EECS 219C: Computer-Aided Verification

# Games and Verification

Sanjit A. Seshia
EECS, UC Berkeley

## Today's Lecture

- The role of Games in Design & Verification
- Safety Games and their solution
- Two applications
  - Controller synthesis
  - Detecting errors before reaching them

# Scenario so far

- 2 (finite-state) machines:
  – M models the system
  – E models the environment
  – Compose M and E to get closed system and check property
- Traditional viewpoint: E is a conservative model of the environment
  – E models a worst-case (adversarial) scenario
  – Pros/cons of this approach?



# An Optimistic View

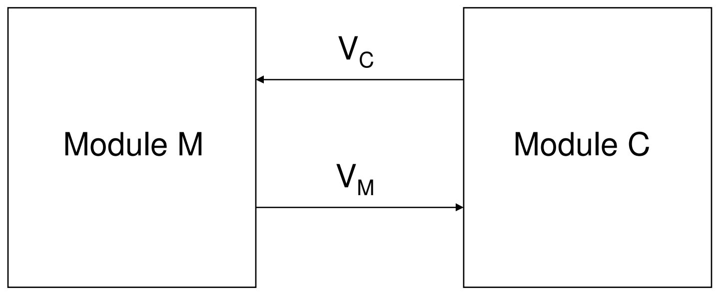
- Instead of asking:

  Does system M work correctly in all environments?
- Consider asking:

  Is there an env E in which M works correctly?
  – If yes, and we had one such E, how could we use it in practice?

# General Setting

State variables $V = V_C \cup V_M$ , $V_C \cap V_M = \phi$




C is "controller"
M's output cannot be controlled.



# An Instance

```
bool l;

lock() {
  assert(!l);
  l := 1;
  /* acquire lock */
  ...
}

unlock() {
  assert(l);
  l := 0;
  /* release lock */
  ...
}
```

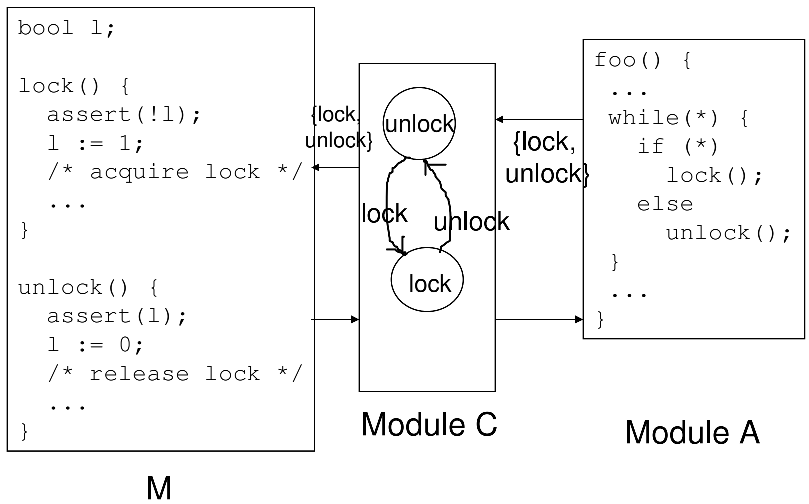
{lock, unlock}

```
foo() {
 ...
 while(*) {
   if (*)
     lock();
   else
     unlock();
 }
 ...
}
```

Module A

M

# An Instance






# Controller Synthesis

- Given finite-state machine M and an LTL formula $\psi$
- Is there a controller C which ensures that M || C satisfies $\psi$ ?
  - If yes, how do we find such a C?
  - If not, M is said to be uncontrollable (from its initial states)

# Controller Synthesis

- Given finite-state machines M and an LTL formula $\psi$
- Is there a controller C which ensures that M || C satisfies $\psi$ ?
  - If yes, how do we find such a C?
  - If not, M is said to be uncontrollable (from its initial states)
    - M is controllable from state s if considering s to be initial, M is controllable



# Games

- We view the problem as a game between the controller C and the system M
- Assume property $\psi$ = G p
- Player M wins if M||C reaches an error ($\neg$ p) state
- C wins if it keeps M||C outside the error states
- Assume perfect information: C and M have perfect knowledge about each other

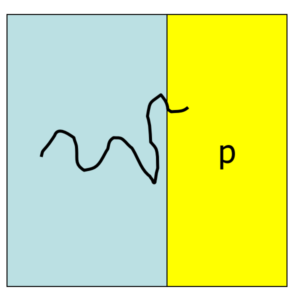
## Games on Graphs

- Defined over the state space S of M || C
- Asynchronous composition
  - Each node/state is either a "M state" or a "C state"
    - Assume one module changes variables at a time
    - "Turn-based" games
- Synchronous composition
  - Both M and C simultaneously decide their next states (moves) and move together



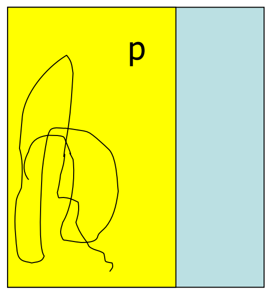
## Reachability Games

- Let p ⊆ S be a set of target states of M||C Reachability objective requires us to visit the set p
  - i.e., find C s.t. M||C satisfies LTL formula ___ ?



