Computation-Tree Logic formulas are evaluated on **state transition systems**
- As before, we define a set At of atoms

Definition (State Transition System)

A **(state) transition system** is a tuple $M = (S, \rightarrow, L)$, where:

- S is a non-empty domain (whose elements are generically called **states**);
- $\rightarrow \subseteq (S \times S)$ is a binary relation on S such that every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
- $L : S \rightarrow \mathscr{P}(\mathrm{At})$ is an atomic **labeling function**

Does it look familiar?

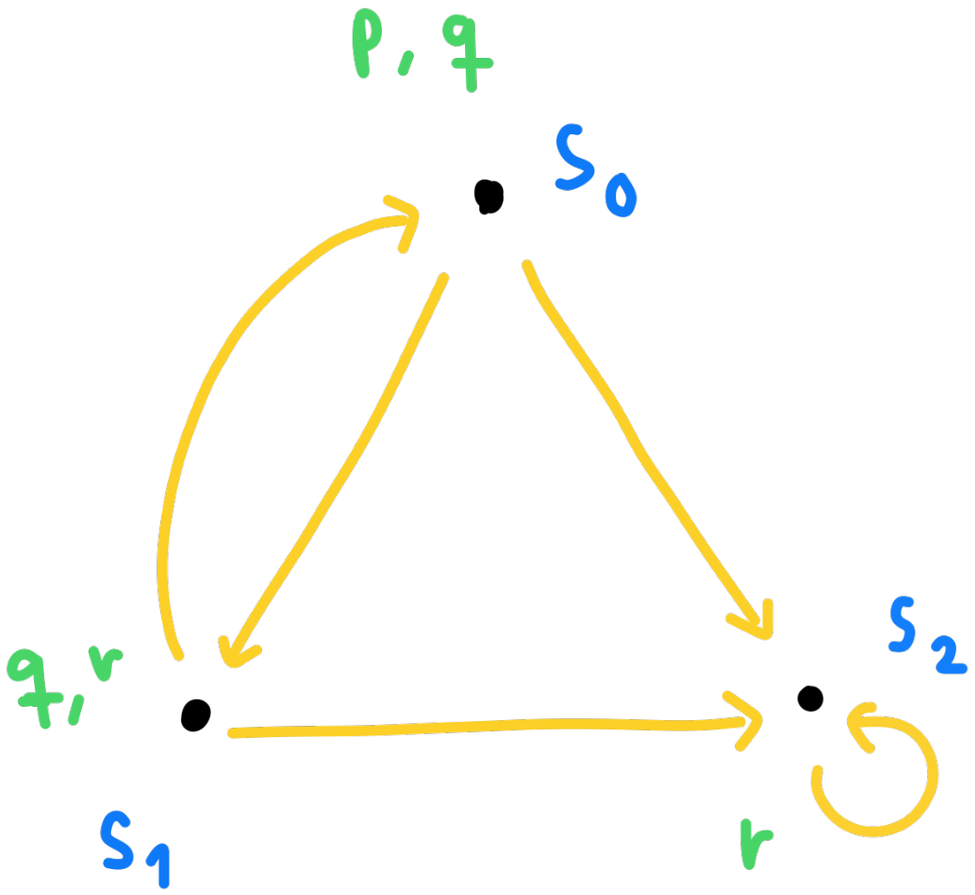
Also note that the labeling function is defined differently

State transition systems are *serial* relational models!

Note on State Transition Systems

- They are serial:
    - For every $s \in S$ there is some $s' \in S$ with $s \to s'$
    - No system can “deadlock”
    - However, you could model a deadlock with a reflexive arrow

Example



$$M = (S, \rightarrow, L)$$

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

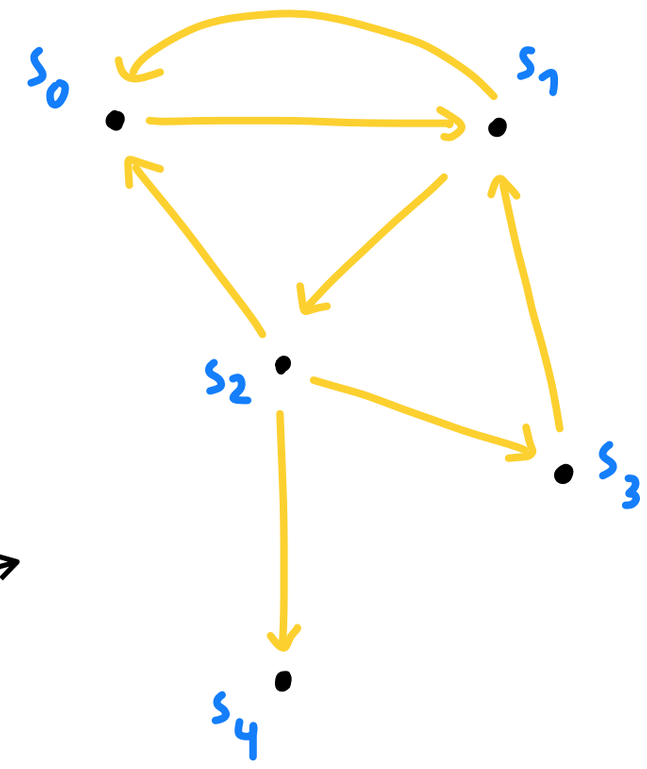
$$s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, \text{ etc.}$$

