Model Checking LTL Specs using Semantic Tableau
Outline

• System descriptions
• LTL model checking problem
• Automata-approach
  – Büchi Automata
  – Overview of MC procedure
  – Construction of BA via semantic tableau
How does LTL formula specify behaviors?

• The system is typically described by one or more finite-state processes (FSPs)

• The FSPs induce a labeled transition system (LTS)
  – paths through the LTS produce state sequences representing possible system behaviors
  – *Stutter* the final state of a finite (terminating) execution

• All such infinite state sequences satisfy the LTL specification at their initial positions (i.e., at 0)
  – \( \rho \models f \) abbreviates \( (\rho, 0) \models f \)
### Example system description

**Process A:**
1. $a := T$
2. $t := \text{'B'}$
3. $a := F$

if not $b$ or $t = \text{'A'}$ then skip

**Process B:**
1. $b := T$
2. $t := \text{'A'}$
3. $b := F$

if not $a$ or $t = \text{'B'}$ then skip

---

**Example execution:**

<table>
<thead>
<tr>
<th>A@1, B@1</th>
<th>A@2, B@1</th>
<th>A@2, B@2</th>
<th>A@2, B@3</th>
<th>A@3, B@3</th>
<th>A@1, B@CS</th>
<th>A@1, B@1</th>
<th>A@CS, B@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a, b$</td>
<td>$a, b$</td>
<td>$a, b$</td>
<td>$a, b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$a$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$t = \text{'A'}$</td>
<td>$t = \text{'B'}$</td>
<td>$t = \text{'B'}$</td>
<td>$t = \text{'B'}$</td>
<td>$t = \text{'B'}$</td>
</tr>
</tbody>
</table>

---

**Example specifications:**

$\Box (a \Rightarrow \Diamond (A@CS))$

$\neg \Diamond (A@CS \land B@CS)$

---

CSE 814  LTL: Model Checking via Semantic Tableau
Example: LTS (reachability graph)

System executions represented by paths of the LTS.

Not usually tractable to grow a full RG.
The LTL Model Checking Problem

• Given
  – A model describing the behaviors of a system (e.g., a set of FSPs or an LTS)
  – A set of specifications expressed in LTL

• Algorithmically
  – Check that every behavior satisfies the specifications
  – Produce a “counterexample”, if not
Automata-theoretic approach

- Finite state automata (FSA) are acceptors for finite strings (sequences):
  
  \[ M = (\Sigma, Q, \delta, I, F) \]

  where \( \Sigma \) is an alphabet, \( Q \) is a set of states, \( \delta \subseteq Q \times \Sigma \times Q \) is a transition relation, and \( I \) and \( F \) are sets of initial and final states, resp.

- We write \( q \xrightarrow{\lambda} q' \) for \((q, \lambda, q') \in \delta\)

- \( M \) accepts \( \lambda_0 \lambda_1 \ldots \lambda_{n-1} \in \Sigma^* \) iff there is a run of the form \( q_0 \xrightarrow{\lambda_0} q_1 \xrightarrow{\lambda_1} q_2 \xrightarrow{\lambda_2} \ldots \xrightarrow{\lambda_{n-1}} q_n \) where \( q_0 \in I \) and \( q_n \in F \).

- \( \mathcal{L}(M) = \{ \lambda_0 \lambda_1 \ldots \lambda_{n-1} \in \Sigma^* \mid M \text{ accepts } \lambda_0 \lambda_1 \ldots \lambda_{n-1} \} \)
Automata-theoretic approach

• A Bucchi automaton is an acceptor for infinite strings (sequences)

\[ B = (\Sigma, Q, \delta, I, \{ F_1, F_2, \ldots, F_k \}) \]

