Lecture 18

Pete Manolios
Northeastern

Schedule

- 11/29: Temporal Logic & Model Checking
- 12/2: Projects, Exam 2 (Take home)
- 12/6: Projects

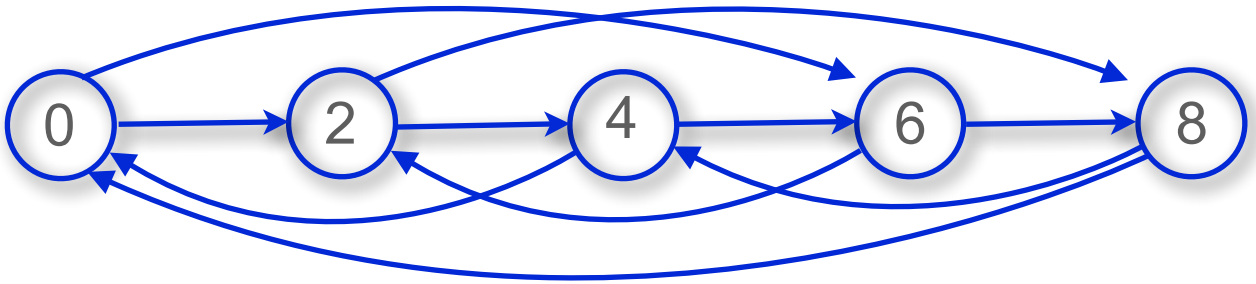
Model Checking

- Does a finite state program satisfy a temporal property?
- Search an implicit graph for errors
- Explicit state model checking
  - Start with initial states
  - Choose an unexplored state and check it
  - If error report; else generate successors & repeat
- Initially: c = 0; m = 10
- Transition relation: c := c+2 mod m || c := c+6 mod m
- Property: G(c < m-1)

finite program

property

implicit graph



Holds?

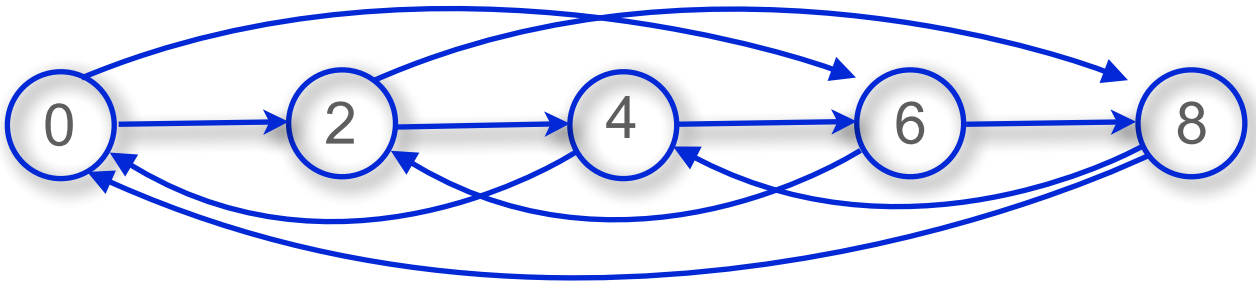
Model Checking

- Does a finite state program satisfy a temporal property?
- Search an implicit graph for errors
- Explicit state model checking
  - Start with initial states
  - Choose an unexplored state and check it
  - If error report; else generate successors & repeat
- Initially: c = 0; m = 10
- Transition relation: c := c+2 mod m || c := c+6 mod m
- Property: G(c < m-1)

finite program

property

implicit graph



Yes!

Model Checking: Concurrency

Initially n = 0 (global, shared variable)

Processes P_i, $1 \le i \le m$

initially reg, counter = 0, 0 (local variables)

```
while counter < 100 {
    reg := n
    reg++
    n := reg
    counter ++ }
```

- What values can n have after all processes terminate?
- G(terminate ⇒ 100≤n≤100*m*) holds? No
- G(terminate ⇒ 2≤n≤100*m*)

