软件分析与架构设计

时序逻辑模型检测

何冬杰
重庆大学

Why Model Checking?

❑Expensive mistakes in Critical Systems

Mars Polar Lander (1999)
landing-logic error

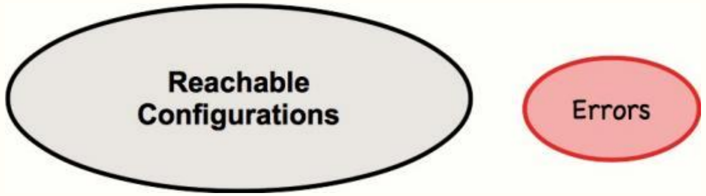
Mission Loss

Spirit Mars Rover (2004)
file-system error

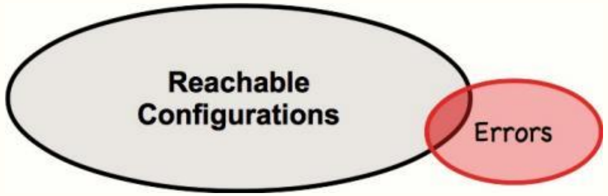
Airbus A380 Flight Deck

Chip Design

❑Want to guarantee safe behavior over unbounded time



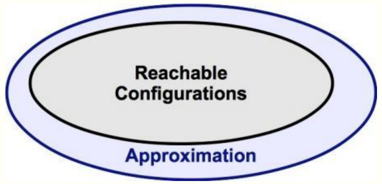
Safe Program



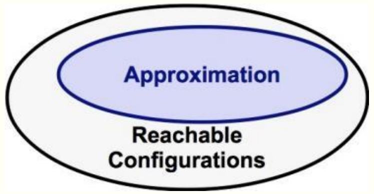
Unsafe Program

Why Model Checking?

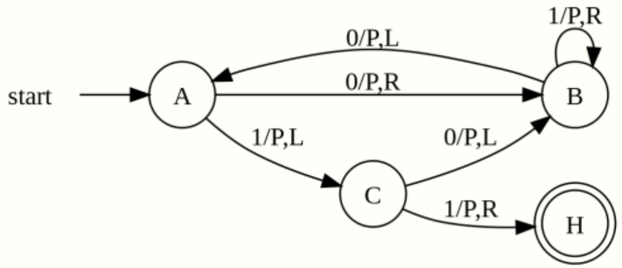
- **The general verification problem is challenging**
  - Deciding whether all possible executions of a program are error-free is hard (**undecidable**). If we could write a program that does it for arbitrary programs to be analyzed then we would always be able to answer whether a Turing machine halts.
- **Solution**
  - Identify sub-problems on which one can decide:
    - e.g. finite state machines, push-down automata, timed automata, Petri nets, well-structured transition systems.
  - Proceed with approximations that will hopefully give some guarantees.



Over-approximation

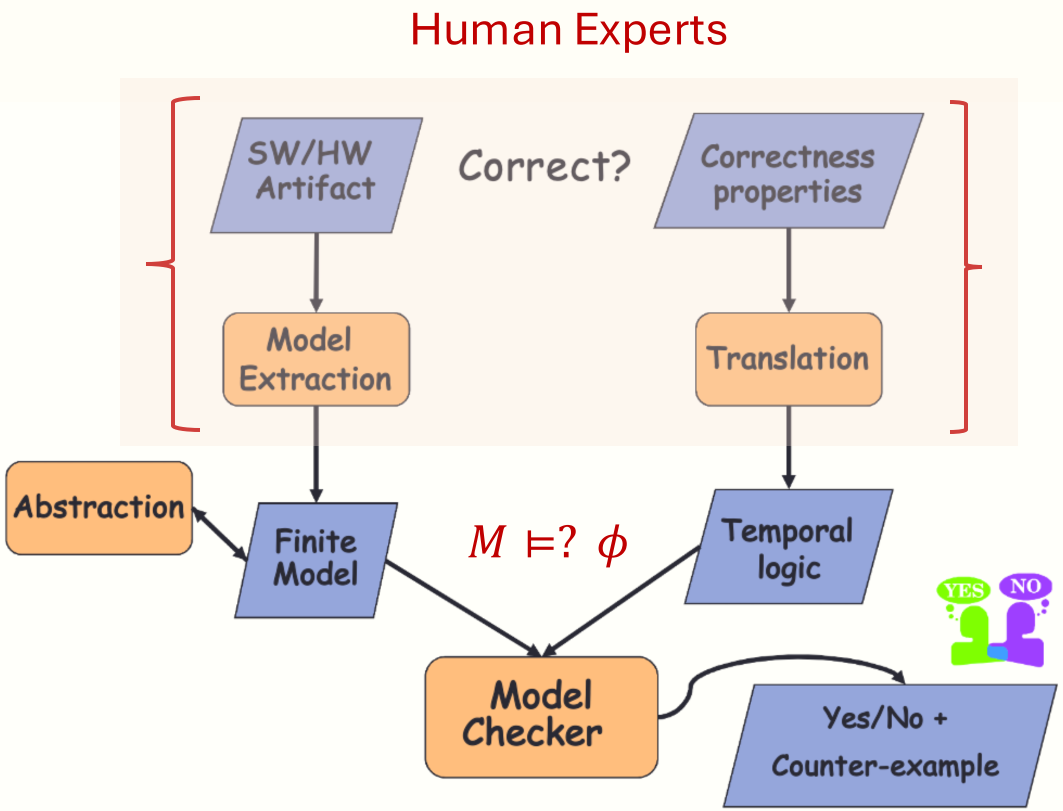


Under-approximation



What is Model Checking?

