CS477 Formal Software Dev 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

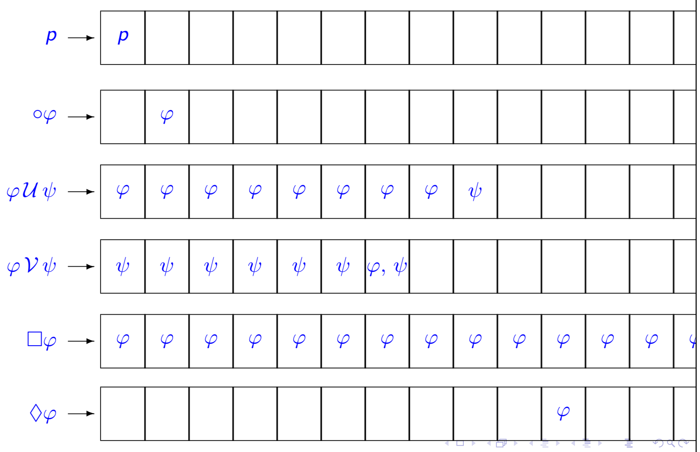
April 3, 2020

Linear Temporal Logic - Syntax

$$\varphi ::= p \mid (\varphi) \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi'$$
$$\mid \; \circ\varphi \mid \varphi\,\mathcal{U}\,\varphi' \mid \varphi\,\mathcal{V}\,\varphi' \mid \Box\varphi \mid \Diamond\varphi$$

- p – a propostion over state variables
- $\circ\varphi$ – "next"
- $\varphi\mathcal{U}\varphi'$ – "until"
- $\varphi\mathcal{V}\varphi'$ – "releases"
- $\Box\varphi$ – "box", "always", "forever"
- $\Diamond\varphi$ – "diamond", "eventually", "sometime"

LTL Semantics: The Idea



Formal LTL Semantics

Given:

- $\mathcal{G} = (V, F, af, R, ar)$ signature expressing state propositions
- Q set of states,
- $\mathcal{M}$ modeling function over Q and $\mathcal{G}$: $\mathcal{M}(q, p)$ is true iff q models p. Write $q \models p$.
- $\sigma = q_0 q_1 \ldots q_n \ldots$ infinite sequence of state from Q.
- $\sigma^i = q_i q_{i+1} \ldots q_n \ldots$ the i^{th} tail of σ

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

- $\sigma \models p$ iff $q_0 \models p$
- $\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$
- $\sigma \models \varphi \wedge \psi$ iff $\sigma \models \varphi$ and $\sigma \models \psi$.
- $\sigma \models \varphi \vee \psi$ iff $\sigma \models \varphi$ or $\sigma \models \psi$.

Formal LTL Semantics

- $\sigma \models \circ\varphi$ iff $\sigma^1 \models \varphi$
- $\sigma \models \varphi\mathcal{U}\psi$ iff for some k, $\sigma^k \models \psi$ and for all $i < k$, $\sigma^i \models \varphi$
- $\sigma \models \varphi\mathcal{V}\psi$ 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

- $\Box\Diamond p$ "p will hold infinitely often"
- $\Diamond\Box p$ "p will continuously hold from some point on"
- $(\Box p) \Rightarrow (\Box q)$ "if p happens infinitely often, then so does q

Some Equivalences

- $\Box(\varphi \wedge \psi) = (\Box\varphi) \wedge (\Box\psi)$
- $\Diamond(\varphi \vee \psi) = (\Diamond\varphi) \vee (\Diamond\psi)$
- $\Box\varphi = \mathbf{F}\,\mathcal{V}\,\varphi$
- $\Diamond\varphi = \mathbf{T}\,\mathcal{U}\,\varphi$
- $\varphi\,\mathcal{V}\,\psi = \neg((\neg\varphi)\,\mathcal{U}\,(\neg\psi))$
- $\varphi\,\mathcal{U}\,\psi = \neg((\neg\varphi)\,\mathcal{V}\,(\neg\psi))$
- $\neg(\Diamond\varphi) = \Box(\neg\varphi)$
- $\neg(\Box\varphi) = \Diamond(\neg\varphi)$

