Model Checking
Principles

Model (System Requirements) → Model Checker → Specification (System Property)

Answer:
Yes, if the model satisfies the specification
Counterexample, otherwise
Kripke Model

- Kripke Structure + Labeling Function
  - Let $\text{AP}$ be a non-empty set of atomic propositions.
  - Kripke Model: $M = (S, s_0, R, L)$

\[
\begin{align*}
S & \quad \text{finite set of states} \\
S_0 \in S & \quad \text{initial state} \\
R \subseteq S \times S & \quad \text{transition relation} \\
L : S \rightarrow 2^{\text{AP}} & \quad \text{labeling function}
\end{align*}
\]
Specification

- Often expressed in temporal logic
  - Propositional logic with temporal aspect
  - Describes ordering of events without explicitly using the concept of time
  - Several variants: LTL, CTL, CTL*
Why Use Temporal Logic?

- Requirements of concurrent, distributed, and reactive systems are often phrased as constraints on *sequences of events or states* or constraints on *execution paths*.

- Temporal logic provides a formal, expressive, and compact notation for realizing such requirements.

- The temporal logics we consider are also strongly tied to various computational frameworks (e.g., automata theory) which provides a foundation for building verification tools.
Temporal Logics

- Express properties of event orderings in time

- **Linear Time**
  - Every moment has a unique successor
  - Infinite sequences (words)
  - Linear Temporal Logic (LTL)

- **Branching Time**
  - Every moment has several successors
  - Infinite tree
  - Computation Tree Logic (CTL)
Computational Tree Logic (CTL)

Syntax

\( \Phi ::= \) P
\( | \) !\( \Phi \) | \( \Phi \) && \( \Phi \) | \( \Phi \) | | \( \Phi \) | \( \Phi \) \( \rightarrow \) \( \Phi \)
\( | \) AG \( \Phi \) | EG \( \Phi \) | AF \( \Phi \) | EF \( \Phi \)
\( | \) AX \( \Phi \) | EX \( \Phi \) | A[\( \Phi \) U \( \Phi \)] | E[\( \Phi \) U \( \Phi \)]

Semantic Intuition

AG \( p \)  ...along All paths \( p \) holds **Globally**
EG \( p \)  ...there Exists a path where \( p \) holds **Globally**
AF \( p \)  ...along All paths \( p \) holds at some state in the **Future**
EF \( p \)  ...there Exists a path where \( p \) holds at some state in the **Future**
Computational Tree Logic (CTL)

Syntax

\[ \Phi ::= \text{P} \]
\[ | \neg \Phi \]
\[ | \Phi \&\& \Phi \]
\[ | \Phi \mid \Phi \]
\[ | \Phi \rightarrow \Phi \]
\[ | \text{AG} \Phi \]
\[ | \text{EG} \Phi \]
\[ | \text{AF} \Phi \]
\[ | \text{EF} \Phi \]
\[ | \text{AX} \Phi \]
\[ | \text{EX} \Phi \]
\[ | \text{A[} \Phi \cup \Phi \text{]} \]
\[ | \text{E[} \Phi \cup \Phi \text{]} \]

Semantic Intuition

\[ \text{AX} \ p \]
...along \textit{All} paths, \( p \) holds in the \textit{neXt} state

\[ \text{EX} \ p \]
...there \textit{Exists} a path where \( p \) holds in the \textit{neXt} state

\[ \text{A[} p \cup q \text{]} \]
...along \textit{All} paths, \( p \) holds \textit{Until} \( q \) holds

\[ \text{E[} p \cup q \text{]} \]
...there \textit{Exists} a path where \( p \) holds \textit{Until} \( q \) holds
Computation Tree Logic

AG p

Diagram of a computation tree logic with AG operator and propositions p.
Computation Tree Logic

EG p
Computation Tree Logic

$\text{AF } p$
Computation Tree Logic

EF p
Computation Tree Logic

AX p
Computation Tree Logic

EX p
Computation Tree Logic

\[ A[p \bigcup q] \]
Computation Tree Logic

$E[p U q]$
Example CTL Specifications

For any state, a request (e.g., for some resource) will eventually be acknowledged

\[ \text{AG(requested} \rightarrow \text{AF acknowledged)} \]

