A labeled transition system (LTS) is a 4-tuple \((Q, \Sigma, \delta, I)\) where

- \(Q\) set of states
  - \(Q\) finite or countably infinite
- \(\Sigma\) set of labels (aka actions)
  - \(\Sigma\) finite or countably infinite
- \(\delta \subseteq Q \times \Sigma \times Q\) transition relation
- \(I \subseteq Q\) initial states

Note: Write \(q \xrightarrow{\alpha} q'\) for \((q, \alpha, q') \in \delta\).

### Example: Candy Machine

- \(Q = \{\text{Start}, \text{Select}, \text{GetMarsBar}, \text{GetKitKatBar}\}\)
- \(I = \{\text{Start}\}\)
- \(\Sigma = \{\text{Pay}, \text{ChooseMarsBar}, \text{ChooseKitKatBar}, \text{TakeCandy}\}\)

\[
\delta = \begin{cases} 
(\text{Start}, \text{Pay}, \text{Select}) & \text{Select, ChooseMarsBar, GetMarsBar} \\
(\text{Select, ChooseMarsBar, GetMarsBar}) & (\text{GetMarsBar, TakeCandy, Start}) \\
(\text{GetKitKatBar, TakeCandy, Start}) & 
\end{cases}
\]

### Predecessors, Successors and Determinism

Let \((Q, \Sigma, \delta, I)\) be a labeled transition system.

- \(\text{In}(q, \alpha) = \{q' | q' \xrightarrow{\alpha} q\}\)
- \(\text{Out}(q, \alpha) = \{q' | q \xrightarrow{\alpha} q'\}\)

A labeled transition system \((Q, \Sigma, \delta, I)\) is deterministic if

- \(|I| \leq 1\)
- \(|\text{Out}(q, \alpha)| \leq 1\)

### Labeled Transition Systems vs Finite State Automata

- LTS have no accepting states
  - Every FSA an LTS - just forget the accepting states
- Set of states and actions may be countably infinite
- May have infinite branching
A partial execution in an LTS is a finite or infinite alternating sequence of states and actions \( \rho = q_0 q_1 \ldots \alpha q_n \ldots \) such that:
- \( q_0 \in f \)
- \( q_i \rightarrow \alpha \rightarrow q_{i+1} \) for all \( i \) with \( q_i \) in sequence

An execution is a maximal partial execution.

A finite or infinite sequence of actions \( \alpha_1 \ldots \alpha_n \ldots \) is a trace if there exist states \( q_0 \ldots q_n \ldots \) such that the sequence \( q_0 \alpha_1 q_1 \ldots \alpha_n q_n \ldots \) is a partial execution.

Let \( \rho = q_0 q_1 \ldots q_n \ldots \) be a partial execution. Then \( \text{trace}(\rho) = \alpha_1 \ldots \alpha_n \ldots \) is a trace.

A finite or infinite sequence of states \( q_0 \ldots q_n \ldots \) is a run if there exist actions \( \alpha_1 \ldots \alpha_n \ldots \) such that the sequence \( q_0 \alpha_1 q_1 \ldots \alpha_n q_n \ldots \) is a partial execution.

Let \( \rho = q_0 q_1 \ldots q_n \ldots \) be a partial execution. Then \( \text{run}(\rho) = q_0 \ldots q_n \ldots \).

Example: Traffic Light

\[ V = \{ \text{NS, EW, Red, Yellow, Green} \} \]
\[ R = \{ \} \]
\[ \NSG \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Red} \rightarrow \text{NSC} := \text{Green} \]
\[ \NSY \quad \text{NS} \land \text{NSC} = \text{Green} \rightarrow \text{NSC} := \text{Yellow} \]
\[ \NSR \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Yellow} \rightarrow (\text{Turn, NSC}) := (\text{EW}, \text{Red}) \]
\[ \EWS \quad \text{NSC} = \text{Red} \land \text{EWC} = \text{Red} \rightarrow \text{EWC} := \text{Green} \]
\[ \EWW \quad \text{NSC} = \text{Red} \land \text{EWC} = \text{Green} \rightarrow \text{EWC} := \text{Yellow} \]
\[ \EWR \quad \text{NSC} = \text{Red} \land \text{EWC} = \text{Yellow} \rightarrow (\text{NS, Red}) \]

\[ \text{init} = (\text{NSC} = \text{Red} \land \text{EWC} = \text{Red} \land (\text{Turn} = \text{NS} \lor \text{Turn} = \text{EW})) \]
Interpreting PTS as LTS

Let \((S, T, init)\) be a program transition system. Assume \(V\) finite, \(D\) at most countable.

