Discrete Event Systems
Verification of Finite Automata (Part 2)

Lana Josipović
Digital Systems and Design Automation Group
dynamo.ethz.ch

ETH Zurich (D-ITET)
December 7, 2023

Most materials from Lothar Thiele and Romain Jacob
Last week in
Discrete Event Systems
Verification Scenarios

Example

\[ y = (x_1 + x_2) \cdot x_3 \]

“The device can always be switched off.”

Comparison of specification and implementation

- reference system → data structure
- system under test → data structure
- comparison

Proving properties

- property
- system under test → data structure
- fixed-point calculation
Comparison using BDDs

• Boolean (combinatorial) circuits: Compare specification and implementation, or compare two implementations.

• Method:
  • Representation of the two systems in ROBDDs, e.g., by applying the APPLY operator repeatedly.
  • Compare the structures of the ROBDDs.

• Example:
Sets and Relations

• Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset $A$ is represented by $a \in A \iff \psi_A(\sigma(a))$

• Relation function: describe state transitions
  $\psi_\delta(\sigma(q), \sigma(q')) = \psi_\delta(q, q')$

\[
\sigma(e_1) = (0, 1, 0) \\
\sigma(e_2) = (0, 0, 0) \\
\psi_A(\sigma(e_1)) = 0 \\
\psi_A(\sigma(e_2)) = 1
\]
Reachability of States

• Problem: Is a state \( q \in Q \) reachable by a sequence of state transitions?

• Method:
  • Represent set of states and the transformation relation as ROBDDs.
  • Use these representations to transform from one set of states to another. Set \( Q_i \) corresponds to the set of states reachable after \( i \) transitions.
  • Iterate the transformation until a fixed-point is reached, i.e., until the set of states does not change anymore (steady-state).

• Example:

\[
Q_0 = \{q_0\} \quad \quad \quad \quad \quad Q_1 = \{q_0, q_1\} \quad \quad \quad \quad \quad Q_2 = \{q_0, q_1, q_2\} \quad \quad \quad \quad \quad Q_3 = \{q_0, q_1, q_2\}
\]
Reachability of States

State encoding

<table>
<thead>
<tr>
<th>σ(q)</th>
<th>x_1</th>
<th>x_0</th>
<th>ψ_{Q_0}</th>
<th>ψ_{Q_1}</th>
<th>ψ_{Q_2}</th>
<th>ψ_{Q_3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>q_1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>q_2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>q_3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Characteristic function: 1 if state in set, 0 otherwise

\[ \psi_{Q_0}(q) = \overline{x_1} \cdot \overline{x_0} \]
\[ Q_0 = \{ q_0 \} \]

\[ \psi_{Q_1}(q) = \overline{x_1} \]
\[ Q_1 = \{ q_0, q_1 \} \]

\[ \psi_{Q_2}(q) = \overline{x_1} + \overline{x_0} \]
\[ Q_2 = \{ q_0, q_1, q_2 \} \]

\[ \psi_{Q_3}(q) = \overline{x_1} + \overline{x_0} \]
\[ Q_3 = \{ q_0, q_1, q_2 \} \]
This week in
Discrete Event Systems
Efficient state representation

- Set of states as Boolean function
- Binary Decision Diagram representation

Computing reachability

- Leverage efficient state representation
- Explore successor sets of states

Proving properties

- Temporal logic (CTL)
- Encoding as reachability problem

Today
Temporal Logic

- Verify properties of a finite automaton, for example
  - Can we always reset the automaton?
  - Is every request followed by an acknowledgement?
  - Are both outputs always equivalent?

<table>
<thead>
<tr>
<th>Formula</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic proposition</td>
<td>The printer is busy.</td>
</tr>
<tr>
<td></td>
<td>The light is on.</td>
</tr>
<tr>
<td>Boolean logic</td>
<td>$\phi_1 + \phi_2 \land \neg \phi_1$</td>
</tr>
</tbody>
</table>
Temporal Logic

• Verify properties of a finite automaton, for example
  • Can we always reset the automaton?
  • Is every request followed by an acknowledgement?
  • Are both outputs always equivalent?

• Specification of the query in a formula of temporal logic.
• We use a simple form called Computation Tree Logic (CTL).

• Let us start with a minimal set of operators.
  • Any atomic proposition is a CTL formula.
  • CTL formula are constructed by composition of other CTL formula.