From any state, it is possible to get to a restart state

\[ \text{AG(EF restart)} \]

An upwards travelling elevator at the second floor does not change its direction when it has passengers waiting to go to the fifth floor

\[ \text{AG((floor=2} \& \& \text{direction=up} \& \& \text{button5pressed) } \rightarrow \text{A[direction=up} \cup \text{ floor=5]}) \]
CTL Example

LEGEND:  
- ● p holds  
- ○ q holds  
- ○ don't care
CTL Semantics

- \( M, s \models p \) if \( p \in L(s) \)
- \( M, s \models \neg p \) if not \( M, s \models p \)
- \( M, s \models p \land q \) if \( M, s \models p \) and \( M, s \models q \)
- \( M, s \models p \lor q \) if \( M, s \models p \) or \( M, s \models q \)
- \( M, s \models Ap \) if \( \forall \pi \in \pi(s) : M, \pi \models p \)
- \( M, s \models Ep \) if \( \exists \pi \in \pi(s) : M, \pi \models p \)
CTL Semantics

• $M, \pi \models Xp$ if $M, \pi_1 \models p$

• $M, \pi \models Fp$ if $\exists i \geq 0: M, \pi_i \models p$

• $M, \pi \models Gp$ if $\forall i \geq 0: M, \pi_i \models p$

• $M, \pi \models pUq$ if $\exists i \geq 0: M, \pi_i \models q$ and $\forall j < i: M, \pi_j \models p$

$M \models p$ if $M, s_0 \models p$
CTL Satisfiability

• If a CTL formula is satisfiable, then the formula is satisfiable by a finite Kripke model.

• CTL Model Checking: $O(|p| \cdot (|S|+|R|))$
Example: traffic light controller

- Guarantee no collisions
- Guarantee eventual service
Specifications

• Safety (no collisions)

  \[ AG \neg (E_{Go} \land (N_{Go} \lor S_{Go})) ; \]

• Liveness

  \[ AG (\neg N_{Go} \land N_{Sense} \Rightarrow AF N_{Go}) ; \]
  \[ AG (\neg S_{Go} \land S_{Sense} \Rightarrow AF S_{Go}) ; \]
  \[ AG (\neg E_{Go} \land E_{Sense} \Rightarrow AF E_{Go}) ; \]

• Fairness constraints

  \[ AF \neg (N_{Go} \land N_{Sense}) ; \]
  \[ AF \neg (S_{Go} \land S_{Sense}) ; \]
  \[ AF \neg (E_{Go} \land E_{Sense}) ; \]
# Equivalence

<table>
<thead>
<tr>
<th>EXp</th>
<th>EGp</th>
<th>E(pUq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXp</td>
<td>$\equiv \neg EX \neg p$</td>
<td></td>
</tr>
<tr>
<td>AFp</td>
<td>$\equiv \neg EG \neg p$</td>
<td></td>
</tr>
<tr>
<td>AGp</td>
<td>$\equiv \neg EF \neg p$</td>
<td></td>
</tr>
<tr>
<td>A(pUq)</td>
<td>$\equiv \neg E(\neg pR \neg q)$</td>
<td></td>
</tr>
<tr>
<td>EFp</td>
<td>$\equiv E(\text{true } U p)$</td>
<td></td>
</tr>
</tbody>
</table>
CTL Model Checking

• Six Cases:
  – p is an atomic proposition
  – p = \neg q
  – p = q \lor r
  – p = \text{EX}q
  – p = \text{EG}q
  – p = \text{E} (q \text{ Ur})
Example: Microwave Oven
CTL Specification

• We would like the microwave to have the following properties (among others):
  – No heat while door is open
    • $\text{AG}( \text{Heat} \rightarrow \text{Close})$:
  – If oven starts, it will eventually start cooking
    • $\text{AG} (\text{Start} \rightarrow \text{AF Heat})$
  – It must be possible to correct errors
    • $\text{AG} (\text{Error} \rightarrow \text{AF } \neg \text{Error})$:

• Does it? How do we prove it?
CTL Model Checking Algorithm

• Iterate over subformulas of $f$ from smallest to largest
  – For each $s \in S$, if subformula is true in $s$, add it to $\text{labels}(s)$

• When algorithm terminates
  – $M,s \models f$ iff $f \in \text{labels}(s)$
Checking Subformulas

• Any CTL formula can be expressed in terms of: \(\neg, \lor, \text{EX}, \text{EU}, \text{and EG}\), therefore must consider 6 cases:

Atomic proposition
  if \(ap \in L(s)\), add to \(labels(s)\)

\(\neg f_1\)
  if \(f_1 \notin labels(s)\), add \(\neg f_1\) to \(labels(s)\)

\(f_1 \lor f_2\)
  if \(f_1 \in labels(s)\) or \(f_1 \in labels(s)\), add \(f_1 \lor f_2\) to \(labels(s)\)

\(\text{EX } f_1\)
  add \(\text{EX } f_1\) to \(labels(s)\) if successor of \(s, s'\), has \(f_1 \in labels(s')\)
Checking Subformulas

- $E[f_1 \cup f_2]$
  - Find all states $s$ for which $f_2 \in \text{labels}(s)$
  - Follow paths backwards from $s$ finding all states that can reach $s$ on a path in which every state is labeled with $f_1$
  - Label each of these states with $E[f_1 \cup f_2]$
Checking Subformulas

• **EG** $f_1$  Basic idea – look for one infinite path on which $f_1$ holds.

• Decompose $M$ into nontrivial strongly connected components
  – A strongly connected component (SCC) $C$ is
    • a maximal subgraph such that every node in $C$ is reachable by every other node in $C$ on a directed path that contained entirely within $C$.
  – $C$ is nontrivial iff either
    • it has more than one node or
    • it contains one node with a self loop

• Create $M' = (S', R', L')$ from $M$ by removing all states $s \in S$ in which $f_1 \notin labels(s)$ and updating $S$, $R$, and $L$ accordingly
Checking Subformulas

- Lemma $M,s \models EG f_1$ iff

1. $s \in S'$

2. There exists a path in $M'$ that leads from $s$ to some node $t$ in a nontrivial strongly connected component of the graph $(S', R', L')$.

- Proof left as exercise, but basic idea is
  - Can’t have an infinite path over finite states without cycles
  - So if we find a path from $s$ to a cycle and $f_1$ holds in every state (by construction), then we’ve found an infinite path over which $f_1$ holds
Checking EG $f_1$

```latex
\begin{procedure}
  \textbf{CheckEG}(f_1)
  \begin{align*}
  S' &= \{ s \mid f_1 \in \text{labels}(s) \}; \\
  \text{SCC} &= \{ C \mid C \text{ is a nontrivial SCC of } S' \}; \\
  T &= \bigcup_{C \in \text{SCC}} \{ s \mid s \in C \}; \\
  \text{for all } s \in T \text{ do } \text{labels}(s) = \text{labels}(s) \cup \{ \text{EG } f_1 \}; \\
  \text{while } T \neq \emptyset \text{ do}
  \begin{align*}
    \text{choose } s \in T; \\
    T &= T \setminus \{ s \}; \\
    \text{for all } t \text{ such that } t \in S' \text{ and } R(t,s) \text{ do}
    \begin{align*}
      \text{if } \text{EG } f_1 \notin \text{labels}(t) \text{ then}
        &\quad \text{labels}(t) = \text{labels}(t) \cup \{ \text{EG } f_1 \}; \\
        &\quad T = T \cup \{ t \};
    \end{align*}
    \end{align*}
  \end{align*}
  \text{end while;}
  \end{procedure}
```
Checking a Property

- Checking $\text{AG}(\text{Start} \rightarrow \text{AF Heat})$
  - Rewrite as $\neg \text{EF}(\text{Start} \land \text{EG} \neg \text{Heat})$
  - Rewrite as $\neg \text{E}[\text{true U} (\text{Start} \land \text{EG} \neg \text{Heat})]$