Model Checking

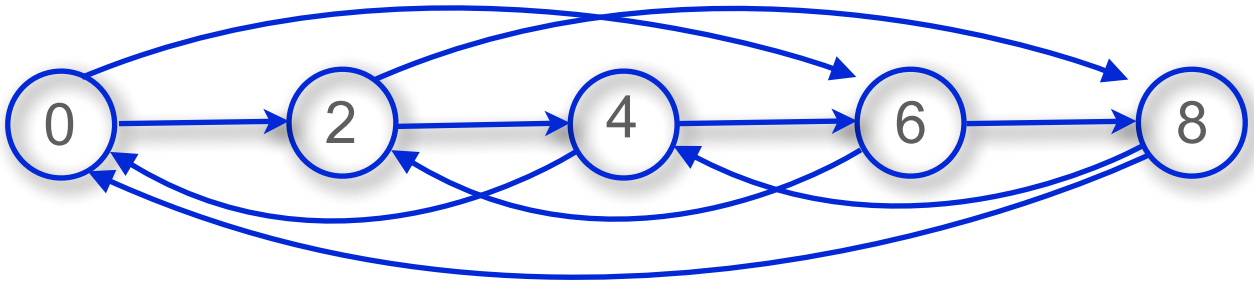
- *Model:* program and graph (can be exponentially bigger)
- *Checking*: temporal logic formula (more later)
- *Nondeterministic: e.g.,* multiple transitions from 0
- *Explicit state*: explicitly represent states
- *On the fly:* only reachable states computed; quit on error
- *Counterexample*: report a path from initial to error state
- *Probabilistic*: use of hashing to store visited states
- *Optimizations*: symmetry & partial order reductions
- *Abstraction*: used to tame complexity
- *Automata:* can represent temporal logic and models
- *Symbolic*: represent states symbolically (BDDs, SAT)
- *Infinite State*: programs are typically infinite state

Transition Systems

- Transition System (TS) $M = \langle S, \dashrightarrow, L \rangle$ where
  - S is a set of states
  - $\dashrightarrow \subseteq S \times S$ is the transition relation (left-total)
  - L is the labeling function: shows what is observable
- A *path* (*trace*) σ is a sequence of states s.t. $\sigma_i \dashrightarrow \sigma_{i+1}$
- A *fullpath* is an infinite path (*ω-trace*)
- The suffix $\langle \sigma_i, \sigma_{i+1}, \ldots \rangle$ of σ is denoted σ^i
- $S_0 \subseteq S$ is the set of initial states (L identifies initial states)
- $L : S \rightarrow \wp(AP)$ is common, for AP a set of atomic prop vars
- Transition systems = Kripke structures = labeled graphs
- Sometimes transitions are also labeled

Transition System Example

- Transition System (TS) $M = \langle S, \dashrightarrow, L \rangle$ where
  - S is a set of states
  - $\dashrightarrow \subseteq S \times S$ is the transition relation (left-total)
  - L is the labeling function: shows what is observable

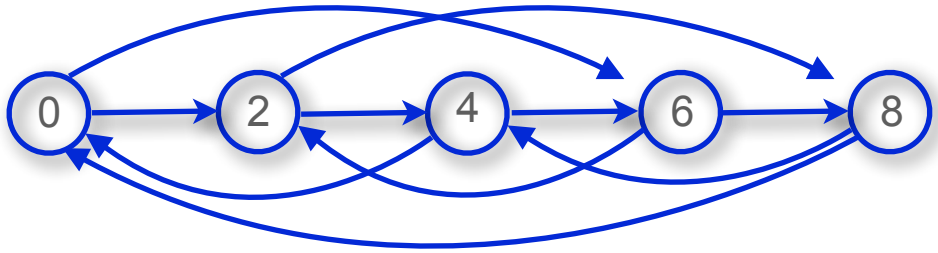


- $S = \{ 0, 2, 4, 6, 8 \}$
- $\dashrightarrow = \{\langle 0, 2\rangle, \langle 0, 6\rangle, \langle 2, 4\rangle, \langle 2, 8\rangle, \langle 4, 6\rangle, \langle 4, 0\rangle, ...\}$
- L = identity = $\{\langle 0, 0\rangle, \langle 2, 2\rangle, \langle 4, 4\rangle, \langle 6, 6\rangle, \langle 8, 8\rangle\}$
- L = div3? = $\{\langle 0, \mathit{true}\rangle, \langle 2, \mathit{false}\rangle, \langle 4, \mathit{false}\rangle, \langle 6, \mathit{true}\rangle, \langle 8, \mathit{false}\rangle\}$

LTL Syntax & Intuition

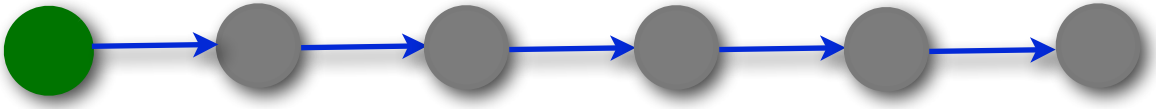
- The syntax of LTL formulas:
  - *e*, where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, where *f*, *g* are formulas

