Outline:
• Transition systems
• Linear-time properties
• Regular properties
This short-course is on this picture applied to a particular class of systems/problems.

- **requirements** (on the system behavior)
- **assumptions** (on the unknowns, e.g., environment behavior)
- **complete system or some of its components**

**verification**
- satisfied (+certificate)
- violated (+counterexample)

**synthesis**
- controller that render the system to satisfy the spec's
- no such controller exists
This short-course is on this picture applied to a particular class of systems/problems.

- **requirements (on the system behavior)**
- **assumptions (on the unknowns, e.g., environment behavior)**
- **complete system or some of its components**

**formal specifications**

**verification**
- satisfied (+certificate)
- violated (+counterexample)

**synthesis**
- controller that render the system to satisfy the spec's
- no such controller exists

**system model**
This short-course is on this picture applied to a particular class of systems/problems.

This lecture is an intro to these.

- requirements (on the system behavior)
- assumptions (on the unknowns, e.g., environment behavior)
- complete system or some of its components

**formal specifications**

**verification**
- satisfied (+certificate)
- violated (+counterexample)

**synthesis**
- controller that render the system to satisfy the spec’s
- no such controller exists

**system model**
Finite transition system

A finite transition system is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)

• inputs,

• outputs, and

• internal states and transitions between the states.
A finite transition system is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)
  • inputs,
  • outputs, and
  • internal states and transitions between the states.

{door is not open}  \quad {door is open}
Finite transition system

A finite transition system is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)
• inputs,
• outputs, and
• internal states and transitions between the states.
Finite transition system

A finite transition system is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)
• inputs,
• outputs, and
• internal states and transitions between the states.

Finite transition system

\[ q_0 \quad \text{\{door is not open\}} \quad q_1 \quad \text{\{door is open\}} \]

\[ q_0 \quad \text{\{door is not open\}} \quad q_1 \quad \text{\{door is open\}} \]
Finite transition system

A finite transition system is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)
- inputs,
- outputs, and
- internal states and transitions between the states.

\[
\begin{align*}
q_0 & \rightarrow \text{front, rear, both} \\
q_1 & \rightarrow \text{rear, both, neither}
\end{align*}
\]

\{\text{door is not open}\} \quad \text{neither} \\
\{\text{door is open}\}
**Finite transition system**

**Example:** Traffic logic planner in Alice.

Partial nomenclature:
- DR = drive.
- STO = stop.
- NP = no passing, no reversing.
- P = passing, no reversing.
- PR = passing, reversing allowed.
- S = safe clearance with obstacle.
- A = aggressive clearance with obstacle.
- B = no clearance with obstacle.
Finite transition system

Example: Traffic lights.
A **proposition** is a statement that can be either true or false, but not both.

Examples:
- “Traffic light is green” is a proposition.
- “The front pad is occupied” is a proposition.
- “Is the front pad?” is **not** a proposition.
Preliminaries

A *proposition* is a statement that can be either true or false, but not both.

Examples:

- “Traffic light is green” is a proposition.
- “The front pad is occupied” is a proposition.
- “Is the front pad?” is *not* a proposition.

An *atomic proposition* is one whose truth or falsity does not depend on the truth or falsity of any other proposition.

Examples:

- All propositions above are atomic propositions.
- “If traffic light is green, the car can drive” is *not* an atomic proposition.
A **proposition** is a statement that can be either true or false, but not both.

Examples:
- “Traffic light is green” is a proposition.
- “The front pad is occupied” is a proposition.
- “Is the front pad?” is **not** a proposition.

An **atomic proposition** is one whose truth or falsity does not depend on the truth or falsity of any other proposition.

Examples:
- All propositions above are atomic propositions.
- “If traffic light is green, the car can drive” is **not** an atomic proposition.

For notational brevity, use propositional variables to abbreviate propositions. For example,

\[ p \equiv \text{Traffic light is green} \]
\[ q \equiv \text{Front pad is occupied} \]
Finite transition system

A transition system $TS$ is a tuple $TS = (S, Act, \rightarrow, I, AP, L)$, where

- $S$ is a set of states,
- $Act$ is a set of actions,
- $\rightarrow \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- $AP$ is a set of atomic propositions,
- $L : S \rightarrow 2^{AP}$ is a labeling function, and

$TS$ is called finite if $S$, $Act$, and $AP$ are finite.

Example

<table>
<thead>
<tr>
<th>State</th>
<th>Actions</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>rear, both, neither</td>
<td>{door is not open}</td>
</tr>
<tr>
<td>$q_1$</td>
<td>front, rear, both</td>
<td>{door is open}</td>
</tr>
</tbody>
</table>

$S = \{q_0, q_1\}$
$Act = \{\text{rear, front, both, neither}\}$
$\rightarrow = \{(q_0, \text{front}, q_1), (q_1, \text{neither}, q_0), (q_1, \text{rear}, q_1), \ldots\}$
$I = \{q_0\}$
$L(q_0) = \{\text{door is not open}\}$
$L(q_1) = \{\text{door is open}\}$
Finite transition system

A transition system $TS$ is a tuple $TS = (S, Act, \rightarrow, I, AP, L)$, where

- $S$ is a set of states,
- $Act$ is a set of actions,
- $\rightarrow \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- $AP$ is a set of atomic propositions,
- $L : S \rightarrow 2^{AP}$ is a labeling function, and

$TS$ is called finite if $S$, $Act$, and $AP$ are finite.

