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 8, 2020

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: $M \models \varphi$?
- If M is a transition system, $M \models \varphi$ if $\sigma \models \varphi$ for every run σ of M.

Model Checking

- Model checkers usually give example of failure if $M \not\models \varphi$, e.g. a run σ of M such that $\sigma \not\models \varphi$
- 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 M amd logical property φ of M, does $M \models \varphi$?
- Given transition system M with states Q, transition relation δ and inital state state I, say $M \models \varphi$ for LTL formula φ if every run σ of $M = (Q, \delta, I)$, σ satisfies φ, that is $\sigma \models \varphi$.

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 $\mathcal{L}(M)$ for short) is set of runs in M
- Language of φ, $\mathcal{L}\varphi = \{\sigma \in Q^\omega \mid \sigma \models \varphi\}$
- Question: $\mathcal{L}(M) \subseteq \mathcal{L}(\varphi)$?
- Same as: $\mathcal{L}(M) \cap \mathcal{L}(\neg\varphi) = \emptyset$?

How to Decide the Model Checking Problem?

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

Reducing LTL

- LTL given by

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

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

  and thus
  - $\neg(\varphi\,\mathcal{V}\,\psi) = (\neg\varphi)\,\mathcal{U}\,(\neg\psi)$
  - $\neg(\varphi\,\mathcal{U}\,\psi) = (\neg\varphi)\,\mathcal{V}\,(\neg\psi)$
- Can eliminate $\Box$ and $\Diamond$, and always move negation down to state predicates p.

Reduced LTL

- LTL given by

$$\begin{array}{rl}\varphi ::= & p \mid (\varphi) \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \\ \mid & \circ\varphi \mid \varphi\,\mathcal{U}\,\varphi' \mid \varphi\,\mathcal{V}\,\varphi' \mid \Box\varphi \mid \Diamond\varphi\end{array}$$

- Equivalent language LTL^r given by

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

LTL_reduce

LTL_reduce$(p) = p$
LTL_reduce$(\neg p) = \neg p$
LTL_reduce$(\,(\varphi)\,) = (\,$LTL_reduce$(\varphi)\,)$
LTL_reduce$(\varphi \wedge \psi) = (LTL_reduce(\varphi)) \wedge (LTL_reduce(\psi))$
LTL_reduce$(\neg(\varphi \wedge \psi)) = (LTL_reduce(\neg(\varphi))) \vee (LTL_reduce(\neg(\psi)))$
LTL_reduce$(\varphi \vee \psi) = ($LTL_reduce$(\varphi)) \vee ($LTL_reduce$(\psi))$
LTL_reduce$(\neg(\varphi \vee \psi)) = (LTL_reduce(\neg(\varphi))) \wedge (LTL_reduce(\neg(\psi)))$
LTL_reduce$(\circ\varphi) = \circ($LTL_reduce$(\varphi))$
LTL_reduce$(\neg(\circ\varphi)) = \circ(LTL_reduce(\neg(\varphi)))$

LTL_reduce

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