M =

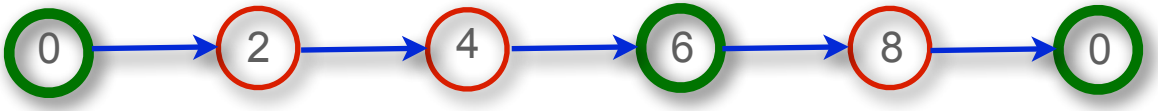


e *e.g.*, div3?; ⟦*e*⟧ = {0, 3, 6, 9, 12, ...} (the predicate denoted by *e*)

σ =

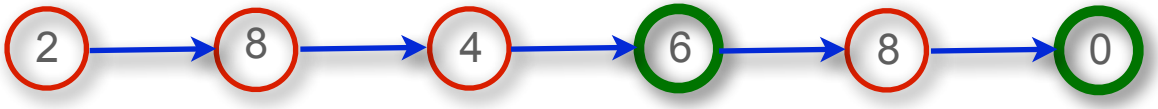


σ =



...

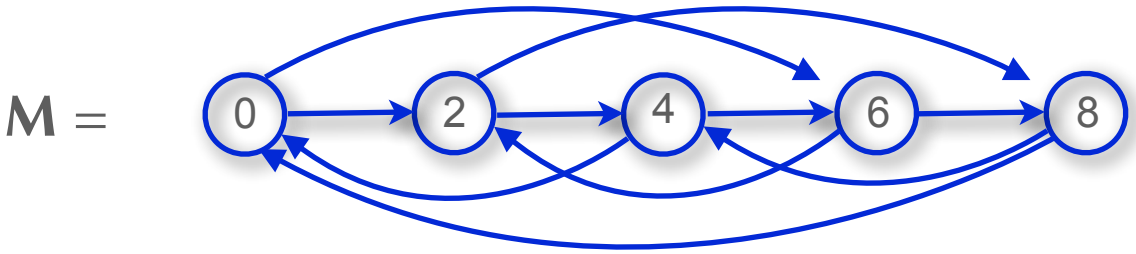
σ =



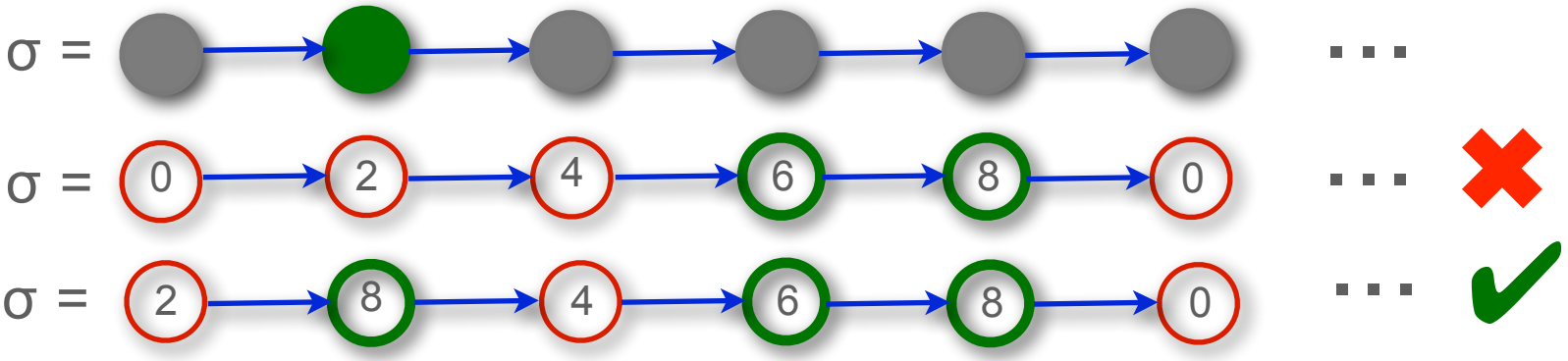
...

LTL **X** Intuition

- The syntax of LTL formulas:
  - *e*, where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, where *f*, *g* are formulas