$$L(s_0) = \{p, q\}$$

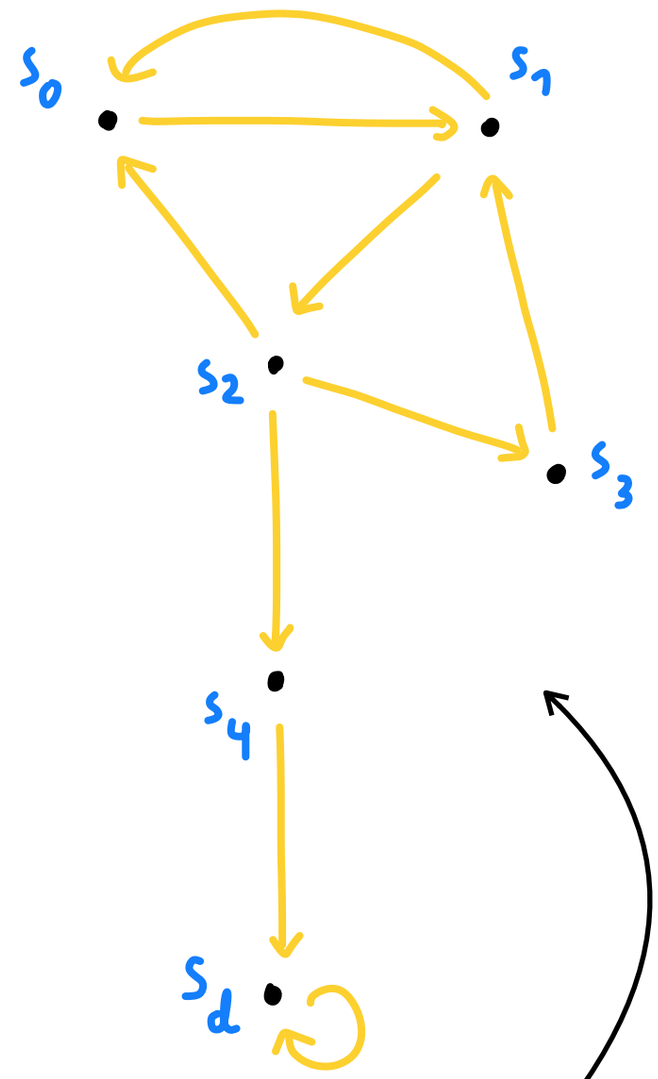
$$L(s_1) = \{q, r\}$$

$$L(s_2) = \{r\}$$

Example: Deadlock



Not a state transition system



But this one is

Paths

- To define the semantics of CTL, we need to specify what a **path** in a state transition system is

Definition (Path)

A path in a state transition system $M = (S, \rightarrow, L)$ is an infinite sequence of states $s_1, s_2, s_3, \ldots$ in S such that, for each $i \leq 1$, $s_i \rightarrow s_{i+1}$. We write the path as $s_1 \rightarrow s_2 \rightarrow \ldots$.

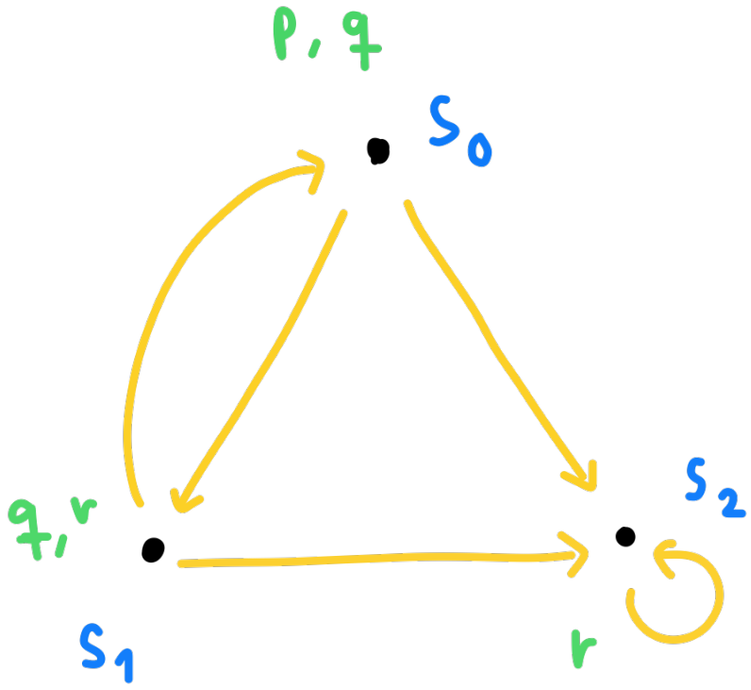
A path $\pi = s_1 \rightarrow s_2 \rightarrow \ldots$ represents a possible future of our system