<table>
<thead>
<tr>
<th>Formula</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic proposition</td>
<td>The printer is busy. The light is on.</td>
</tr>
<tr>
<td>Boolean logic</td>
<td>$\phi_1 + \phi_2 ; \neg \phi_1$</td>
</tr>
<tr>
<td>CTL logic</td>
<td>EX $\phi_1$</td>
</tr>
</tbody>
</table>

There exists other logics (e.g. LTL, CTL*)
Formulation of CTL properties

Based on atomic propositions ($\phi$) and quantifiers

$A \phi \rightarrow \langle \text{All } \phi \rangle$, $\phi$ holds on all paths

$E \phi \rightarrow \langle \text{Exists } \phi \rangle$, $\phi$ holds on at least one path

$X \phi \rightarrow \langle \text{Next } \phi \rangle$, $\phi$ holds on the next state

$F \phi \rightarrow \langle \text{Finally } \phi \rangle$, $\phi$ holds at some state along the path

$G \phi \rightarrow \langle \text{Globally } \phi \rangle$, $\phi$ holds on all states along the path

$\phi_1 U \phi_2 \rightarrow \langle \phi_1 \text{ Until } \phi_2 \rangle$, $\phi_1$ holds until $\phi_2$ holds

implies that $\phi_2$ has to hold eventually

CTL quantifiers work in pairs: we need one of each! $\{A,E\} \{X,F,G,U\}$
A \phi \rightarrow \text{All } \phi, \quad \phi \text{ holds on all paths}

E \phi \rightarrow \text{Exists } \phi, \quad \phi \text{ holds on at least one path}

X \phi \rightarrow \text{NeXt } \phi, \quad \phi \text{ holds on the next state}

F \phi \rightarrow \text{Finally } \phi, \quad \phi \text{ holds at some state along the path}

G \phi \rightarrow \text{Globally } \phi, \quad \phi \text{ holds on all states along the path}

\phi_1 U \phi_2 \rightarrow \text{Until } \phi_2, \quad \phi_1 \text{ holds until } \phi_2 \text{ holds}

\text{implies that } \phi_2 \text{ has to hold eventually}

\text{CTL quantifiers work in pairs: we need one of each!} \quad \{A,E\} \{X,F,G,U\} \phi
CTL works on computation trees

Over paths:
\( A\phi \rightarrow \text{All } \phi \)
\( E\phi \rightarrow \text{Exists } \phi \)
\( E\phi \rightarrow \text{exists } \phi \)
\( F\phi \rightarrow \text{Finally } \phi \)
\( G\phi \rightarrow \text{Globally } \phi \)
\( \phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2 \)

Path-specific:
\( X\phi \rightarrow \text{Next } \phi \)

Automaton

Computation tree
CTL works on computation trees

Automaton of interest

Requires fully-defined transition functions

Automaton to work with

Each state has at least one successor (can be itself)

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Visualizing CTL formula

- We use this computation tree as a running example.

- We suppose that the black and red states satisfy atomic properties $p$ and $q$, respectively.

- The topmost state is the initial state; in the examples, it always satisfies the given formula.

$M$ satisfies $\phi \iff q_0 \models \phi$ where $q_0$ is the initial state of $M$
Visualizing CTL formula

Over paths:

$A\phi \rightarrow \text{All } \phi$

$E\phi \rightarrow \text{Exists } \phi$

Path-specific:

$X\phi \rightarrow \text{Next } \phi$

$F\phi \rightarrow \text{Finally } \phi$

$G\phi \rightarrow \text{Globally } \phi$

$\phi_1 \mathbin{\mathcal{U}} \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Visualizing CTL formula

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 \cup \phi_2 \rightarrow \text{Until } \phi_2$
Visualizing CTL formula

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

$AX p$

...
Visualizing CTL formula

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U\phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Visualizing CTL formula

\[ EG \, p \]

Over paths:
- \( A\phi \rightarrow \text{All } \phi \)
- \( E\phi \rightarrow \text{Exists } \phi \)

Path-specific:
- \( X\phi \rightarrow \text{Next } \phi \)
- \( F\phi \rightarrow \text{Finally } \phi \)
- \( G\phi \rightarrow \text{Globally } \phi \)
- \( \phi_1 \cup \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2 \)
Visualizing CTL formula

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Visualizing CTL formula

EX p
Visualizing CTL formula

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{NeXt } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

EX p

p EU q
Formulation of CTL properties

Can be more than one pair

\[ \text{AG } \phi_1 \text{ where } \phi_1 = \text{EF } \phi_2 \equiv \text{AG EF } \phi_2 \]

A and F are convenient, but not necessary

\[ \text{AF } \phi \equiv \neg \text{EG } (\neg \phi) \]
\[ \text{AG } \phi \equiv \neg \text{EF } (\neg \phi) \]
\[ \text{AX } \phi \equiv \neg \text{EX } (\neg \phi) \]
\[ \text{EF } \phi \equiv \text{true EU } \phi \]