**Example**

- $S = \{q_0, q_1\}$
- $Act = \{\text{rear, front, both, neither}\}$
- $\rightarrow = \{(q_0, \text{front}, q_1), (q_1, \text{neither}, q_0), (q_1, \text{rear}, q_1), \ldots\}$
- $I = \{q_0\}$
- $L(q_0) = \{\text{door is not open}\}$
- $L(q_1) = \{\text{door is open}\}$
Propositional logic

Given finite set $AP$ of atomic propositions, the set of propositional logic formulas is inductively defined by:
- true is a formula;
- any $a \in AP$ is a formula;
- if $\phi_1$, $\phi_2$, and $\phi$ are formulas, so are $\neg \phi$ and $\phi_1 \land \phi_2$; and
- nothing else is a formula.

From “Specifying Systems” by L. Lamport: Propositional logic is the math of the Boolean values, true and false, and the operators $\neg, \land, \lor, \to$
Propositional logic

Given finite set $AP$ of atomic propositions, the set of propositional logic formulas is inductively defined by:
- true is a formula;
- any $a \in AP$ is a formula;
- if $\phi_1$, $\phi_2$, and $\phi$ are formulas, so are $\neg\phi$ and $\phi_1 \land \phi_2$; and
- nothing else is a formula.

Notation

• Connectives:
  - $\neg$ (negation), $\land$ (and), $\lor$ (or), $\rightarrow$ (implies)

  • 1 for “true” and 0 for “false.”

Example propositional logic formulas obtained by applying the above four rules:

$$\phi_1 \lor \phi_2 := \neg(\neg\phi_1 \land \neg\phi_2)$$

$$\phi_1 \rightarrow \phi_2 := \neg\phi_1 \lor \phi_2$$
**Propositional logic**

Given finite set $AP$ of atomic propositions, the set of propositional logic formulas is inductively defined by:
- true is a formula;
- any $a \in AP$ is a formula;
- if $\phi_1$, $\phi_2$, and $\phi$ are formulas, so are $\neg \phi$ and $\phi_1 \land \phi_2$; and
- nothing else is a formula.

**Notation**

- Connectives:
  - $\neg$ (negation), $\land$ (and), $\lor$ (or), $\rightarrow$ (implies)
- 1 for “true” and 0 for “false.”

Example propositional logic formulas obtained by applying the above four rules:

$$\phi_1 \lor \phi_2 := \neg(\neg \phi_1 \land \neg \phi_2)$$
$$\phi_1 \rightarrow \phi_2 := \neg \phi_1 \lor \phi_2$$

The evaluation function $\mu : AP \rightarrow \{0, 1\}$ assigns a truth value to each $a \in AP$.

The truth value $\mu(\Phi)$ of a formula $\Phi$ is determined by substituting the values for the atomic propositions specified by $\mu$.

Given: $AP = \{a, b, c\}$, $\mu(a) = 0$ and $\mu(b) = \mu(c) = 1$.

$$\Phi_1 = (a \land \neg b) \lor c, \quad \mu(\Phi_1) = 1$$
$$\Phi_2 = (a \land \neg b) \land c, \quad \mu(\Phi_2) = 0$$
Logical dynamical system as a finite transition system

\[
x_1[k+1] = x_2[k] \lor u[k], \quad x_1[0] = 0,
\]
\[
x_2[k+1] = x_1[k] \land u[k], \quad x_2[0] = 1,
\]
\[
y[k] = x_1[k] \oplus x_2[k]
\]

\[\phi_1 \oplus \phi_2 := (\neg \phi_1 \land \phi_2) \lor (\phi_1 \land \neg \phi_2)\]

XOR (exclusive or) gives true only if exactly one of the operands is true.

\[
S = \{0, 1\}^2
\]
\[
Act = \{0, 1\}
\]
\[
I = \{(0, 1)\}
\]
\[
AP = \{y\}
\]

\[
L(x_1, x_2) = \begin{cases} 
\{y\} \text{ (indicating 1) if } x_1 \oplus x_2 = 1 \\
\emptyset \text{ (indicating 0) otherwise}
\end{cases}
\]
**Concurrent systems**

Systems in which multiple tasks can be executed at the same time potentially with inter-task communication and resource sharing.
**Concurrent systems**

Systems in which multiple tasks can be executed at the same time potentially with inter-task communication and resource sharing.

**Example:** multi-threaded control

- Separate code into independent threads
- Switch between threads, allowing each to run simultaneously
- Potential problems: deadlocks, race conditions

Modes of communication between the subsystems:

- hand-shaking (leads to synchrony)
- changing the values of shared variables (leads to asynchrony)
**Composition of transition systems (by handshaking)**

Let $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ and $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ be transition systems. Their parallel composition, $TS_1 || TS_2$ is the transition system defined by

$$TS_1 || TS_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by the following rules:

- If $\alpha \in Act_1 \cap Act_2$, $s_1 \xrightarrow{\alpha_1} s_1'$, and $s_2 \xrightarrow{\alpha_2} s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle$.
- If $\alpha \in Act_1 \setminus Act_2$ and $s_1 \xrightarrow{\alpha_1} s_1'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle$.
- If $\alpha \in Act_2 \setminus Act_1$ and $s_2 \xrightarrow{\alpha_2} s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle$.
Composition of transition systems (by handshaking)