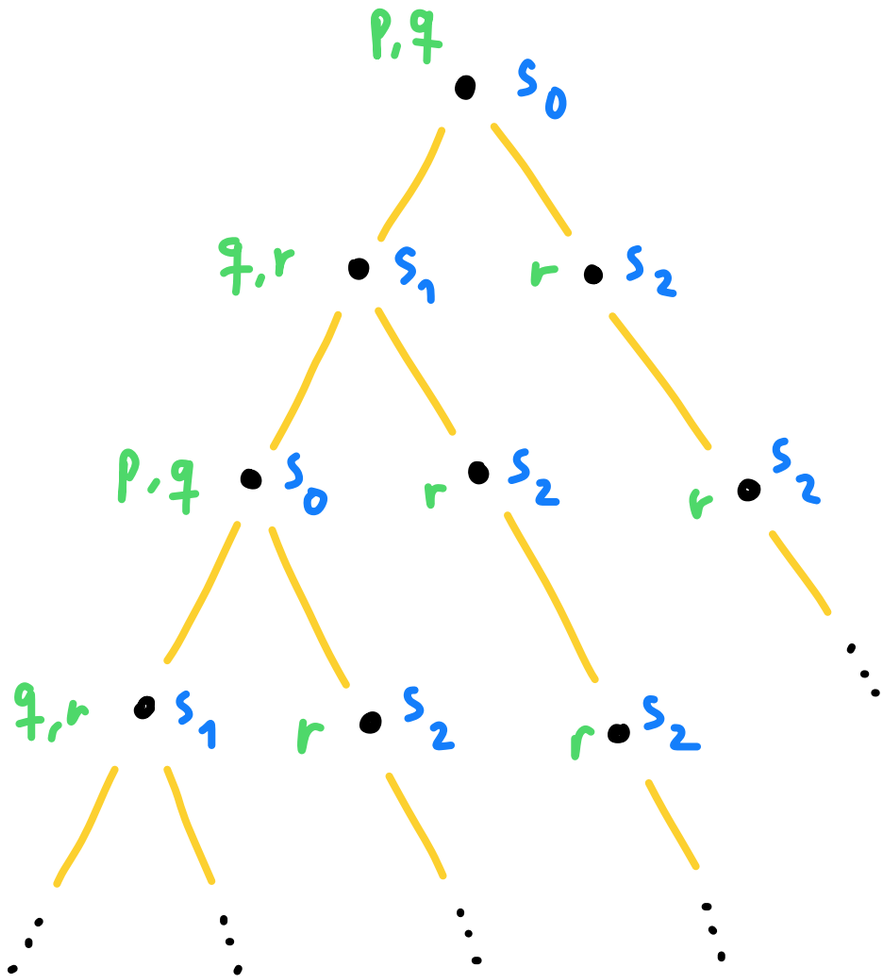
We write π^i for the suffix starting at s_i

For example, π^3 is $s_3 \rightarrow s_4 \rightarrow \ldots$

Unwinding

- It might be useful to visualize all possible computation paths from a given state s by **unwinding** the transition system to obtain an infinite computation tree





Language of CTL

- CTL includes temporal connectives *X*, *F*, *G*, *U*
    - *X* means “neXt state”
    - *F* means “some Future state”
    - *G* means “all future states (Globally)”
    - *U* means “Until”
- CTL also includes quantifiers *A* and *E*
    - Which express “along All paths” and “along at least (there Exists) one path”, respectively

Language of CTL

- Here are some examples of statements we can write in CTL:
    - There is a reachable state satisfying *q*: EFq
    - From all reachable states satisfying *p*, it is possible to maintain *p* continuously until reaching a state satisfying *q*: $AG(p \rightarrow E[pUq])$
    - Whenever a state satisfying *p* is reached, the system can exhibit *q* continuously forevermore: $AG(p \rightarrow EGq)$
    - There is a reachable state from which all reachable states satisfy *p*: $EFAGp$