It is like a FSA but with a different acceptance criterion:
\( \lambda_0 \lambda_1 \lambda_2 \ldots \in \Sigma^\omega \) is accepted by \( B \) if there an infinite run

\[ q_0 \xrightarrow{\lambda_0} q_1 \xrightarrow{\lambda_1} q_2 \xrightarrow{\lambda_2} \ldots \]

where \( q_0 \in I \) and \( \{ q_0, q_1, \ldots \} \cap F_j \) is infinite, for \( j = 1..k \)

• General approach:
  – Represent both the system and the specification as BA
  – Use theory of BA to solve the MC problem
Automata-theoretic approach

General approach:

• Describe the system as BA, $M$

• Describe the specification $f$ as BA, $B_f$, such that
  – labels are LTS states, i.e., $\Sigma = 2^P$
  – for all $\rho \in \Sigma^\omega$, $\rho \models f$ iff $\rho \in L(B_f)$

• Use theory of BA to determine if $L(M) \subseteq L(B_f)$
Automata-theoretic approach

• Given $B_1 = (\Sigma, Q_1, \delta_1, I_1, \{F_{1,1}, \ldots, F_{1,k_1}\})$ and $B_2 = (\Sigma, Q_2, \delta_2, I_2, \{F_{2,1}, \ldots, F_{1,k_2}\})$, the synchronous product of $B_1$ and $B_2$ is defined:

$$B_1 \times B_2 = (\Sigma, Q_1 \times Q_2, \delta', I_1 \times I_2, F)$$

where

$$\delta' = \{ (q_1, q_2) \xrightarrow{\lambda} (q_1', q_2') \mid q_1 \xrightarrow{\lambda} q_1' \land q_2 \xrightarrow{\lambda} q_2' \}$$

$$F = \{ F_{1,1} \times Q_2, \ldots, F_{1,k_1} \times Q_2, Q_1 \times F_{2,1}, \ldots, Q_1 \times F_{1,k_2} \}$$

• Trivial to show that $\mathcal{L}(B_1 \times B_2) = \mathcal{L}(B_1) \cap \mathcal{L}(B_2)$
Automata-theoretic approach

• Observations:

\[ \mathcal{L}(B_f) = \mathcal{L}(B_{\neg f}) \]

\[ \mathcal{L}(B_f \times B_g) = \mathcal{L}(B_f) \cap \mathcal{L}(B_g) = \mathcal{L}(B_f \land g) \]

• Thus:

\[ \mathcal{L}(M) \subseteq \mathcal{L}(B_f) \text{ iff } \mathcal{L}(M) \cap \overline{\mathcal{L}(B_f)} = \emptyset \]

\[ \text{iff } \mathcal{L}(M) \cap \mathcal{L}(B_{\neg f}) = \emptyset \]

\[ \text{iff } \mathcal{L}(M \times B_{\neg f}) = \emptyset \]

• Reduces MC problem to the problem of checking emptiness of a BA
Automata-theoretic approach

• The emptiness problem for BA is solved via graph theory
  – Acceptance means passing through some final state infinitely often
  – Look for a strongly connected component that is reachable from an initial state

• Additionally, a witness, $\lambda_0 \lambda_1 \lambda_2 \ldots \in \mathcal{L}(M \times B_{\neg f})$, is a counterexample
Automata-theoretic approach

Represent an LTS model of a system as a BA:
- Alphabet $\Sigma$ is the set of LTS states
- All BA state are accepting

\[
\begin{align*}
x = 0, & \quad y = 0 \\
x = 1, & \quad y = 0 \\
x = 1, & \quad y = 1
\end{align*}
\]
Automata-theoretic approach

• Key result for MC LTL: For any LTL $f$ there is a Buechi automaton, $B_f$, whose language is the sequence of LTS states that satisfy $f$

$$\mathcal{L}(B_f) = \{ \rho \in (2^P)^\omega \mid \rho \models f \}$$

Example: BA for $\Box \neg (inCSA \land inCSB)$

Abbreviation: show multiple edges between BA states as a single edge, labeled by a formula describing a multiple LTS states
Automata-theoretic approach

\[ \Box (\Diamond a \land \Diamond b) \]