Let $TS_1 = (S_1, \text{Act}_1, \rightarrow_1, I_1, AP_1, L_1)$ and $TS_2 = (S_2, \text{Act}_2, \rightarrow_2, I_2, AP_2, L_2)$ be transition systems. Their parallel composition, $TS_1||TS_2$ is the transition system defined by

$$TS_1||TS_2 = (S_1 \times S_2, \text{Act}_1 \cup \text{Act}_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by the following rules:

- If $\alpha \in \text{Act}_1 \cap \text{Act}_2$, $s_1 \xrightarrow{\alpha_1} s_1'$, and $s_2 \xrightarrow{\alpha_2} s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle$.
- If $\alpha \in \text{Act}_1 \setminus \text{Act}_2$ and $s_1 \xrightarrow{\alpha_1} s_1'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle$.
- If $\alpha \in \text{Act}_2 \setminus \text{Act}_1$ and $s_2 \xrightarrow{\alpha_2} s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle$.

![Diagram](attachment:diagram.png)
Composition of transition systems (by handshaking)

Let $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ and $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ be transition systems. Their parallel composition, $TS_1 \parallel TS_2$ is the transition system defined by

$$TS_1 \parallel TS_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where $L(⟨s_1, s_2⟩) = L_1(s_1) \cup L_2(s_2)$ and $→$ is defined by the following rules:

- If $α ∈ Act_1 \cap Act_2$, $s_1 \xrightarrow{α} s'_1$, and $s_2 \xrightarrow{α} s'_2$, then $⟨s_1, s_2⟩ \xrightarrow{α} ⟨s'_1, s'_2⟩$.
- If $α ∈ Act_1 \setminus Act_2$ and $s_1 \xrightarrow{α} s'_1$, then $⟨s_1, s_2⟩ \xrightarrow{α} ⟨s'_1, s_2⟩$.
- If $α ∈ Act_2 \setminus Act_1$ and $s_2 \xrightarrow{α} s'_2$, then $⟨s_1, s_2⟩ \xrightarrow{α} ⟨s_1, s'_2⟩$.

![Diagrams of traffic light 1, traffic light 2, and "controller" systems with unreachable states.](image-url)
Paths of a finite transition system

Given a transition system $TS = (S, Act, \rightarrow, I, AP, L)$. For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}$$

- Example: $Post((0,0)) = \{(0,0),(1,0)\}$.
- A state $s$ is terminal iff $Post(s)$ is empty.
**Paths of a finite transition system**

Given a transition system \( TS = (S, Act, \rightarrow, I, AP, L) \). For \( s \in S \),

\[
Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}
\]

- Example: \( Post((0,0)) = \{(0,0),(1,0)\} \).
- A state \( s \) is *terminal* iff \( Post(s) \) is empty.
**Paths of a finite transition system**

Given a transition system $TS = (S, Act, \to, I, AP, L)$. For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \; s.t. \; s \xrightarrow{a} s' \right\}$$

- **Example:** $Post((0,0)) = \{(0,0),(1,0)\}$.
- A state $s$ is **terminal** iff $Post(s)$ is empty.
- A sequence of states, either finite $\pi = s_0 s_1 s_2 \ldots s_n$ or infinite $\pi = s_0 s_1 s_2 \ldots$, is a **path fragment** if $s_{i+1} \in Post(s_i), \; \forall i \geq 0$.

\[
\begin{align*}
(0,1) \xrightarrow{1} (1,0) \xrightarrow{1} (1,1) \xrightarrow{1} (1,1) \xrightarrow{0} \cdots \\
(1,0) \xrightarrow{0} (0,0) \xrightarrow{0} (0,0) \xrightarrow{1} (1,0) \xrightarrow{0} \cdots \\
(0,1) \xrightarrow{1} (1,0) \xrightarrow{1} (1,1).
\end{align*}
\]
Paths of a finite transition system

Given a transition system $TS = (S, Act, \rightarrow, I, AP, L)$. For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}$$

- Example: $Post((0,0)) = \{(0,0),(1,0)\}$.
- A state $s$ is terminal iff $Post(s)$ is empty.

- A sequence of states, either finite $\pi = s_0s_1s_2 \ldots s_n$ or infinite $\pi = s_0s_1s_2 \ldots$, is a path fragment if $s_{i+1} \in Post(s_i)$, $\forall i \geq 0$.

- A path is a path fragment s.t. $s_0 \in I$ and it is
  - either finite with terminal $s_n$
  - or infinite.
- Denote the set of paths in $TS$ by $Path(TS)$.

A path:

$$(0, 1) \xrightarrow{1} (1, 0) \xrightarrow{1} (1, 1) \xrightarrow{1} (1, 1) \xrightarrow{0} \ldots$$

Not a path:

$$(1, 0) \xrightarrow{0} (0, 0) \xrightarrow{0} (0, 0) \xrightarrow{1} (1, 0) \xrightarrow{0} \ldots$$

Not a path:

$$(0, 1) \xrightarrow{1} (1, 0) \xrightarrow{1} (1, 1).$$
Traces of a finite transition system

Consider a finite transition system

\[ TS = (S, Act, \rightarrow, I, AP, L) \]
with no terminal states (wlog).

The trace of an infinite path fragment \( \pi = s_0s_1s_2 \ldots \) is defined by

\[ \text{trace}(\pi) = L(s_0)L(s_1)L(s_2) \ldots \]

The set, \( \text{Traces}(TS) \), of traces of TS is defined by

\[ \text{Traces}(TS) = \{\text{trace}(\pi) : \pi \in \text{Paths}(TS)\} \]
Traces of a finite transition system

Consider a finite transition system

\[ TS = (S, Act, \rightarrow, I, AP, L) \]

with no terminal states (wlog).