No need to know that one

\[ \phi_1 \text{AU } \phi_2 \equiv \neg([\neg \phi_1 ] \text{EU } \neg(\phi_1 + \phi_2)) + \text{EG } (\neg \phi_2) \]
Intuition for “AF p = ¬ EG (¬ p)”
Intuition for “AF p = ¬ EG (¬ p)”
Evaluating a CTL formula

$EF \phi$ : “There exists a path along which at some state $\phi$ holds.”

```plaintext
Over paths:
A$\phi$ → All $\phi$
E$\phi$ → Exists $\phi$
F$\phi$ → Finally $\phi$
G$\phi$ → Globally $\phi$
$\phi_1 U \phi_2$ → $\phi_1$ Until $\phi_2$
```

Path-specific:
X$\phi$ → Next $\phi$

$q \models EF \phi$

$r \not\models ?$

$s \not\models ?$

28
Evaluating a CTL formula

$\text{EF } \phi : \text{“There exists a path along which at some state } \phi \text{ holds.”}$

Over paths:

- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 \cup \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

Path-specific:

- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$

$q \models \phi$

$r \not\models \text{EF } \phi$

$s \models ?$
Evaluating a CTL formula

$\text{EF } \phi$: “There exists a path along which at some state $\phi$ holds.”

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 \cup \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$

Diagram:

- $s \models \phi$
- $q \models \text{EF } \phi$
- $r \not\models \text{EF } \phi$
- $s \not\models \text{EF } \phi$
Evaluating a CTL formula
$\text{AF } \phi$ : “On all paths, at some state $\phi$ holds.”
Evaluating a CTL formula
AF $\phi$ : “On all paths, at some state $\phi$ holds.”

Over paths:  
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$
Path-specific:  
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Evaluating a CTL formula
AF $\phi$: "On all paths, at some state $\phi$ holds."

- $q \models AF \phi$
- $r \models AF \phi$
- $s \not\models AF \phi$
Evaluating a CTL formula

$\text{AG } \phi$: “On all paths, for all states $\phi$ holds.”

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 \cup \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

$q \models \phi$

$q \models \text{AG } \phi$

$r \models ?$

$s \models ?$
Evaluating a CTL formula

$\text{AG } \phi$: “On all paths, for all states $\phi$ holds.”

Over paths:

$A\phi \rightarrow \text{All } \phi$

$E\phi \rightarrow \text{Exists } \phi$

Path-specific:

$X\phi \rightarrow \text{Next } \phi$

$F\phi \rightarrow \text{Finally } \phi$

$G\phi \rightarrow \text{Globally } \phi$

$\phi_1 \text{ Until } \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

$q \models A\text{G } \phi$

$r \models A\text{G } \phi$

$s \models ?$
Evaluating a CTL formula
AG \phi: “On all paths, for all states \phi holds.”
Evaluating a CTL formula

$\text{EG } \phi : \text{“There exists a path along which for all states } \phi \text{ holds.”} \)
Evaluating a CTL formula

EG $\phi$ : “There exists a path along which for all states $\phi$ holds.”

Over paths:
- $A\phi \rightarrow$ All $\phi$
- $E\phi \rightarrow$ Exists $\phi$
- $G\phi \rightarrow$ Globally $\phi$
- $\phi_1 U \phi_2 \rightarrow$ $\phi_1$ Until $\phi_2$

Path-specific:
- $X\phi \rightarrow$ NeXt $\phi$
- $F\phi \rightarrow$ Finally $\phi$

Diagram:

- $q \models \phi$
- $q \models EG \phi$
- $r \models EG \phi$
- $s \models ?$
Evaluating a CTL formula

\( \text{EG } \phi : \text{“There exists a path along which for all states } \phi \text{ holds.”} \)
Evaluating a CTL formula $\phi \mathcal{E} \mathcal{U} \Psi$ : “There exists a path along which $\phi$ holds until $\Psi$ holds.”

\[ q \models \phi \]
\[ s \models \]

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 \mathcal{U} \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$

\[ q \models \phi \mathcal{E} \mathcal{U} \Psi \]
\[ r \models ? \]
\[ s \models ? \]
Evaluating a CTL formula $\phi E U \Psi$ : “There exists a path along which $\phi$ holds until $\Psi$ holds.”
Evaluating a CTL formula $\phi E \Psi$ : “There exists a path along which $\phi$ holds until $\Psi$ holds.”
Evaluating a CTL formula $\varphi \mathbf{A} \mathbf{U} \varPsi$ : “On all paths, $\varphi$ holds until $\varPsi$ holds.”

