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 1, 2022

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
- fixed-point calculation

- system under test → data structure
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 \( a \in A \Leftrightarrow \psi_A(\sigma(a)) \)

- Relation function: describe state transitions
  \[
  \psi_\delta(\sigma(q), \sigma(q')) = \psi_\delta(q, q')
  \]

\[
\begin{align*}
\sigma(e_1) &= (0, 1, 0) \\
\sigma(e_2) &= (0, 0, 0) \\
\psi_A(\sigma(e_1)) &= 0 \\
\psi_A(\sigma(e_2)) &= 1
\end{align*}
\]
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:
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
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?
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. The light is on.</td>
</tr>
<tr>
<td>Boolean logic</td>
<td>$\phi_1 + \phi_2 ; \neg \phi_1$</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.</td>
</tr>
<tr>
<td></td>
<td>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>$\text{EX } \phi_1$</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.</td>
</tr>
<tr>
<td></td>
<td>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
Formulation of CTL properties

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

$A\phi \rightarrow \text{«All } \phi\text{»}$, $\phi$ holds on all paths

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

Quantifiers over paths
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
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
- $\phi_1 U \phi_2 \rightarrow \langle\text{Until } \phi_2\rangle$, $\phi_1$ holds until $\phi_2$ holds
  - implies that $\phi_2$ has to hold eventually

CTL quantifiers work in pairs: we need one of each! $\{A,E\}$ $\{X,F,G,U\} \phi$
Formulation of CTL properties

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

\[ A\phi \rightarrow \text{All } \phi, \]

\[ E\phi \rightarrow \text{Exists } \phi, \]

\[ X\phi \rightarrow \text{NeXt } \phi, \]

\[ F\phi \rightarrow \text{Finally } \phi, \]

\[ G\phi \rightarrow \text{Globally } \phi, \]

\[ \phi_1 U \phi_2 \rightarrow \text{Until } \phi_2, \]

implies that $\phi_2$ has to hold eventually

Quantifiers over paths

\begin{align*}
A\phi & \rightarrow \text{All } \phi \\
E\phi & \rightarrow \text{Exists } \phi \\
X\phi & \rightarrow \text{NeXt } \phi \\
F\phi & \rightarrow \text{Finally } \phi \\
G\phi & \rightarrow \text{Globally } \phi
\end{align*}

Path-specific quantifiers

\begin{align*}
\phi_1 U \phi_2 & \rightarrow \text{Until } \phi_2
\end{align*}

CTL quantifiers work in pairs: we need one of each!

$\{A,E\} \{X,F,G,U\} \phi$
CTL works on computation trees

Automaton

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

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

Over paths:
- $\forall \phi \rightarrow \text{All } \phi$
- $\exists \phi \rightarrow \text{Exists } \phi$
- $\exists_1 \phi \rightarrow \text{Exists } \phi_2$

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

Automaton

Computation tree
CTL works on computation trees

Automaton of interest

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

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

Requires fully-defined transition functions
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.

\[ M \text{ satisfies } \phi \iff q_0 \models \phi \text{ where } q_0 \text{ is the initial state of } M \]
Visualizing CTL formula

AG p

Over paths:
\[ A\phi \rightarrow \text{All} \phi \]
\[ E\phi \rightarrow \text{Exists} \phi \]

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

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

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

AG $p$

AF $p$
Visualizing CTL formula

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

Path-specific:
- $X\phi \rightarrow \text{NeXt } \phi$
Visualizing CTL formula

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

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

$AX\ p$

$p \text{ AU } q$
Visualizing CTL formula

EG p

Over paths:
- $\forall \phi \rightarrow \text{All } \phi$
- $\exists \phi \rightarrow \text{Exists } \phi$

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

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

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

EG $p$

EF $p$
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

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

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

EX $p$

$p \text{ EU } q$

3G
Formulation of CTL properties

Can be more than one pair

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

A and F are convenient, but not necessary

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

No need to know that one

\[ \phi_1 AU\phi_2 \equiv \neg([\neg \phi_1)EU(\neg(\phi_1 + \phi_2)] + EG(\neg \phi_2)) \]