The trace of an infinite path fragment \( \pi = s_0 s_1 s_2 \ldots \) is defined by

\[ \text{trace}(\pi) = L(s_0) L(s_1) L(s_2) \ldots \]

The set, \( Traces(TS) \), of traces of TS is defined by

\[ Traces(TS) = \{ \text{trace}(\pi) : \pi \in Paths(TS) \} \]

Equivalent FSMs w/ and w/o terminal state

Sequence of sets of atomic propositions that are valid in the states along the path

Actions: \( f, f, n, b, f, f, b, \ldots \)
Path: \( q_0 q_1 q_1 q_0 q_0 q_1 q_1 q_1 \ldots \)
Trace: \( \neg o, o, o, \neg o, \neg o, o, o, o, o, \ldots \)

(with some abuse of notation)
Linear-time properties

A linear-time (LT) property $P$ over atomic propositions in $AP$ is a set of infinite sequences over $2^{AP}$.

Let $P$ be an LT property over $AP$ and $TS = (S, Act, \to, I, AP, L)$ be a transition system.

$TS$ satisfies $P$, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$. 
Linear-time properties

A linear-time (LT) property $P$ over atomic propositions in $AP$ is a set of infinite sequences over $2^{AP}$.

Let $P$ be an LT property over $AP$ and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

$TS$ satisfies $P$, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$. 

traces of $TS$

admissible, desired, undesired, etc. behavior
Linear-time properties

A linear-time (LT) property $P$ over atomic propositions in $AP$ is a set of infinite sequences over $2^{AP}$.

Let $P$ be an LT property over $AP$ and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

$TS$ satisfies $P$, denoted as $TS \models P$, iff $\text{Traces}(TS) \subseteq P$.

Example: $AP = \{\text{red}1, \text{green}1, \text{red}2, \text{green}2\}$

$P1 = \text{“The first light is infinitely often green.”}$

$[A_0A_1A_2\ldots \text{ with \text{green}1} \in A_i \subseteq 2^{AP} \text{ holds for infinitely many } i]$  

$\checkmark \{r1, g2\}\{g1, r2\}\{r1, g2\}\{g1, r2\}\ldots$  

$\checkmark \emptyset \{g1\}\emptyset \{g1\}\emptyset \{g1\}\emptyset \ldots$  

$\checkmark \{g1, g2\}\{g1, g2\}\{g1, g2\}\ldots$  

$\times \{r1, g2\}\{r1g1\}\emptyset \emptyset \ldots$

$P2 = \text{“The lights are never both green simultaneously.”}$

$[A_0A_1A_2\ldots \text{ with \text{green}1} \notin A_i \text{ or \text{green}2} \notin A_i, \text{ for all } i \geq 0]$
Linear-time properties

A linear-time (LT) property $P$ over atomic propositions in $AP$ is a set of infinite sequences over $2^{AP}$.

Let $P$ be an LT property over $AP$ and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

$TS$ satisfies $P$, denoted as $TS \models P$, iff $\text{Traces}(TS) \subseteq P$.

Example: $AP = \{\text{red}_1, \text{green}_1, \text{red}_2, \text{green}_2\}$

$P_1$ = “The first light is infinitely often green.”

$[A_0A_1A_2\ldots \text{ with } \text{green}_1 \in A_i \subseteq 2^{AP} \text{ holds for infinitely many } i]$ 

$\checkmark \ \{r_1, g_2\}\{g_1, r_2\}\{r_1, g_2\}\{g_1, r_2\}\ldots$

$\checkmark \ \emptyset\{g_1\}\emptyset\{g_1\}\emptyset\{g_1\}\emptyset\ldots$

$\checkmark \ \{g_1, g_2\}\{g_1, g_2\}\{g_1, g_2\}\ldots$

$\times \ \{r_1, g_2\}\{r_1g_1\}\emptyset\emptyset\ldots$

$P_2$ = “The lights are never both green simultaneously.”

$[A_0A_1A_2\ldots \text{ with } \text{green}_1 \notin A_i \text{ or } \text{green}_2 \notin A_i, \text{ for all } i \geq 0]$
Linear-time properties

A linear-time (LT) property $P$ over atomic propositions in $AP$ is a set of infinite sequences over $2^{AP}$.

Let $P$ be an LT property over $AP$ and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

$TS$ satisfies $P$, denoted as $TS \models P$, iff $\text{Traces}(TS) \subseteq P$.

Example: $AP = \{\text{red}1, \text{green}1, \text{red}2, \text{green}2\}$

$P1 =$ “The first light is infinitely often green.”

$[A_0A_1A_2\ldots \text{ with } \text{green}1 \in A_i \subseteq 2^{AP} \text{ holds for infinitely many } i]$

$\sqrt{\{r1, g2\}\{g1, r2\}\{r1, g2\}\{g1, r2\}\ldots}$

$\sqrt{\emptyset\{g1\}\emptyset\{g1\}\emptyset\{g1\}\emptyset\ldots}$

$\sqrt{\{g1, g2\}\{g1, g2\}\{g1, g2\}\ldots}$

$\times \{r1, g2\}\{r1g1\}\emptyset\emptyset\ldots$

$P2 =$ “The lights are never both green simultaneously.”