Language of CTL: Formally

- We can now define the language of CTL properly

Definition (Language of CTL)

Let At be a set of atomic propositions. Formulas ϕ, ψ of the **CTL language** are given inductively as follows:

$$\phi, \psi ::= \text{At} \mid \neg\phi \mid (\phi \wedge \psi) \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U \psi] \mid E[\phi U \psi]$$

We define ($\vee$, $\rightarrow$, $\leftrightarrow$, $\top$, $\bot$) **as standard.**

Binding Priorities

- To avoid writing parenthesis, we assume a convention for **binding priorities** of the connectives:
    - The unary connectives bind most tightly (consisting of ¬ and the temporal connectives AG, EG, AF, EF, AX, EX)
- Next in the order come $\wedge$ and $\vee$
- After that come $\rightarrow$, AU and EU

Language of CTL

$$\phi, \psi ::= \text{At} \mid \neg\phi \mid (\phi \wedge \psi) \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid$$
$$AG\phi \mid EG\phi \mid A[\phi U \psi] \mid E[\phi U \psi]$$

Are these (well-formed) formulas of the CTL language?

$AG(q \rightarrow EGr)$ ✓ (not the same as $AGq \rightarrow EGr$)

$A[pU\,EFr]$ ✓

$A\,\neg G\,\neg p$ ✗

$EF(rUq)$ ✗

$EF\,E[rUq]$ ✓

$F[rUq]$ ✗

$A[(rUq) \wedge (pUr)]$ ✗

$E[A[p_1Up_2]Up_3]$ ✓

$EF\,EGp \rightarrow AFr$ ✓

(not the same as $EF(EGp \rightarrow AFr)$ or $EF\,EG(p \rightarrow AFr)$)

Semantics of CTL

Definition (Semantic Interpretation/Truth, pt.1)

Let $M = (S, \rightarrow, L)$ be a state transition system and $s \in S$ be a state. **Truth of a CTL formula ϕ at state s in M, written $M, s \Vdash \phi$, is defined inductively as follows:**

$M, s \Vdash p$ iff $p \in L(s)$

$M, s \Vdash \neg\phi$ iff $M, s \nVdash \phi$

$M, s \Vdash \phi \wedge \psi$ iff $M, s \Vdash \phi$ and $M, s \Vdash \psi$

$M, s \Vdash AX\phi$ iff for all s_1 such that $s \rightarrow s_1$ we have $M, s_1 \Vdash \phi$

Thus, AX says "in every next state"

$M, s \Vdash EX\phi$ iff for some s_1 such that $s \rightarrow s_1$ we have $M, s_1 \Vdash \phi$

Thus, EX says "in some next state"

$M, s \Vdash AG\phi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where $s_1 = s$, and for all s_i along the path, we have $M, s_i \Vdash \phi$

Intuitively, for all paths beginning in s, the property ϕ holds globally

Semantics of CTL

Definition (Semantic Interpretation/Truth, pt.2)

$M, s \Vdash EG\phi$ iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where $s_1 = s$, and for all s_i along the path, we have $M, s_i \Vdash \phi$

Intuitively, there exists a path beginning in s such that the property ϕ holds globally along that path

$M, s \Vdash AF\phi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where $s_1 = s$, there is some s_i such that $M, s_i \Vdash \phi$

Intuitively, for all paths beginning in s, there will be some future state where ϕ holds

$M, s \Vdash EF\phi$ iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where $s_1 = s$, and for some s_i along the path, we have $M, s_i \Vdash \phi$

Intuitively, there exists a path beginning in s, such that ϕ holds in some future state

Semantics of CTL

Definition (Semantic Interpretation/Truth, pt.3)