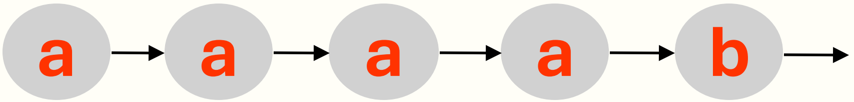
- An approach for verifying the temporal behavior of a (*reactive*) system
- Primarily fully-automated techniques
- **Model**
  - Representation of the system
  - Need to decide the right level of granularity
- **Specification**
  - High-level desired property of system
  - Considers infinite sequences
- **Model Checker**
  - Either a counter-example or a proof
  - PSPACE-complete for FSMs (Finite State Machines)



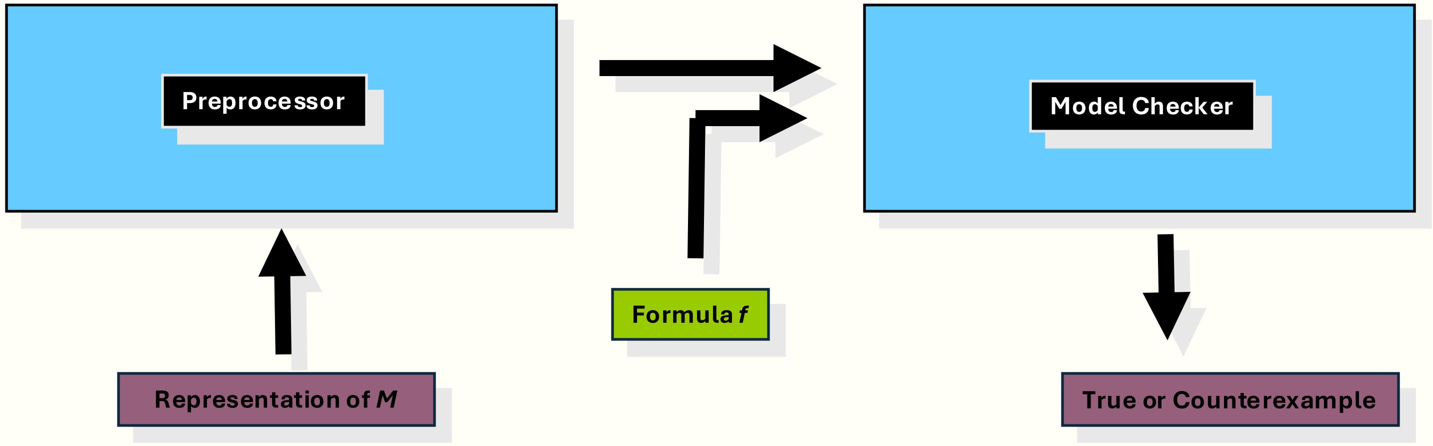
时序逻辑模型检测

❑时序逻辑模型检测(Temporal logic model checking)问题

- ➢Let M be a state-transition graph
- ➢Let f be a formula of temporal logic
  - ○ e.g., $a\ U\ b$ means “a holds true Until b becomes true”

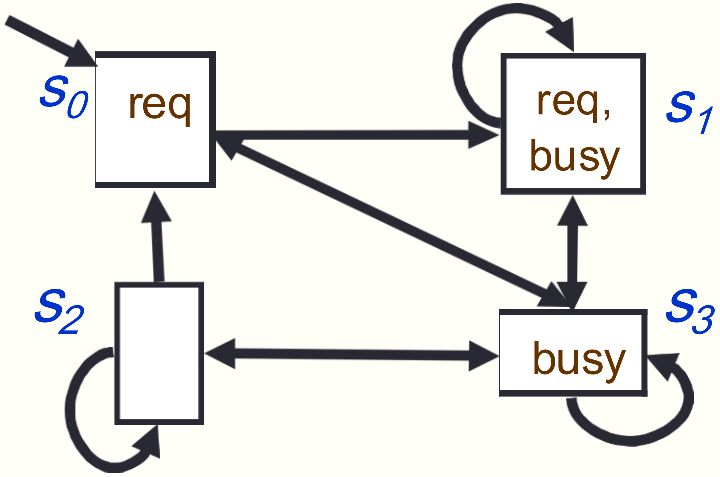


- ➢Does f hold along all paths that start at initial state of M ?



Models: Kripke Structures

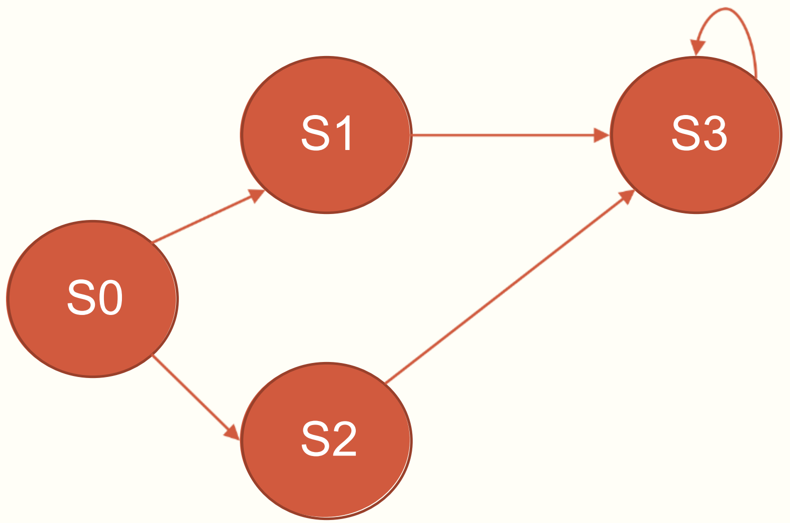
- **A Kripke structure M is a tuple $(AP, S, S_0\,, R, L)$ where:**
  - AP is a set of atomic propositions
  - S is a finite set of states
  - $S_0 \subseteq S$ is the set of initial states
  - $R \subseteq S \times S$ is the transition relation s.t. for any $s \in S,\ R(s, s')$ holds for some
  - labels each state with the atomic propositions that hold on it.



