Discrete Event Systems
Verification of Finite Automata (Part 2)

Romain Jacob
www.romainjacob.net

ETH Zurich (D-ITET)
December 2, 2021

Most materials from Lothar Thiele
Thank you for your feedback!

- Slightly too fast
- Reachability was covered too quickly
- More examples would be nice
- More interaction would be nice

Will hopefully improve already today 😊
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 \rightarrow data structure \rightarrow comparison

System under test \rightarrow data structure

Proving properties

Property \rightarrow fixed-point calculation

System under test \rightarrow 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 using Boolean Expressions

- Representation of a relation $R \subseteq A \times B$
  - Binary encoding $\sigma(a)$, $\sigma(b)$ of all elements $a \in A$, $b \in B$
  - Representation of $R$
    $$(a, b) \in R \iff \psi_R(\sigma(a), \sigma(b))$$

- Example finite automaton:

  finite automaton

characteristic function of the relation $R$

- we remove the binary encoding for convenience in our notation; but $u$, $q$, $q'$ are actually represented as binary vectors:

$\psi_\delta(u, q, q') = 1$
$\psi_\omega(u, q, y) = 1$
Reachability of States – State Diagram

**Question**

Is a state $q \in Q$ reachable by a sequence of state transitions?

$Q_0 = \{q_0\}$

$Q_1 = Q_0 \cup \{q_1\}$

$Q_2 = Q_1 \cup \{q_1, q_2\}$

$Q_3 = Q_2 \cup \{q_1, q_2\}$

**Problem**

Drawing state diagrams is not feasible in general.
Reachability of States – Boolean Expressions

**Fixed-point computation**
- Start with the initial state
- Determine the set of states that can be reached in one
- Take the union and iterate until a fixed-point is reached

\[ Q_0 = \{q_0\} \]

\[ Q_{i+1} = Q_i \cup Suc(Q_i, \delta) \]

\[ \psi_{Q_{i+1}}(q') = \psi_{Q_i}(q') + (\exists q : \psi_{Q_i}(q) \cdot \psi_\delta(q, q')) \]

Test by comparing the ROBDDs of \( Q_{i+1} = Q_i \)

**Finite union** if model is finite

**\( Q_R \): set of reachable states**

\[ Q_R = Q_0 \cup \bigcup_{i \geq 0} Suc(Q_i, \delta) \]

\[ \psi_{Q_R}(q') = \psi_{Q_0}(q') \sum_{i \geq 0} (\exists q : \psi_{Q_i}(q) \cdot \psi_\delta(q, q')) \]
Reachability of States – Example

State encoding

\[(x_1, x_0) = \sigma(q)\]

$$
\begin{array}{c|cc}
\sigma(q) & x_1 & x_0 \\
\hline
q_0 & 0 & 0 \\
q_1 & 0 & 1 \\
q_2 & 1 & 0 \\
q_3 & 1 & 1 \\
\end{array}
$$

Transition relation encoding

\[\psi_\delta(q, q')\]

entries where \[\psi_\delta(q, q') = 1\] only

As a Boolean function

\[\psi_\delta(q, q') = \overline{x_0'} \cdot (x_0 \cdot (x_1 + x_1') + x_1 \cdot x_1') + \overline{x_0} \cdot x_0' \cdot \overline{x_1'}\]

\[
\begin{array}{c|ccc}
x_1 & x_0 & x_1' & x_0' \\
\hline
0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

e.g.

\[q_0 \rightarrow q_1\]

\[q_2 \rightarrow q_2\]
$\psi_{Q_{i+1}}(q') = \psi_{Q_i}(q') + (\exists q : \psi_{Q_i}(q) \cdot \psi_\delta(q, q'))$

**States**

<table>
<thead>
<tr>
<th>$\sigma(q)$</th>
<th>$x_1$</th>
<th>$x_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$q_1$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$q_2$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$q_3$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Transitions**

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_0$</th>
<th>$x_1'$</th>
<th>$x_0'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Comparison of Finite Automata

For simplicity, we only consider Moore automata, i.e., the output depends on the current state only. The output function is $\omega : Q \rightarrow \Sigma$ and $y = \omega(q)$.