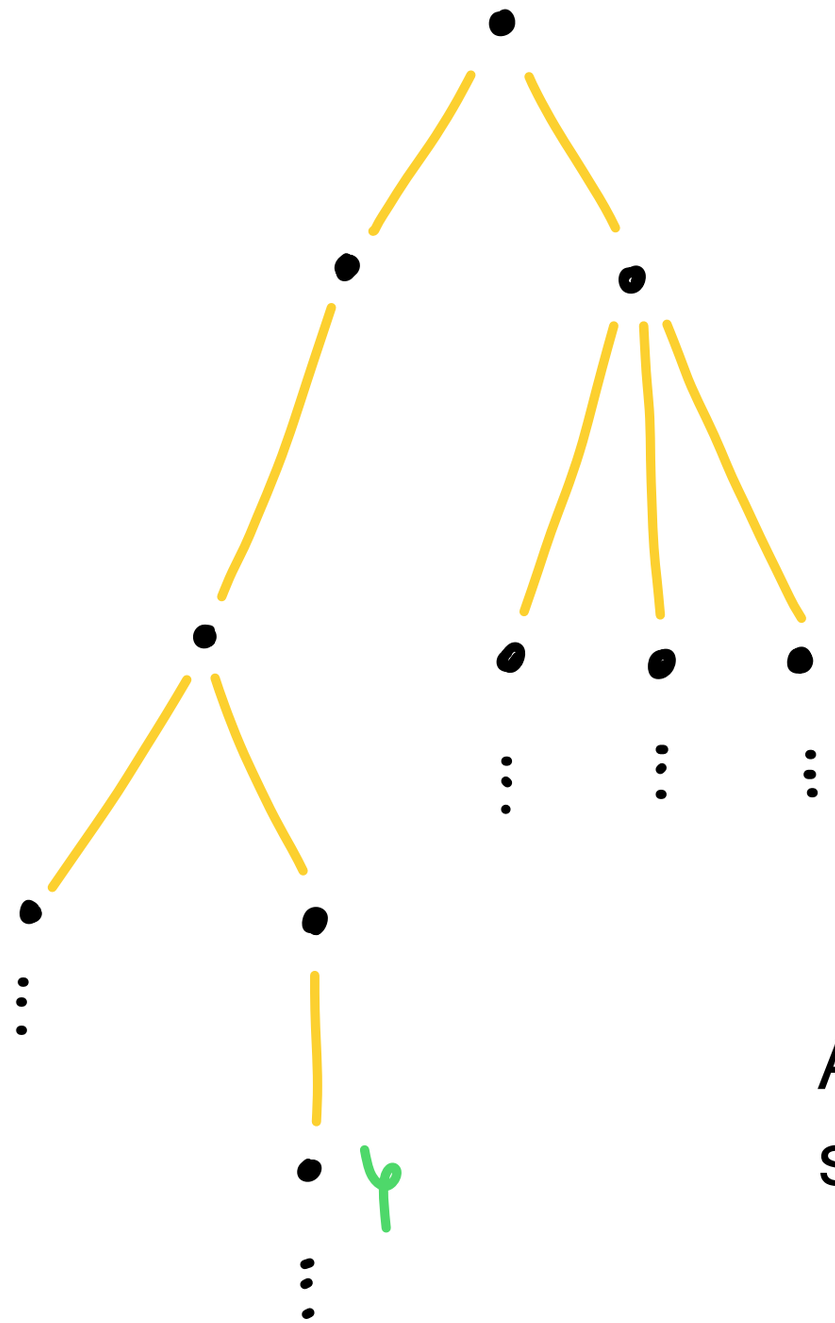
$M, s \Vdash A[\phi_1 U \phi_2]$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where $s_1 = s$, there is some s_i along the path, such that $M, s_i \Vdash \phi_2$, and, for each $j < i$, we have $M, s_j \Vdash \phi_1$

Intuitively, all paths beginning in s satisfy that ϕ_1 until ϕ_2 holds on it

$M, s \Vdash E[\phi_1 U \phi_2]$ iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where $s_1 = s$, and in that path there is some s_i along the path, such that $M, s_i \Vdash \phi_2$, and, for each $j < i$, we have $M, s_j \Vdash \phi_1$