E,G,X,U are sufficient to define the whole logic.
Intuition for “$\text{AF } p = \neg \text{EG } (\neg p)$”

<table>
<thead>
<tr>
<th>Over paths:</th>
<th>Path-specific:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A\phi \rightarrow \text{All } \phi$</td>
<td>$X\phi \rightarrow \text{NeXt } \phi$</td>
</tr>
<tr>
<td>$E\phi \rightarrow \text{Exists } \phi$</td>
<td>$F\phi \rightarrow \text{Finally } \phi$</td>
</tr>
<tr>
<td>$G\phi \rightarrow \text{Globally } \phi$</td>
<td>$\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$</td>
</tr>
</tbody>
</table>

\[\neg \text{ AF } (p)\]
Intuition for “AF p = ¬ EG (¬ p)”
Interpreting 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:
- $X\phi \rightarrow \text{Next } \phi$
- $F\phi \rightarrow \text{Finally } \phi$
- $G\phi \rightarrow \text{Globally } \phi$
- $\phi_1 U \phi_2 \rightarrow \phi_1 \text{ Until } \phi_2$
## Interpreting 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 \( p \)
- EF \( p \)
- AF EG \( p \)
- EG AF \( p \)
- \( p \) AU \( q \)
**EF \( \phi \): “There exists a path along which at some state \( \phi \) holds.”**
AF $\phi$: "On all paths, at some state $\phi$ holds."
AG $\phi$ : “On all paths, for all states $\phi$ holds.”
EG $\phi$ : “There exists a path along which for all states $\phi$ holds.”
\[ \phi \mathsf{EU} \Psi : \text{“There exists a path along which } \phi \text{ holds until } \Psi \text{ holds.”} \]
$\phi \text{AU}\Psi$ : “On all paths, $\phi$ holds until $\Psi$ holds.”

\[
\begin{array}{l}
\text{q} \vDash \phi \\
\text{s} \vDash \phi \\
\text{r} \vDash \phi \\
\text{q} \text{ AU } \Psi \\
\end{array}
\]
$\exists \phi$ : "There exists a path along which the next state satisfies $\phi$."

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

$\exists \phi \rightarrow \exists \phi$

$\text{Path-specific:}$

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

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

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

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

$q \models \exists \phi$

$r \models ?$

$s \models ?$
AG EF $\phi$: “On all paths and for all states, there exists a path along which at some state $\phi$ holds.”
\textbf{AG EF }\phi : \textbf{“On all paths and for all states, there exists a path along which at some state }\phi \textbf{ holds.”}
Specifying using 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 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.”
Computing CTL formula

• Define $\llbracket \phi \rrbracket$ as the set of all initial states of the finite automaton for which CTL formula $\phi$ is true. A finite automaton with initial state $q_0$ satisfies $\phi$ iff

$$q_0 \in \llbracket \phi \rrbracket$$

• Now, we can use our “trick”: computing with sets of states!
  • $\psi_{\llbracket \phi \rrbracket}(q)$ is true if the state $q$ is in the set $\llbracket \phi \rrbracket$, i.e., it is a state for which the CTL formula is true.
  • Therefore, we can also say

$$q_0 \in \llbracket \phi \rrbracket \iff \psi_{\llbracket \phi \rrbracket}(q_0) \quad \text{characteristic function of the set} \ \llbracket \phi \rrbracket$$
Computing CTL formula: EX φ

Over paths:
Aφ → All φ
Eφ → Exists φ
Path-specific:
Xφ → NeXt φ
Fφ → Finally φ
Gφ → Globally φ
φ₁Uφ₂ → φ₁ Until φ₂
Computing CTL formula: EX $\phi$

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

Sets

$Q = \llbracket \phi \rrbracket$

Characteristic functions

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

- Suppose that $Q$ is the set of initial 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