**Strategy**

1. Compute the set of jointly reachable states.
2. Compare the output values of the two finite automata.

$y = 1 \iff y_1 = y_2$
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

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

• 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>
Formulation of CTL properties

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

$A\phi \rightarrow \langle\text{All } \phi\rangle$, \hspace{1cm} $\phi$ holds on all paths
$E\phi \rightarrow \langle\text{Exists } \phi\rangle$, \hspace{1cm} $\phi$ holds on at least one path

$X\phi \rightarrow \langle\text{NeXt } \phi\rangle$, \hspace{1cm} $\phi$ holds on the next state
$F\phi \rightarrow \langle\text{Finally } \phi\rangle$, \hspace{1cm} $\phi$ holds at some state along the path
$G\phi \rightarrow \langle\text{Globally } \phi\rangle$, \hspace{1cm} $\phi$ holds on all states along the path
$\phi_1 U \phi_2 \rightarrow \langle\phi_1 \text{ Until } \phi_2\rangle$, \hspace{1cm} $\phi_1$ holds until $\phi_2$ holds

implies that $\phi_2$ has to hold eventually

Quantifiers over paths

Path-specific quantifiers
Formulation of CTL properties

CTL quantifiers work in pairs

\{A,E\} \{X,F,G,U\} \phi

You need one of each!

Can be more than one pair

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

E,G,X,U are sufficient to define the whole logic.
A and F are convenient, but not necessary

No need to know that one

\[ \phi_1 \text{AU} \phi_2 \equiv \neg[(\neg \phi_1) \text{EU} \neg(\phi_1 + \phi_2)] + \text{EG}(\neg \phi_2) \]
CTL works on computation trees

Automaton

Computation tree
CTL works on computation trees

Required fully-defined transition functions

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

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

AG p

AF p
Visualizing CTL formula

AX p

p AU q
Visualizing CTL formula

$EG \, p$

$EF \, p$
Visualizing CTL formula

EX p

p EU q
Intuition for “AF p = ¬ EG (¬ p)”
Interpreting CTL formula

- AG p
- EF p
- AF EG p
- EG AF p

- p AU q
Interpreting CTL formula

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

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

$\begin{array}{c}
q \models \phi \\
q \models \text{AF } \phi \\
r \models ? \\
s \models ?
\end{array}$
AF $\phi$: “On all paths, at some state $\phi$ holds.”

$q \models AF \phi$

$r \models AF \phi$

$s \not\models AF \phi$
AG $\phi$: “On all paths, for all states $\phi$ holds.”
AG $\phi$: “On all paths, for all states $\phi$ holds.”

$q \models \phi$

$q \models AG \phi$

$r \models AG \phi$

$s \not\models AG \phi$
EG $\phi$: “There exists a path along which for all states $\phi$ holds.”
$\text{EG } \phi : \text{ “There exists a path along which for all states } \phi \text{ holds.”}$

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

$r \models \text{EG } \phi$

$s \not\models \text{EG } \phi$
\( \phi \text{EU} \Psi \): “There exists a path along which \( \phi \) holds until \( \Psi \) holds.”
\( \phi \mathbin{\text{EU}} \Psi \): “There exists a path along which \( \phi \) holds until \( \Psi \) holds.”
\( \phi \mathsf{A} \mathsf{U} \Psi : \text{“On all paths, } \phi \text{ holds until } \Psi \text{ holds.”} \)
$\phi A U \Psi$ : “On all paths, $\phi$ holds until $\Psi$ holds.”

$q \models \phi A U \Psi$

$r \models \phi A U \Psi$

$s \models \phi A U \Psi$
EX$\phi$ : “There exists a path along which the next state satisfies $\phi$.”
EX\(\phi\) : “There exists a path along which the next state satisfies \(\phi\).”
**AG EF \( \phi \): “On all paths and for all states, there exists a path along which at some state \( \phi \) holds.”**
\(\mathsf{AG\ EF}\ \phi\ : \ "On\ all\ paths\ and\ for\ all\ states,\ there\ exists\ a\ path\ along\ which\ at\ some\ state\ \phi\ holds."\)
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.”
Specifying using 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 CTL formula

- In order to compute CTL formula, we first define $\llbracket \phi \rrbracket$ as the set of all initial states of the finite automaton for which CTL formula $\phi$ is true. Then we can say that 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

$$ q_0 \in \llbracket \phi \rrbracket \equiv \psi_{\llbracket \phi \rrbracket}(q_0) $$

- Remember: We suppose that every state has at least one successor state (could be itself).
Computing CTL formula

- We now show how to compute some operators in CTL. All others can be determined using the equivalence relations between operators that we listed earlier.
  - **EX φ** : Let us first define the set of predecessor states of \( Q \), i.e., the set of states that lead in one transition to a state in \( Q \):
    \[
    Q' = \text{Pre}(Q, \delta) = \{q' \mid \exists q : \psi_\delta(q', q) \cdot \psi_Q(q)\}
    \]
    Suppose that \( Q \) is the set of initial states for which the formula \( \phi \) is true. Then we can write

\[
\begin{align*}
Q &= \llbracket \phi \rrbracket \\
\psi_Q(q) &\quad \rightarrow \\
\begin{array}{c}
\downarrow \\
Q' &= \llbracket \text{EX} \phi \rrbracket = \text{Pre}(\llbracket \phi \rrbracket, \delta) \\
\psi_{Q'}(q') &= (\exists q : \psi_Q(q) \cdot \psi_\delta(q', q))
\end{array}
\end{align*}
\]
Computing CTL formula
Computing CTL formula

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

$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0$

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

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

$\{q' | \exists q : \psi_\delta(q', q) \cdot \psi_Q(q)\}$

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

• EF $\phi$: The idea here is to start with the set of initial states for which the formula $\phi$ is true. Then we add to this set the set of predecessor states. For the resulting set of states we do the same, ..., until we reach a fixed-point. The corresponding operations can be done using BDDs (as described before).