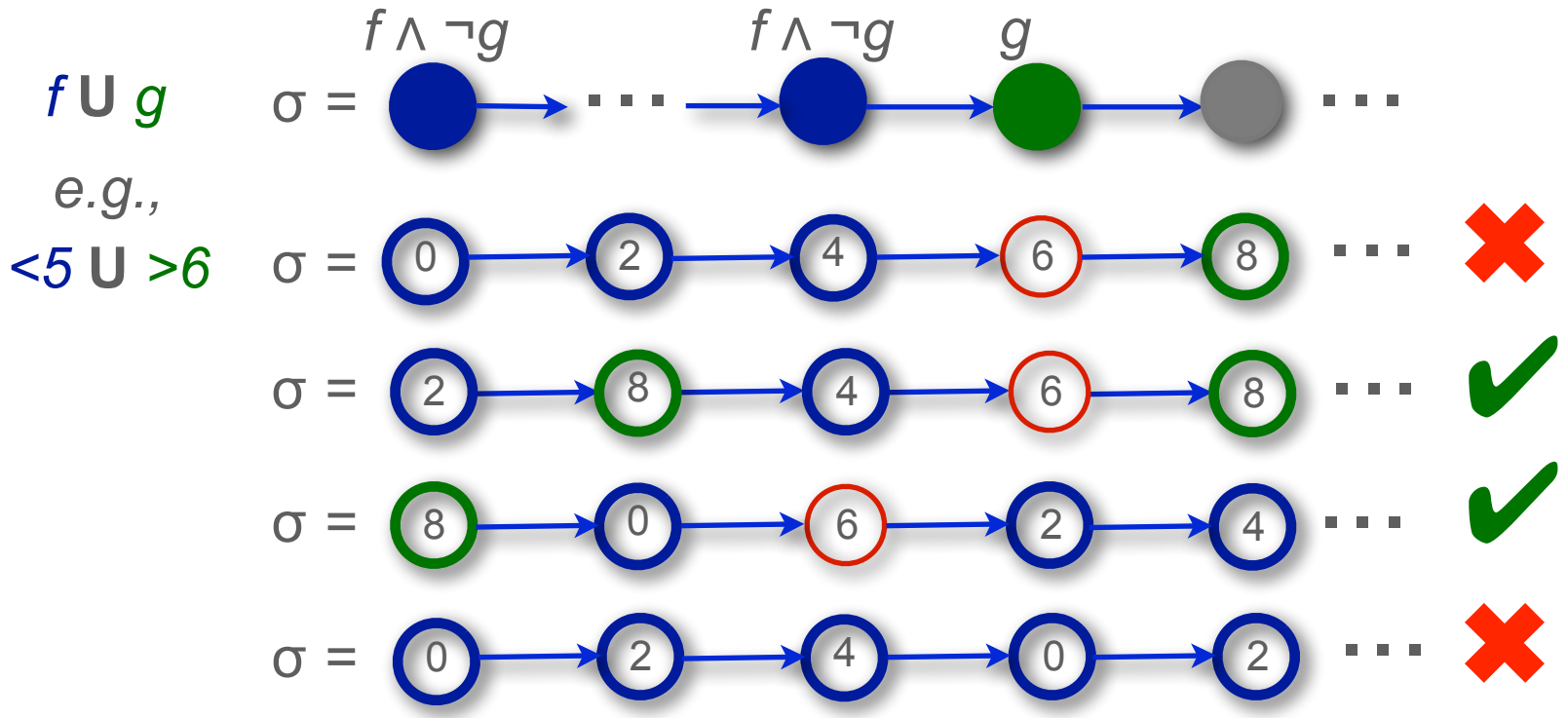


X *f* *e.g.*, **X** >5



LTL **U** Intuition

- The syntax of LTL formulas:
  - *e*, where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, where *f*, *g* are formulas