Over paths:
- $A\varphi \rightarrow \text{All } \varphi$
- $E\varphi \rightarrow \text{Exists } \varphi$
- $F\varphi \rightarrow \text{Finally } \varphi$
- $G\varphi \rightarrow \text{Globally } \varphi$
- $\varphi_1 \mathbf{U} \varphi_2 \rightarrow \varphi_1 \text{ Until } \varphi_2$

Path-specific:
- $X\varphi \rightarrow \text{Next } \varphi$

$\{ \square \models \varPsi \}
\{ \bigcirc \models \varphi \}
q \models \varphi \mathbf{A} \mathbf{U} \varPsi
r \models ?
s \models ?$
Evaluating a CTL formula \( \phi \text{A} \text{U} \Psi \): “On all paths, \( \phi \) holds until \( \Psi \) holds.”
Evaluating a CTL formula $\phi A U \Psi$: “On all paths, $\phi$ holds until $\Psi$ holds.”
Evaluating a CTL formula $\text{EX}\phi$ : “There exists a path along which the next state satisfies $\phi$.”
Evaluating a CTL formula $\text{EX}\phi$: “There exists a path along which the next state satisfies $\phi$."

\begin{itemize}
  \item $A\phi \rightarrow \text{All } \phi$
  \item $E\phi \rightarrow \text{Exists } \phi$
  \item $F\phi \rightarrow \text{Finally } \phi$
  \item $G\phi \rightarrow \text{Globally } \phi$
  \item $\phi_1 \text{U}\phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
\end{itemize}
Evaluating a CTL formula $\text{EX}\varphi$: “There exists a path along which the next state satisfies $\varphi$."

Over paths: $A\varphi \rightarrow A\text{ll} \varphi$, $E\varphi \rightarrow E\text{xists} \varphi$, $F\varphi \rightarrow F\text{inally} \varphi$, $G\varphi \rightarrow G\text{lobally} \varphi$, $\varphi_1 U \varphi_2 \rightarrow \varphi_1 U\text{ntil} \varphi_2$

Path-specific: $X\varphi \rightarrow \text{NeXt} \varphi$

q $\models \varphi$
q $\models \text{EX}\varphi$
r $\models \text{EX}\varphi$
s $\not\models \text{EX}\varphi$
Evaluating a CTL formula

\( AG \ EF \ \phi \): “On all paths and for all states, there exists a path along which at some state \( \phi \) holds.”

\[ q \vDash AG EF \phi \]

\[ r \vdash ? \]

\[ s \vdash ? \]
Evaluating a CTL formula
\( \text{AG EF } \phi : \text{“On all paths and for all states, there exists a path along which at some state } \phi \text{ holds.”} \)
Evaluating a CTL formula

**AG EF ϕ**: “On all paths and for all states, there exists a path along which at some state ϕ holds.”
Evaluating a CTL formula

$\text{AG}\ EF\ \phi : \ "\text{On all paths and for all states, there exists a path along which at some state }\phi\ \text{holds.}"$

What if we remove this edge?
Evaluating a CTL formula \( \mathbf{AG \ EF \ \phi} \): “On all paths and for all states, there exists a path along which at some state \( \phi \) holds.”
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \text{Until } \phi_2$
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- AG p

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- $\text{AG } p$  
  I will like chocolate from now on, no matter what happens.

Over paths:
- $\text{A} \phi \rightarrow \text{All } \phi$
- $\text{E} \phi \rightarrow \text{Exists } \phi$

Path-specific:
- $\text{X} \phi \rightarrow \text{Next } \phi$
- $\text{F} \phi \rightarrow \text{Finally } \phi$
- $\text{G} \phi \rightarrow \text{Globally } \phi$
- $\phi_1 \text{U} \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- AG p  I will like chocolate from now on, no matter what happens.
- EF p

Over paths:
- A\(\phi\) → All \(\phi\)
- E\(\phi\) → Exists \(\phi\)

Path-specific:
- X\(\phi\) → NeXt \(\phi\)
- F\(\phi\) → Finally \(\phi\)
- G\(\phi\) → Globally \(\phi\)
- \(\phi_1\) U\(\phi_2\) → \(\phi_1\) Until \(\phi_2\)
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- **AG p**  
  I will like chocolate from now on, no matter what happens.

- **EF p**  
  It's possible I may like chocolate someday, at least for one day.
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- **AG p**  
  I will like chocolate from now on, no matter what happens.

- **EF p**  
  It's possible I may like chocolate someday, at least for one day.

- **AF EG p**
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- AG p  
  I will like chocolate from now on, no matter what happens.

- EF p  
  It's possible I may like chocolate someday, at least for one day.