(All states are initial.)
Automata-theoretic approach

\(\Box(\Diamond a \land \Diamond b)\)

(All states are initial.)
Construction of $B_f$

Intuition:

- Each state $s$ of $B_f$ “checks” a subset $A_s$ of “sub-formulas” of $f$

- Obtain $A_s$ incrementally by decomposing $f$ according to its Boolean structure
  - Start with a set of sub-formulas of $f$ that must be checked
  - Expand temporal operators—separate what needs to be true “now” from what has to be true in the future, e.g.,
    \[
    \text{if } \square g \in A_s \text{ then } A_s := A_s \cup \{ g, O \square g \}
    \]
  - “Split” $A_s$ to express alternatives (disjunction), e.g.,
    \[
    \text{if } \Diamond g \in A_s \text{ then } A_s := A_s \cup \{ g \} \text{ and } A_s' := A_s \cup \{ O \Diamond g \}
    \]

Produces a *semantic tableau*
Construction of $B_f$

- Start with a specification in negation normal form
- Use semantics-preserving reduction rules to decompose it

<table>
<thead>
<tr>
<th>formula</th>
<th>alternative 1</th>
<th>alternative 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f \lor g$</td>
<td>${f}$</td>
<td>${g}$</td>
</tr>
<tr>
<td>$f \land g$</td>
<td>${f, g}$</td>
<td></td>
</tr>
<tr>
<td>$\Diamond g$</td>
<td>${g}$</td>
<td>${\Diamond \Diamond g}$</td>
</tr>
<tr>
<td>$\square f$</td>
<td>${f, \Diamond \square f}$</td>
<td></td>
</tr>
<tr>
<td>$f \cup g$</td>
<td>${g}$</td>
<td>${f, \Diamond (f \cup g)}$</td>
</tr>
</tbody>
</table>

- Iteratively decompose sub-formulas, down to atoms & Next-formulas
- Prune locally inconsistent branches (containing $f$ and $\neg f$)
- Next-formulas yield sub-formulas for a successor to check
Example: Iterative decomposition

\[\square(\Diamond a \land \Diamond b) \checkmark\]
\[\Diamond a \land \Diamond b \checkmark\]
\[\Diamond a\checkmark\]
\[\Diamond b\checkmark\]
Ex: Alternative sub-formulas to check ($A_s$)

\[ \Box(\Diamond a \land \Diamond b) \]
\[ \Diamond a \land \Diamond b \]
\[ \bigcirc \Box(\Diamond a \land \Diamond b) \]
\[ \Diamond a \]
\[ \Diamond b \]

Each path is a locally consistent set of sub-formulas

A BA state checks each such set

Next-formulas seed expansion of the successor state(s)
\( \Box(\Diamond a \land \Diamond b) \)
\( \Diamond a \land \Diamond b \)
\( \bigcirc \Box(\Diamond a \land \Diamond b) \)
\( \Diamond a \)
\( \Diamond b \)
\( \bigcirc \Diamond a \)
\( \bigcirc \Diamond b \)

\[ s_1 : \]
\( \Box(\Diamond a \land \Diamond b) \)
\( \Diamond a \land \Diamond b \)
\( \Diamond a, \Diamond b \)
\( a, b \)
\( \bigcirc \Box(\Diamond a \land \Diamond b) \)
LTL: Model Checking via Semantic Tableau

$s_1$: 
- $\square(\Diamond a \land \Diamond b)$
- $\Diamond a \land \Diamond b$
- $\Diamond a$, $\Diamond b$
- $a$, $b$
- $\Diamond b$