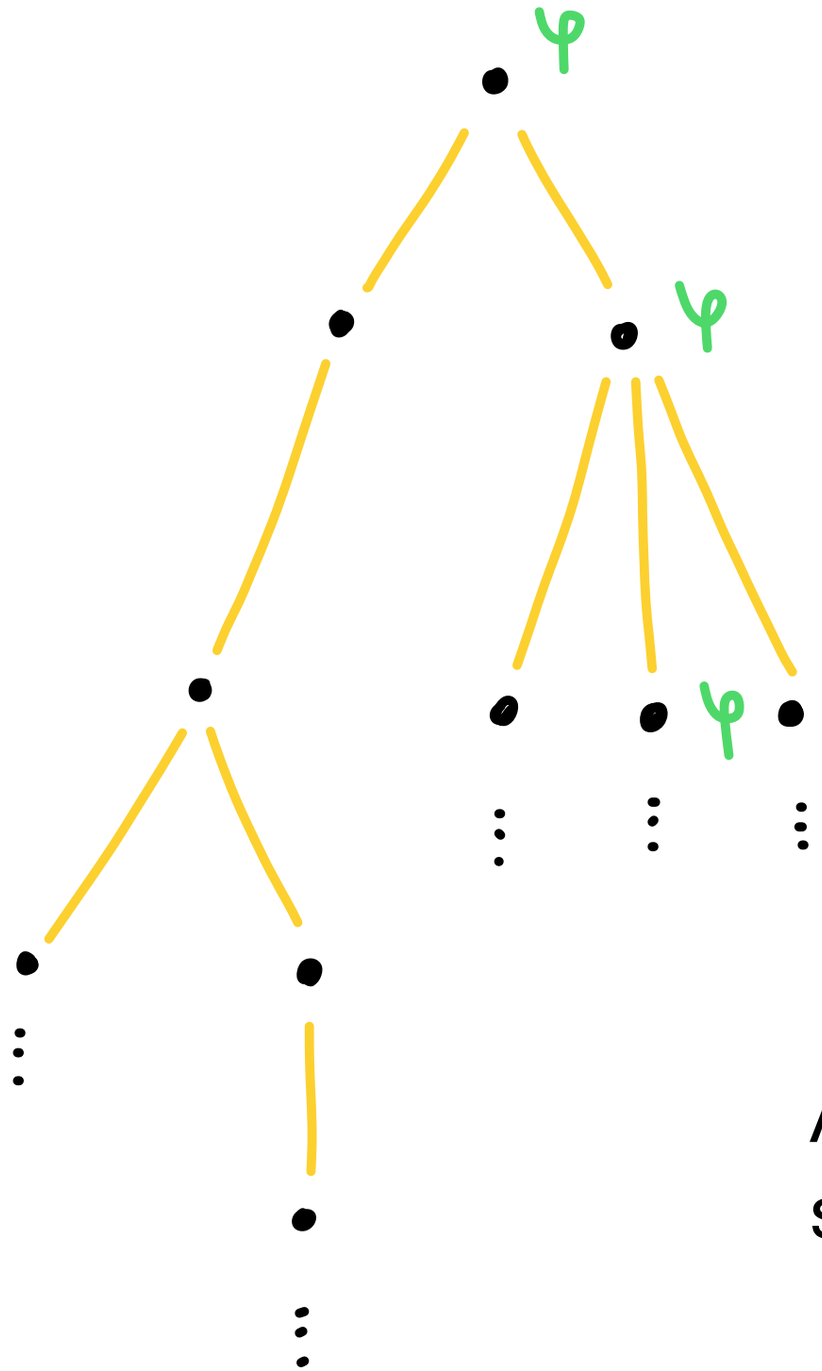
Intuitively, there exists a path beginning in s, such that ϕ_1 until ϕ_2 holds on it

Example



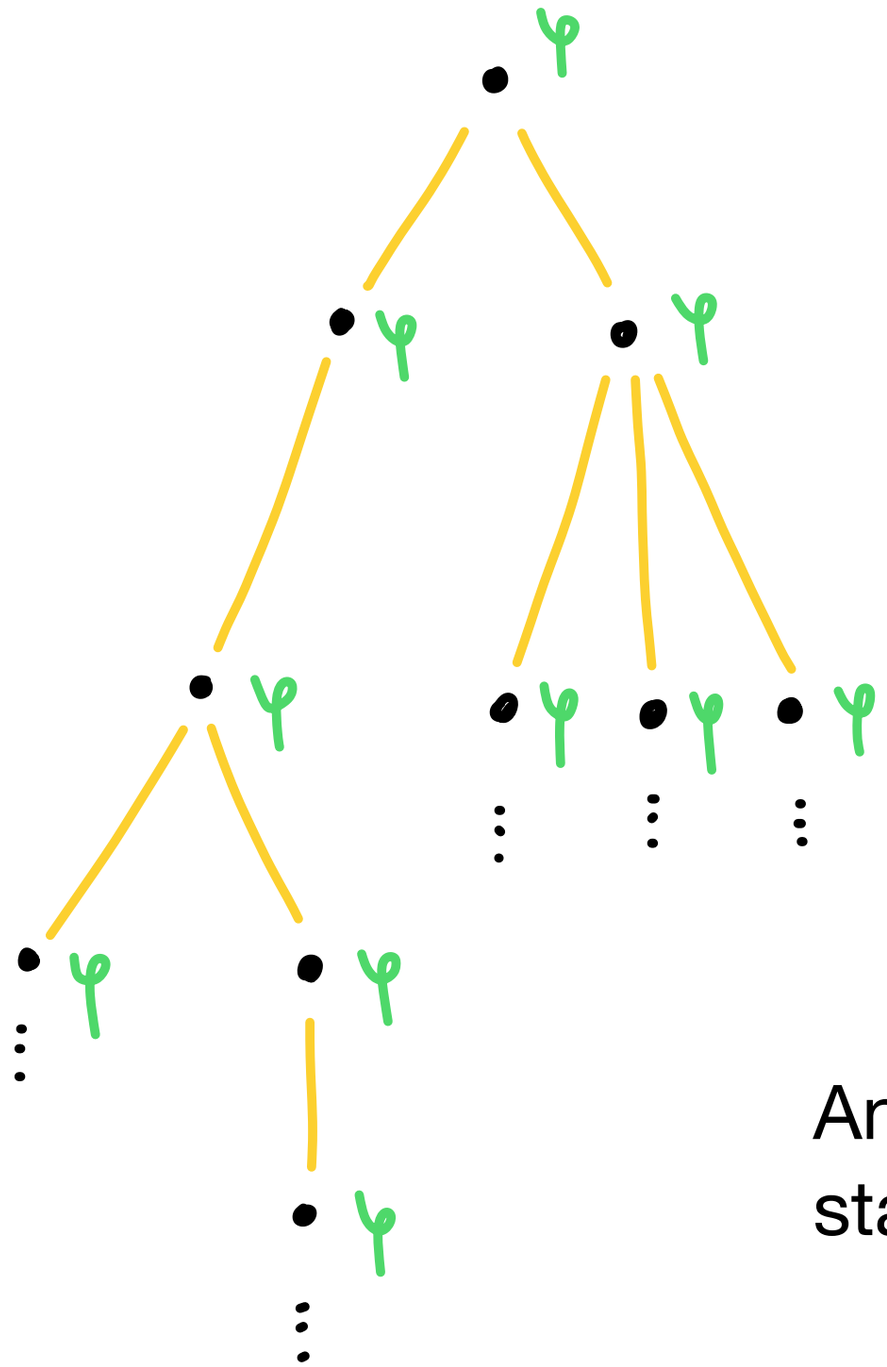
An (unwinded) system whose starting state satisfies $EF\phi$