- AF EG p  
  There will be always sometime in the future (AF) that I may suddenly start liking chocolate for the rest of time (EG).
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

| AG p      | I will like chocolate from now on, no matter what happens. |
| EF p      | It's possible I may like chocolate someday, at least for one day. |
| AF EG p   | There will be always sometime in the future (AF) that I may suddenly start liking chocolate for the rest of time (EG). |
| EG AF p   | |

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- **AG p**
  I will like chocolate from now on, no matter what happens.

- **EF p**
  It's possible I may like chocolate someday, at least for one day.

- **AF EG p**
  There will be always sometime in the future (AF) that I may suddenly start liking chocolate for the rest of time (EG).

- **EG AF p**
  This is a critical time in my life. Depending on what happens (E), it's possible that for the rest of time (G), there will always be some time in the future (AF) when I will like chocolate. However, if the wrong thing happens next, then all bets are off and there's no guarantee about whether I will ever like chocolate.
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- **AG p**  
  I will like chocolate from now on, no matter what happens.

- **EF p**  
  It’s possible I may like chocolate someday, at least for one day.

- **AF EG p**  
  There will be always sometime in the future (AF) that I may suddenly start liking chocolate for the rest of time (EG).

- **EG AF p**  
  This is a critical time in my life. Depending on what happens (E), it's possible that for the rest of time (G), there will always be some time in the future (AF) when I will like chocolate. However, if the wrong thing happens next, then all bets are off and there's no guarantee about whether I will ever like chocolate.

- **p AU q**
Interpreting a CTL formula

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Proposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>I like chocolate</td>
</tr>
<tr>
<td>q</td>
<td>It's warm outside</td>
</tr>
</tbody>
</table>

- **AG p**  
  I will like chocolate from now on, no matter what happens.

- **EF p**  
  It's possible I may like chocolate someday, at least for one day.

- **AF EG p**  
  There will be always sometime in the future (AF) that I may suddenly start liking chocolate for the rest of time (EG).

- **EG AF p**  
  This is a critical time in my life. Depending on what happens (E), it's possible that for the rest of time (G), there will always be some time in the future (AF) when I will like chocolate. However, if the wrong thing happens next, then all bets are off and there's no guarantee about whether I will ever like chocolate.

- **p AU q**  
  No matter what happens, I will like chocolate from now on. But when it gets warm outside, I don’t know whether I still like it. And it will get warm outside someday.
Specifying using a CTL formula

Famous problem

Dining Philosophers

- Five philosophers are sitting around a table, taking turns at thinking and eating.
- Each needs two forks to eat.
- They put down forks only once they have eaten.
- There are only five forks.

Atomic proposition

\( e_i \): Philosopher \( i \) is currently eating.
Specifying using a CTL formula

- “Philosophers 1 and 4 will never eat at the same time.”

- “Every philosopher will get infinitely many turns to eat.”

- “Philosopher 2 will be the first to eat.”
Specifying using a CTL formula

- “Philosophers 1 and 4 will never eat at the same time.”
  
  \[ \text{AG} \neg (e_1 \cdot e_4) \]

- “Every philosopher will get infinitely many turns to eat.”

- “Philosopher 2 will be the first to eat.”

Over paths:
- \( A\phi \rightarrow \text{All } \phi \)
- \( E\phi \rightarrow \text{Exists } \phi \)

Path-specific:
- \( X\phi \rightarrow \text{Next } \phi \)
- \( F\phi \rightarrow \text{Finally } \phi \)
- \( G\phi \rightarrow \text{Globally } \phi \)
- \( \phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2 \)
Specifying using a CTL formula

- “Philosophers 1 and 4 will never eat at the same time.”
  \[ AG\neg(e_1 \cdot e_4) \]

- “Every philosopher will get infinitely many turns to eat.”
  \[ AG(AFe_1 \cdot AFe_2 \cdot AFe_3 \cdot AFe_4 \cdot AFe_5) \]

- “Philosopher 2 will be the first to eat.”
Specifying using a CTL formula

- “Philosophers 1 and 4 will never eat at the same time.”
  \[ \text{AG} \neg (e_1 \cdot e_4) \]

- “Every philosopher will get infinitely many turns to eat.”
  \[ \text{AG} (\text{AF} e_1 \cdot \text{AF} e_2 \cdot \text{AF} e_3 \cdot \text{AF} e_4 \cdot \text{AF} e_5) \]

- “Philosopher 2 will be the first to eat.”
  \[ \neg (e_1 + e_3 + e_4 + e_5) \text{ AU } e_2 \]
Computing a CTL formula

