Temporal Logics

CS560: Reasoning About Programs

Roopsha Samanta
PURDUE UNIVERSITY

Based on slides by Georg Weissenbacher

Roadmap

Previously

- Propositional logic and SAT Solving
- First-order logic, first-order theories and SMT Solving

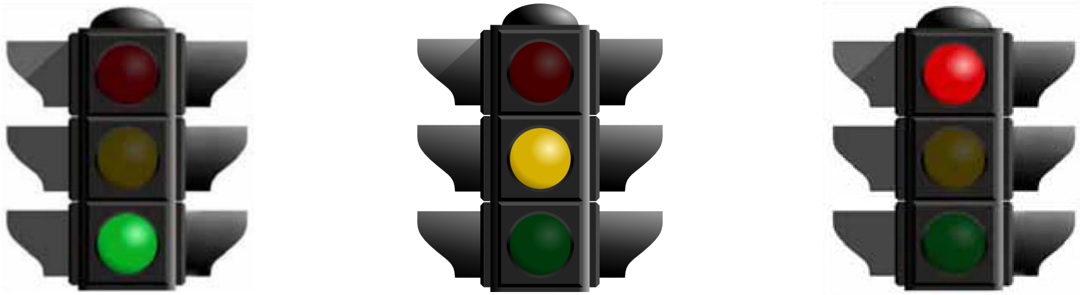
Today

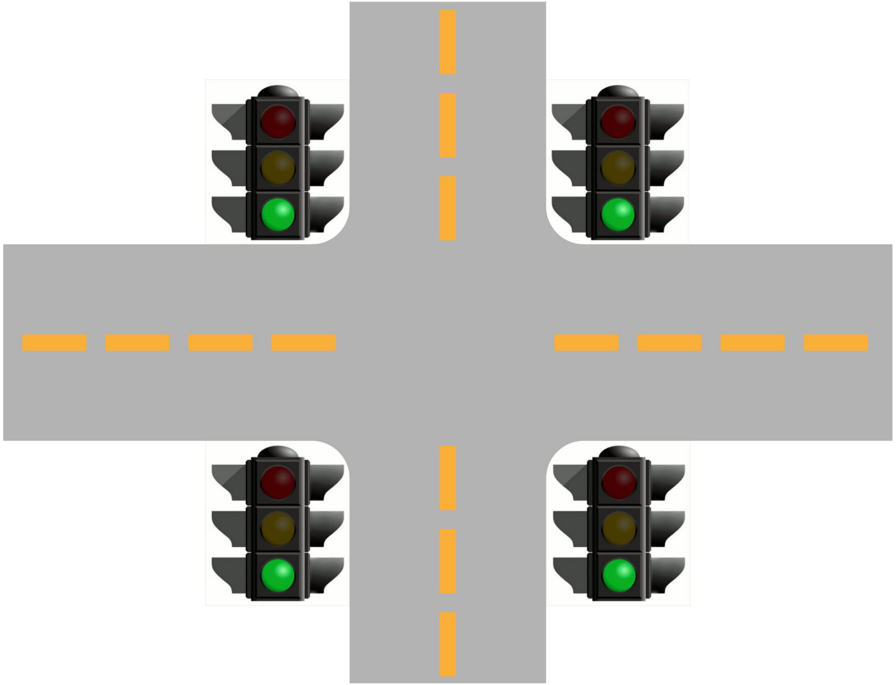
- Temporal logic!

Specifying Correctness for Ongoing Systems

A software system controlling traffic lights

Each traffic light in the system can be in one of three states





Consider a crossing with two traffic lights  $_1$ and  $_2$

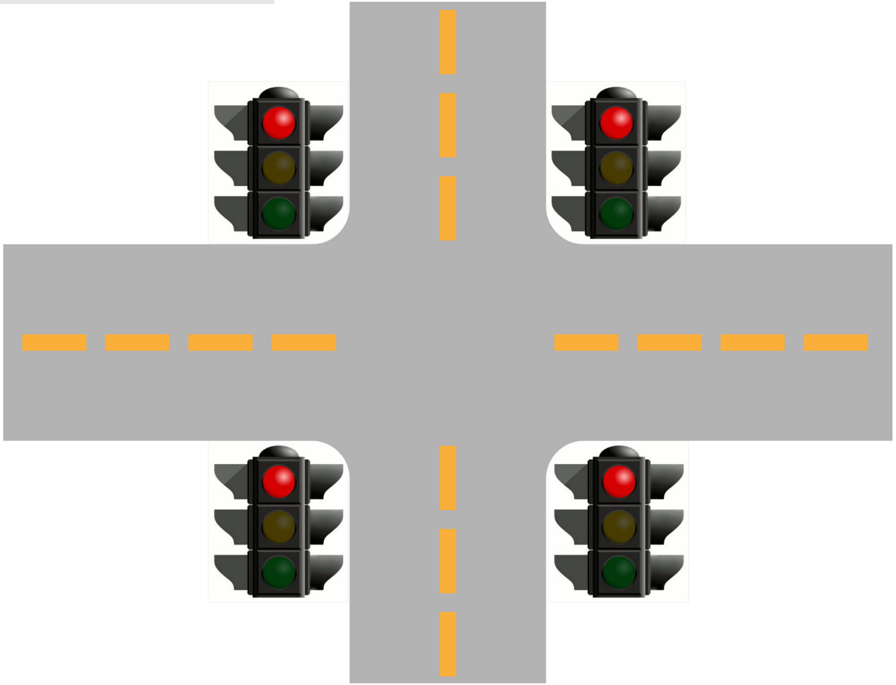
assert (¬  $_1$ ∨ ¬  $_2$)