Modeling: Transition System Executions

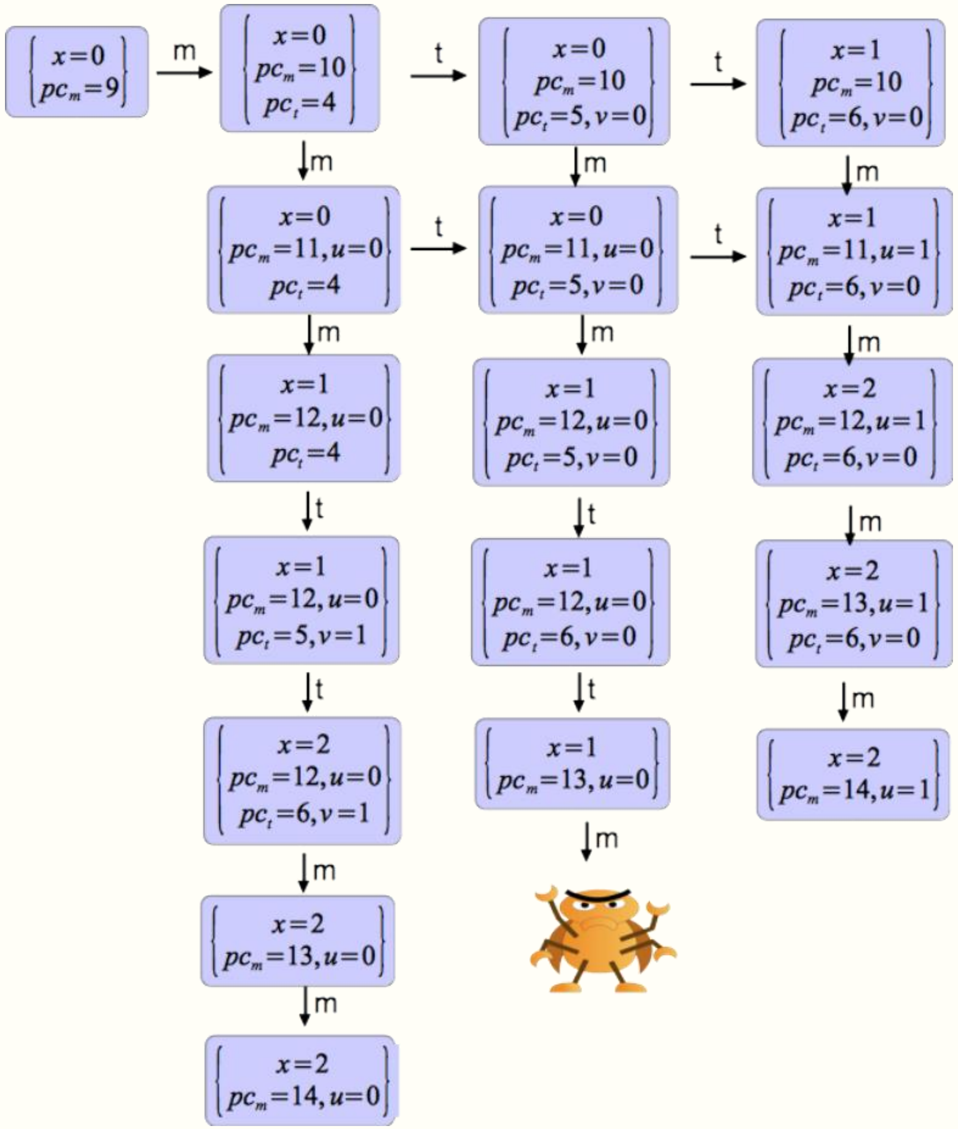
- An ***execution*** **is a sequence of states that respects S_0, in the first state and R between every adjacent pair**

- $\pi := s_0 \; s_1 \dots s_n$ **is a finite sequence**
  - if $s_0 \in S_0 \wedge \bigwedge_{\mathrm{i}=1}^{\mathrm{n}} R(s_{i-1}, s_i)$



Example: Programs as Kripke structures

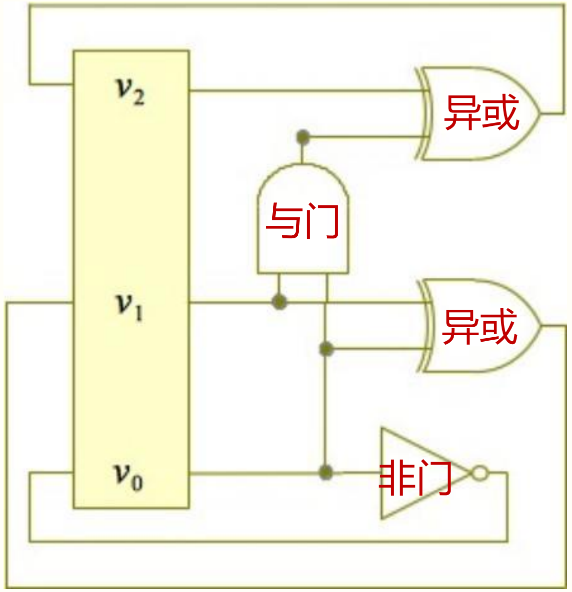
```
int x = 0;

void thread(){
    int v = x;
    x = v + 1;
}

void main(){
    fork(thread); int u
    = x;
    x = u + 1;
    join(thread);
            assert(x == 2);
}
```

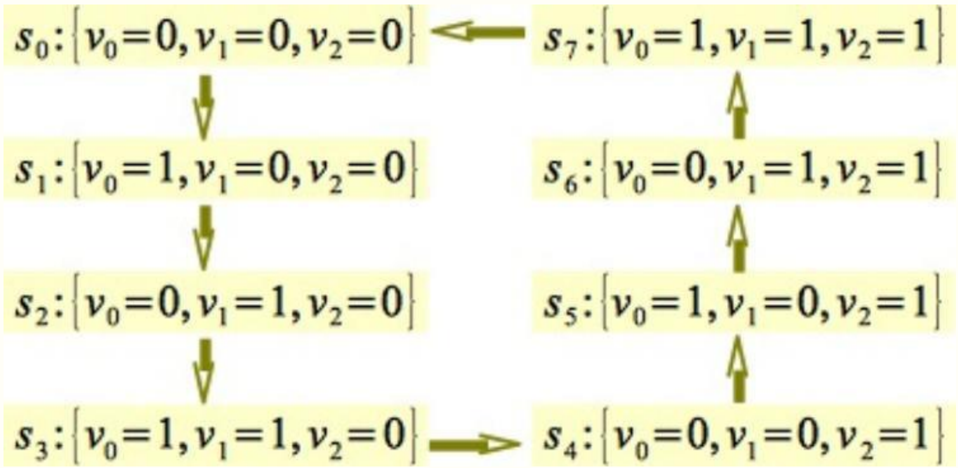


Example: circuits as Kripke structures

❑Synchronous circuits:

➢所有存储元件的更新都由一个共同的时钟信号控制





$v_2 = \quad (v_0 \wedge v_1) \oplus v_2 \quad (3)\, v_1$

$= \quad v_0 \oplus v_1 \quad (2)\, v_0$

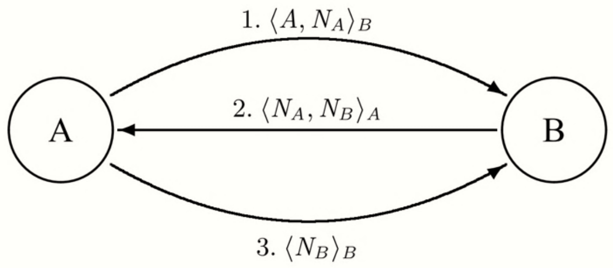
$= \quad \neg v_0 \quad (1)$

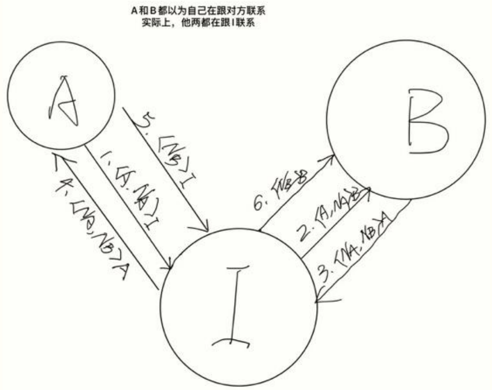
Example：Needham-Schroeder Public-Key Protocol

