Beyond Temporal Logics

Temporal logic expresses properties of infinite sequences of states, but there are interesting properties that cannot be expressed, e.g.,

“\( p \) is true only (at most) at even positions.”

Questions (foundational/practical):

- What other languages can we use to express properties of sequences (\( \Rightarrow \) properties of programs)?
- How do their expressive powers compare?
- How do their computational complexities (for the decision problems) compare?
\( \Sigma \): nonempty finite set (alphabet) of characters

\( \Sigma^* \): set of all finite strings of characters in \( \Sigma \)

finite word \( w \in \Sigma^* \)

\( \Sigma^\omega \): set of all infinite strings of characters in \( \Sigma \)

\( \omega \)-word \( w \in \Sigma^\omega \)

(finitary) language: \( L \subseteq \Sigma^* \)

\( \omega \)-language: \( L \subseteq \Sigma^\omega \)
States

Propositional LTL (PLTL) formulas are constructed from the following:

• propositions $p_1, p_2, \ldots, p_n$.
• boolean/temporal operators.
• a state $s \in \{f, t\}^n$
  i.e., every state $s$ is a truth-value assignment to all $n$ propositional variables.

Example:

If $n = 3$, then

$$s : \langle p_1 : t, p_2 : f, p_3 : t \rangle$$

corresponds to state $tft$.

$p_1 \leftrightarrow p_2$ denotes the set of states

$$\{fff, fft, ttf, ttt\}$$

• alphabet $\Sigma = \{f, t\}^n$
  i.e, $2^n$ strings, one string for every state.

Note: $T, F =$ formulas (syntax)

$t, f =$ truth values (semantics)
Models of PLTL $\mapsto \omega$-languages

- A **model** of PLTL for the language with $n$ propositions

  \[ \sigma : s_0, s_1, s_2, \ldots \]

  can be viewed as an infinite string $s_0s_1s_2\ldots$, i.e.,

  \[ \sigma \in (\{f, t\}^n)^\omega \]

- A PLTL **formula** $\varphi$ denotes an $\omega$-language

  \[ \mathcal{L} = \{ \sigma | \sigma \models \varphi \} \subseteq (\{f, t\}^n)^\omega \]

**Example:**

If $n = 3$, then

\[ \varphi : \square(p_1 \leftrightarrow p_2) \] denotes the $\omega$-language

\[ \mathcal{L}(\varphi) = \{fff, fft, ttf, ttt\}^\omega \]
Other Languages to Talk about Infinite Sequences

- $\omega$-regular expressions
- $\omega$-automata
Regular Expressions

Syntax:

\[ r ::= \emptyset \mid \varepsilon \mid a \mid r_1 r_2 \mid r_1 + r_2 \mid r^* \]

(\(\varepsilon\) = empty word, \(a \in \Sigma\))

Semantics:
A regular expression \(r\) (on alphabet \(\Sigma\)) denotes a finitary language \(\mathcal{L}(r) \subseteq \Sigma^*\):

\[
\begin{align*}
\mathcal{L}(\emptyset) & = \emptyset \\
\mathcal{L}(\varepsilon) & = \{\varepsilon\} \\
\mathcal{L}(a) & = \{a\} \\
\mathcal{L}(r_1 r_2) & = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2) = \\
& \quad \{xy \mid x \in \mathcal{L}(r_1), y \in \mathcal{L}(r_2)\} \\
\mathcal{L}(r_1 + r_2) & = \mathcal{L}(r_1) \cup \mathcal{L}(r_2) \\
\mathcal{L}(r^*) & = \mathcal{L}(r)^* = \{x_1 x_2 \cdots x_n \mid n \geq 0, \\
& \quad x_1, x_2, \ldots, x_n \in \mathcal{L}(r)\}
\end{align*}
\]
ω-regular expressions

Syntax:

\[ \omega r := r_1(s_1)^{\omega} + r_2(s_2)^{\omega} + \cdots + r_n(s_n)^{\omega} \]

\[ n \geq 1, \ r_i, s_i = \text{regular expressions} \]

Semantics:

\[ \mathcal{L}(r s^{\omega}) = \{ x y_1 y_2 \cdots | x \in \mathcal{L}(r), y_1, y_2, \ldots \in \mathcal{L}(s) \setminus \{ \varepsilon \} \} \]

\( r s^{\omega} \) denotes all infinite strings with an initial prefix in \( \mathcal{L}(r) \), followed by a concatenation of infinitely many nonempty words in \( \mathcal{L}(s) \).
\(\omega\)-regular expressions (cont.)