$\neg(g_1 \wedge g_2)$

Safety specification

Expresses *something bad should not happen*

A perfectly *safe* scenario



“not indefinitely ($_1$ $\wedge$ $_2$)”

Liveness specification

Expresses *something good will eventually happen*

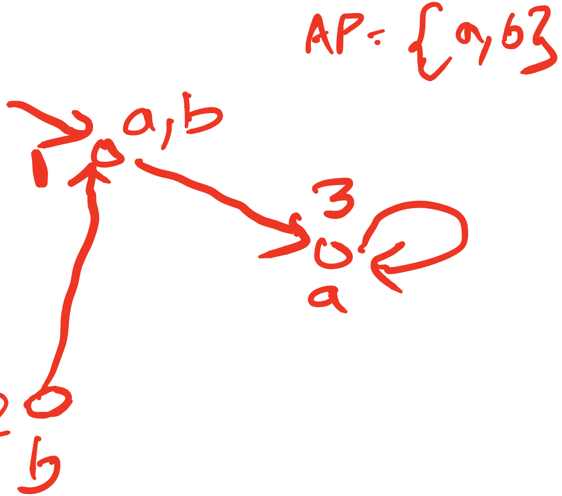
Temporal logics can express safety and liveness specifications for ongoing systems

Modeling Ongoing Systems

Finite State Transition system $\langle S, T, I \rangle$

- A finite set of states S
- A set of initial states $I \subseteq S$
- A **total** transition relation $\mathrm{T} \subseteq S \times S$

$\forall s \in S. \exists s' \in S. T(s, s')$



$\pi = 1, 3, 3, 3, 3, \ldots$

Modeling Ongoing Systems

Kripke structure $\langle S, T, I, L \rangle$

- A finite set of states S
- A set of initial states $I \subseteq S$
- A total transition relation $\mathrm{T} \subseteq S \times S$
- A *labeling function* $L : S \rightarrow 2^{AP}$

set of subsets of AP

AP: set of atomic propositions

- Properties of states
- *Abstracts* values of variables

State Formulas

State Formula
Boolean combination of atomic propositions in AP

Given Kripke structure M, state s and state formula F, we write $M, s \vDash F$ if F holds in s

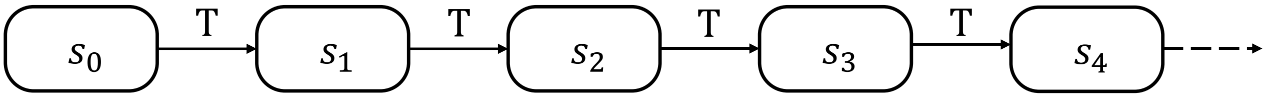
satisfies

$M, s \vDash p$	iff	$p \in L(s)$	
$M, s \vDash \neg F$	iff	$M, s \nvDash F$	
$M, s \vDash F_1 \vee F_2$	iff	$M, s \vDash F_1$ or	$M, s \vDash F_2$
$M, s \vDash F_1 \wedge F_2$	iff	$M, s \vDash F_1$ and	$M, s \vDash F_2$

Path Formulas

A path π is a

- Sequence of states $\mathrm{s}_0, \mathrm{s}_1, \ldots$
- Such that $\mathrm{T}(\mathrm{s}_\mathrm{i}, \mathrm{s}_{\mathrm{i}+1})$ (where $0 \leq i$)



We use π^i to denote the *suffix* of π starting at s_i

- In particular, $\pi = \pi^0$

$\pi^0 = \pi$

$\pi^1 = s_1, s_2, s_3,$

$\pi^3 = s_3, s_4, \ldots$

Path Formulas

Given Kripke structure M, path π and path formula ϕ, we write $M, \pi \vDash \phi$ if ϕ holds for $\pi \in M$

A state formula F is also a path formula

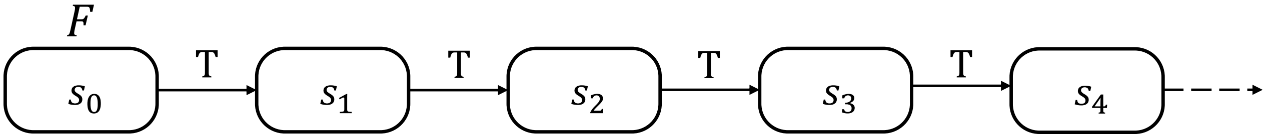
$$M, \pi \vDash F \quad \text{iff} \quad ??$$

Path Formulas

Given Kripke structure M, path π and path formula ϕ, we write $M, \pi \models \phi$ if ϕ holds for $\pi \in M$