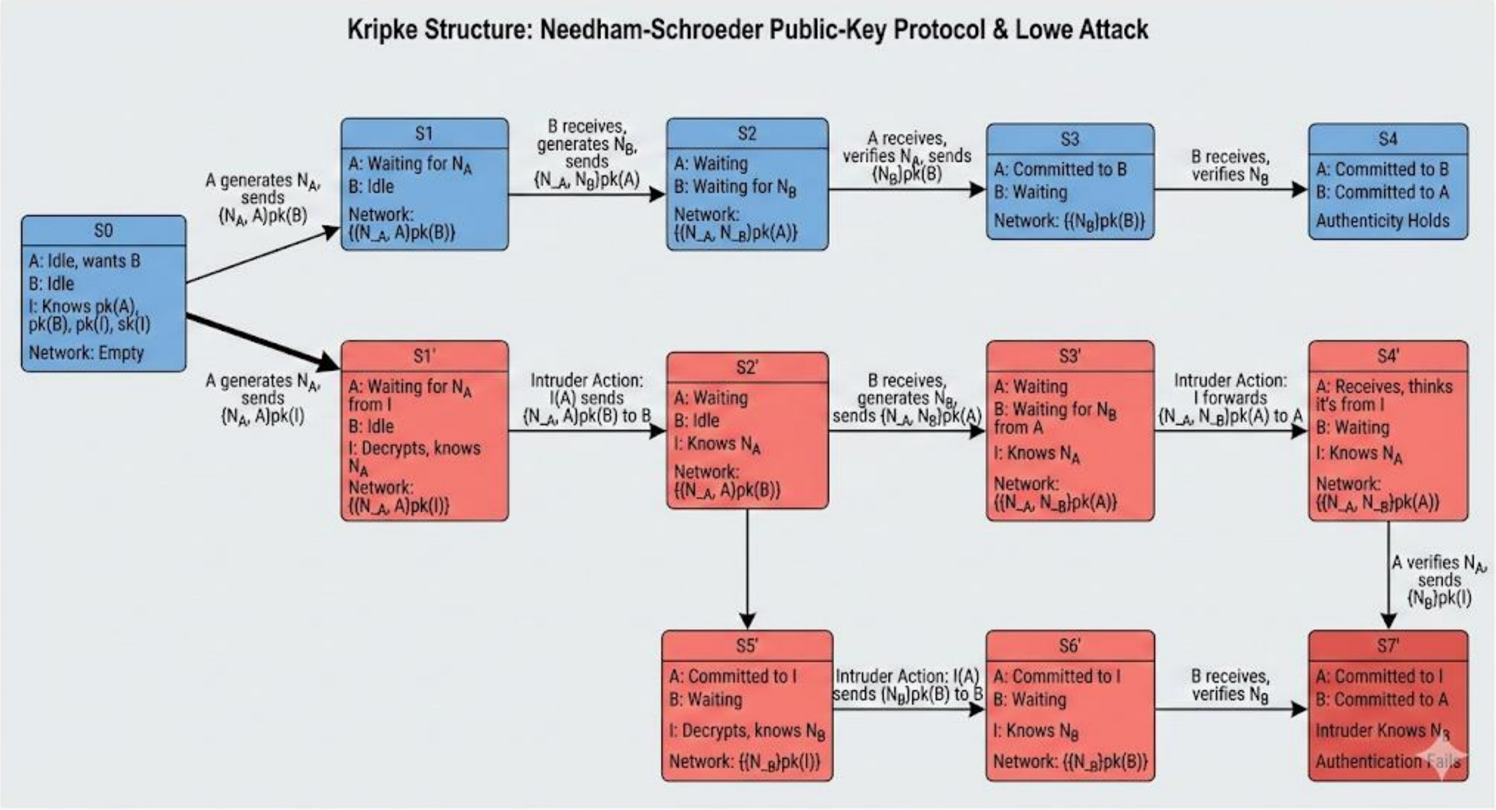
❑**NSPK 协议** Anthenticate first, then switch to symmetric crpto

- ➢$\langle M\rangle_C$:用C的公钥加密消息M
- ➢私钥用于解密
- ➢认证协议，非完整通信协议

$K_{\text{sess}} = \text{KDF}(N_A \parallel N_B)$ $\quad A \leftrightarrow B : \{\text{data}\}_{K_{\text{sess}}}$







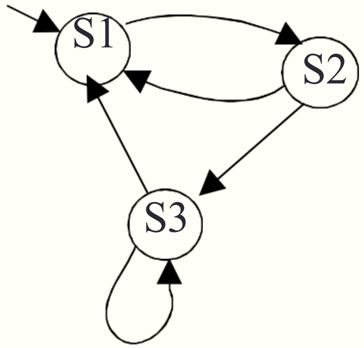
$$S_i = \langle \text{State}_A, \text{State}_B, \text{State}_{\text{Intruder}}, \text{NetworkBuffer}\rangle$$