$[A_0A_1A_2\ldots \text{ with } \text{green}1 \notin A_i \text{ or } \text{green}2 \notin A_i, \text{ for all } i \geq 0]$
An LT property $P_\Phi$ over $AP$ is an invariant with respect to a propositional logic formula $\Phi$ over $AP$ if

$$P_\Phi = \{ A_0 A_1 A_2 \ldots \in (2^AP)^\omega : A_j \models \Phi \forall j \geq 0 \}.$$
Invariants

An LT property $P_\Phi$ over $AP$ is an invariant with respect to a propositional logic formula $\Phi$ over $AP$ if

$$P_\Phi = \{A_0A_1A_2\ldots \in (2^{AP})^\omega : A_j \models \Phi \ \forall j \geq 0\}.$$
Invariants

An LT property $P_\Phi$ over $AP$ is an invariant with respect to a propositional logic formula $\Phi$ over $AP$ if

$$P_\Phi = \{A_0A_1A_2\ldots \in (2^{AP})^\omega : A_j \models \Phi \ \forall j \geq 0\}.$$ 

**Example:** The LT property “the lights are never both green simultaneously” is an invariant with respect to $\Phi = \neg green_1 \lor \neg green_2$. 

Notation: repeat infinitely many times

For $A \subseteq AP$, let the evaluation $\mu_A$ be the characteristic function of $A$.

$A \models \Phi$ iff $\mu_A(\Phi) = 1$.
Invariants

An LT property \( P_\Phi \) over \( AP \) is an invariant with respect to a propositional logic formula \( \Phi \) over \( AP \) if

\[
P_\Phi = \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega : A_j \models \Phi \ \forall j \geq 0 \}.
\]

**Example:** The LT property “the lights are never both green simultaneously” is an invariant with respect to \( \Phi = \neg \text{green}1 \lor \neg \text{green}2 \).

Given \( TS, \Phi, \) and \( P_\Phi \), \( TS \models P_\Phi \)?

The following four statements are equivalent.
1. \( TS \models P_\Phi \)
2. \( \text{trace}(\pi) \in P_\Phi, \ \forall \pi \in \text{Path}(TS) \)
3. \( L(s) \models \Phi, \ \forall s \in S \) on a path of \( TS \)
4. \( L(s) \models \Phi, \ \forall s \in \text{Reach}(TS) \)

A state \( s \) is reachable if there exists an execution fragment s.t. \( s_0 \in I \) and

\[
s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n = s
\]

\( \text{Reach}(TS) \) : set of reachable states in TS

Invariants are state properties. That is, for verification, find the reachable states and check \( \Phi \).
Safety properties

An LT property $P_{safe}$ is a safety property if for all words $\sigma \in (2^{AP})^\omega \setminus P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of $\sigma$ s.t.

$$P_{safe} \cap \{ \sigma' \in (2^{AP})^\omega : \hat{\sigma} \text{ is a finite prefix of } \sigma' \} = \emptyset.$$ 

Bad things have happened in the bad prefix $\hat{\sigma}$. Hence, no infinite word that starts with $\hat{\sigma}$ satisfies $P_{safe}$. 
Safety properties

An LT property $P_{safe}$ is a safety property if for all words $\sigma \in (2^{AP})^\omega \setminus P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of $\sigma$ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega : \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset.$$  

Bad things have happened in the bad prefix $\hat{\sigma}$. Hence, no infinite word that starts with $\hat{\sigma}$ satisfies $P_{safe}$.

Example: $AP = \{\text{red, green, yellow}\}$

• “At least one of the lights is always on” is a safety property.
  
  $$\{\sigma = A_0A_1\ldots : A_j \subseteq AP \land A_j \neq \emptyset\}$$
  
  Bad prefixes: finite words that contain $\emptyset$.

• “Two lights are never on at the same time” is a safety property.
  
  $$\{\sigma = A_0A_1\ldots : A_j \subseteq AP \land \text{card}(A_j) \leq 1\}$$
  
  Bad prefixes: finite words that contain $\{\text{red,green}\}, \{\text{red,yellow}\}$, and so on.
Safety properties

An LT property $P_{safe}$ is a safety property if for all words $\sigma \in (2^{AP})^\omega \setminus P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of $\sigma$ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega : \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset.$$ 

Bad things have happened in the bad prefix $\hat{\sigma}$. Hence, no infinite word that starts with $\hat{\sigma}$ satisfies $P_{safe}$.

Example: $AP = \{\text{red, green, yellow}\}$

- “At least one of the lights is always on” is a safety property.
  $$\{\sigma = A_0A_1 \ldots : A_j \subseteq AP \land A_j \neq \emptyset\}$$
  Bad prefixes: finite words that contain $\emptyset$.

- “Two lights are never on at the same time” is a safety property.
  $$\{\sigma = A_0A_1 \ldots : A_j \subseteq AP \land card(A_j) \leq 1\}$$
  Bad prefixes: finite words that contain $\{\text{red,green}\}$, $\{\text{red,yellow}\}$, and so on.

Any invariant is a safety property. There are safety properties that are not invariant.

Example: $AP = \{\text{red, yellow}\}$

“Each red is immediately preceded by a yellow” is a safety property, but not invariant (because it is not a state property).