A state formula F is also a path formula

$M, \pi \models F$ iff F holds in the first state s_0 of π



- From now on, we use
  - F to denote a *state formula*
  - ϕ to denote a *path formula*

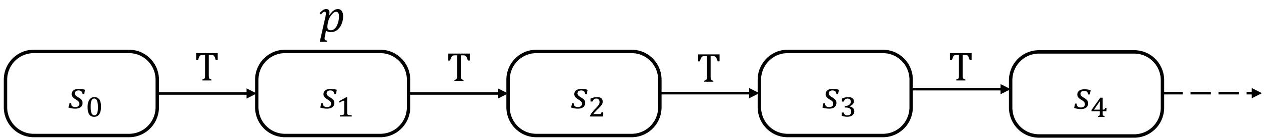
- We introduce a number of temporal operators
  - Allow us to specify what’s supposed to happen *along a path*

Temporal Operators: Next

$$M, \pi \models \mathbf{X}\phi \quad \text{iff} \quad M, \pi^{1} \models \phi$$

For instance: $M, \pi \models \mathbf{X}p$

$s_1 \models p \Rightarrow \pi^1 \models p \Rightarrow \pi \models \mathbf{X}p$

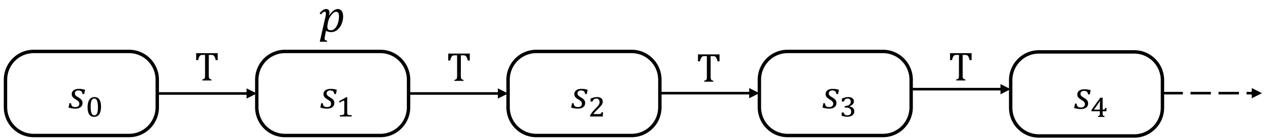


It *doesn't matter* whether or not p holds in s_0 or s_2, s_3, …

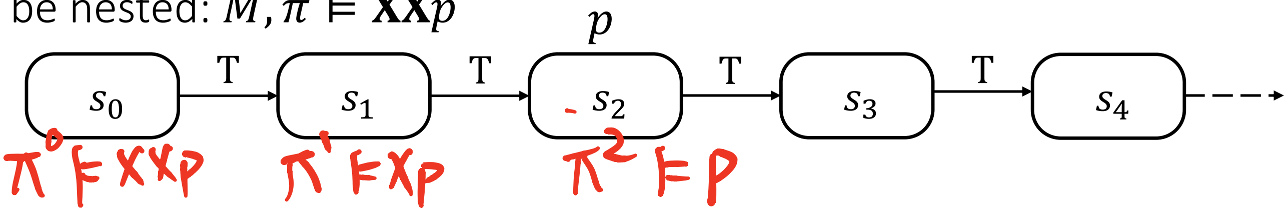
Temporal Operators: Next

$$M, \pi \vDash \mathbf{X}\phi \qquad \text{iff} \qquad M, \pi^{\mathbf{1}} \vDash \phi$$

For instance: $M, \pi \vDash \mathbf{X}p$



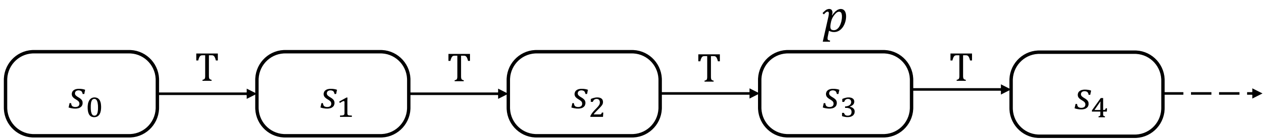
$\mathbf{X}$ can be nested: $M, \pi \vDash \mathbf{XX}p$



Temporal Operators: Eventually

$$M, \pi \vDash \boldsymbol{F}\phi \qquad \Leftrightarrow \qquad \exists k \geq 0.\, M, \pi^k \vDash \phi$$

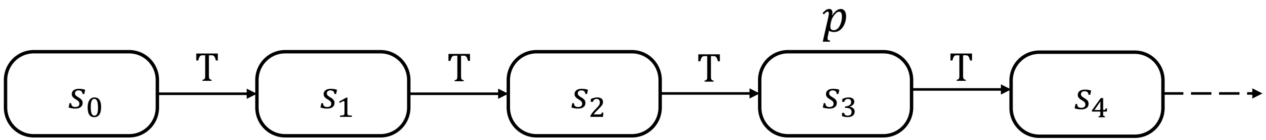
- Basic liveness property
- For instance: $M, \pi \vDash \boldsymbol{F}p$



Temporal Operators: Eventually

$$M, \pi \vDash \boldsymbol{F}\phi \quad \Leftrightarrow \quad \exists k \geq 0.\, M, \pi^k \vDash \phi$$

- Basic liveness property
- For instance: $M, \pi \vDash \boldsymbol{F}p$
  - p holds after a *finite* number of steps