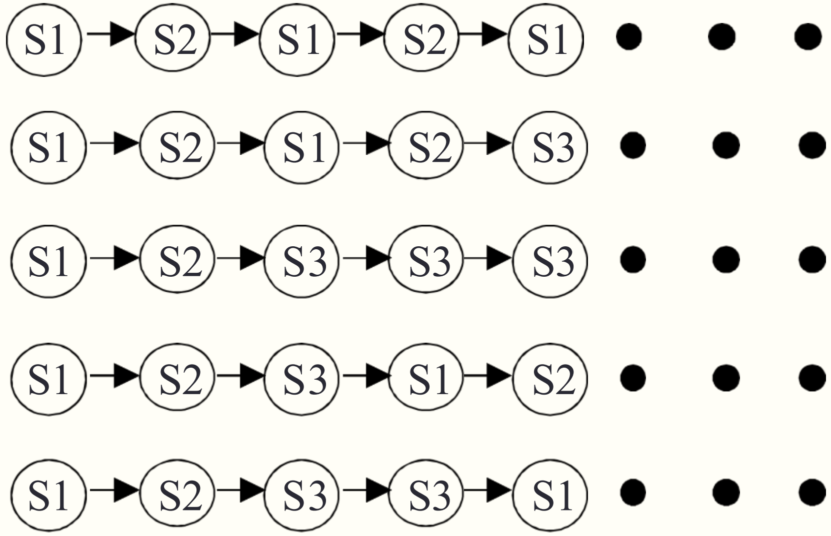
Specification: Temporal Logics

- **Propositional Logic:**
  - Proposition:
    - Fixed set of ***atomic propositions***, e.g, $\{p, q, r\}$
      - *"Printer is busy"*
      - *"There are currently no requested jobs for the printer"*
      - *"Conveyer belt is stopped"*
    - Logical connectives:
      - Not(¬), And(∧), Or(∨), If-then(→), if-and-only-if(↔)
  - Do not involve time
- **Temporal Logic: describing sequences**
  - express that certain properties in AP are:
    - never reached
    - eventually reached
    - more complex combinations of those

Linear Temporal Logic (LTL)

- Reasoning about complete traces through the system

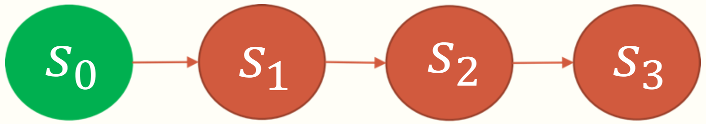




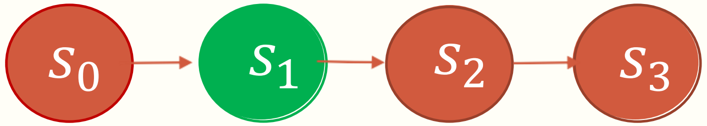
- Allows to make statements about a **trace**

Linear Temporal Logic (LTL)

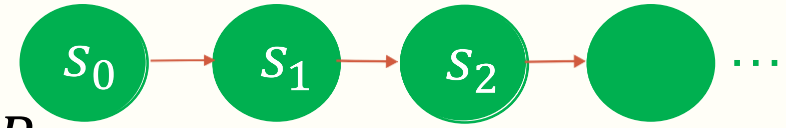
- State formula $P \subseteq S$: Holds iff



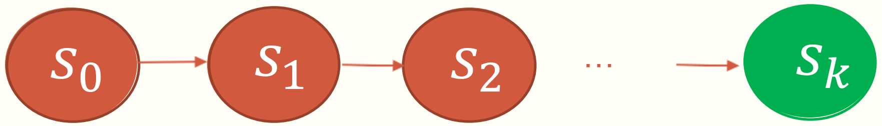
- (for **Next** time) operator: $X(P)$
  - Holds iff the next state meets property P



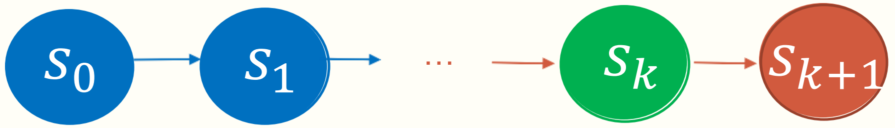
- G (for **Global**) operator: $G(P)$
  - True iff every reachable state meets property P



- F (for **Finally**) operator: $F(P)$
  - True iff P eventually holds



- U (for **Until**) operator: $P_1\ U\ P_2$
  - True iff P_1 holds up until (but not necessarily including) a state where P_2 holds
  - P_2 must hold at some point



LTL Syntax and Properties

- **φ is an LTL formula if it is an atomic propositional formula**
- **If φ and ψ are LTL formulas,**
  - so are $\varphi \wedge \psi$, $\varphi \vee \psi$, φ, φ U ψ, X φ, Fφ, G φ
- **Interpretation over computations** $\pi\colon \omega \to 2^{AP}$
  - which assigns truth values to the elements of V at each time instant
  - $\pi \models X\,\varphi$ iff $\pi^1 \models \varphi$; $\pi \models G\,\varphi$ *iff* $\forall i. \pi^i \models \varphi$; $\pi \models F\,\varphi$ *iff* $\exists \mathrm{i}. \pi^i \models \varphi$
  - $\pi \models \varphi \,\mathrm{U}\, \psi$ iff $\exists i. \pi^i \models \psi \wedge \forall j. 0 \le j < i \Rightarrow \pi^j \models \varphi$
  - π^i is the $i-$th state on path π
- Properties: a property holds in a model if it holds on every path starting from the initial state

$\neg$ X φ = X $\neg$ φ

F φ = true U φ

G φ = $\neg$ F $\neg$ φ

G φ = φ $\wedge$ X G φ

F φ = φ $\vee$ X F φ

φ W ψ = G φ $\vee$ (φ U ψ) (weak until)

Expressing Properties in LTL

- $\text{Req} \rightarrow \text{F}(\text{Ack})$:
  - When a request occurs, it will eventually be acknowledged
- $\text{G}(\text{Req} \rightarrow \text{F}(\text{Ack}))$:
  - Every request eventually acknowledged
- $G(F(DeviceEnabled))$:
  - The device is enabled infinitely often
- $F\big(G(\neg Initializing)\big)$:
  - Eventually it is not initializing
- $\neg q \text{ U } (q \wedge [\neg p \text{ U } s])$:
  - Action s precedes p after q

Expressing Properties in LTL

- invariance: $G(Error)$
- guarantee: $F(Ok)$
- response: $Req \Rightarrow F(Ack)$
- precedence:
  - $Req \Rightarrow (Busy\ U\ Ack)$

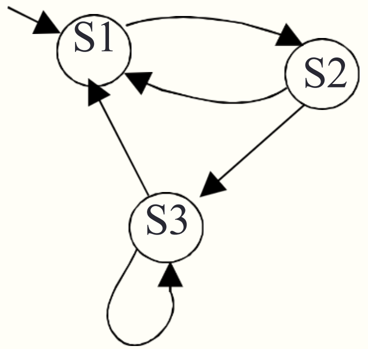
- progress: $GF(Move)$
- stability: $FG(Stable)$
- weak fairness:
  - $GF(\neg Enabled \wedge Executed)$
- strong fairness:
  - $GF(Enabled) \Rightarrow GF(Executed)$