Example



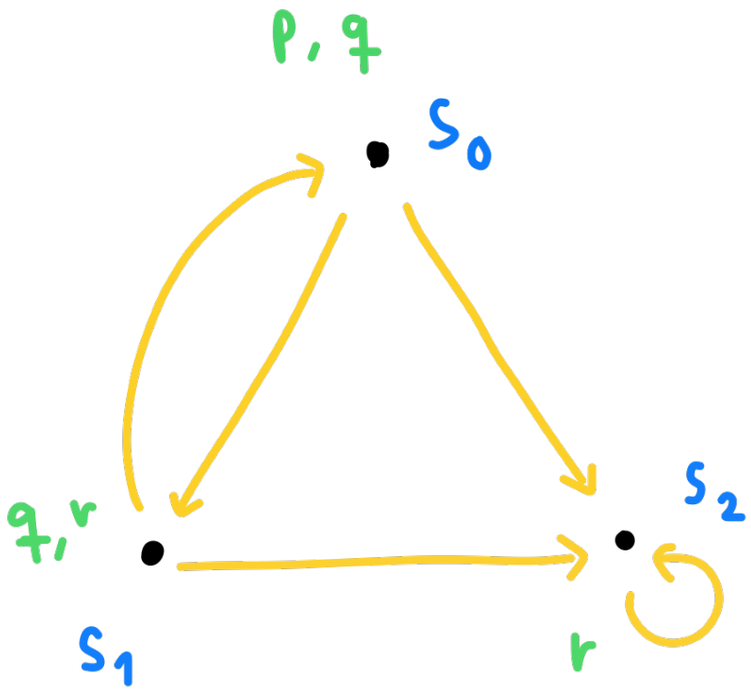
An (unwinded) system whose starting state satisfies $EG\phi$