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

Characteristic functions

$\psi_Q(q) \implies \psi_{Q'}(q') = (\exists q : \psi_Q(q) \cdot \psi_\delta(q', q))$
Computing CTL formula: EX \( \phi \)

- Example for EX \( \phi \): Compute EX \( q_2 \)

\[
[q_2] = \{q_2\}
\]
Computing CTL formula: EX ϕ

- Example for EX ϕ: Compute EX q_2

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

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

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

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

\[
\llbracket q_2 \rrbracket = \{q_2\}
\]

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

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

As $q_0 \notin \llbracket \text{EX } q_2 \rrbracket = \{q_1, q_2, q_3\}$, the CTL formula EX $q_2$ is not true.
Computing CTL formula: EF $\phi$

- Start with the set of initial 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.

$Q_0 = [[\phi]]$

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

$[[EF\phi]] = Q'$
Computing CTL formula: EF \( \phi \)

- Example for EF\( \phi \): Compute EF \( q_2 \)

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

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

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

\[
\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_1, q_2, q_3\}
\]
Computing CTL formula: EF $\phi$

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

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

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

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

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

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

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

$$Q_0 = [q_2] = \{q_2\}$$
$$Q_1 = \{q_2\} \cup \text{Pre}(\{q_2\}, \delta) = \{q_1, q_2, q_3\}$$
$$Q_2 = \{q_1, q_2, q_3\} \cup \text{Pre}(\{q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$$
$$Q_3 = \{q_0, q_1, q_2, q_3\} \cup \text{Pre}(\{q_0, q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$$
$$[\text{EF}q_2] = Q_3 = \{q_0, q_1, q_2, q_3\}$$
Computing CTL formula: EF $\phi$

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

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

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

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

- Start with the set of initial 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.

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

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

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

$$Q_0 = \text{[[}q_2\text{]]} = \{q_2\}$$
Computing CTL formula: EG $\phi$

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

\[ Q_0 = [q_2] = \{q_2\} \]
\[ Q_1 = \{q_2\} \cap \text{Pre}(\{q_2\}, \delta) = \{q_2\} \]
\[ [\text{EG}q_2] = Q_2 = \{q_2\} \]

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

- Start with the set of initial 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.

$$Q_0 = \bbox{\phi_2}$$
$$Q_i = Q_{i-1} \cup (\text{Pre}(Q_{i-1}, \delta) \cap \bbox{\phi_1}) \quad \text{for all } i > 1 \text{ until a fixed point is reached}$$
Computing CTL formula: $\phi_1 EU \phi_2$

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

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

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

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

$$\{q' \mid \exists q \text{ with } \psi_Q(q) \cdot \psi_\delta(q', q)\} = \{q_0, q_2\}$$
Computing CTL formula: $\phi_1 EU \phi_2$

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

- Diagram:

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

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

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

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

$Q_0 = [q_1] = \{q_1\}$

$Q_1 = \{q_1\} \cup (\text{Pre}(\{q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\}$

$Q_2 = \{q_0, q_1\} \cup (\text{Pre}(\{q_0, q_1\}, \delta) \cap \{q_0\}) = \{q_0, q_1\}$

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

$\{q', \exists q \text{ with } \psi_\alpha(q) \cdot \psi_\delta(q', q)\} = \{q_0, q_2\}$

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:

$\text{AF} \phi \equiv \neg \text{EG} (\neg \phi)$  \hspace{1cm} \text{AG} \phi \equiv \neg \text{EF} (\neg \phi)$  \hspace{1cm} $\text{AX} \phi \equiv \neg \text{EX} (\neg \phi)$
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 $M$. 

Finite automata
Petri nets
Kripke machine
... 

CTL, LTL, ...
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 automata
Petri nets
Kripke machine
CTL, LTL, ...
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!)
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
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?
Thanks for your attention and see you next week! 😊

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

ETH Zurich (D-ITET)
December 1, 2022

Most materials from Lothar Thiele and Romain Jacob