- Safety: "something bad does not happen" $G(\neg bad)$
- Liveness: "something good eventually happens" $GF(good)$

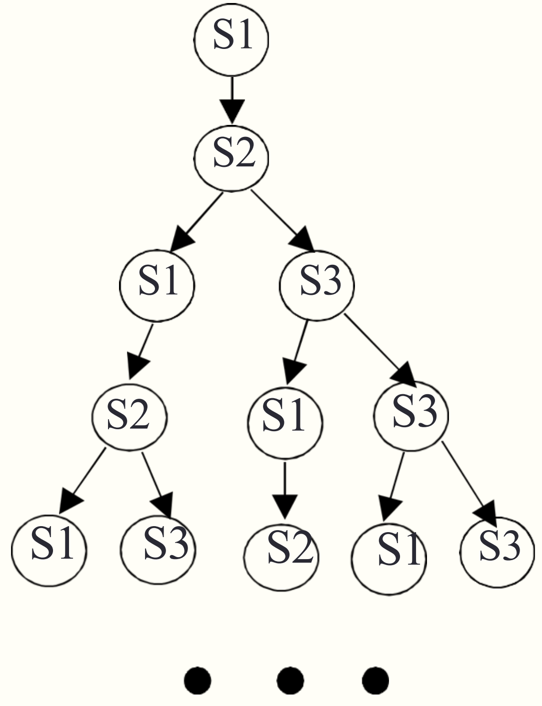
Every property can be written as a conjunction of a safety and liveness property.

Computation Tree Logic (CTL)

- ❑**CTL: Branching-time propositional temporal logic**
  - ➢Propositional temporal logic with explicit quantification over possible futures
- ❑**Model - a tree of computation paths**



Kripke Structure



Tree of computation

Computation Tree Logic (CTL)

❑CTL Syntax

- $True$ and $False$ are CTL formulas;
- propositional variables are CTL formulas;
- If φ and ψ are CTL formulae, then so are: φ , $\varphi \wedge \psi$, $\varphi \vee \psi$
- Existential quantification:
  - $EX\ \varphi$: φ holds in some next state
  - $EF\ \varphi$: along some path, φ holds in a future state
  - $E[\varphi\ \mathrm{U}\ \psi]$: along some path, φ holds until ψ holds
  - $EG\ \varphi$: along some path, φ holds in every state
- Universal quantification:
  - $AX\varphi, AF\varphi, A[\varphi U \psi], AG\varphi$

❑Each of X, F, G, U is immediately preceded by E or A

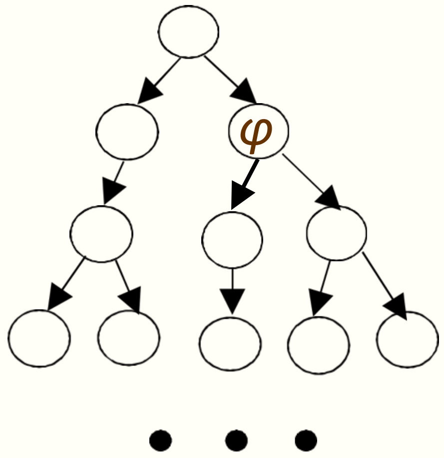
Semantics of CTL

- $K, s \vDash \varphi$: means that formula φ is true in state s.
  - K is often omitted since we often talk about the same Kripke structure
- $\Pi = \pi^0\pi^1 \cdots$ is a path
  - π^0 is the current state (root)
  - π^{i+1} is a successor state of π^i

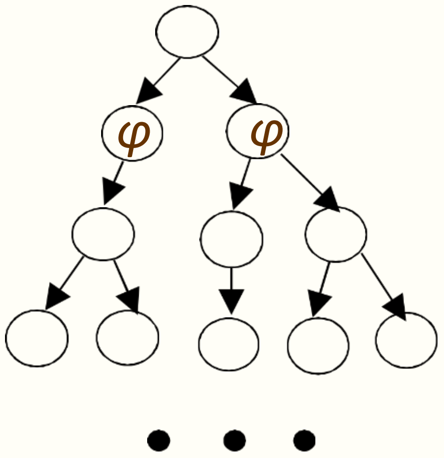
- $AX\ \varphi = \forall\pi.\pi^1 \vDash \varphi$
- $AG\ \varphi = \forall\pi.\forall i.\pi^i \vDash \varphi$
- $AF\ \varphi = \forall\pi.\exists i.\pi^i \vDash \varphi$
- $EX\ \varphi = \exists\pi.\pi^1 \vDash \varphi$
- $EG\ \varphi = \exists\pi.\forall i.\pi^i \vDash \varphi$
- $EF\varphi = \exists\pi.\exists i.\pi^i \vDash \varphi$

- $A[\varphi\ \mathrm{U}\ \psi] = \forall\pi.\exists i.\pi^i \vDash \psi \wedge \forall j. 0 \le j < i \Rightarrow \pi^j \vDash \varphi$
- $\mathrm{E}[\varphi\ \mathrm{U}\ \psi] = \exists\pi.\exists i.\pi^i \vDash \psi \wedge \forall j. 0 \le j < i \Rightarrow \pi^j \vDash \varphi$

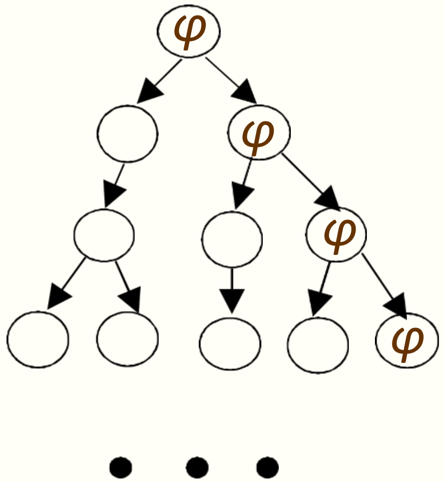
CTL operator illustration



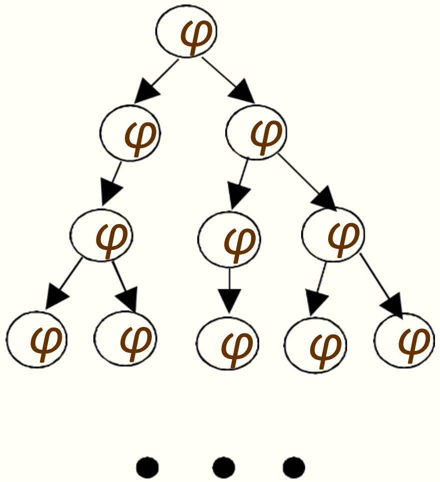
EX φ (exists next)