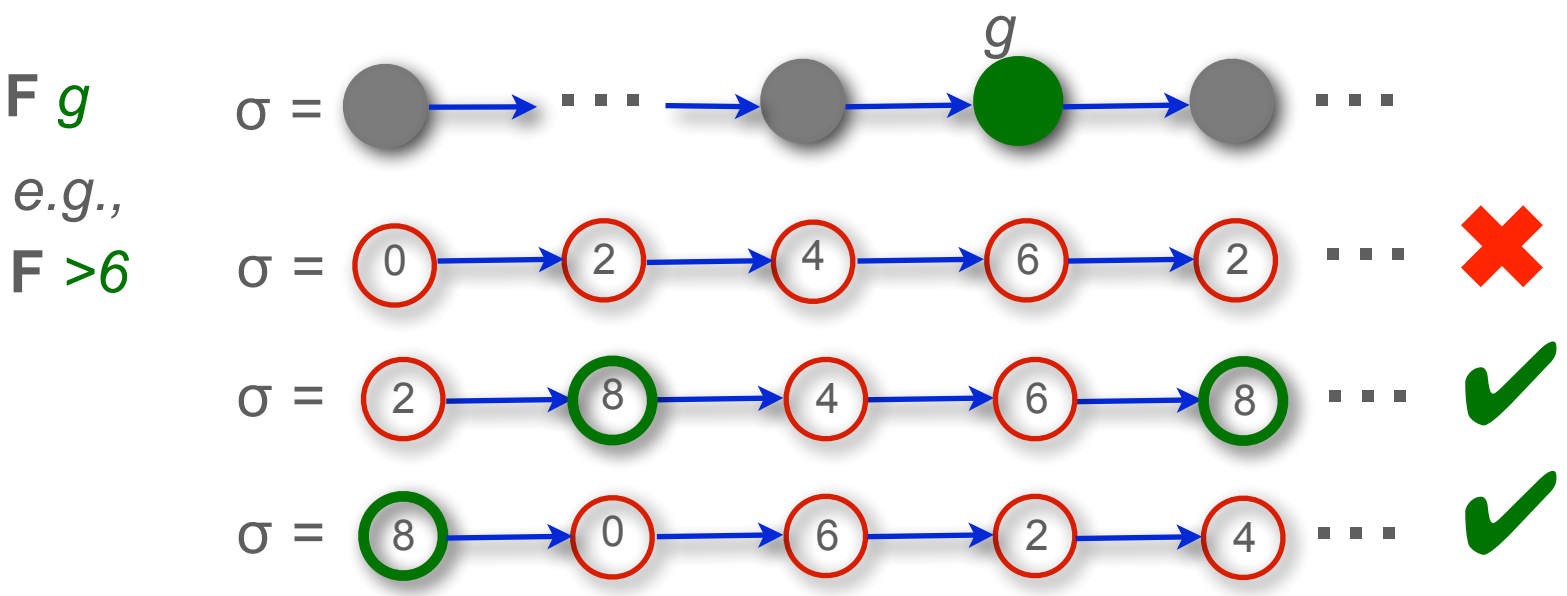


LTL Semantics

- The syntax of LTL formulas:
  - *e,* where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, where *f*, *g* are formulas
- The semantics of LTL formulas wrt M, σ
  - $M, \sigma \models e$ iff $L(\sigma_0) \in [\![e]\!]$
  - $M, \sigma \models f \wedge g$ iff $M, \sigma \models f$ and $M, \sigma \models g$
  - $M, \sigma \models \neg f$ iff it is not the case that $M, \sigma \models f$
  - $M, \sigma \models \mathbf{X}\, f$ iff $M, \sigma^1 \models f$
  - $M, \sigma \models f\ \mathbf{U}\ g$ iff $\exists\ i$ s.t. $M, \sigma^i \models g$ and $\forall\ j < i$, $M, \sigma^j \models f$
- $M \models f$ iff $\forall$ fullpaths σ starting from an initial state: $M, \sigma \models f$

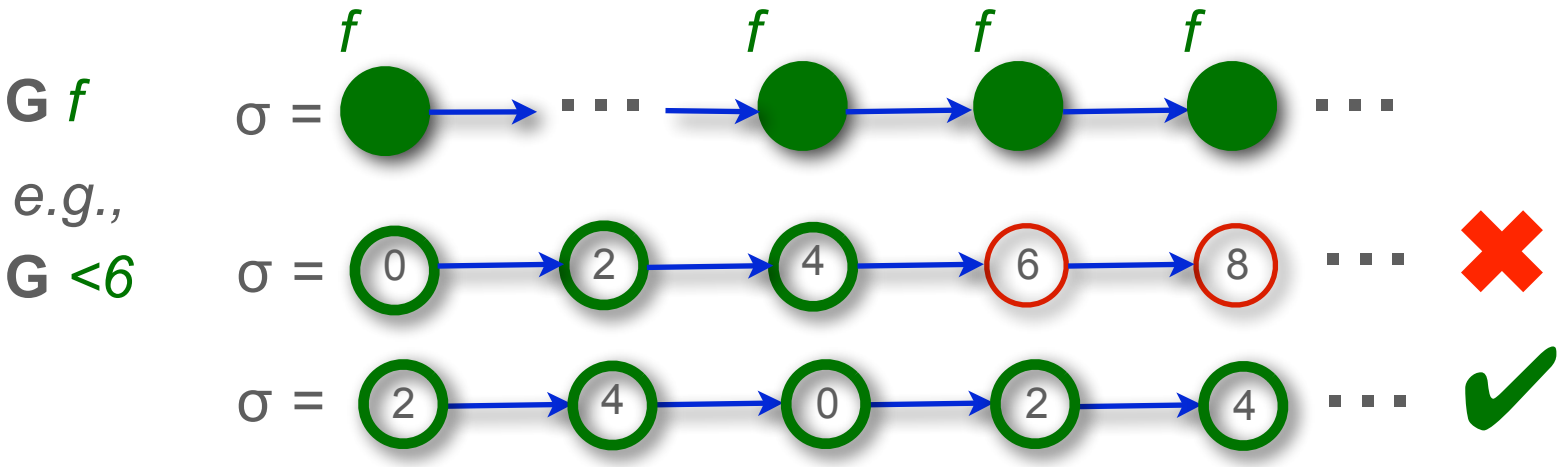
LTL **F** Intuition

- **F** *g* means eventually *g*
- Formally, **F** *g* is an abbreviation for *true* **U** *g*