- Let \(Q = V \rightarrow D\), interpreted as all assignments of values to variables
- Let \(\Sigma = T\)
- Let \(\delta = \{(q, g) \rightarrow (v_1, \ldots, v_n) := (e_1, \ldots, e_n), q'\} | M_q(g) \land (\forall i \leq n. q(v_i) = T_q(e_i)) \land (\forall v \notin \{v_1, \ldots, v_n\}. q'(v) = q(v))\)
- \(I = \{q | T_q(init) = T\}\)

Example: Traffic Light

\(V = \{\text{Turn}, \text{NSC}, \text{EWC}\}, F = \{\text{NS}, \text{EW}, \text{Red}, \text{Yellow}, \text{Green}\}\) (all arity 0), \(R = \{\}\)

- \(\text{NSC} \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Red} \rightarrow \text{NSC} = \text{Green}\)
- \(\text{NSY} \quad \text{NSC} = \text{Green} \rightarrow \text{NSC} = \text{Yellow}\)
- \(\text{NSR} \quad \text{NSC} = \text{Yellow} \rightarrow (\text{Turn, NSC}) = (\text{EW, Red})\)
- \(\text{EWG} \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Red} \rightarrow \text{EWC} = \text{Green}\)
- \(\text{EYW} \quad \text{EWC} = \text{Green} \rightarrow \text{EWC} = \text{Yellow}\)
- \(\text{EWR} \quad \text{EWC} = \text{Yellow} \rightarrow (\text{Turn, EWC}) = (\text{NS, Red})\)

\(\text{init} = (\text{NSC} = \text{Red} \land \text{EWC} = \text{Red} \land (\text{Turn} = \text{NS} \lor \text{Turn} = \text{EW})\)

Examples (cont)

- LTS for traffic light has \(3 \times 3 \times 2 = 18\) possible well typed states
  - Is it possible to reach a state where \(\text{NSC} \neq \text{Red} \land \text{EWC} \neq \text{Red}\) from an initial state?
  - If so, what sequence of actions allows this?
  - Do all the immediate predecessors of a state where \(\text{NSC} = \text{Green} \lor \text{EWC} = \text{Green}\) satisfy \(\text{NSC} = \text{Red} \land \text{EWC} = \text{Red}\)?
  - If not, are any of those offended states reachable from and initial state, and if so, how?
- LTS for Mutual Exclusion has \(6 \times 6 \times 2 = 144\) possible well-typed states.
  - Is it possible to reach a state where \(pc1 = m5 \land pc2 = n5\)?
  - How can we state these questions rigorously, formally?
  - Can we find an algorithm to answer these types of questions?

Linear Temporal Logic - Syntax

\[ \varphi ::= p(\varphi) \mid \Diamond \varphi \mid \Box \varphi \mid p \varphi \land q \varphi \mid p \varphi \lor q \varphi \mid p \varphi \land q \varphi \mid \varphi \land q \varphi \mid p \varphi \lor q \varphi \mid (\varphi \land q \varphi) \mid (\varphi \lor q \varphi)

- \(p\) - a proposition over state variables
- \(\Diamond \varphi\) - “next”\)
- \(\varphi \land q \varphi\) - “until”\)
- \(\varphi \lor q \varphi\) - “releases”\)
- \(\Box \varphi\) - “box”, “always”, “forever”\)
- \(\Diamond \varphi\) - “diamond”, “eventually”, “some time”\)

LTL Semantics: The Idea

- \(p \rightarrow p\)
- \(\Diamond \varphi \rightarrow \varphi\)
- \(\varphi \land q \varphi \rightarrow \varphi \land q \varphi\)
- \(\varphi \lor q \varphi \rightarrow \varphi \lor q \varphi\)
- \(\Box \varphi \rightarrow \Box \varphi\)
- \(\Diamond \varphi \rightarrow \Diamond \varphi\)

Examples (cont)

- \(\text{NSC} \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Red} \rightarrow \text{NSC} = \text{Green}\)
- \(\text{NSY} \quad \text{NSC} = \text{Green} \rightarrow \text{NSC} = \text{Yellow}\)
- \(\text{NSR} \quad \text{NSC} = \text{Yellow} \rightarrow (\text{Turn, NSC}) = (\text{EW, Red})\)
- \(\text{EWG} \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Red} \rightarrow \text{EWC} = \text{Green}\)
- \(\text{EYW} \quad \text{EWC} = \text{Green} \rightarrow \text{EWC} = \text{Yellow}\)
- \(\text{EWR} \quad \text{EWC} = \text{Yellow} \rightarrow (\text{Turn, EWC}) = (\text{NS, Red})\)

\(\text{init} = (\text{NSC} = \text{Red} \land \text{EWC} = \text{Red} \land (\text{Turn} = \text{NS} \lor \text{Turn} = \text{EW})\)

Examples (cont)

- LTS for traffic light has \(3 \times 3 \times 2 = 18\) possible well typed states
  - Is it possible to reach a state where \(\text{NSC} \neq \text{Red} \land \text{EWC} \neq \text{Red}\) from an initial state?
  - If so, what sequence of actions allows this?
  - Do all the immediate predecessors of a state where \(\text{NSC} = \text{Green} \lor \text{EWC} = \text{Green}\) satisfy \(\text{NSC} = \text{Red} \land \text{EWC} = \text{Red}\)?
  - If not, are any of those offended states reachable from and initial state, and if so, how?
- LTS for Mutual Exclusion has \(6 \times 6 \times 2 = 144\) possible well-typed states.
  - Is it possible to reach a state where \(pc1 = m5 \land pc2 = n5\)?
  - How can we state these questions rigorously, formally?
  - Can we find an algorithm to answer these types of questions?
### Some Common Combinations