**Example:**

Take \(A = \{a, b\}\). What languages do the following \(\omega\)-r.e.’s denote?

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(aa ; b^\omega)</td>
<td>(\omega)-word starting with two (a)’s, followed by (b)’s</td>
</tr>
<tr>
<td>(a^* ; b^\omega)</td>
<td>all (\omega)-words starting with a finite string of (a)’s, followed by (b)’s</td>
</tr>
<tr>
<td>((a + b)^* ; b^\omega)</td>
<td>all (\omega)-words with only finitely many (a)’s</td>
</tr>
<tr>
<td>(((a + b)^*b)^\omega)</td>
<td>all (\omega)-words containing infinitely many (b)’s</td>
</tr>
</tbody>
</table>
Example:

- $p$ is an abbreviation for $tt + tf$
- $q$ is an abbreviation for $tt + ft$
- $T$ is an abbreviation for $tt + tf + ft + ff$

\[ \Downarrow \]

- $\Box p$: $p^\omega$
- $\Diamond q$: $T^* q \ T^\omega$
- $p \mathcal{U} q$: $p^* q \ T^\omega$
- $p \Rightarrow \Box q$: $(\neg p)^* q^\omega + (\neg p)^\omega$
- $\Box \Diamond p$: $(T^* p)^\omega$
- $\Diamond \Box q$: $T^* q^\omega$
Expressive Power

• Every PLTL formula has an equivalent $\omega$-r.e.

• PLTL is strictly weaker than $\omega$-r.e.’s:
  “$p$ is true only (at most) at even positions.”
  – not expressible in PLTL (Pierre Wolper, 1983)
  – $\omega$-r.e.: $(T(\neg p))^\omega$

• $\omega$-r.e.’s are equivalent to $\omega$-automata.
Finite alphabet $\Sigma$.

**Automaton $A$:** $\langle N, N_0, E, \mu, F \rangle$, where
- $N$: nodes
- $N_0 \subseteq N$: initial nodes
- $E \subseteq N \times N$: edges
- $\mu: N \rightarrow 2^\Sigma$: node labeling function
- $F \subseteq N$: final nodes

**Note:** We label the nodes and not the edges.
Main question:
Given a string

$$\sigma: s_0 \ldots s_k$$

over $$\Sigma$$, is $$\sigma$$ accepted by $$\mathcal{A}$$?

- path
  A sequence of nodes

$$\pi: n_0, \ldots, n_k$$

is a path of $$\mathcal{A}$$ if

- $$n_0 \in N_0$$
- for every $$i: 0 \ldots k-1$$, $$\langle n_i, n_{i+1} \rangle \in E.$$
Finite-State Automata (Cont’d)

• **trail**
  A path
  \[ \pi : n_0, \ldots, n_k \]
  of \( A \) is a trail of a string
  \[ \sigma : s_0, \ldots, s_k \]
  in \( A \) if for every \( i : 0 \ldots k \),
  \[ s_i \in \mu(n_i). \]

• **accepted**
  A string
  \[ \sigma : s_0 \ldots s_k \]
  is accepted by \( A \) if it has a trail
  \[ \pi : n_0, \ldots, n_k \]
  in \( A \) such that
  \[ n_k \in F. \]
Finite-State Automata (Cont’d)

• \( L(A) \)
The set of all strings ("languages") accepted by \( A \).

• **deterministic**
  An automaton \( A \) is called **deterministic** if every string has exactly one (not necessarily accepting) trail in \( A \).

• **total**
  An automaton \( A \) is called **total** if every string has at least one (not necessarily accepting) trail in \( A \).
Finite-State Automata: Decision Problems

- **Emptiness:**
  Is any string accepted?
  \[ \mathcal{L}(A) \nsubseteq \emptyset \]

- **Universality:**
  Are all strings accepted?
  \[ \mathcal{L}(A) \supseteq \Sigma^* \]

- **Inclusion:**
  Are all strings accepted by \( A_1 \) accepted by \( A_2 \)?
  \[ \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) \]
Finite-State Automata: Operations

• Complementation: \( \overline{A} \)
  \[ \mathcal{L}(\overline{A}) = \Sigma^* - \mathcal{L}(A) \]