Sample bad prefixes:

- $\emptyset \emptyset \{r\}$
- $\{y\} \{y\} \{r\} \{r\} \emptyset \{r\}$


Liveness properties

An LT property $P$ is a liveness property if and only if for each finite word $w$ of $2^{AP}$ there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying $w\sigma \in P$.

**Example:** Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will eventually turn green
- First light will turn green *infinitely often*
Liveness properties

An LT property $P$ is a liveness property if and only if for each finite word $w$ of $2^{AP}$ there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying $w\sigma \in P$.

**Example:** Two traffic lights with $AP = \{red_1, green_1, red_2, green_2\}$

- First light will eventually turn green
- First light will turn green infinitely often

**Use of liveness properties:**

- specify the absence of (undesired) infinite loops or progress toward a goal.
- rule out executions that cannot realistically occur (fairness), e.g., in an asynchronous execution, every process is activate infinitely often.
Liveness properties

An LT property $P$ is a liveness property if and only if for each finite word $w$ of $2^{AP}$ there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying $w\sigma \in P$.

**Example:** Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will eventually turn green
- First light will turn green infinitely often

**Use of liveness properties:**

- specify the absence of (undesired) infinite loops or progress toward a goal.
- rule out executions that cannot realistically occur (fairness), e.g., in an asynchronous execution, every process is activate infinitely often.

**Example:** Is the following a safety or liveness property?

"the first light is eventually green after it is initially red three time instances in a row"
Liveness properties

An LT property $P$ is a liveness property if and only if for each finite word $w$ of $2^{AP}$ there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying $w\sigma \in P$.

**Example:** Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will eventually turn green
- First light will turn green infinitely often

Use of liveness properties:

- specify the absence of (undesired) infinite loops or progress toward a goal.
- rule out executions that cannot realistically occur (fairness), e.g., in an asynchronous execution, every process is activate infinitely often.

Example: Is the following a safety or liveness property?

“the first light is eventually green after it is initially red three time instances in a row”

Answer: It is a combination of a safety and a liveness property.

- Liveness: any finite word can be extended by an infinite word $A_0A_1A_2\ldots$ with $green1 \in A_j$ for some $j \geq 0$.
- Safety: any finite word $A_0A_1A_2$ with $red1 \notin A_i$ for any $i \in \{0, 1, 2\}$ is a bad prefix.
<table>
<thead>
<tr>
<th>Invariant</th>
<th>Safety</th>
<th>Liveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>state condition</td>
<td>something bad</td>
<td>something good</td>
</tr>
<tr>
<td></td>
<td>never happens</td>
<td>will happen</td>
</tr>
<tr>
<td>violated at</td>
<td>any infinite run</td>
<td>violated only by infinite</td>
</tr>
<tr>
<td>individual states</td>
<td>violating the property</td>
<td>runs</td>
</tr>
<tr>
<td></td>
<td>has a finite prefix</td>
<td></td>
</tr>
<tr>
<td>verification: find</td>
<td>verification:</td>
<td>verification:</td>
</tr>
<tr>
<td>the reachable states and check</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>the invariant condition</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Nondeterministic finite automaton (NFA)

A nondeterministic finite automaton $A = (Q, \Sigma, \delta, Q_0, F)$ is a tuple with
- $Q$ is a set of states,
- $\Sigma$ is an alphabet,
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function,
- $Q_0 \subseteq Q$ is a set of initial states, and
- $F \subseteq Q$ is a set of accept (or: final) states.

$\delta(q_0, A) = \{q_0\}$, $\delta(q_0, B) = \{q_0, q_1\}$
$\delta(q_1, A) = \{q_2\}$, $\delta(q_1, B) = \{q_2\}$
$\delta(q_2, A) = \emptyset$, $\delta(q_0, B) = \emptyset$

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{A, B\}$
$Q_0 = \{q_0\}$, $F = \{q_2\}$
A nondeterministic finite automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ is a tuple with

- $Q$ is a set of states,
- $\Sigma$ is an alphabet,
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function,
- $Q_0 \subseteq Q$ is a set of initial states, and
- $F \subseteq Q$ is a set of accept (or: final) states.

Let $w = A_1 \ldots A_n \in \Sigma^*$ be a finite word. A run for $w$ in $\mathcal{A}$ is a finite sequence of states $q_0q_1 \ldots q_n$ s.t.

- $q_0 \in Q_0$
- $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leq i < n$.

**set of finite words**

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{A, B\}$
$Q_0 = \{q_0\}$, $F = \{q_2\}$

$\delta(q_0, A) = \{q_0\}$, $\delta(q_0, B) = \{q_0, q_1\}$
$\delta(q_1, A) = \{q_2\}$, $\delta(q_1, B) = \{q_2\}$
$\delta(q_2, A) = \emptyset$, $\delta(q_0, B) = \emptyset$

**word**
**run**

empty word

<table>
<thead>
<tr>
<th>word</th>
<th>run</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$ABA$</td>
<td>$q_0q_0q_0q_0$</td>
</tr>
<tr>
<td>$BBA$</td>
<td>$q_0q_0q_0q_0$</td>
</tr>
<tr>
<td>$BA$</td>
<td>$q_0q_1q_2$</td>
</tr>
<tr>
<td>$BBA$</td>
<td>$q_0q_0q_1q_2$</td>
</tr>
</tbody>
</table>
Nondeterministic finite automaton (NFA)