1. Define $[\phi]$ as the set of all states of the finite automaton for which CTL formula $\phi$ is true.
2. A finite automaton with initial state $q_0$ satisfies $\phi$ iff
   \[ q_0 \in [\phi] \]

- Now, we can use our “trick”: computing with sets of states!
  - $\psi_{[\phi]}(q)$ is true if the state $q$ is in the set $[\phi]$, i.e., it is a state for which the CTL formula is true.
  - Therefore, we can also say
    \[ q_0 \in [\phi] \equiv \psi_{[\phi]}(q_0) \]  
    characteristic function of the set $[\phi]$
Computing a CTL formula: EX $\phi$

Over paths:
- $A\phi \rightarrow \text{All } \phi$
- $E\phi \rightarrow \text{Exists } \phi$

Path-specific:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
Computing a CTL formula: EX $\phi$

- Suppose that $Q$ is the set of states for which the formula $\phi$ is true.

Sets

$Q = \llbracket \phi \rrbracket$

Characteristic functions

$\psi_Q(q)$
Computing a CTL formula: EX \( \phi \)

- Suppose that \( Q \) is the set of states for which the formula \( \phi \) is true.
- \( Q' \) is the set of predecessor states of \( Q \), i.e., the set of states that lead in one transition to a state in \( Q \):

\[
Q' = \text{Pre}(Q, \delta) = \{ q' \mid \exists q : \psi_\delta(q', q) \cdot \psi_Q(q) \}
\]

Sets

\[
Q = [\phi] \quad \rightarrow \quad Q' = [\text{EX}\phi] = \text{Pre}([\phi], \delta)
\]

Characteristic functions

\[
\psi_Q(q) \quad \rightarrow \quad \psi_{Q'}(q') = (\exists q : \psi_Q(q) \cdot \psi_\delta(q', q))
\]
Computing a CTL formula: EX φ

- Example for EX φ: Compute EX q₂

1. Define $\llbracket EX q₂ \rrbracket$: set of all states of the finite automaton for which CTL formula EX $q₂$ is true.

$\llbracket q₂ \rrbracket = \{q₂\}$
Computing a CTL formula: EX \( \phi \)

- Example for \( \text{EX} \ \phi \): Compute \( \text{EX} \ q_2 \)

1. Define \([\text{EX} \ q_2]\): set of all states of the finite automaton for which CTL formula \( \text{EX} \ q_2 \) is true.

\[
[q_2] = \{q_2\}
\]

\[
Q' = [\text{EX} \ q_2] = \text{Pre}([q_2], \delta) = \{q_1, q_2, q_3\}
\]

\[
\{q' \mid \exists q : \psi_\delta(q', q) \cdot \psi_Q(q)\}
\]
Computing a CTL formula: \( \text{EX} \ \phi \)

- Example for \( \text{EX} \ \phi \): Compute \( \text{EX} \ q_2 \)

1. Define \([\text{EX} \ q_2]\): set of all states of the finite automaton for which CTL formula \( \text{EX} \ q_2 \) is true.

\[
[q_2] = \{q_2\}
\]

\[
Q' = [\text{EX} \ q_2] = \text{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\}
\]

\[
\{q' \mid \exists q : \psi_\delta(q', q) \cdot \psi_Q(q)\}
\]

2. A finite automaton with initial state \( q_0 \) satisfies \( \text{EX} \ q_2 \) iff \( q_0 \in [\text{EX} \ q_2] \)

As \( q_0 \notin [\text{EX} \ q_2] = \{q_1, q_2, q_3\} \), the CTL formula \( \text{EX} \ q_2 \) is not true.
Computing a CTL formula: EF $\phi$

- Start with the set of states for which the formula $\phi$ is true.
- Add to this set the set of predecessor states. Repeat for the resulting set of states,..., until we reach a fixed point.

$$Q_0 = \llbracket \phi \rrbracket$$

$$Q_i = Q_{i-1} \cup \text{Pre}(Q_{i-1}, \delta) \quad \text{for all } i > 1 \text{ until a fixed point } Q' \text{ is reached}$$

$$\llbracket \text{EF} \phi \rrbracket = Q'$$
Computing a CTL formula: EF $\phi$

- Example for EF$\phi$: Compute EF $q_2$

1. Define $[\{EF \; q_2\}]$: set of all states of the finite automaton for which CTL formula EF $q_2$ is true.

$$Q_0 = [\{q_2\}] = \{q_2\}$$
Computing a CTL formula: EF $\phi$

- Example for $\text{EF}\phi$: Compute $\text{EF} \, q_2$

1. Define $\llbracket \text{EF} \, q_2 \rrbracket$: set of all states of the finite automaton for which CTL formula $\text{EF} \, q_2$ is true.