• Product: \( A_1 \times A_2 \)
  \[ \mathcal{L}(A_1 \times A_2) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2) \]

• Union: \( A_1 + A_2 \)
  \[ \mathcal{L}(A_1 + A_2) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2) \]

Using complementation and product construction, we only need a decision procedure for emptiness to decide universality and inclusion:

• Universality:
  \[ \mathcal{L}(A) = \Sigma^* \iff \mathcal{L}(\overline{A}) = \emptyset \]

• Inclusion:
  \[ \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) \iff \mathcal{L}(A_1 \times \overline{A_2}) = \emptyset \]
Finite-State Automata: Determinization

For every nondeterministic automaton $A_N$, there exists a deterministic automaton $A_D$ such that

$$\mathcal{L}(A_N) = \mathcal{L}(A_D).$$

(May cause exponential blowup in size.)
ω-Automata

Finite-state automata over infinite strings.

Main question:
Given an infinite sequence of states

\[ \sigma : s_0, s_1, s_2, \ldots \]

is \( \sigma \) accepted by \( \mathcal{A} \)?

Additional references:
\[ n_1 : p_1 \]

\[ n_2 : \neg p_1 \land p_2 \]

\( n_1 \) represents all states in which \( p_1 \) is true; i.e. \( tf \) and \( tt \).

\[ \mu(n_1) = \{ tf, tt \} \]

\( n_2 \) represents all states in which \( p_1 \) is false and \( p_2 \) is true.

\[ \mu(n_2) = \{ ft \} \]
ω-Automata (Definition)

Set of propositions: \( p_1, \ldots, p_n \).
Alphabet \( \Sigma = \{ t, f \}^n \).

Automaton \( \mathcal{A} \): \( \langle N, N_0, E, \mu, F \rangle \), where

- \( N \): finite set of nodes
- \( N_0 \subseteq N \): initial nodes
- \( E \subseteq N \times N \): edges
- \( \mu : N \rightarrow 2^\Sigma \): node labeling function (assertions)
- \( F \): acceptance condition

Note: Most of the literature on \( \omega \)-automata uses edge labeling, similarly to automata on finite strings. However, we use node labeling to ease the transition to diagrams. The two approaches are equally expressive and can easily be translated into each other.
\( \omega \)-Automata: Trails

**Definition**: A path

\[
\pi : n_0, n_1, \ldots
\]

of \( \mathcal{A} \) is a trail of an infinite sequence of states

\[
\sigma : s_0, s_1, \ldots
\]

if for every \( i \geq 0 \),

\[
s_i \models \mu(n_i) \quad (or \ s_i \in \mu(n_i)).
\]

**Example:**

The sequence of states

\[
\sigma : t t, \ tf, \ ft, \ tt, \ tf, \ ft, \ \ldots
\]

has trail

\[
\pi : n_1, n_1, n_2, n_1, n_1, n_2, \ldots
\]

Note: no trail for \( \sigma : \ldots, f f, \ldots \).

- In general, \( \mathcal{A} \) is **nondeterministic** i.e., trail \( \pi \) is not necessarily unique for \( \sigma \).

- \( \mathcal{A} \) is **deterministic** if for every \( \sigma \), there is exactly one trail \( \pi \) of \( \sigma \).
\( \text{Inf}(\pi) \)

infinite sequence of states \( \sigma : s_0, s_1, s_2, \ldots \)

\downarrow

infinite trail \( \pi : n_0, n_1, n_2, \ldots \)

\( \text{inf}(\pi) \): The set of nodes appearing infinitely often in \( \pi \).

Observe:

- \( \text{inf}(\pi) \) is nonempty since the set of nodes of the automaton is finite.
- The nodes in \( \text{inf}(\pi) \) form a Strongly Connected Subgraph (SCS) in \( A \).