A nondeterministic finite automaton \( \mathcal{A} = (Q, \Sigma, \delta, Q_0, F) \) is a tuple with
- \( Q \) is a set of states,
- \( \Sigma \) is an alphabet,
- \( \delta : Q \times \Sigma \rightarrow 2^Q \) is a transition function,
- \( Q_0 \subseteq Q \) is a set of initial states, and
- \( F \subseteq Q \) is a set of accept (or: final) states.

\[
\delta(q_0, A) = \{q_0\}, \quad \delta(q_0, B) = \{q_0, q_1\}
\]
\[
\delta(q_1, A) = \{q_2\}, \quad \delta(q_1, B) = \{q_2\}
\]
\[
\delta(q_2, A) = \emptyset, \quad \delta(q_0, B) = \emptyset
\]

Let \( w = A_1 \ldots A_n \in \Sigma^* \) be a finite word. A run for \( w \) in \( \mathcal{A} \) is a finite sequence of states \( q_0q_1 \ldots q_n \) s.t.
- \( q_0 \in Q_0 \)
- \( q_i \xrightarrow{A_{i+1}} q_{i+1} \) for all \( 0 \leq i < n \).

A run \( q_0q_1 \ldots q_n \) is called accepting if \( q_n \in F \).

A finite word in accepted if it leads to an accepting run.

The accepted language \( \mathcal{L}(\mathcal{A}) \) of \( \mathcal{A} \) is the set of finite words in \( \Sigma^* \) accepted by \( \mathcal{A} \).
Regular safety properties

A set $\mathcal{L} \subseteq \Sigma^*$ of finite strings is called a regular language if there is a nondeterministic finite automaton $\mathcal{A}$ s.t. $\mathcal{L} = \mathcal{L}(\mathcal{A})$. 

NFA: $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$

language (set of finite words) accepted by the NFA
Regular safety properties

A set $\mathcal{L} \subseteq \Sigma^*$ of finite strings is called a regular language if there is a nondeterministic finite automaton $\mathcal{A}$ s.t. $\mathcal{L} = \mathcal{L}(\mathcal{A})$.

A safety property $P_{safe}$ over $AP$ is called regular if its set of bad prefixes constitutes a regular language over $2^AP$.

That is: $\exists$ NFA $\mathcal{A}$ s.t. $\mathcal{L}(\mathcal{A})$ = bad prefixes of $P_{safe}$
Regular safety properties

A set \( \mathcal{L} \subseteq \Sigma^* \) of finite strings is called a regular language if there is a nondeterministic finite automaton \( A \) s.t. \( \mathcal{L} = \mathcal{L}(A) \).

A safety property \( P_{safe} \) over \( AP \) is called regular if its set of bad prefixes constitutes a regular language over \( 2^{AP} \).

That is: \( \exists \) NFA \( A \) s.t. \( \mathcal{L}(A) = \) bad prefixes of \( P_{safe} \)

Example: \( AP = \{\text{red}, \text{green}, \text{yellow}\} \)
“Each red must be preceded immediately by a yellow” is a regular safety property.

Sample bad prefixes:
- \{\}\{\}\{\text{red}\}
- \{\}\{\text{red}\}
- \{\text{yellow}\}\{\text{yellow}\}\{\text{green}\}\{\text{red}\}
- A_0 A_1 \ldots A_n \text{ s.t. } n > 0, \text{red} \in A_n, \text{and} \text{yellow} \notin A_{n-1}

General form of minimal bad prefixes
Verifying regular safety properties

Given a transition system $TS$ and a regular safety property $P_{safe}$, both over the atomic propositions $AP$.

Let $A$ be an NFA s.t. $\mathcal{L}(A) = BadPref(P_{safe})$. 

Verifying regular safety properties

Given a transition system $TS$ and a regular safety property $P_{safe}$, both over the atomic propositions $AP$.

Let $A$ be an NFA s.t. $\mathcal{L}(A) = BadPref(P_{safe})$. 

\begin{center}
\begin{tikzpicture}
  \node[draw, ellipse, fill=blue!50] at (0,0) {$Traces(TS)$};
  \node[draw, ellipse, fill=white!50] at (3,0) {$P_{safe}$};
\end{tikzpicture}
\end{center}
Verifying regular safety properties

Given a transition system $TS$ and a regular safety property $P_{safe}$, both over the atomic propositions $AP$.

Let $A$ be an NFA s.t. $\mathcal{L}(A) = \text{BadPref}(P_{safe})$.

$$TS \models P_{safe} \iff \text{Traces}(TS) \subseteq P_{safe}$$
$$\iff \text{Traces}(TS) \cap ((2^AP)^\omega \setminus P_{safe}) = \emptyset$$
$$\iff \text{Traces}(TS) \cap \text{BadPref}(P_{safe}).(2^AP)^\omega = \emptyset$$
$$\iff \text{pref}(\text{Traces}(TS)) \cap \text{BadPref}(P_{safe}) = \emptyset$$
$$\iff \text{pref}(\text{Traces}(TS)) \cap \mathcal{L}(A) = \emptyset$$

finite prefixes

For words $w$ and $\sigma$, $w.\sigma$ denotes their concatenation.
Verifying regular safety properties

Given a transition system $TS$ and a regular safety property $P_{safe}$, both over the atomic propositions $AP$.

