CS477 Formal Software Development Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

Slides based in part on previous lectures by Mahesh Vishwanathan, and by Gul Agha

April 4, 2014
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

- $Q = \{\text{Start, Select, GetMarsBar, GetKitKatBar}\}$
- $I = \{\text{Start}\}$
- $\Sigma = \{\text{Pay, ChooseMarsBar, ChooseKitKatBar, TakeCandy}\}$
- $\delta = \left\{ \begin{array}{l}
(\text{Start, Pay, Select}) \\
(\text{Select, ChooseMarsBar, GetMarsBar}) \\
(\text{Select, ChooseKitKatBar, GetKitKatBar}) \\
(\text{GetMarsBar, TakeCandy, Start}) \\
(\text{GetKitKatBar, TakeCandy, Start})
\end{array} \right\}$
Example: Candy Machine

- Start
  - Pay
    - TakeCandy
      - Select
        - ChooseMarsBar
          - GetMarsBar
        - ChooseKitKatBar
          - GetKitKatBar
Let \((Q, \Sigma, \delta, I)\) be a labeled transition system.

\[
\begin{align*}
\ln(q, \alpha) &= \{ q' | q' \xrightarrow{\alpha} q \} \\
\ln(q) &= \bigcup_{\alpha \in \Sigma} \ln(q, \alpha) \\
\out(q, \alpha) &= \{ q' | q \xrightarrow{\alpha} q' \} \\
\out(q) &= \bigcup_{\alpha \in \Sigma} \out(q, \alpha)
\end{align*}
\]

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

\[
|I| \leq 1 \text{ and } |\out(q, \alpha)| \leq 1
\]
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 $\rho = q_0\alpha_1 q_1 \ldots \alpha_n q_n \ldots$ such that

- $q_0 \in I$
- $q_{i-1} \xrightarrow{\alpha_i} q_i$ 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 $\rho = q_0\alpha_1 q_1 \ldots \alpha_n q_n \ldots$ be a partial execution. Then $\text{trace}(\rho) = \alpha_1 \ldots \alpha_n \ldots$.

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

Example: Candy Machine

- **Partial execution:**
  \[ \rho = \text{Start} \cdot \text{Pay} \cdot \text{Select} \cdot \text{ChooseMarsBar} \cdot \text{GetMarsBar} \cdot \text{TakeCandy} \cdot \text{Start} \]

- **Trace:**  \[ \text{trace}(\rho) = \text{Pay} \cdot \text{ChooseMarsBar} \cdot \text{TakeCandy} \]

- **Run:**  \[ \text{run}(\rho) = \text{Start} \cdot \text{Select} \cdot \text{GetMarsBar} \cdot \text{Start} \]
A Program Transition System is a triple \((S, T, init)\)

- \(S = (G, D, F, \phi, R, \rho)\) is a first-order structure over signature \(G = (V, F, af, R, ar)\), used to interpret expressions and conditionals.
- \(T\) is a finite set of conditional transitions of the form
  \[ g \rightarrow (v_1, \ldots, v_n) := (e_1, \ldots, e_n) \]
  where \(v_i \in V\) distinct, and \(e_i\) term in \(G\), for \(i = 1 \ldots n\).
- \(init\) initial condition asserted to be true at start of program.
Example: Traffic Light

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

\[ R = \{ = \} \]

\[ \text{NSG} \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Red} \rightarrow \text{NSC} := \text{Green} \]

\[ \text{NSY} \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Green} \rightarrow \text{NSC} := \text{Yellow} \]

\[ \text{NSR} \quad \text{Turn} = \text{NS} \land \text{NSC} = \text{Yellow} \rightarrow (\text{Turn, NSC}) := (\text{EW, Red}) \]

\[ \text{EWG} \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Red} \rightarrow \text{EWC} := \text{Green} \]

\[ \text{EWY} \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Green} \rightarrow \text{EWC} := \text{Yellow} \]

\[ \text{EWR} \quad \text{Turn} = \text{EW} \land \text{EWC} = \text{Yellow} \rightarrow (\text{Turn, EWC}) := (\text{NS, Red}) \]

\[ \text{init} = (\text{NSC} = \text{Red} \land \text{EWC} = \text{Red} \land (\text{Turn} = \text{NS} \lor \text{Turn} = \text{EW})) \]
Mutual Exclusion (Attempt)

\[\begin{align*}
P1 :: & \quad m1 : \text{ while true do} \\
& \quad m2 : \quad p1(\text{*not in crit sect*}) \\
& \quad m3 : \quad c1 := 0 \\
& \quad m4 : \quad \text{wait}(c2 = 1) \\
& \quad m5 : \quad r1(\text{*in crit sect*}) \\
& \quad m6 : \quad c1 := 1 \\
& \quad m7 : \quad \text{od} \\
\end{align*}\]