$s_2$: 
- $\square(\Diamond a \land \Diamond b)$
- $\Diamond a \land \Diamond b$
- $\Diamond a$, $\Diamond b$
- $a$
- $\Diamond b$
- $\Diamond b$
\[
\begin{align*}
\square(\Diamond a \land \Diamond b) \\
\Diamond a \land \Diamond b \\
\bigcirc \square(\Diamond a \land \Diamond b) \\
\Diamond a
\end{align*}
\]

\[
\begin{align*}
S_1: \\
\square(\Diamond a \land \Diamond b) \\
\Diamond a \land \Diamond b \\
\Diamond a, \Diamond b \\
a, \ b \\
\bigcirc \square(\Diamond a \land \Diamond b) \\
\Diamond a
\end{align*}
\]

\[
\begin{align*}
S_2: \\
\square(\Diamond a \land \Diamond b) \\
\Diamond a \land \Diamond b \\
\Diamond a, \Diamond b \\
a \\
\bigcirc \square(\Diamond a \land \Diamond b) \\
\bigcirc \Diamond b
\end{align*}
\]

\[
\begin{align*}
S_3: \\
\square(\Diamond a \land \Diamond b) \\
\Diamond a \land \Diamond b \\
\Diamond a, \Diamond b \\
b \\
\bigcirc \square(\Diamond a \land \Diamond b) \\
\bigcirc \Diamond b \\
\bigcirc \Diamond a
\end{align*}
\]

CSE 814

LTL: Model Checking via Semantic Tableau

23
Doron & Peled paper:

\[\square(\Diamond a \land \Diamond b)\]
\[\Diamond a \land \Diamond b\]
\[\bigcirc \square (\Diamond a \land \Diamond b)\]
\[\Diamond a\]
\[\Diamond b\]

\[a\]
\[b\]

\[\bigcirc \Diamond a\]
\[\bigcirc \Diamond b\]

\[\text{Old}\]

\[\text{Next}\]

\[s_1:\]
\[\square(\Diamond a \land \Diamond b)\]
\[\Diamond a \land \Diamond b\]
\[\Diamond a, \Diamond b\]
\[a, b\]

\[s_2:\]
\[\square(\Diamond a \land \Diamond b)\]
\[\Diamond a \land \Diamond b\]
\[\Diamond a, \Diamond b\]
\[a\]

\[s_3:\]
\[\square(\Diamond a \land \Diamond b)\]
\[\Diamond a \land \Diamond b\]
\[\Diamond a, \Diamond b\]
\[b\]

\[s_4:\]
\[\square(\Diamond a \land \Diamond b)\]
\[\Diamond a \land \Diamond b\]
\[\Diamond a, \Diamond b\]
\[\Diamond a\]
\[\Diamond b\]
To obtain successors, repeat this process with the sub-formulas that must be true in the future (Doron & Peled: \textit{Next})
\( \Box (\Diamond a \land \Diamond b) \)
\[ \square(\lozenge a \land \lozenge b) \]
\[ \lozenge b \]
\[ \lozenge a \land \lozenge b \]
\[ \bigcirc \square(\lozenge a \land \lozenge b) \]
\[ \lozenge a \]
\[ \lozenge b \]
\[ a \]
\[ \bigcirc \lozenge a \]
\[ b \]
\[ \bigcirc \lozenge b \]
\[ b \]
\[ \bigcirc \lozenge b \]
\( \Box (\Diamond a \land \Diamond b) \)

Similarly …
BA states containing the original formula are initial states

In this case, all are initial states

\[ \square (\Diamond a \land \Diamond b) \]
Label edges leaving each BA state with the propositional formulas that must be true now.
Local automaton can postpone eventualities indefinitely, here: \( \Diamond a \) and \( \Diamond b \)

For each eventuality \( e \) define \( F_e : \) states not checking \( e \) or in which \( e \) is satisfied now.
Checking “On-the-fly”

• Construct BA for negated spec on demand while intersecting with the process description (LTS or multiple FSPs)
• Use DFS strategy while checking for an accepting cycle
  – Only have to keep frontier of the product in memory
  – Violation of a property can be detected w/o growing full automaton