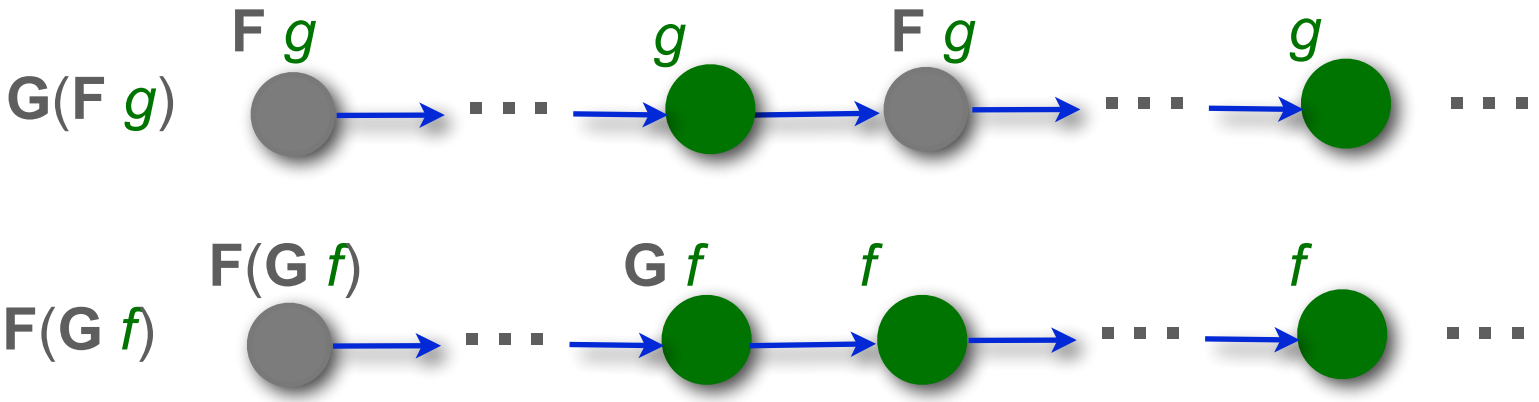
LTL **G** Intuition

- **G** *f* means always *f*
- Formally, **G** *f* is an abbreviation for ¬(**F** ¬*f*)



LTL Fairness

- Express infinitely often (*e.g.*, P_i executes infinitely often)
- Express eventually always



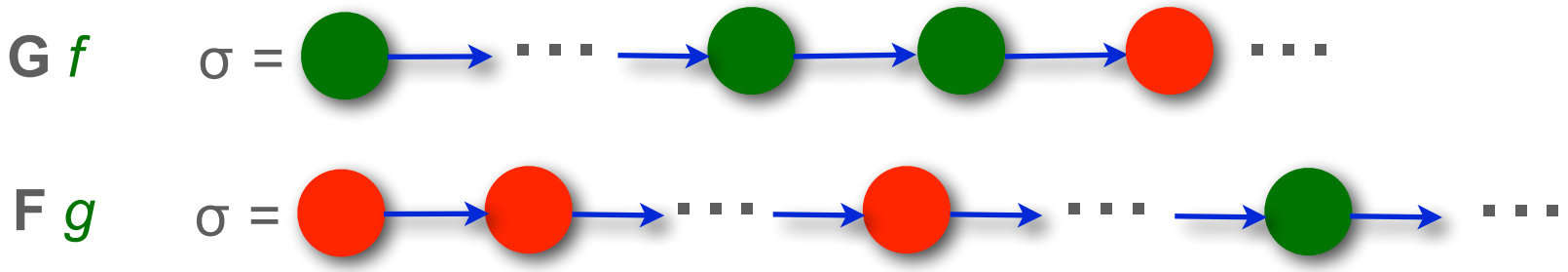
- Does **G** (**F** *g*) ⇒ **F** (**G** *f*)? **No**
- Does **F** (**G** *f*) ⇒ **G** (**F** *g*)? **Yes**

Model Checking LTL

- Model M is given implicitly as a program
  - this is the Kripke structure/ graph/ transition system
  - can be concurrent, nondeterministic, reactive
  - gives rise to the set M of fullpaths (traces) from S_0
- Property is given as an LTL formula f
  - usually the conjunction of formulae
  - can include fairness constraints
  - gives rise to the set P of fullpaths satisfying f
- Model checking means
  - checking $M \models f$ (as defined before)
  - equivalent to checking whether $M \subseteq P$

Safety and Liveness

- Lamport classified properties as:
  - Safety: nothing bad ever happens
  - Liveness: something good eventually happens
  - Neither: neither of the above
- Safety properties: can always be falsified with a finite trace
- Liveness properties: can never be falsified with a finite trace



G *f* ∧ **F** *g* is neither a safety nor a liveness property. Why?