- Compute labels for smallest subformulas
  - Start, Heat
  - $\neg$ Heat

<table>
<thead>
<tr>
<th>Formulas/States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Start}$</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Heat}$</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>$\neg \text{Heat}$</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{EG} \neg \text{Heat}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Start} \land \text{EG} \neg \text{Heat}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{E[true U}(\text{Start} \land \text{EG} \neg \text{Heat})\text{]}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\neg \text{E[true U}(\text{Start} \land \text{EG} \neg \text{Heat})\text{]}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Checking a Property

- Compute labels for $\text{EG} \neg \text{Heat}$
- $S' = \{1,2,3,5,6\}$
- $\text{SCC} = \{\{1,2,3,5\}\}$
- $T = \{1,2,3,5\}$
- No other state in $S'$ can reach a state in $T$ along a path in $S'$.
- Computation terminates. States 1, 2, 3, and 5 labelled with $\text{EG} \neg \text{Heat}$

<table>
<thead>
<tr>
<th>Formulas/States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Start}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\text{Heat}$</td>
<td></td>
<td></td>
<td></td>
<td>$\times$</td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\neg \text{Heat}$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$\text{EG} \neg \text{Heat}$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Start} \land \text{EG} \neg \text{Heat}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{E}[\text{true U} (\text{Start} \land \text{EG} \neg \text{Heat})]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\neg \text{E}[\text{true U} (\text{Start} \land \text{EG} \neg \text{Heat})]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Checking a Property

- Compute labels for \( \text{Start} \land \text{EG} \, \neg \text{Heat} \)

<table>
<thead>
<tr>
<th>Formulas/States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Start}</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>\text{Heat}</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>\neg \text{Heat}</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>\text{EG} \neg \text{Heat}</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>\text{Start} \land \text{EG} \neg \text{Heat}</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\text{E}[\text{true} , U(\text{Start} \land \text{EG} \neg \text{Heat})]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\neg \text{E}[\text{true} , U(\text{Start} \land \text{EG} \neg \text{Heat})]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Checking a Property

- \( E[true \ U(Start \land EG \neg Heat)] \)
- Start with set of states in which \( Start \land EG \neg Heat \) holds i.e., \{2,5\}
- Work backwards marking every state in which \( true \) holds

<table>
<thead>
<tr>
<th>Formulas/States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Heat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>( \neg Heat )</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>EG ( \neg Heat )</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>( Start \land EG \neg Heat )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( E[true \ U(Start \land EG \neg Heat)] )</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>( \neg E[true \ U(Start \land EG \neg Heat)] )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Checking a Property

- Check $\neg E[true \cup (Start \land EG \neg Heat)]$
- Leaves us with the empty set, so this property doesn’t hold over our microwave oven

<table>
<thead>
<tr>
<th>Formulas/States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Start$</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Heat$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>$\neg Heat$</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>$EG \neg Heat$</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>$Start \land EG \neg Heat$</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>$E[true \cup (Start \land EG \neg Heat)]$</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>$\neg E[true \cup (Start \land EG \neg Heat)]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Genealogy

- **Floyd/Hoare** late 60s
- **Aristotle 300’s BCE**
- **Kripke 59**

- **CTL Model Checking**
  - **Clarke/Emerson** Early 80’s
  - **Pnueli** late 70’s
  - **ω-automata** late 60s
  - **LTL Model Checking**
  - **Logics of Programs**
    - **Büchi, 60**
    - **Kurshan** mid 80’s
    - **Vardi/Wolper** mid 80’s
  - **ATV**
  - **BDD** mid 80’s

- **Temporal/Modal Logics**
  - **Clarke/Emerson** Early 80’s
  - **Clarke/Emerson** Early 80’s

- **Symbolic Model Checking**
  - **Clarke/Emerson** Early 80’s
  - **Büchi, 60**
  - **Kurshan** mid 80’s
  - **Vardi/Wolper** mid 80’s
  - **ATV**
  - **BDD** mid 80’s

- **Tarski** 50’s
  - **Park, 60’s**
  - **μ-Calculus**
  - **QBF** late 80’s
Turing Awards in Verification

   Temporal logics for specifying system behavior

2. Edmund Clarke, Allen Emerson, and Joseph Sifakis (2007)
   Development of model checking