$$Q_0 = \llbracket q_2 \rrbracket = \{q_2\}$$

$$Q_1 = \{q_2\} \cup \text{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\}$$
Computing a CTL formula: EF $\phi$

- Example for EF$\phi$: Compute EF $q_2$

1. Define $\llbracket EF q_2 \rrbracket$: set of all states of the finite automaton for which CTL formula EF $q_2$ is true.

$$Q_0 = \llbracket q_2 \rrbracket = \{q_2\}$$

$$\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q) \} = \{q_1, q_2, q_3\}$$

$$Q_1 = \{q_2\} \cup \text{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\}$$

$$Q_2 = \{q_1, q_2, q_3\} \cup \text{Pre}(\{q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$$
Computing a CTL formula: EF $\phi$

- Example for EF$\phi$: Compute EF $q_2$

1. Define $[EF q_2]$: set of all states of the finite automaton for which CTL formula EF $q_2$ is true.

$$\begin{align*}
Q_0 &= [q_2] = \{q_2\} = \{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_{\delta}(q', q) \} = \{q_1, q_2, q_3\} \\
Q_1 &= \{q_2\} \cup \text{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\} \\
Q_2 &= \{q_1, q_2, q_3\} \cup \text{Pre}(\{q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\} \\
Q_3 &= \{q_0, q_1, q_2, q_3\} \cup \text{Pre}(\{q_0, q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\} \\
[EF q_2] &= Q_3 = \{q_0, q_1, q_2, q_3\}
\end{align*}$$
Computing a CTL formula: EF $\phi$

- **Example for EF$\phi$:** Compute EF $q_2$
  
  1. Define $[EF q_2]$: set of all states of the finite automaton for which CTL formula EF $q_2$ is true.

  $Q_0 = [q_2] = \{q_2\}$
  
  $Q_1 = \{q_2\} \cup \text{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\}$
  
  $Q_2 = \{q_1, q_2, q_3\} \cup \text{Pre}(\{q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$
  
  $Q_3 = \{q_0, q_1, q_2, q_3\} \cup \text{Pre}(\{q_0, q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$

  $[EF q_2] = Q_3 = \{q_0, q_1, q_2, q_3\}$

  2. A finite automaton with initial state $q_0$ satisfies EF $q_2$ iff $q_0 \in [EF q_2]$

  As $q_0 \in [EF q_2] = \{q_0, q_1, q_2, q_3\}$, the CTL formula EF $q_2$ is true.
Computing a CTL formula: EG \( \phi \)

- Start with the set of states for which the formula \( \phi \) is true.
- Cut this set with the set of predecessor states. Repeat for the resulting set of states,..., until we reach a fixed point.

\[
Q_0 = \llbracket \phi \rrbracket \\
Q_i = Q_{i-1} \cap \text{Pre}(Q_{i-1}, \delta) \quad \text{for all } i > 1 \text{ until a fixed point } Q' \text{ is reached}
\]
Computing a CTL formula: EG $\phi$

- Example for EG $\phi$: Compute $\text{EG } q_2$

1. Define $\llbracket \text{EG } q_2 \rrbracket$: set of all states of the finite automaton for which CTL formula EG $q_2$ is true.

$Q_0 = \llbracket q_2 \rrbracket = \{ q_2 \}$
Computing a CTL formula: $\text{EG } \phi$

- Example for $\text{EG } \phi$: Compute $\text{EG } q_2$

1. Define $[\text{EG } q_2]$: set of all states of the finite automaton for which CTL formula $\text{EG } q_2$ is true.

   $Q_0 = [q_2] = \{q_2\}$

   $\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_1, q_2, q_3\}$

   $Q_1 = \{q_2\} \cap \text{Pre}(\{q_2\}, \delta) = \{q_2\}$

2. A finite automaton with initial state $q_0$ satisfies $\text{EG } q_2$ iff $q_0 \in [\text{EG } q_2]$

   As $q_0 \not\in [\text{EG } q_2] = \{q_2\}$, the CTL formula $\text{EG } q_2$ is not true.
Computing a CTL formula: $\phi_1 EU \phi_2$

- Start with the set of states for which the formula $\phi_2$ is true.
- Add to this set the set of predecessor states for which the formula $\phi_1$ is true. Repeat for the resulting set of states we do the same, ..., until we reach a fixed point.
- Like EF $\phi_2$; the only difference is that, on our path backwards, we always make sure that also $\phi_1$ holds.