Safety

- Transformational systems
  - type/stack/memory safety
  - no reachable structures are deallocated
  - partial correctness
- Reactive systems
  - only one process is in its critical section at any point in time
  - transactions appear to be atomic
  - messages are authenticated
  - requests are processed within *k* steps

Liveness

- Transformational systems
  - termination
  - unreachable structures are deallocated eventually
- Reactive systems
  - requests are eventually processed
  - weak fairness (eventually always enabled ⇒ taken)
  - strong fairness (infinitely often enabled ⇒ taken)

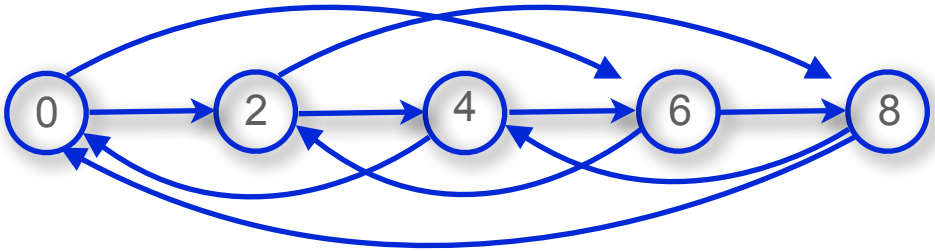
Safety and Liveness

- Specification: partial/total correctness, fairness, etc.
- Different proof methods employed
  - safety: proofs by induction
  - liveness: construction of well-founded relations
- For some problems, safety is decidable but not liveness
- Model checking safety is easier
- Security: enforceable security properties = safety properties
- Topological & lattice-theoretic characterizations
- Decomposition theorem
  - every property is the conjunction of a safety and liveness prop
  - extremal: strongest safety and weakest liveness

Temporal Logic: CTL*

- The syntax of CTL* formulas:
  - *e*, where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, **E** *f*, where *f*, *g* are formulas

M =

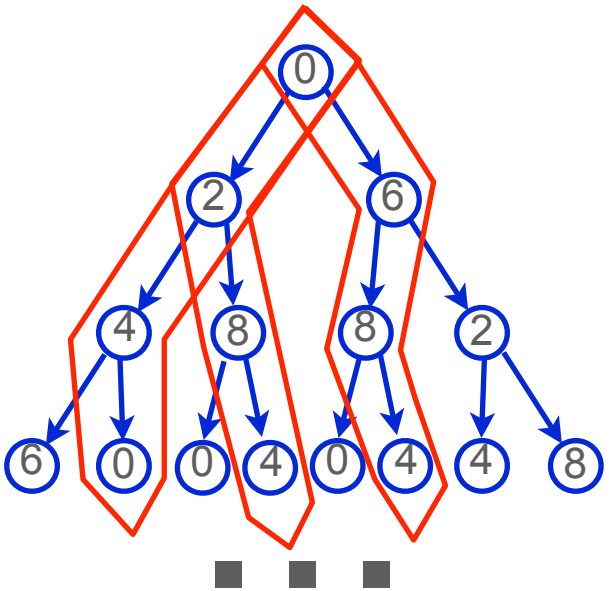


Consider **EF** >6 ✔

Consider **EG** <6 ✔

Consider **EG** >2 ✖

Consider **EFG** >2 ✔

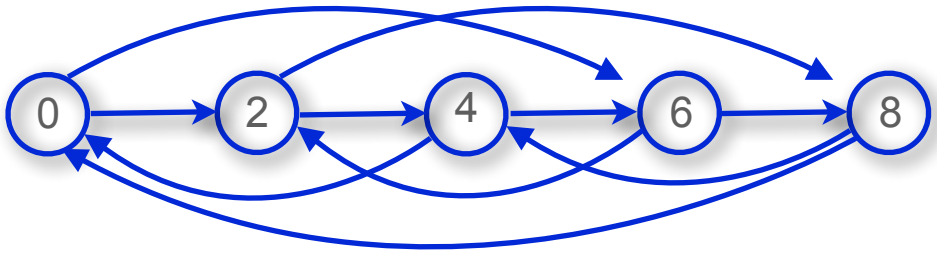


Computation tree obtained by unrolling transition relation

A Intuition CTL*

- **A** *g* means for all paths, *g*
- Formally, **A** *g* is an abbreviation for ¬**E**¬*g*