Temporal Operators: Globally

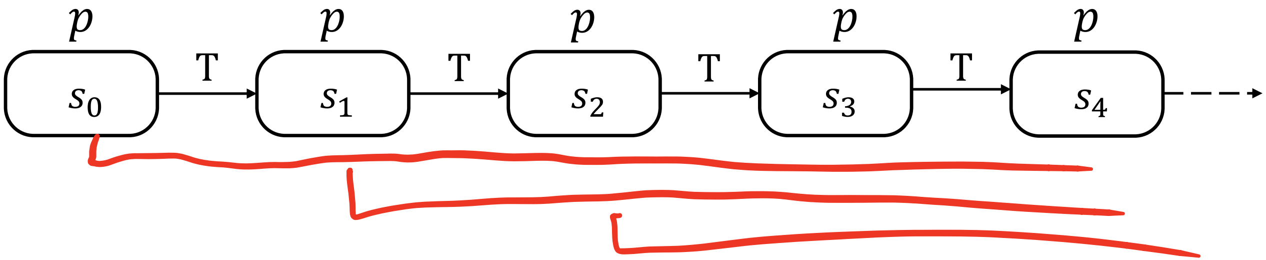
$X p \Rightarrow F p$

$G p \Rightarrow F p$

$$M, \pi \vDash \boldsymbol{G}\phi \quad \Leftrightarrow \quad \forall i \geq 0. M, \pi^i \vDash \phi$$

- Basic safety property
- For instance: $M, \pi \vDash \boldsymbol{G}p$

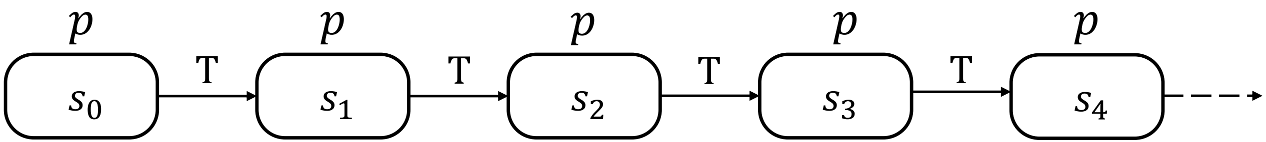
$G(Xp)$ $\quad$ $G(Fp)$



Temporal Operators: Globally

$$M, \pi \vDash \boldsymbol{G}\phi \qquad \Leftrightarrow \qquad \forall i \geq 0.\, M, \pi^i \vDash \phi$$

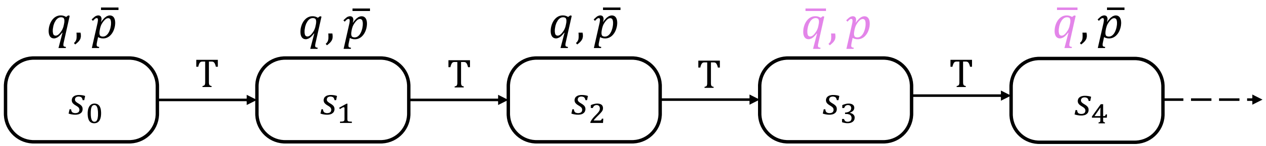
- Basic safety property
- For instance: $M, \pi \vDash \boldsymbol{G}p$
  - p holds after *any* number of steps



Temporal Operators: Until

$$M, \pi \vDash \phi_1 \boldsymbol{U} \phi_2 \quad \Leftrightarrow \quad \exists k \geq 0. M, \pi^k \vDash \phi_2$$
$$\forall j \in \{0..k-1\}. M, \pi^j \vDash \phi_1$$

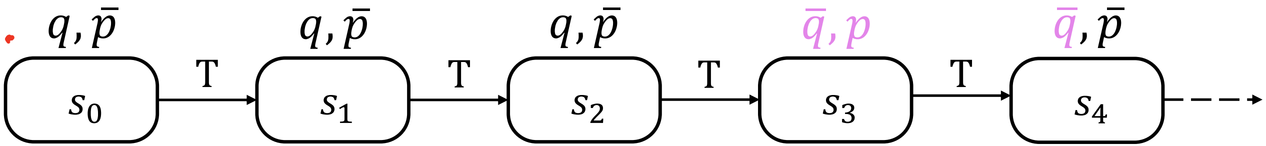
- ϕ_1 holds until ϕ_2 holds
- Also: ϕ_2 has to hold eventually!
- For instance: $M, \pi \vDash q\boldsymbol{U}p$



Temporal Operators: Until

$$M, \pi \vDash \phi_1 \boldsymbol{U} \phi_2 \quad \Leftrightarrow \quad \exists k \geq 0.\, M, \pi^k \vDash \phi_2$$
$$\forall\, j \in \{0..k-1\}.\, M, \pi^j \vDash \phi_1$$

- ϕ_1 holds until ϕ_2 holds
- Also: ϕ_2 has to hold eventually!
- For instance: $M, \pi \vDash q\boldsymbol{U}p$