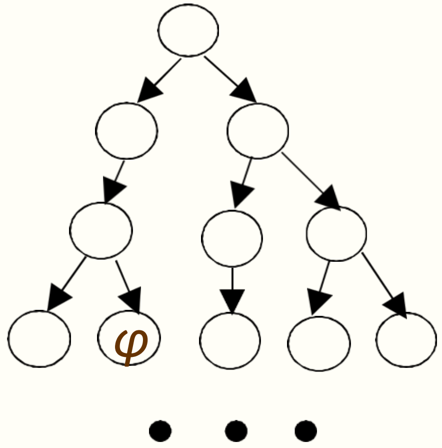
AX φ (all next)



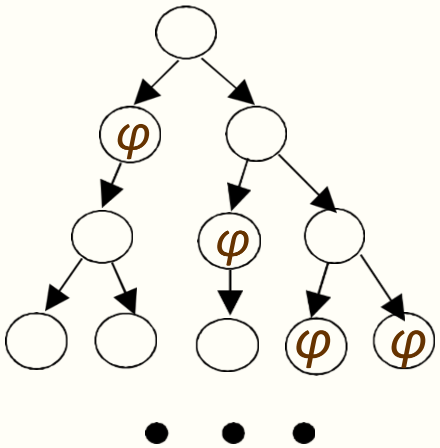
EG φ (exists global)



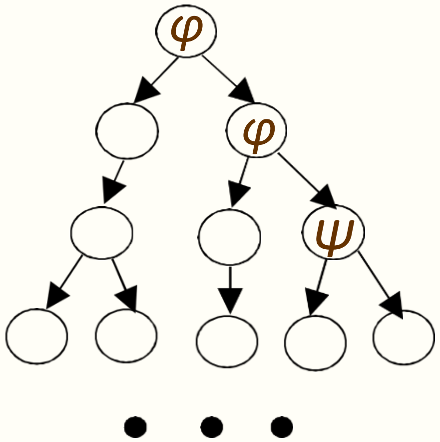
AG φ (all global)



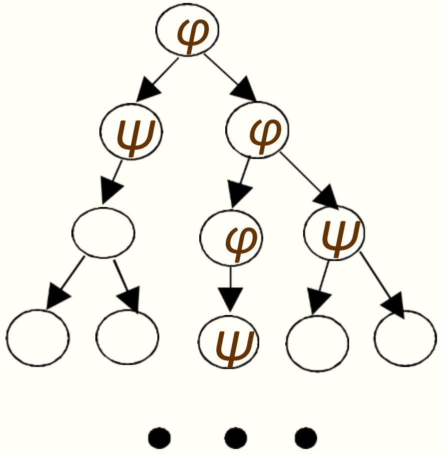
EF φ (exists future)



AF φ (all future)



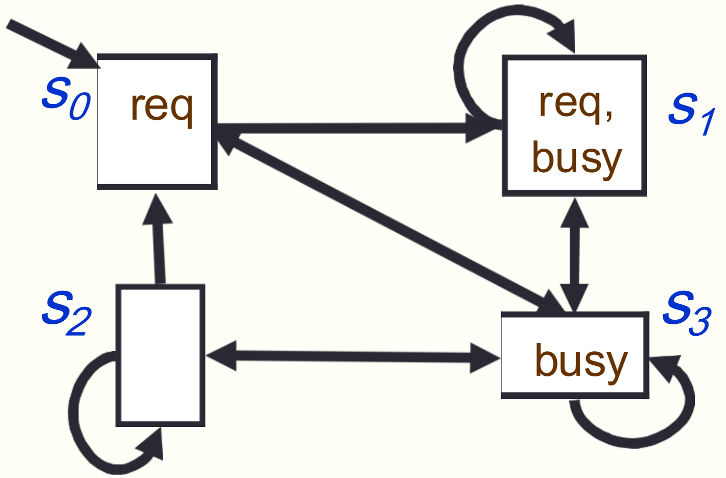
E[φ U ψ] (exists until)



A[φ U ψ] (all until)

CTL Examples

- **Properties that hold:**
  - (AX busy)(s_0)
  - (EG busy)(s_3)
  - A (req U busy) (s_0)
  - E (req U busy) (s_1)
  - AG (req → AF busy) (s_0)
- Properties that fail:
  - (AX (req ∨ busy))(s_3)



Universal and Existential CTL

- ACTL: also called universal CTL formulas
  - uses only universal temporal connectives (AX, AF, AU, AG) with negation applied to the level of atomic propositions
  - $A\ [p \cup AX\ \neg q]$
- ECTL: also called existential CTL formulas
  - uses only existential temporal connectives (EX, EF, EU, EG) with negation applied to the level of atomic propositions
  - e.g., $E\ [p \cup EX\ \neg q]$
- Mixed CTL: CTL formulas not in $\mathrm{ECTL} \cup \mathrm{ACTL}$
  - e.g., $E\ [p \cup AX\ \neg q]$ and $A\ [p \cup EX\ \neg q]$

Theorem. The set of operators {false, ¬, ∧} together with EX, EG, and EU is adequate for CTL.

Relationship Between CTL Operators

$\neg AX\varphi = EX\ \neg\varphi$

$\neg AF\varphi = EG\ \neg\varphi$

$AF\varphi = A[\text{true}\ U\ \varphi]$

$$\boxed{\begin{aligned} AG\ \varphi &= \varphi \wedge AX\ AG\ \varphi \\ AF\ \varphi &= \varphi \vee AX\ AF\ \varphi \end{aligned}}$$

$\neg EF\varphi = AG\ \neg\varphi$

$EF\varphi = E[\text{true}\ U\ \varphi]$

$EG\ \varphi = \varphi \wedge EX\ EG\ \varphi$

$EF\ \varphi = \varphi \vee EX\ EF\ \varphi$

$A[\text{false}\ U\ \varphi] = E[\text{false}\ U\ \varphi] = \varphi$

$$\boxed{A[\varphi\ U\ \psi] = \neg E[\neg\psi\ U\ (\neg\varphi \wedge \neg\psi)] \wedge \neg EG\ \neg\psi}$$

$A[\varphi\ U\ \psi] = \psi \vee (\varphi \wedge AX\ A[\varphi\ U\ \psi])$

$E[\varphi\ U\ \psi] = \psi \vee (\varphi \wedge EX\ E[\varphi\ U\ \psi])$

$A[\varphi\ W\ \psi] = \neg E[\neg\psi\ U\ (\neg\varphi \wedge \neg\psi)]$ (weak until)

$E[\varphi\ U\ \psi] = \neg A[\neg\psi\ W\ (\neg\varphi \wedge \neg\psi)]$

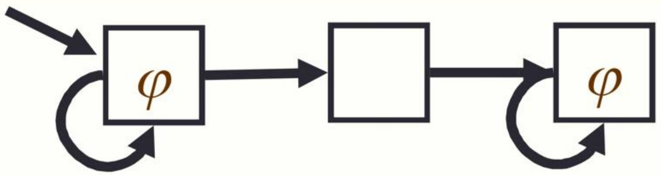
Safety and Liveness in CTL

- Safety ($AG\ \neg bad$): Finite counterexample if fail
- Liveness ($AG\ AF\ good$): Infinite counterexample if fail

Every universal temporal logic formula can be decomposed into a conjunction of safety and liveness.