M =

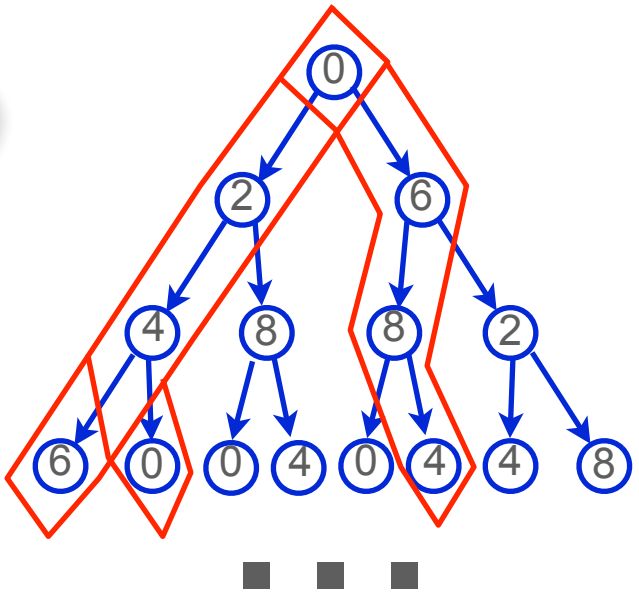


Consider **AX** *>4* ✖

Consider **AF** *>4* ✖

Consider **AF** *=0* ✔

Consider **AXGF** *=0* ✖



Temporal Logic: CTL*

- The syntax of CTL* formulas:
  - *e,* where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, **E** *f*, where *f*, *g* are formulas
- The semantics of CTL* formulas wrt M, σ
  - $M, \sigma \models e$ iff $L(\sigma_0) \; [\![e]\!]$
  - $M, \sigma \models f \wedge g$ iff $M, \sigma \models f$ and $M, \sigma \models g$
  - $M, \sigma \models \neg f$ iff it is not the case that $M, \sigma \models f$
  - $M, \sigma \models \mathbf{X}\, f$ iff $M, \sigma^1 \models f$
  - $M, \sigma \models f \,\mathbf{U}\, g$ iff $\exists\, i$ s.t. $M, \sigma^i \models g$ and $\forall\, j < i$, $M, \sigma^j \models f$
  - $M, \sigma \models \mathbf{E}\, f$ iff $\exists\, \delta = \langle \sigma_0, \ldots \rangle$ in M s.t. $M, \delta \models f$
- $M \models f$ iff $\forall$ fullpaths σ starting from an initial state: $M, \sigma \models f$

State formulas: formulas depending only on first state

Can write $M, s \models f$

Temporal Logic: CTL

- The syntax of CTL* formulas:
  - *e,* where *e* is an *expression*
  - $f \wedge g$ and $\neg f$, where *f*, *g* are formulas
  - **X** *f*, *f* **U** *g*, **E** *f*, where *f*, *g* are formulas
- The syntax of CTL formulas: replace third line with:
  - **EX** *f*, **E**(*f* **U** *g*), **E**¬(*f* **U** *g*), where *f*, *g* are formulas
- Note:
  - Path quantifiers are paired with temporal operators
  - Use CTL to represent **AX** *f*
  - Use CTL can represent **A**(*f* **U** *g*)
  - CTL can represent **EF***f*, **AF***f*, **EG***f*, **AG***f*

(¬**EX** ¬*f*)

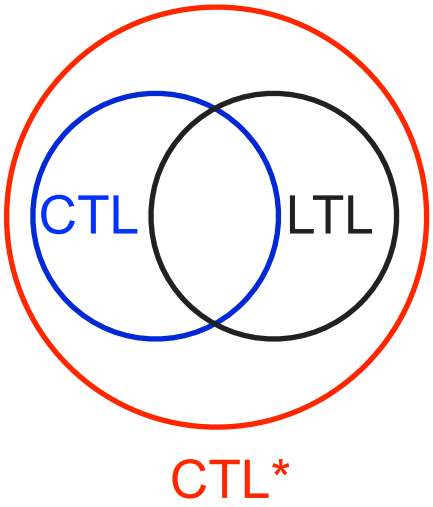
¬(**E**¬(*f* **U** *g*))

E(*true* **U** *f*), **A**(true **U** *f*), **E**¬(*true* **U** ¬*f*), **A**¬(*true* **U** ¬*f*)

Temporal Logic Hierarchy

- CTL* subsumes CTL and LTL
- Find an LTL formula not expressible in CTL **A(FG** p) **??**
- Find a CTL formula not expressible in LTL **AG(EF** p) **??**
- Find a CTL* formula not expressible in LTL or CTL

A(FG p) ∨ **AG(EF** p)



A(FG p) holds

AF(AG p) does not

AG(EF p) holds

LTL only considers fullpaths, so $\neg p, \neg p, \neg p, \ldots$ can provide no evidence for **EF**p