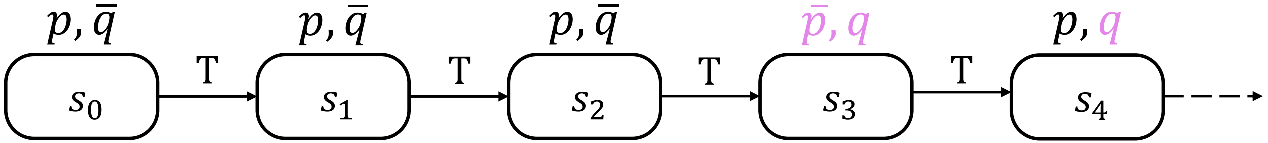
- Note: q doesn't have to hold anymore once discharged by p

Temporal Operators: More Examples

$$M, \pi \vDash p\ \boldsymbol{U}\ (\boldsymbol{G} q)$$

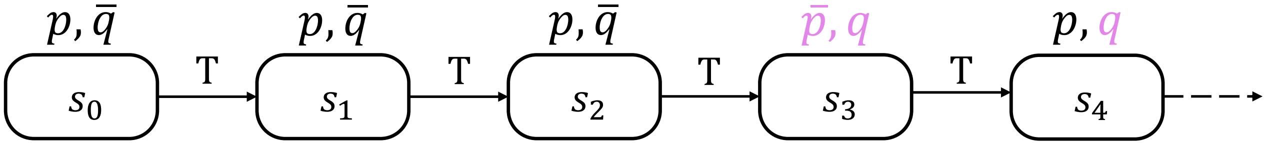
Temporal Operators: More Examples

$$M, \pi \vDash p\ \boldsymbol{U}\ (\boldsymbol{G}q)$$



Temporal Operators: More Examples

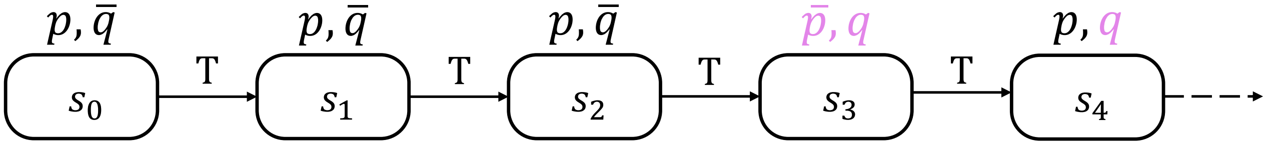
$$M, \pi \vDash p\ \boldsymbol{U}\ (\boldsymbol{G}q)$$



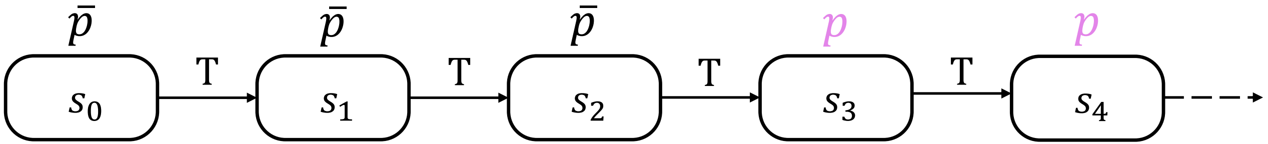
$$M, \pi \vDash \boldsymbol{F}(\boldsymbol{G}p)$$

Temporal Operators: More Examples

$$M, \pi \models p\ \boldsymbol{U}\ (\boldsymbol{G}q)$$



$$M, \pi \models \boldsymbol{F}(\boldsymbol{G}p)$$



Temporal Operators: More Examples

“not indefinitely ($_1$ $\wedge$ $_2$)”

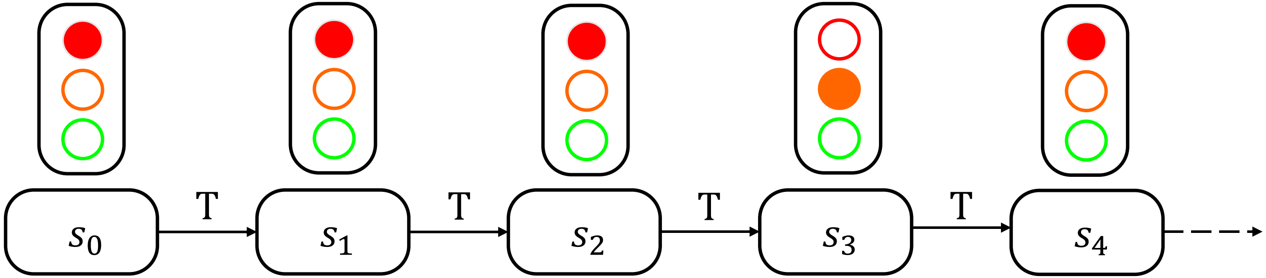
$M, \pi \vDash \boldsymbol{F}\,(\neg$ $)$ or $M, \pi \vDash \neg \boldsymbol{G}\,($$)$

Temporal Operators: More Examples

“not indefinitely ($_1$ ∧ $_2$)”

not indefinitely (R)

$M, \pi \vDash \boldsymbol{F}\,(\neg$ $)$ or $M, \pi \vDash \neg \boldsymbol{G}\,($$)$