\[\begin{align*}
P2 :: & \quad n1 : \text{ while true do} \\
& \quad n2 : \quad p2(\text{*not in crit sect*}) \\
& \quad n3 : \quad c2 := 0 \\
& \quad n4 : \quad \text{wait}(c1 = 1) \\
& \quad n5 : \quad r2(\text{*in crit sect*}) \\
& \quad n6 : \quad c2 := 1 \\
& \quad n7 : \quad \text{od} \\
\end{align*}\]
Mutual Exclusion PTS

\[ V = \{ pc_1, pc_2, c_1, c_2 \}, \quad F = \{ m_1, \ldots, m_6, n_1, \ldots, n_6, 0, 1 \} \]

\[ T = \begin{align*}
pc_1 &= m_1 \quad \rightarrow \quad pc_1 := m_2 \\
pc_1 &= m_2 \quad \rightarrow \quad pc_1 := m_3 \\
pc_1 &= m_3 \quad \rightarrow \quad (pc_1, c_1) := (m_4, 0) \\
pc_1 &= m_4 \land c_2 = 1 & \text{to} & \quad pc_1 := m_5 \\
pc_1 &= m_5 \quad \rightarrow \quad pc_1 := m_6 \\
pc_1 &= m_6 \quad \rightarrow \quad (pc_1, c_1) := (m_1, 1) \\
pc_2 &= n_1 \quad \rightarrow \quad pc_2 := n_2 \\
pc_2 &= n_2 \quad \rightarrow \quad pc_2 := n_3 \\
pc_2 &= n_3 \quad \rightarrow \quad (pc_2, c_2) := (n_4, 0) \\
pc_2 &= n_4 \land c_1 = 1 & \text{to} & \quad pc_2 := n_5 \\
pc_2 &= n_5 \quad \rightarrow \quad pc_2 := n_6 \\
pc_2 &= n_6 \quad \rightarrow \quad (pc_2, c_2) := (n_1, 1)
\end{align*} \]

\[ \text{init} = (pc_1 = m_1 \land pc_2 = n_1 \land c_1 = 1 \land c_2 = 1) \]
Let \((S, T, \text{init})\) 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.
  - Can restrict to mappings \(q\) where \(v\) and \(q(v)\) have same type.
- Let \(\Sigma = T\).
- Let \(\delta = \{(q, g \rightarrow (v_1, \ldots, v_n) := (e_1, \ldots, e_n), q') | M_q(g) \land (
\forall i \leq n. q'(v_i) = T_q(e_i)) \land \n\forall v \notin \{v_1, \ldots, v_n\}. q'(v) = q(v))\}\).
- \(I = \{q | T_q(\text{init}) = T\}\).
Example: Traffic Lights

1GG

1GY

1YY

1YG

1GR

2RG

2RY

2GY

2YY

2GG

NSY

NSR

EWR

NSG

EWR

NSY

EWR

NSY
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 $NSC \neq Red \land EWC \neq Red$ from an initial state?
  - If so, what sequence of actions allows this?
  - Do all the immediate predecessors of a state where $NSC = Green \lor EWC = Green$ satisfy $NSC = Red \land EWC = Red$?
  - If not, are any of those offend states reachable from an initial state, and if so, how?

- LTS for Mutual Exclusion has $6 \times 6 \times 2 \times 2 = 144$ 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

\[ \varphi ::= p | (\varphi) | \neg \varphi | \varphi \land \varphi' | \varphi \lor \varphi' | \circ \varphi | \varphi U \varphi' | \varphi V \varphi' | \Box \varphi | \Diamond \varphi \]

- **p** – a proposition over state variables
- **\circ \varphi** – “next”
- **\varphi U \varphi'** – “until”
- **\varphi V \varphi'** – “releases”
- **\Box \varphi** – “box”, “always”, “forever”
- **\Diamond \varphi** – “diamond”, “eventually”, “sometime”
### LTL Semantics: The Idea

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$p$</td>
</tr>
<tr>
<td>$\circ \varphi$</td>
<td>$\varphi$</td>
</tr>
<tr>
<td>$\varphi U \psi$</td>
<td>$\varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \psi$</td>
</tr>
<tr>
<td>$\varphi V \psi$</td>
<td>$\psi \psi \psi \psi \psi \psi \varphi, \psi$</td>
</tr>
<tr>
<td>$\square \varphi$</td>
<td>$\varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi \varphi$</td>
</tr>
<tr>
<td>$\diamond \varphi$</td>
<td>$\varphi$</td>
</tr>
</tbody>
</table>
Formal LTL Semantics

Given:

- $G = (V, F, af, R, ar)$ signature expressing state propositions
- $Q$ set of states,
- $M$ modeling function over $Q$ and $G$: $M(q, p)$ is true iff $q$ models $p$.
  Write $q \models p$.
- $\sigma = q_0q_1 \ldots q_n \ldots$ infinite sequence of state from $Q$.
- $\sigma^i = q_iq_{i+1} \ldots q_n \ldots$ the $i^{th}$ tail of $\sigma$

Say $\sigma$ models LTL formula $\varphi$, write $\sigma \models \varphi$ as follows:

- $\sigma \models p$ iff $q_0 \models p$
- $\sigma \models \neg \varphi$ iff $\sigma \not\models \varphi$
- $\sigma \models \varphi \land \psi$ iff $\sigma \models \varphi$ and $\sigma \models \psi$.
- $\sigma \models \varphi \lor \psi$ iff $\sigma \models \varphi$ or $\sigma \models \psi$.  