- □p "p will hold infinitely often"
- ♦p "p will continuously hold from some point on"
- (□p) ⇒ (♦q) "if p happens infinitely often, then so does q"

### Some More Equivalences

- □ϕ = ϕ ∧ ♦ϕ
- ♦ϕ = ϕ ∨ ◦♦ϕ
- ϕ V ψ = (ϕ ∧ ψ) ∨ (ψ ∧ ◦(ϕ V ψ))
- ϕ U ψ = ψ ∨ (ϕ ∧ ◦(ϕ V ψ))
- □, ◦, U, V may all be understood recursively, by what they state about right now, and what they state about the future
- Caution: □ vs ◦, U vs V differ in their limit behavior

### Formal LTL Semantics

Given:
- G = (V, F, af, R, ar) signature expressing state propositions
- Q set of states,
- M modeling function over Q and G; M(q, ρ) is true iff q models ρ.
- Write q |= ρ.
- σ = q₀q₁...qᵣ... infinite sequence of state from Q.
- σᵢ = qᵢqᵢ₊₁...qᵣ... the iᵗʰ tail of σ

Say σ models LTL formula ϕ, write σ |= ϕ as follows:
- σ |= ϕ iff q₀ |= ϕ
- σ |= ¬ϕ iff σ |= ϕ
- σ |= ϕ ∧ ψ iff σ |= ϕ and σ |= ψ.
- σ |= ϕ ∨ ψ iff σ |= ϕ or σ |= ψ.

### Traffic Light Example

Basic Behavior:
- □((NSC = Red) ∨ (NSC = Green) ∨ (NSC = Yellow))
- □((NSC = Red) ⇒ ((NSC ≠ Green) ∧ (NSC ≠ Yellow)))
- Similarly for Green and Red
- □(((NCS = Red) ∧ o(NCS ≠ Red)) ⇒ o(NCS = Green))
- Same as □((NCS = Red) ⇒ ((NCS = Red) U (NCS = Green)))
- □(((NCS = Green) ∧ o(NCS ≠ Green)) ⇒ o(NCS = Yellow))
- □(((NCS = Yellow) ∧ o(NCS ≠ Yellow)) ⇒ o(NCS = Red))
- Same for EWC
**Traffic Light Example**

**Basic Safety**
- $\square((\text{NSC} = \text{Red}) \lor (\text{EWC} = \text{Red}))$
- $\square((\text{NSC} = \text{Red}) \land (\text{EWC} = \text{Red})) \lor
  ((\text{NSC} \neq \text{Green}) \Rightarrow (\diamond (\text{NSC} = \text{Green}))))$

**Basic Liveness**
- $\langle (\text{NSC} = \text{Red}) \rangle \land (\langle (\text{NSC} = \text{Green}) \rangle \land (\langle (\text{NSC} = \text{Yellow}) \rangle)$
- $\langle (\text{EWC} = \text{Red}) \rangle \land (\langle (\text{EWC} = \text{Green}) \rangle \land (\langle (\text{EWC} = \text{Yellow}) \rangle)$

---

**Proof System for LTL**

- **First step:** View $\varphi \lor \psi$ as macro: $\varphi \lor \psi = \neg((\neg \varphi) \mathcal{U} (\neg \psi))$
- **Second Step:** Extend all rules of Prop Logic to LTL
- **Third Step:** Add one more rule: $\square \varphi \Rightarrow \square \varphi$
- **Fourth Step:** Add a collection of axioms (a sufficient set of 8 exists)
  - A1: $\square \varphi \Rightarrow \diamond (\neg \varphi)$
  - A2: $\square (\varphi \Rightarrow \psi) \Rightarrow (\square \varphi \Rightarrow \square \psi)$
  - A3: $\square \varphi \Rightarrow (\varphi \land \square \varphi)$
  - A4: $\varphi \Rightarrow \diamond \varphi$
  - A5: $\diamond (\varphi \Rightarrow \psi) \Rightarrow (\varphi \Rightarrow \diamond \psi)$
  - A6: $\square (\varphi \Rightarrow \diamond \psi) \Rightarrow (\varphi \Rightarrow \square \psi)$
  - A7: $\varphi \mathcal{U} \psi \Rightarrow (\varphi \land \psi) \lor (\varphi \land \diamond (\varphi \lor \psi))$
  - A8: $\varphi \mathcal{U} \psi \Rightarrow \diamond \psi$
- **Result:** A sound and relatively complete proof system
- **Can implement in Isabelle in much the same way as we did Hoare Logic**