Comparison between LTL and CTL

- Syntactically, LTL is simpler than CTL
- Semantically: incomparable!
  - CTL formula $AG\ EF\ \varphi$ (always can reach) is not expressible in LTL
  - LTL formula $F\ G\ \varphi$ (eventually always) is not expressible in CTL
    - LTL formula $F\ G\ \varphi$ vs. CTL $AF\ \mathrm{AG}\ \varphi$?
    - Different interpretation on the following state machine:
      - $AF\ \mathrm{AG}\ \varphi = \mathrm{false}$
      - $F\ G\ \varphi = \mathrm{true}$



- The logic CTL* is a super-set of both CTL and LTL
- LTL and CTL coincide if the model has only one path!

(Optional) The CTL* Temporal Logic

❑CTL* formula:

- Path quantifiers: **A** (for all paths), **E** (there Exists a path)
- Temporal operators: **X**(next), **F**(finally, future), **G**(globally), **U**(until)

❑CTL* vs CTL:

- CTL: each of X, F, G, U is immediately preceded by E or A
- CTL*: no such restriction
- $\mathrm{LTL} \subset \mathrm{CTL}*$; $\mathrm{CTL} \subset \mathrm{CTL}*$; $\mathrm{LTL} \not\subset \mathrm{CTL}$; $\mathrm{CTL} \not\subset \mathrm{LTL}$

❑CTL* Syntax:

- State formulas:
  - Atomic formulas
  - $\neg \mathrm{f}, \mathrm{f} \wedge \mathrm{g}$, and $\mathrm{f} \vee \mathrm{g}$ if f, g are state formulas
  - Af, Ef if f is a path formula
- Path formulas:
  - State formulas
  - $\neg \mathrm{f}, \mathrm{f} \wedge \mathrm{g}, \mathrm{f} \vee \mathrm{g}, \mathrm{X\,f}, \mathrm{F\,f}, \mathrm{G\,f}$, and $\mathrm{f\,U\,G}$ if f, g are path formulas

CTL* is the set of state formulas generated by the above rules.

(Optional) CTL* Semantics

f_1 and f_2 are state formulas, g_1 and g_2 are path formulas.

$$
\begin{array}{lcl}
M, s \models p & \Leftrightarrow & p \in L(s) \\
M, s \models \neg f_1 & \Leftrightarrow & M, s \not\models f_1 \\
M, s \models f_1 \vee f_2 & \Leftrightarrow & M, s \models f_1 \text{ or } M, s \models f_2 \\
M, s \models f_1 \wedge f_2 & \Leftrightarrow & M, s \models f_1 \text{ and } M, s \models f_2 \\
M, s \models \mathbf{E}\ g_1 & \Leftrightarrow & \text{there is a path } \pi \text{ from } s \text{s.t. } M, \pi \models g_1 \\
M, s \models \mathbf{A}\ g_1 & \Leftrightarrow & \text{for every path } \pi \text{ starting from } s,\ M, \pi \models g_1
\end{array}
$$

$$
\begin{array}{lcl}
M, \pi \models f_1 & \Leftrightarrow & \text{if } \pi = s_0 s_1 \ldots \text{ then } M, s_0 \models f_1 \\
M, \pi \models \neg g_1 & \Leftrightarrow & M, \pi \not\models g_1 \\
M, \pi \models g_1 \vee g_2 & \Leftrightarrow & M, \pi \models g_1 \text{ or } M, \pi \models g_2 \\
M, \pi \models g_1 \wedge g_2 & \Leftrightarrow & M, \pi \models g_1 \text{ and } M, \pi \models g_2 \\
M, \pi \models \mathbf{X}\ g_1 & \Leftrightarrow & M, \pi^1 \models g_1 \\
M, \pi \models \mathbf{F}\ g_1 & \Leftrightarrow & \text{there exists a } k \geq 0 \text{ s.t. } M, \pi^k \models g_1 \\
M, \pi \models \mathbf{G}\ g_1 & \Leftrightarrow & \text{for all } k \geq 0 \text{ s.t. } M, \pi^k \models g_1 \\
M, \pi \models g_1 \mathbf{U}\ g_2 & \Leftrightarrow & \text{there exists a } k \geq 0 \text{ s.t. } M, \pi^k \models g_2 \\
 & & \text{and for all } 0 \leq j < k,\ M, \pi^j \models g_1
\end{array}
$$

$\boldsymbol{M, s} \models \boldsymbol{f}$: state formula f holds at state s in the Kripke structure M

$\boldsymbol{M, \pi} \models \boldsymbol{f}$: state formula f holds along path π in the Kripke structure M

(Optional) 作业：用时序逻辑写规约

- **用CTL表示下面语句：**
  - An elevator can remain idle on the third floor with its doors closed
  - When a request occurs, it will eventually be acknowledged
  - A process is enabled infinitely often on every computation path
  - A process will eventually be permanently deadlocked
  - Action s precede p after q

- **CTL*练习**：Express each of the following using f, g, $\neg$, **U**, **E**:
  - $(\mathrm{F}\ \mathrm{f}) = ?$
  - $(\mathrm{G}\ \mathrm{f}) = ?$
  - $(\mathrm{A}\ \mathrm{f}) = ?$
  - $(\mathrm{f}\ \mathrm{R}\ \mathrm{g}) = ?$

$$M, \pi \models g_1 \mathbf{R}\ g_2 \quad \Leftrightarrow \quad \text{for all } j \geq 0 \text{ if for every } i < j, M, \pi^i \not\models g_1 \text{ then } M, \pi^j \models g_2$$