$Q_0 = \llbracket \phi_2 \rrbracket$

$Q_i = Q_{i-1} \cup (\text{Pre}(Q_{i-1}, \delta) \cap \llbracket \phi_1 \rrbracket)$ for all $i > 1$ until a fixed point is reached
Computing a CTL formula: $\phi_1 EU \phi_2$

- Example for $\phi_1 EU \phi_2$: Compute $q_0 EU q_1$

1. Define $[q_0 EU q_1]$: set of all states of the finite automaton for which CTL formula $q_0 EU q_1$ is true.

$$Q_0 = [q_1] = \{q_1\}$$
Computing a CTL formula: $\phi_1 EU \phi_2$

- Example for $\phi_1 EU \phi_2$: Compute $q_0 EU q_1$

1. Define $[q_0 \ EU \ q_1]$: set of all states of the finite automaton for which CTL formula $q_0 \ EU \ q_1$ is true.

$$Q_0 = [q_1] = \{q_1\} \quad \{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_0, q_2\}$$

$$Q_1 = \{q_1\} \cup (\text{Pre}(\{q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\}$$
Computing a CTL formula: $\phi_1 EU \phi_2$

- **Example for $\phi_1 EU \phi_2$:** Compute $q_0 EU q_1$

  1. Define $[q_0 EU q_1]$: set of all states of the finite automaton for which CTL formula $q_0 EU q_1$ is true.

     \[
     Q_0 = [q_1] = \{q_1\} \\
     Q_1 = \{q_1\} \cup (\text{Pre}(\{q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\} \\
     Q_2 = \{q_0, q_1\} \cup (\text{Pre}(\{q_0, q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\} \\
     \]

     \[
     [q_0 EU q_1] = Q_2 = \{q_0, q_1\} \\
     \{q_0, q_2, q_3\}
     \]

  2. A finite automaton with initial state $q_0$ satisfies $q_0 EU q_1$ iff $q_0 \in [q_0 EU q_1]$

    As $q_0 \in [q_0 EU q_1] = \{q_0, q_1\}$, the CTL formula $q_0 EU q_1$ is true.
Computing a CTL formula: \( \phi_1 EU \phi_2 \)

- **Example for \( \phi_1 EU \phi_2 \): Compute \( q_0 \) \( EU \) \( q_1 \)

1. Define \([q_0 \) \( EU \) \( q_1]\): set of all states of the finite automaton for which CTL formula \( q_0 \) \( EU \) \( q_1 \) is true.

\[
\begin{align*}
Q_0 &= [q_1] = \{q_1\} \\
Q_1 &= \{q_1\} \cup (\text{Pre}(\{q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\} \\
Q_2 &= \{q_0, q_1\} \cup (\text{Pre}(\{q_0, q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\} \\
\end{align*}
\]

\[
[q_0 EU q_1] = Q_2 = \{q_0, q_1\} \\
\]

2. A finite automaton with initial state \( q_0 \) satisfies \( q_0 \) \( EU \) \( q_1 \) iff \( q_0 \in [q_0 \) \( EU \) \( q_1]\)

As \( q_0 \in [q_0 EU q_1] = \{q_0, q_1\} \), the CTL formula \( q_0 \) \( EU \) \( q_1 \) is true.

**Compute other CTL expressions as:**

\[
\begin{align*}
AF\phi & \equiv \neg EG(\neg \phi) & AG\phi & \equiv \neg EF(\neg \phi) & AX\phi & \equiv \neg EX(\neg \phi)
\end{align*}
\]
So... what is model checking exactly?

Model checking is an algorithm which takes two inputs
- a DES model $M$
- a formula $\phi$

It explores the state space of $M$ such as to either
- prove that $M \models \phi$, or
- return a trace where the formula does not hold in $M$. — a counter-example

Extremely useful!
- Debugging the model
- Searching a specific execution sequence

Finite automaton
Petri net
Kripke machine
...
Efficient state representation
- Set of states as Boolean function
- Binary Decision Diagram representation

Computing reachability
- Leverage efficient state representation
- Explore successor sets of states

Proving properties
- Temporal logic (CTL)
- Encoding as reachability problem

Today
Conclusion and perspectives

Next week(s)  Petri Nets

- asynchronous DES model
- tailored model concurrent distributed systems
- capture an infinite state space with a finite model

How they work?
How to use them for modeling systems?
How to verify them?
Your turn to practice!

after the break

1. Familiarise yourself with CTL logic and how to compute sets of states satisfying a given formula

2. Convert a concrete problem into a state reachability question
   (adapted from state-of-the-art research!)
Any feedback?
Please fill out this short (anonymous) form!

The form will be available throughout the lecture—feel free to provide feedback at any point.

https://forms.gle/7VUaidEVreS9uswa9

Thanks for your attention and see you next week! 😊