Temporal Operators: Redundancies

$$M, \pi \vDash \phi_1 \boldsymbol{U} \phi_2$$

- Last example shows:
  - Some temporal operators can be expressed in terms of others

$$\boldsymbol{G}\,\phi \equiv \neg \boldsymbol{F}(\neg \phi)$$

$$\neg G \phi \equiv F(\neg \phi)$$

$$\boldsymbol{F}\,\phi \equiv true\ \boldsymbol{U}\ \phi$$

Temporal Operators: Redundancies

$$M, \pi \vDash \phi_1 \boldsymbol{U} \phi_2$$

- Last example shows:
  - Some temporal operators can be expressed in terms of others

$$\boldsymbol{G}\, \phi \equiv \neg \boldsymbol{F}(\neg \phi)$$

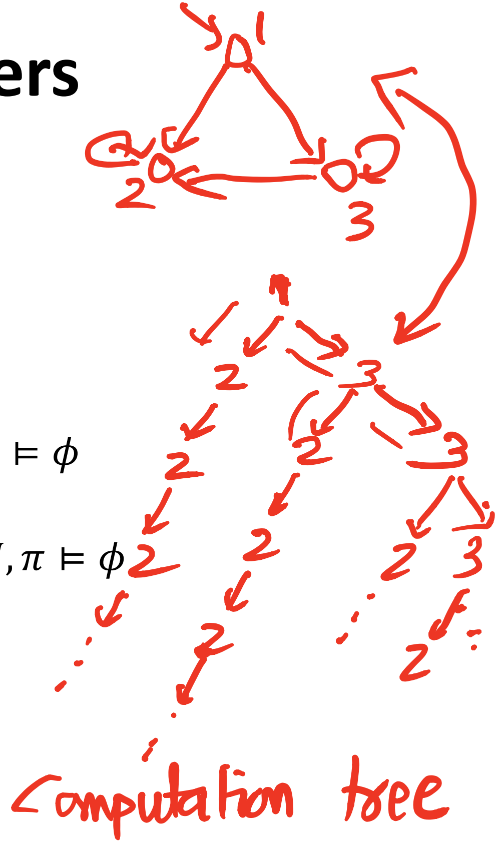
$$\boldsymbol{F}\, \phi \equiv true\ \boldsymbol{U}\ \phi$$

- $\neg, \boldsymbol{X}, \boldsymbol{U}$ are sufficient to express $\boldsymbol{G}$ and $\boldsymbol{F}$
- (c.f. “basis” (¬,∨) in propositional logic)

Temporal Operators: Path Quantifiers

$$M, \pi \vDash \phi_1 \boldsymbol{U} \phi_2$$

- So far, we can only talk about individual paths
- To amend this, we introduce *path quantifies*
  - $M, s \vDash \boldsymbol{E}\,\phi \quad \Leftrightarrow \quad \exists\pi$ starting at s such that $M, \pi \vDash \phi$
  - $M, s \vDash \boldsymbol{A}\,\phi \quad \Leftrightarrow \quad \forall\pi$ starting at s it holds that $M, \pi \vDash \phi$



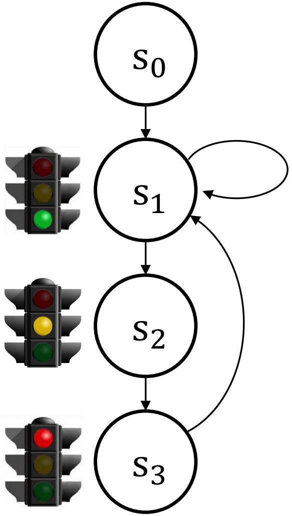
Unwinding Transition Relations

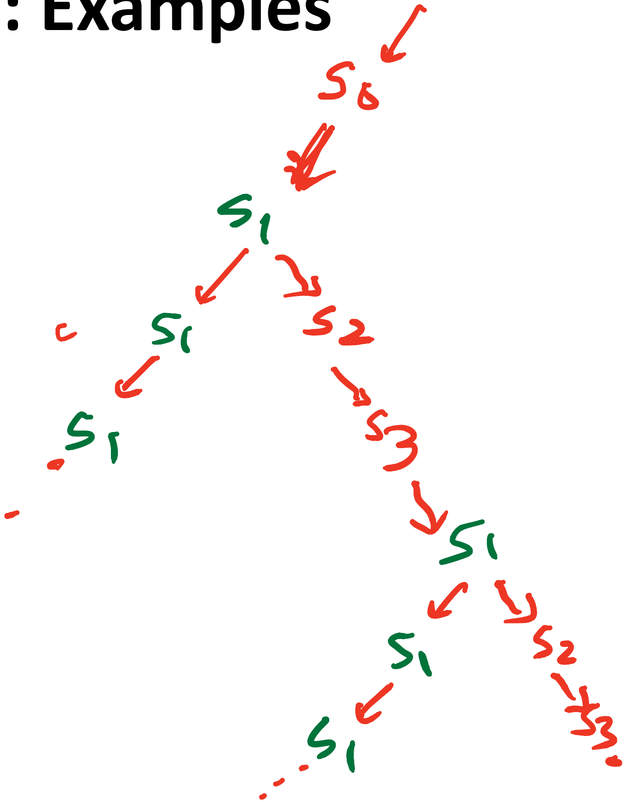
- Remember:
  - Unwinding transition function results in infinite tree