**SCS \( S \):** Every node in \( S \) is reachable from every other node in \( S \).

**MSCS \( S \):** a maximal SCS;
i.e., \( S \) is not contained in any larger SCS.

**Definition:** An infinite sequence of states \( \sigma \) is accepted by \( A \) if it has a trail \( \pi \) such that \( \text{inf}(\pi) \) is accepted by the acceptance condition.
ω-Automata: Acceptance Conditions

\[
A: \quad n_1 : \neg p \quad \xrightarrow{} \quad n_2 : p
\]

<table>
<thead>
<tr>
<th>Name</th>
<th>Büchi</th>
<th>Muller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of</td>
<td>( F \subseteq N )</td>
<td>( F \subseteq 2^N )</td>
</tr>
<tr>
<td>acceptance</td>
<td>a set of nodes</td>
<td>a set of subsets of nodes</td>
</tr>
<tr>
<td>condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>( \inf(\pi) \cap F \neq \emptyset )</td>
<td>( \inf(\pi) \in F )</td>
</tr>
<tr>
<td>for acceptance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>To accept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( L(\Box \Diamond p) )</td>
<td>( F = {n_2} )</td>
<td>( F = {{n_1, n_2}, {n_2}} )</td>
</tr>
<tr>
<td>with ( A )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>To accept</td>
<td>no deterministic Büchi</td>
<td></td>
</tr>
<tr>
<td>( L(\Diamond \Box p) )</td>
<td>accepts this language</td>
<td></td>
</tr>
<tr>
<td>with ( A )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( F = \{\{n_2\}\} \)
\(\omega\)-Automata: Acceptance Conditions (Cont’d)

\[ A: \quad \overset{n_1: \neg p}{\longrightarrow} \overset{n_2: p}{\longrightarrow} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Streett</th>
<th>Rabin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of acceptance condition</td>
<td>(F \subseteq 2^N \times 2^N) a set of pairs ({(P_1, R_1), \ldots, (P_n, R_n)}) where each (P_i, R_i) is a set of nodes</td>
<td></td>
</tr>
<tr>
<td>Condition for acceptance for every (i : [1..n]) (\inf(\pi) \subseteq P_i) or (\inf(\pi) \cap R_i \neq \emptyset) for some (i : [1..n]) (\inf(\pi) \subseteq P_i) and (\inf(\pi) \cap R_i \neq \emptyset)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To accept \(L(\Box \Diamond p)\) with \(A\)

\[ F = \{(\emptyset, \{n_2\})\} \quad \{(\{n_1, n_2\}, \{n_2\})\} \]

To accept \(L(\Diamond \Box p)\) with \(A\)

\[ F = \{\{\{n_2\}, \emptyset\}\} \quad F = \{\{\{n_2\}, \{n_2\}\}\} \]
Automata

Automaton for $\square \Diamond p \rightarrow \Box \Diamond q$
(if $p$ happens infinitely often, then $q$ happens infinitely often)

$\Diamond \Box \neg p \lor \Box \Diamond q$

Deterministic:

\[
\begin{array}{c}
n_1 : p \land q \\
n_2 : p \land \neg q \\
n_3 : \neg p \land q \\
n_4 : \neg p \land \neg q
\end{array}
\]

Muller acceptance condition ($\mathcal{P} = \text{powerset}$):

\[
F = \mathcal{P}(\{n_1, n_2, n_3, n_4\}) - \{\{n_2\}, \{n_2, n_4\}\}
\]

Streett acceptance condition:

\[
F = \left\{ ((\{n_3, n_4\}, \{n_1, n_3\}) \right\}
\]

11-26
Automata (Cont’d)

Automaton for $\Box \Diamond p \rightarrow \Box \Diamond q$

$\Diamond \Box \neg p \lor \Box \Diamond q$

Nondeterministic:

Muller acceptance condition:

$$F = \{\{n_2\}, \{n_4\}, \{n_3, n_4\}\}$$

Streett acceptance condition:

$$F = \{((\{n_2\}, \{n_4\}))\}$$
More Examples: Muller/Streett

\[ \square p \]

\[ n_1 : p \]

\[ p \Rightarrow \Diamond q \]

\[ n_2 : \neg q \]

\[ n_3 : q \]

\[ F_M = \{ \{n_1\}\} \]

\[ F_S = \{(\{n_1\}, \emptyset)\} \]

\[ F_M = \{\{n_1\}, \{n_3\}, \{n_1,n_3\}, \{n_2,n_3\}, \{n_1,n_2,n_3\}\} \]

\[ F_S = \{\emptyset, \{n_1,n_3\}\} \]

