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 16, 2014
Formal LTL Semantics

- $\sigma \models p$ 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 $\sigma \models \psi$.
- $\sigma \models \circ \varphi$ 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 More Equivalences

- $\square \varphi \iff \varphi \land \circ \square \varphi$
- $\Diamond \varphi \iff \varphi \lor \circ \Diamond \varphi$
- $\varphi \mathcal{U} \psi \iff \phi \lor (\psi \land \circ (\varphi \mathcal{U} \psi))$
- $\varphi \mathcal{V} \psi \iff (\varphi \land \psi) \lor (\varphi \land \circ (\varphi \mathcal{V} \psi))$

- $\square$, $\Diamond$, $\mathcal{U}$, $\mathcal{V}$ may all be understood recursively, by what they state about right now, and what they state about the future.

- Caution: $\square$ vs $\Diamond$, $\mathcal{U}$ vs $\mathcal{V}$ differ in their limit behavior.
Traffic Light Example

Basic Behavior:

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

Basic Safety

- $\Box((\text{NSC} = \text{Red}) \lor (\text{EWC} = \text{Red}))$
- $\Box((\text{NSC} = \text{Red}) \land (\text{EWC} = \text{Red})) \lor ((\text{NSC} \neq \text{Green}) \Rightarrow (\lozenge(\text{NSC} \neq \text{Green}))))$

Basic Liveness

- $(\lozenge(\text{NSC} = \text{Red})) \land (\lozenge(\text{NSC} = \text{Green})) \land (\lozenge(\text{NSC} = \text{Yellow}))$
- $(\lozenge(\text{EWC} = \text{Red})) \land (\lozenge(\text{EWC} = \text{Green})) \land (\lozenge(\text{EWC} = \text{Yellow}))$
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: $\frac{\varphi}{\Box \varphi}$ (Gen)
- Fourth Step: Add a collection of axioms (a sufficient set of 8 exists)
  1. A1: $\Box \varphi \iff \neg(\Diamond(\neg \varphi))$
  2. A2: $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box \varphi \Rightarrow \Box \psi)$
  3. A3: $\Box \varphi \Rightarrow (\varphi \land \circ \Box \varphi)$
  4. A4: $\circ \neg \varphi \iff \neg \circ \varphi$
  5. A5: $\circ(\varphi \Rightarrow \psi) \Rightarrow (\circ \varphi \Rightarrow \circ \psi)$
  6. A6: $\Box(\varphi \Rightarrow \circ \varphi) \Rightarrow (\varphi \Rightarrow \Box \varphi)$
  7. A7: $\varphi U \psi \iff (\varphi \land \psi) \lor (\varphi \land \circ(\varphi \lor \psi))$
  8. A8: $\varphi U \psi \Rightarrow \Diamond \psi$

- Result: a sound and relatively complete proof system
- Can implement in Isabelle in much the same way as we did Hoare Logic
Important Meta-Definitions

- **A** is sound with respect to **B** if things that are “true” according to **A** are things that are “true” according to **B**.
- **A** is complete with respect to **B** if things that are “true” according to **B** are things that are “true” according to **A**.
- **A** is sound if things that are “true” according to **A** are true.
- **A** is complete everything that is true (that is in the scope of **A**) is “true” according to **A**.
- **A** is relatively complete with respect to **B** if **A** is complete when **B** is.

Think: **A** proof system; **B** mathematical model
Exercise: \((\varphi \land \psi) \Rightarrow (\Box \varphi \land \Box \psi)\)
What is Model Checking?

Most generally **Model Checking** is

- an **automated** technique, that given
- a **finitely presented** (think **finite-state**) model $M$ of a system
- and a **logical** property $\varphi$,
- **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
- **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?
What is Model Checking?

- **Clarke & Emerson 1981**: Model checking is an automated technique that, given a finite-state model of a system and a logical property, systematically checks whether this property holds for (a given initial state in) that model.

Although finite-state, the model of a system typically grows exponentially.

- Based on Vardi & Wolper 1986.

**System Development**

- System Engineering
- Analysis
- Design
- Code
- Testing
- Maintenance

- “Classic” Model Checking
- “Modern” Model Checking

Classic “waterfall model” [Pressman 1996]
"Classic" Model Checking

1. (initial) Design
2. (manual) abstractions
3. Abstract Verification Model
   - refinement techniques
4. Implementation
5. Model Checker

- Abstraction is the key activity in both approaches.
- This talk deals with pure SPIN, i.e., the "classic" model checking approach.
LTL Model Checking

- **Model Checking Problem**: Given model $M$ and logical property $\varphi$ of $M$, does $M \models \varphi$?

- Given transition system with states $Q$, transition relation $\delta$ and initial state state $I$, say $(Q, \delta, I) \models \varphi$ for LTL formula $\varphi$ if every run of $(Q, \delta, I)$, $\sigma$ satisfies $\sigma \models \varphi$.

**Theorem**

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

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

- How to answer $L(Q) \cap L(\neg \varphi) = \emptyset$?
  - Common approach:
    - Build automaton $A$ such that $L(A) = L(Q) \cap L(\neg \varphi)$
    - 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 $L(B) = L(\neg \varphi)$; take $A = B \times Q$

- Will do at least one approach if time after Spin