Computation Tree Logic CTL*

- Accordingly, our logic is appropriately called Computation Tree Logic*
- More specifically: CTL*

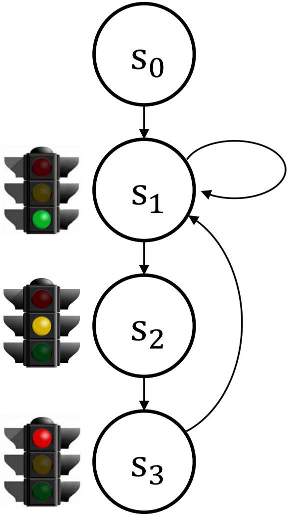
Computation Tree Logic CTL*: Examples





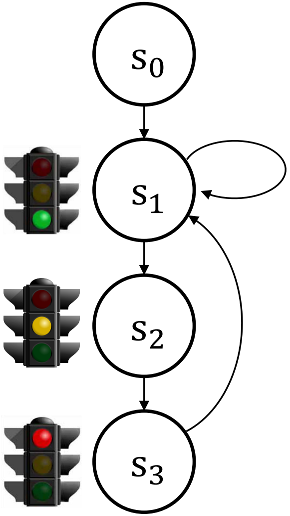
- $M, s_0 \vDash \boldsymbol{AF}$ ()

Computation Tree Logic CTL*: Examples



- $M, s_0 \models \boldsymbol{AF}$ () ✓

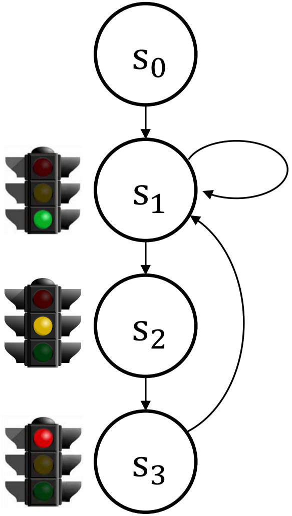
Computation Tree Logic CTL*: Examples



- $M, s_0 \vDash \boldsymbol{AF}$ () ✓

- $M, s_0 \vDash \boldsymbol{AX}(\boldsymbol{EG}($  $))$

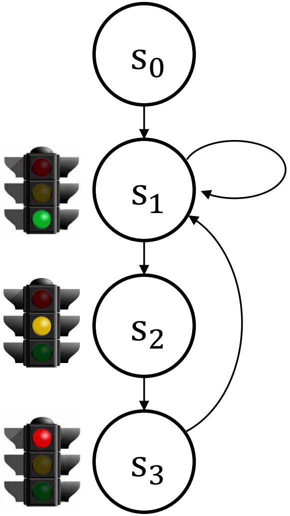
Computation Tree Logic CTL*: Examples



- $M, s_0 \vDash \boldsymbol{AF}$ () ✓

- $M, s_0 \vDash \boldsymbol{AX}(\boldsymbol{EG}($  $))$ ✓

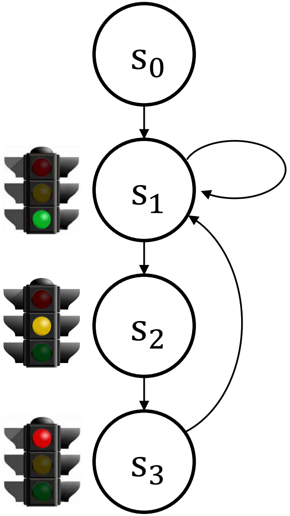
Computation Tree Logic CTL*: Examples



- $M, s_0 \vDash \boldsymbol{AF}$ () ✓
- $M, s_0 \vDash \boldsymbol{AX}(\boldsymbol{EG}(\ a\ ))$ ✓

$M, s_0 \vDash \boldsymbol{EGX}$ ()

Computation Tree Logic CTL*: Examples

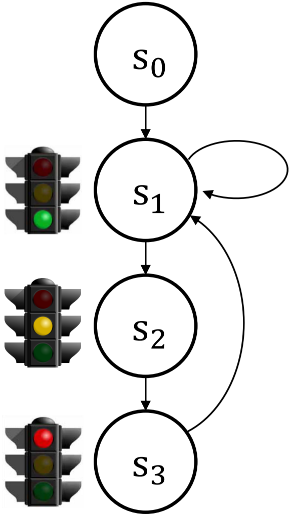


- $M, s_0 \models \boldsymbol{AF}$ () ✓

$M, s_0 \models \boldsymbol{EGX}$ () ✓

- $M, s_0 \models \boldsymbol{AX}(\boldsymbol{EG}($$))$ ✓

Computation Tree Logic CTL*: Examples



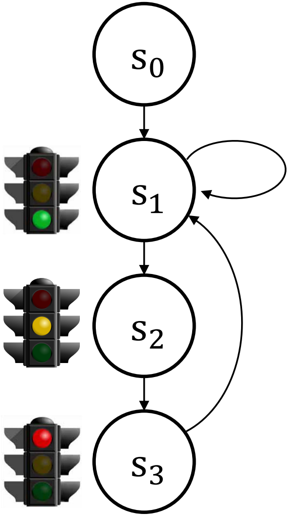
- $M, s_0 \vDash \boldsymbol{AF}$ () ✓

$M, s_0 \vDash \boldsymbol{EGX}$ () ✓

- $M, s_0 \vDash \boldsymbol{AX}(\boldsymbol{EG}($  $))$ ✓

$M, s_0 \vDash \boldsymbol{AGX}$ ()