Elsa L Gunter ()
CS477 Formal Software Development Method
Formal LTL Semantics

- $\sigma \models \circ \varphi$ iff $\sigma^1 \models \varphi$
- $\sigma \models \varphi U \psi$ iff for some $k$, $\sigma^k \models \psi$ and for all $i < k$, $\sigma^i \models \varphi$
- $\sigma \models \varphi V \psi$ iff for some $k$, $\sigma^k \models \varphi$ and for all $i \leq k$, $\sigma^i \models \psi$, or for all $i$, $\sigma^i \models \psi$.
- $\sigma \models \Box \varphi$ if for all $i$, $\sigma^i \models \psi$
- $\sigma \models \Diamond \varphi$ if for some $i$, $\sigma^i \models \psi$
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 Equivalences

- \( \Box(\varphi \land \psi) = (\Box \varphi) \land (\Box \psi) \)
- \( \Diamond(\varphi \lor \psi) = (\Diamond \varphi) \lor (\Diamond \psi) \)
- \( \Box \varphi = \mathbf{F} \lor \varphi \)
- \( \Diamond \varphi = \mathbf{T} \lor \varphi \)
- \( \varphi \lor \psi = \neg((\neg \varphi) \lor (\neg \psi)) \)
- \( \varphi \lor \psi = \neg((\neg \varphi) \lor (\neg \psi)) \)
- \( \neg(\Diamond \varphi) = \Box(\neg \varphi) \)
- \( \neg(\Box \varphi) = \Diamond(\neg \varphi) \)
Some More Equivalences

- $\square \varphi = \varphi \land \circ \square \varphi$
- $\Diamond \varphi = \varphi \lor \circ \Diamond \varphi$
- $\varphi \lor \psi = (\varphi \land \psi) \lor (\psi \land \circ (\varphi \lor \psi))$
- $\varphi \lor \psi = \psi \lor (\varphi \land \circ (\varphi \lor \psi))$
- $\square$, $\Diamond$, $\lor$, $\lor$ may all be understood recursively, by what they state about right now, and what they state about the future
- Caution: $\square$ vs $\Diamond$, $\lor$ vs $\lor$ differ in there limit behavior
Traffic Light Example

Basic Behavior:

- $\Box((\text{NSC} = \text{Red}) \lor (\text{NSC} = \text{Green}) \lor (\text{NSC} = \text{Yellow}))$
- $\Box((\text{NSC} = \text{Red}) \Rightarrow ((\text{NSC} \neq \text{Green}) \land (\text{NSC} \neq \text{Yellow})))$
- Similarly for Green and Red
- $\Box(((\text{NCS} = \text{Red}) \land \circ(\text{NCS} \neq \text{Red})) \Rightarrow \circ(\text{NCS} = \text{Green}))$
- Same as $\Box((\text{NCS} = \text{Red}) \Rightarrow ((\text{NCS} = \text{Red}) \cup (\text{NCS} = \text{Green})))$
- $\Box(((\text{NCS} = \text{Green}) \land \circ(\text{NCS} \neq \text{Green})) \Rightarrow \circ(\text{NCS} = \text{Yellow}))$
- $\Box(((\text{NCS} = \text{Yellow}) \land \circ(\text{NCS} \neq \text{Yellow})) \Rightarrow \circ(\text{NCS} = \text{Red}))$
- Same for EWC
Traffic Light Example

Basic Safety

- □((\textit{NSC} = \textit{Red}) \lor (\textit{EWC} = \textit{Red}))
- □( (\textit{NSC} = \textit{Red}) \land (\textit{EWC} = \textit{Red})) \lor
  \ ((\textit{NSC} \neq \textit{Green}) \Rightarrow (\Diamond(\textit{NSC} = \textit{Green}))))

Basic Liveness

- (\Diamond(\textit{NSC} = \textit{Red})) \land (\Diamond(\textit{NSC} = \textit{Green})) \land (\Diamond(\textit{NSC} = \textit{Yellow}))
- (\Diamond(\textit{EWC} = \textit{Red})) \land (\Diamond(\textit{EWC} = \textit{Green})) \land (\Diamond(\textit{EWC} = \textit{Yellow}))
Proof System for LTL

First step: View $\varphi \lor \psi$ as macro: $\varphi \lor \psi = \neg((\neg \varphi) U (\neg \psi))$

Second Step: Extend all rules of Prop Logic to LTL

Third Step: Add one more rule: $\square \varphi$ Gen $\varphi$

Fourth Step: Add a collection of axioms (a sufficient set of 8 exists)
Result: a sound and relatively complete proof system