Question: Why is \( \{n_1, n_2\} \) not in \( F_M \)?
More Examples: Muller/Streett

\[ p \lor q \]

\[ n_1 : p \]
\[ n_2 : q \]
\[ n_3 : T \]

\[ p \lor q \]

\[ n_1 : p \land \neg q \]
\[ n_2 : q \]
\[ n_3 : T \]

\[ F_M = \{ \{n_1\}, \{n_3\} \} \]
\[ F_M = \{ \{n_3\} \} \]

\[ F_S = \{(\{n_1, n_3\}, \emptyset)\} \]
\[ F_S = \{(\{n_3\}, \emptyset)\} \]

Question: Why \( n_1 : p \land \neg q \) and not \( n_1 : p \)?
More Examples: Muller/Streett

\[ p \Rightarrow \Diamond \Box q \]

\[ p \Rightarrow q \land r \]

\[ F_M = \{\{n_1\}, \{n_3\}\} \]

\[ F_S = \{(\{n_1, n_3\}, \emptyset)\} \]

\[ F_M = \mathcal{P}(\{n_1, n_2, n_3\}) - \{n_1, n_2\} \]

\[ F_S = \{\emptyset, \{n_1, n_2, n_3\}\} \]
More Examples: Muller/Streett

\[ p \Rightarrow q_m \mathcal{W} q_{m-1} \cdots q_1 \mathcal{W} q_0 \]

\[ F_M = \mathcal{P}(\{n_1, \ldots, n_{m+2}\}) \]

\[ F_S = \{(\emptyset, \{n_1, \ldots, n_{m+2}\})\} \]
$n_1 : \neg p$

$\vdash n_2 : q_m$

$\vdash n_3 : q_{m-1}$

$\vdash n_4 : q_{m-2}$

$\vdots$

$\vdash n_{m+1} : q_1$

$\vdash n_{m+2} : q_0$
Existence of $\omega$-Automaton

**Theorem**: For every PLTL formula $\varphi$, there exists an $\omega$-automaton $A_\varphi$ such that $L(\varphi) = L(A_\varphi)$.

**Question**: Does the converse also hold?

- Consider $A$:

  ![Diagram](image)

  $F_M = \{n_1, n_2\}$

  $L(A) = \text{all sequences of form }$

  \[
  p \quad \neg p \quad p \quad \neg p \quad p \quad \neg p \quad p \quad \neg p \quad p \quad \neg p \quad \vdots
  \]

  Is there a PLTL formula $\varphi$ such that $L(A) = L(\varphi)$?
Existence of $\omega$-Automaton (Cont’d)

- **First attempt**: $\square p \land \square (p \leftrightarrow \square \neg p)$

  - Not good because it only accepts

    $\neg p \ p \ \neg p \ p \ \neg p \ldots$

  - That is, it accepts $\mathcal{L}(\mathcal{A}_1)$, with $\mathcal{A}_1$:

    ![Diagram](image)

    $F_M = \{\{n_1, n_2\}\}$
Existence of $\omega$-Automaton (Cont’d)

- **Second attempt**: $\bigcirc p \land \Box (p \equiv \bigcirc \bigcirc p)$

  - Not good because it accepts only

    \[
    \neg p \quad p \quad \neg p \quad p \quad \neg p \quad \ldots
    \]

  and

    \[
    p \quad p \quad p \quad p \quad p \quad \ldots
    \]

  - That is, it accepts $\mathcal{L}(A_2)$, with $A_2$:

    \[
    F_M = \{\{n_1, n_2\}, \{n_3\}\}
    \]
\textbf{ω-Automaton Expressibility}

It was shown by Wolper (1982) that there does not exist a PLTL formula $\varphi$ such that $L(\varphi) = L(\mathcal{A})$ for the automaton $\mathcal{A}$ shown above.

**Theorem:** ω-automata are strictly more expressive than PLTL.

**Theorem:** For every ω-automaton $\mathcal{A}$ there exists an existentially quantified formula $\varphi$ such that $L(\mathcal{A}) = L(\varphi)$. 
Example:

\[
F = \{n_1, n_2\}
\]

\[
\exists k. \left( \lozenge k \land \Box(k \leftrightarrow \lozenge \neg k) \land \Box(k \rightarrow p) \right)
\]

\(k\) holds in the second position \(\land\) \(k\)-positions alternate \(\land\) whenever \(k\) holds, \(p\) also holds

\(k\) is a **flexible**, auxiliary boolean variable:
its value may be different in different positions.

Note: \(\neg k\) at position 0. Why?