Let $A$ be an NFA s.t. $\mathcal{L}(A) = \text{BadPref}(P_{safe})$.

$$TS \models P_{safe} \text{ iff } \text{Traces}(TS) \subseteq P_{safe}$$
$$\text{iff } \text{Traces}(TS) \cap ((2^AP)^\omega \setminus P_{safe}) = \emptyset$$
$$\text{iff } \text{Traces}(TS) \cap \text{BadPref}(P_{safe}).(2^AP)^\omega = \emptyset$$
$$\text{iff } \text{pref(Traces}(TS)) \cap \text{BadPref}(P_{safe}) = \emptyset$$
$$\text{iff } \text{pref(Traces}(TS)) \cap \mathcal{L}(A) = \emptyset$$

finite prefixes

For words $w$ and $\sigma$, $w.\sigma$ denotes their concatenation.
<table>
<thead>
<tr>
<th><strong>Invariant</strong></th>
<th><strong>Safety</strong></th>
<th><strong>Liveness</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>state condition</td>
<td>something bad never happens</td>
<td>something good will happen eventually</td>
</tr>
<tr>
<td>violated at individual states</td>
<td>any infinite run violating the property has a finite prefix</td>
<td>violated only by infinite runs</td>
</tr>
<tr>
<td>verification: find the reachable states and check the invariant condition</td>
<td>verification: based on nondeterministic finite automaton which accepts “finite runs”</td>
<td>verification: ?</td>
</tr>
</tbody>
</table>
Nondeterministic Buchi automaton (NBA)

A nondeterministic Buchi automaton is same as an NFA \( A = (Q, \Sigma, \delta, Q_0, F) \) with its runs interpreted differently.

Let \( w = A_1A_2 \ldots \in \Sigma^\omega \) be an infinite string. A run for \( w \) in \( A \) is an infinite sequence \( q_0q_1 \ldots \) of states s.t.

- \( q_0 \in Q_0 \) and
- \( q_0 \xrightarrow{A_1} q_1 \xrightarrow{A_2} q_2 \xrightarrow{A_3} \ldots \).
A nondeterministic Buchi automaton is same as an NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ with its runs interpreted differently.

Let $w = A_1 A_2 \ldots \in \Sigma^\omega$ be an infinite string. A run for $w$ in $\mathcal{A}$ is an infinite sequence $q_0 q_1 \ldots$ of states s.t.
- $q_0 \in Q_0$ and
- $q_0 \xrightarrow{A_1} q_1 \xrightarrow{A_2} q_2 \xrightarrow{A_3} \ldots$.

A run is accepting if $q_j \in F$ for infinitely many $j$.

A string $w$ is accepted by $\mathcal{A}$ if there is an accepting run of $w$ in $\mathcal{A}$.

$L^\omega(\mathcal{A})$: set of infinite strings accepted by $\mathcal{A}$.
Nondeterministic Buchi automaton (NBA)

A nondeterministic Buchi automaton is same as an NFA \( \mathcal{A} = (Q, \Sigma, \delta, Q_0, F) \) with its runs interpreted differently.

Let \( w = A_1 A_2 \ldots \in \Sigma^\omega \) be an infinite string. A run for \( w \) in \( \mathcal{A} \) is an infinite sequence \( q_0 q_1 \ldots \) of states s.t.
- \( q_0 \in Q_0 \) and
- \( q_0 \xrightarrow{A_1} q_1 \xrightarrow{A_2} q_2 \xrightarrow{A_3} \ldots \).

A run is accepting if \( q_j \in F \) for infinitely many \( j \).

A string \( w \) is accepted by \( \mathcal{A} \) if there is an accepting run of \( w \) in \( \mathcal{A} \).

\( \mathcal{L}_\omega(\mathcal{A}) \): set of infinite strings accepted by \( \mathcal{A} \).

A set of infinite string \( \mathcal{L}_\omega \subseteq \Sigma^\omega \) is called an \( \omega \)-regular language if there is an NBA \( \mathcal{A} \) s.t. \( \mathcal{L}_\omega = \mathcal{L}_\omega(\mathcal{A}) \).

The NBA on the right accepts the infinite words satisfying the LT property: “infinitely often green.”
ω-Regular Properties

An LT property $P$ over $AP$ is called ω-regular if $P$ is an ω-regular language over $2^{AP}$.

Invariant, regular safety, and various liveness properties are ω-regular.

Let $P$ be an ω-regular property and $A$ be an NBA that represents the "bad traces" for $P$.

Basic idea behind model checking ω-regular properties:

$$TS \not\models P \quad \text{if and only if} \quad \text{Traces}(TS) \not\subseteq P$$

$$\quad \text{if and only if} \quad \text{Traces}(TS) \cap (2^{AP})^\omega \setminus P \neq \emptyset$$

$$\quad \text{if and only if} \quad \text{Traces}(TS) \cap \overline{P} \neq \emptyset$$

$$\quad \text{if and only if} \quad \text{Traces}(TS) \cap L_\omega(A) \neq \emptyset$$
<table>
<thead>
<tr>
<th>Invariant</th>
<th>Safety</th>
<th>Liveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>state condition</td>
<td>something bad never happens</td>
<td>something good will happen eventually</td>
</tr>
<tr>
<td>violated at individual states</td>
<td>any infinite run violating the property has a finite prefix</td>
<td>violated only by infinite runs</td>
</tr>
<tr>
<td>verification: find the reachable states and check the invariant condition</td>
<td>verification: based on nondeterministic finite automaton which accepts “finite runs”</td>
<td>verification: based on nondeterministic Buchi automaton which accepts infinite runs</td>
</tr>
</tbody>
</table>