Computation Tree Logic CTL*: Examples



- $M, s_0 \vDash \boldsymbol{AF}$ () ✓
- $M, s_0 \vDash \boldsymbol{AX}(\boldsymbol{EG}($  $))$ ✓

$M, s_0 \vDash \boldsymbol{EGX}$ () ✓

$M, s_0 \vDash \boldsymbol{AGX}$ () ×

Branching Time and Linear Time Logic

- Commonly used subsets of CTL*:
  - *branching-time* logic   CTL

    quantifies over paths possible from a given state

  - *linear-time* logic   LTL

    for events along a single computation path only

Model Checking

Clarke

Emerson

Sifakis

Clarke & Emerson, *Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic, 1981*

Algorithmic framework for exhaustive exploration of finite-state transition systems to check temporal properties

Branching Time Logic: Computation Tree Logic

- Computation Tree Logic CTL
- CTL ⊂ CTL*
- Restriction:
  $\mathbf{X}$, $\mathbf{F}$, $\boldsymbol{G}$, and $\mathbf{U}$, must be immediately preceded by $\boldsymbol{A}$ or $\boldsymbol{E}$

AX
EF
~~FG~~
AF EG
AF AG

Branching Time Logic: Computation Tree Logic

- Computation Tree Logic CTL
- CTL ⊂ CTL*
- Restriction:
  X, **F**, ***G***, and **U**, must be immediately preceeded by **A** or ***E***
- Examples:

safety violation	**EF**(start ∧ ¬ready)	There's a path on which we start at some point despite not being ready
Liveness	**AG**(req **A** ⇒ **AF** ack)	Each request eventually acknowledged
	AG EX progress	No deadlocks

Branching Time Logic: Computation Tree Logic

- What are the restrictions?
  - Some properties can't be expressed!
- $\mathbf{A}(\mathbf{FG}\, p)$ can't be expressed in CTL!

- And the advantages?
  - More efficient to check than cull CTL*
    - Checking CTL-formula ϕ for $\langle S, T, I, L\rangle$ is O($|\phi| \cdot (|\mathrm{S}| + |\mathrm{T}|)$)
    - Checking CTL* lies in PSPACE
  - Can be checked *using fixed points*!

Linear Temporal Logic

- Linear Temporal Logic: Another subset of CTL*
  for events along a single computation path only

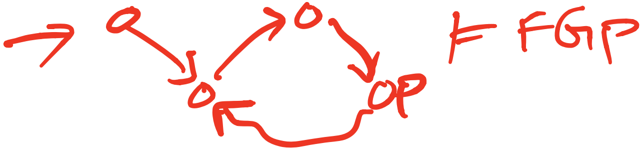
- Formulas have the form $\mathbf{A}\phi$
  - *State formulas can only be atomic propositions*
  - In particular, ϕ doesn't contain $\mathbf{A}$, $\mathbf{E}$, conjunctions, or disjunctions of path formulas

Linear Temporal Logic

- Linear Temporal Logic: Another subset of CTL*
  for events along a single computation path only

- Formulas have the form $\mathbf{A}\phi$
  - *State formulas can only be atomic propositions*
  - ϕ doesn't contain $\mathbf{A}$ or $\mathbf{E}$
- Intuitively, ϕ is always interpreted over all paths

Linear Temporal Logic

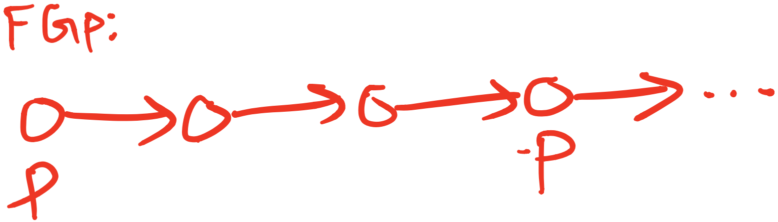


Examples for LTL formulas:

- $\mathbf{A}(\mathbf{FG}\ p)$ "all paths eventually stabilize with property p"
  - This can't be expressed in CTL
- $\mathbf{A}(\mathbf{GF}\ p)$ "p is visited infinitely often"
- $\mathbf{AG}(\text{try} \Rightarrow \mathbf{F}\ \text{succeed})$ "every attempt eventually succeeds"

We can't express

- $\mathbf{AG}(\mathbf{EF}\ p)$
  - This *can* be expressed in CTL



Summary

Today

- Temporal logic as a specification language
- Branching time logic CTL
- Linear time logic LTL

Next

- Odds and Ends