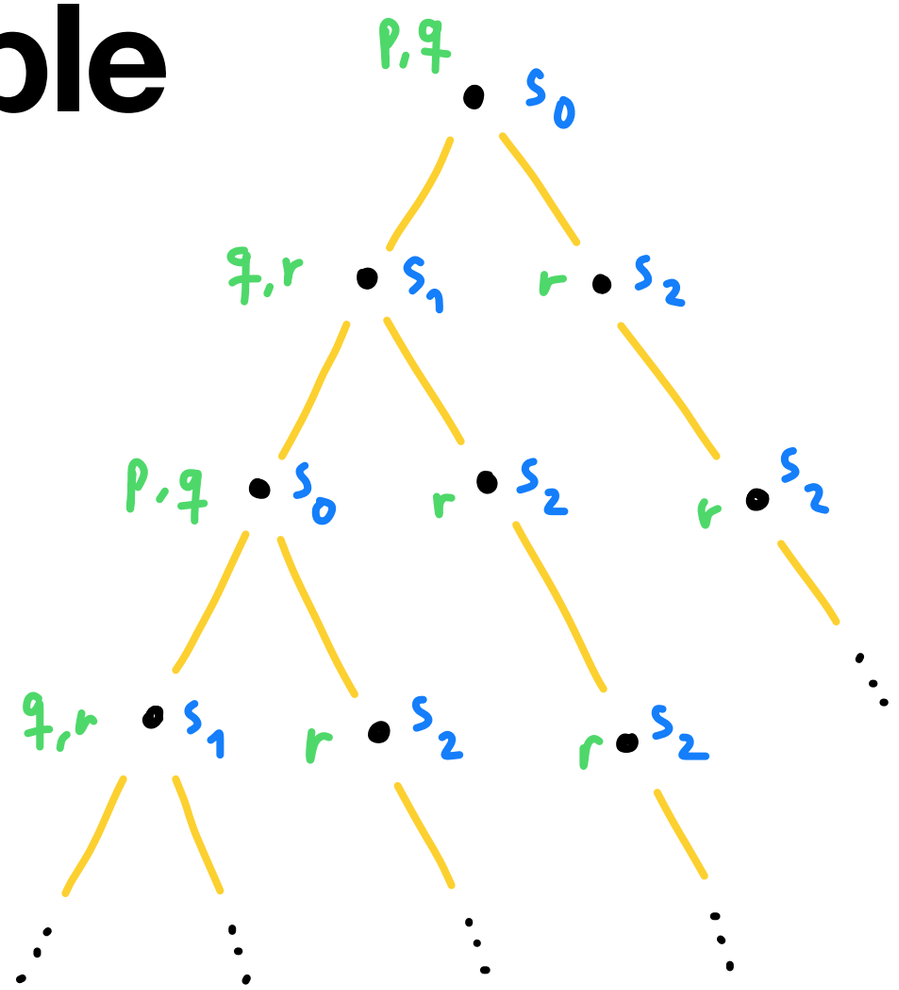
Example



An (unwinded) system whose starting state satisfies $AG\phi$

Return to the Example





Which formulas are true at (M, s_0)?

$p \wedge q$ ✓

r ✗

$\top$ ✓

$EX(q \wedge r)$ ✓

$AX(q \wedge r)$ ✗

$EF(p \wedge r)$ ✗

$EG\, r$ ✗

$AF\, r$ ✓

$E[(p \wedge q) U\, r]$ ✓

$A[p\, U\, r]$ ✓

$AG(p \vee q \vee r \rightarrow EF\, EG\, r)$ ✓

Practical Patterns of Specifications

- Suppose we have a set of atomic descriptions which include words such as “start” and “request”

- Here are some examples of useful things we can say in the CTL-language:

“It is possible to get to a state where *started* holds, but *ready* doesn’t”

$$EF(\text{started} \wedge \neg \text{ready})$$

“For any state, if a *request* (of some resource) occurs, then it will eventually be acknowledged”

$$AG(\text{request} \rightarrow AF\,\text{acknowledged})$$

“An upwards traveling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor”

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

Important Equivalences Between CTL Formulas

- There turns out to be some important equivalences between formulas in CTL
    - Like in, e.g., the basic modal logic

Definition (Semantic Equivalence)

Two (CTL) formulas ϕ and ψ are said to be **semantically equivalent** if any state in any state transition system which satisfies one of them also satisfies the other; we denote this by $\phi \equiv \psi$ (or $\phi := \psi$).

Adequate Sets of CTL Connectives

- We have some redundancy among the CTL connectives:

$$AX\phi \equiv \neg EX \neg \phi$$

$$AG\phi \equiv \neg EF \neg \phi$$

$$EG\phi \equiv \neg AF \neg \phi$$

$$AF\phi \equiv A[\top \, U \, \phi]$$

$$EF\phi \equiv E[\top \, U \, \phi]$$

So, AU, EU and EX form an adequate set of connectives

Adequate Sets of CTL Connectives

However, EG, EU and EX also form an adequate set of connectives, because:

$$A[\phi U \psi] \equiv \neg(E[\neg\psi U(\neg\phi \wedge \neg\psi)] \vee EG\psi)$$

More generally, we have:

Theorem

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

Some Other Equivalences Between CTL Formulas

$$AG\phi \equiv \phi \wedge AX\, AG\phi$$

$$EG\phi \equiv \phi \wedge EX\, EG\phi$$

$$AF\phi \equiv \phi \vee AX\, AF\phi$$

$$EF\phi \equiv \phi \vee EX\, EF\phi$$

$$A[\phi U \psi] \equiv \psi \vee (\phi \wedge AX\, A[\phi U \psi])$$

$$E[\phi U \psi] \equiv \psi \vee (\phi \wedge EX\, E[\phi U \psi])$$

Additional Reading Material

- For additional reading material, I advise:
    - Michael Huth & Mark Ryan: *Logic in Computer Science.* Cambridge University Press, 2004.