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

EF p
Computing CTL formula

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

$Q_0 = \lfloor q_2 \rfloor = \{q_2\}$

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

$Q_2 = \{q_1, q_2, q_3\} \cup \text{Pre}(\{q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$

$Q_3 = \{q_0, q_1, q_2, q_3\} \cup \text{Pre}(\{q_0, q_1, q_2, q_3\}, \delta) = \{q_0, q_1, q_2, q_3\}$

$\lceil \text{EF}q_2 \rceil = Q_3 = \{q_0, q_1, q_2, q_3\}$

As $q_0 \in \lceil \text{EF}q_2 \rceil = \{q_0, q_1, q_2, q_3\}$, the CTL formula $\text{EF } q_2$ is true.
Computing CTL formula

- \( \text{EG } \phi \): The idea here is to start with the set of initial states for which the formula \( \phi \) is true. Then we cut this set with the set of predecessor states. For the resulting set of states we do the same, ..., until we reach a fixed-point. The corresponding operations can be done using BDDs (as described before).

\[
Q_0 = \lbrack \phi \rbrack \\
Q_i = Q_{i-1} \cap \text{Pre}(Q_{i-1}, \delta) \quad \text{for all } i > 1 \text{ until a fixed-point is reached}
\]
Computing CTL formula

$\text{EG } p$
Computing CTL formula

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

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

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

As $q_0 \notin [\text{EG}q_2] = \{ q_2 \}$, the CTL formula EG $q_2$ is not true.
Computing CTL formula

- $\phi_1 EU \phi_2$: The idea here is to start with the set of initial states for which the formula $\phi_2$ is true. Then we add to this set the set of predecessor states for which the formula $\phi_1$ is true. For the resulting set of states we do the same, ..., until we reach a fixed-point. The corresponding operations can be done using BDDs (as described before).

\[
Q_0 = \llbracket \phi_2 \rrbracket
\]

\[
Q_i = Q_{i-1} \cup (\text{Pre}(Q_{i-1}, \delta) \cap \llbracket \phi_1 \rrbracket) \quad \text{for all } i > 1 \text{ until a fixed-point is reached}
\]

Like EF $\phi_2$, the only difference is that on our path backwards, we always make sure that also $\phi_1$ holds.
Computing CTL formula

\[ p \text{ EU } q \]
Computing CTL formula

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

As $q_0 \in \llbracket q_0 \ EU q_1 \rrbracket = \{q_0, q_1\}$, the CTL formula $q_0 \ EG \ q_1$ is true.
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 \( M \vDash \phi \), or
- return a trace where the formula does not hold in \( M \).
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 $M \vDash \phi$, or
- return a trace where the formula does not hold in $M$.  

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

Finite automata
Petri nets
Kripke machine
CTL, LTL, ...
Let’s see how it works in practice...

**UPPAAL model-checker**

- free for academia
- (much) more general than what we show here
- can verify the timed behavior of communicating finite automata

Example

Modeling and verification of a simple protocol for ATM-Money-Withdrawal
Step 1. ATM without Cancel

- **Sending event “bank_card”**

- **Initial state**

- **Enabled by event “cash”**

- **Communicating finite automata**

```plaintext
A[] Eric.IDLE imply (Bank.Money imply Pocket.No_money)  
E<> Pocket.Money  
A[] Eric.IDLE imply (Bank.No_money imply Pocket.Money)  
```
Step 2. ATM with Cancel

A[Eric.IDLE imply (Bank.Money imply Pocket.No_money)]
E<> Pocket.Money
A[Eric.IDLE imply (Bank.No_money imply Pocket.Money)]
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!)
Conclusion and perspectives

Example

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

```
 x1
 | +
 x2
 | +
 x3
 | +
   \rightarrow y
```

“The device can always be switched off.”

Comparison of specification and implementation

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

Proving properties

- property
- fixed-point calculation
- system under test → data structure
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?
See you next week!
in Discrete Event Systems

Romain Jacob
www.romainjacob.net

ETH Zurich (D-ITET)
December 2, 2021

Most materials from Lothar Thiele