Some More Eqivalences

- $\Box\varphi = \varphi \wedge \circ\Box\varphi$
- $\Diamond\varphi = \varphi \vee \circ\Diamond\varphi$
- $\varphi\,\mathcal{V}\,\psi = (\varphi \wedge \psi) \vee (\psi \wedge \circ(\varphi\,\mathcal{V}\,\psi))$
- $\varphi\,\mathcal{U}\,\psi = \psi \vee (\varphi \wedge \circ(\varphi\,\mathcal{U}\,\psi)$
- $\Box$, $\Diamond$, $\mathcal{U}$, $\mathcal{V}$ may all be understood recursively, by what they state about right now, and what they state about the future
- Caution: $\Box$ vs $\Diamond$, $\mathcal{U}$ vs $\mathcal{V}$ differ in there limit behavior

Traffic Light Example

Basic Behavior:

- $\Box((NSC = Red) \vee (NSC = Green) \vee (NSC = Yellow))$
- $\Box((NSC = Red) \Rightarrow ((NSC \neq Green) \wedge (NSC \neq Yellow))$
- Similarly for *Green* and *Red*
- $\Box(((NCS = Red) \wedge \circ(NCS \neq Red)) \Rightarrow \circ(NCS = Green))$
- Same as $\Box((NCS = Red) \Rightarrow ((NCS = Red)\,\mathcal{U}\,(NCS = Green)))$
- $\Box(((NCS = Green) \wedge \circ(NCS \neq Green)) \Rightarrow \circ(NCS = Yellow))$
- $\Box(((NCS = Yellow) \wedge \circ(NCS \neq Yellow)) \Rightarrow \circ(NCS = Red))$
- Same for *EWC*

Traffic Light Example

Basic Safety

- $\Box((NSC = Red) \vee (EWC = Red)$
- $\Box(\,((NSC = Red) \wedge (EWC = Red))\,\mathcal{V}$
  $((NSC \neq Green) \Rightarrow (\circ(NSC = Green))))$

Basic Liveness

- $\Box((\Diamond(NSC = Red)) \wedge (\Diamond(NSC = Green)) \wedge (\Diamond(NSC = Yellow)))$
- $\Box((\Diamond(EWC = Red)) \wedge (\Diamond(EWC = Green)) \wedge (\Diamond(EWC = Yellow)))$

What is Model Checking?

Most generally Model Checking is

- an automated technique, that given
- a finite-state model M of a system
- and a logical property φ,
- checks whether the property holds of model: $M \models \varphi$?

Model Checking

- Model checkers usually give example of failure if $M \not\models \varphi$.
- This makes them useful for debugging.
- Problem: Can only handle finite models: unbounded or continuous data sets can't be directly handled
  - Symbolic model checking can handle limited cases of finitely presented models
- Problem: Number of states grows exponentially in the size of the system.
- Answer: Use abstract model of system
- Problem: Relationship of results on abstract model to real system?

LTL Model Checking

- Model Checking Problem: Given model $\mathcal{M}$ amd logical property φ of $\mathcal{M}$, does $\mathcal{M} \models \varphi$?
- Given transition system with states Q, transition relation δ and inital state state I, say $(Q, \delta, I) \models \varphi$ for LTL formula φ if every run of (Q, δ, I), σ satisfies $\sigma \models \varphi$.

Theorem

The Model Checking Problem for finite transition systems and LTL formulae is decideable.

- Treat states $q \in Q$ as letters in an alphabet.
- Language of (Q, δ, I), $\mathcal{L}(Q, \delta, I)$ (or L(Q) for short) is set of runs in Q
- Language of φ, $\mathcal{L}\varphi = \{\sigma | \sigma \models \varphi\}$
- Question: $\mathcal{L}(Q) \subseteq \mathcal{L}(\varphi)$?
- Same as: $\mathcal{L}(Q) \cap \mathcal{L}(\neg\varphi) = \emptyset$?

How to Decide the Model Checking Problem?

- How to answer $\mathcal{L}(Q) \cap \mathcal{L}(\neg\varphi) = \emptyset$?
- Common approach:
  - Build automaton A such the $\mathcal{L}(A) = \mathcal{L}(Q) \cap \mathcal{L}(\neg\varphi)$
  - Are accepting states of A reachable? (Infinitely often?)
- How to build A?
  - One possible answer: Build a series of automata by recursion on structure of $\neg\varphi$.
  - Another possible answer: Build an automaton B such $\mathcal{L}(B) = \mathcal{L}(\neg\varphi)$; take $A = B \times Q$