## Safety Games

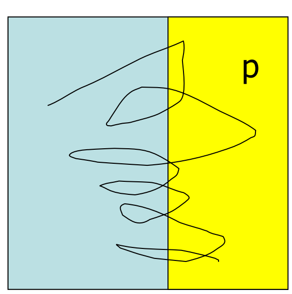
- Let p ⊆ S be the set of safe states
  Safety objective requires us never to visit any vertex outside p
  - i.e., find C s.t. M||C satisfies LTL formula ___






## Games with Buchi Objectives

- Let p ⊆ S be a set of states
  Buchi objective requires that the set p is visited infinitely often
  - i.e., find C s.t. M||C satisfies LTL formula ___



## Solving Safety Games

- Given: M, C, property Gp
  - Assume synchronous composition
- What we want:

  A strategy for C s.t. no matter what M does, C can keep M||C within the region satisfying p
- What is a "strategy for C" (informally)?

S. A. Seshia 15

## Strategy $\sigma$

- For C: Mapping from a finite history of states to next state values of $V_C$

  $\sigma_C : Val(V)^+ \rightarrow Val(V_C)$
- Similarly, strategy for M is

  $\sigma_M : Val(V)^+ \rightarrow Val(V_M)$
- Taken together, $\sigma_C$ and $\sigma_M$ define the next state for C||M
- C wins from initial state s if for every $\sigma_M$ it has a $\sigma_C$ that keeps C||M in the safe states
  - Note that initial state is important

S. A. Seshia 16

## Memoryless Strategy $\sigma$

- For C: Mapping from current state to next state values of $V_C$
  $\sigma_C : Val(V) \rightarrow Val(V_C)$
- Similarly, strategy for M is
  $\sigma_M : Val(V) \rightarrow Val(V_M)$
- Taken together, $\sigma_C$ and $\sigma_M$ define the next state for C||M

S. A. Seshia 17

## Local Strategy

- The overall strategy comprises many "local" decisions
  - which state to go to next
- Given a state s = ($s_M$, $s_C$) how should M and C choose their next states?

S. A. Seshia 18

# Local Strategy

- The overall strategy comprises many "local" decisions
  - which state to go to next
- Given a state $s = (s_M, s_C)$ how should M and C choose their next states?
  - No matter what C does, M wants to force it into an error state ($\neg$ p)
  - No matter what M does, C wants to continue satisfying p



# Controller Synthesis for Gp

- M chooses its next state according to its transition relation R
- We want to compute a transition relation (strategy) for C, $\sigma_C$ so that p is always true
- Given a state $s = (s_M, s_C)$,
  What is $\sigma_C(s, s_C')$ ?

## Controller Synthesis for Gp

- M chooses its next state according to its transition relation R
- We want to compute a transition relation (strategy) for C, $\sigma_C$ so that p is always true
- Given a state s = $(s_M, s_C)$,

$\sigma_C(s, s_C')$

= $\forall s_M' \; R(s, s_M') \rightarrow p(s')$

= Set of all pairs $(s, s_C')$ s.t. no matter what M does in s, p holds in s'



## Solving Safety Games backwards

- We can work backwards from error states
- $Pre_M(s)$

= set of states from which, regardless of the controller, M can enter an error ($\neg$ p) state

= $\forall s_C' \; \exists s_M' \; ( R(s, s_M') \wedge \neg p(s') )$

- Note: Pre is used above in a different sense from the normal pre operator
- If least fixed point of the following operator is B, then controllable states are $\neg$ B
  - $\tau(Z) = \neg p(s) \;\; \vee \;\; \forall s_C' \; \exists s_M' \; ( R(s, s_M') \wedge Z )$

# Early Error Detection

[de Alfaro, Henzinger, Mang, CAV'00]

- We can use the game formulation to speed up symbolic model checking of LTL properties
- Idea: (for Gp)
  - Given modules A and B
  - Find all states of A that are controllable w.r.t. Gp and similarly for B
    - Denote by $C_A$ and $C_B$
    - Then check if A||B satisfies $G(C_A \wedge C_B)$
    - Suppose this check fails. What do we know?

S. A. Seshia

23

# Early Error Detection

- Idea: (for Gp)
  - Given modules A and B
  - Find all states of A that are controllable w.r.t. Gp and similarly for B
    - Denote by $C_A$ and $C_B$
    - Then check if A||B satisfies $G(C_A \wedge C_B)$
    - Suppose this check fails. What do we know?
      - Either $C_A$ or $C_B$ is not satisfied in some state s of A||B
      - Say $C_A$: Thus, A is not controllable from s – no environment can prevent it from reaching a $\neg$ p state!
      - So we know that "A is doomed to fail" even before it fails!

S. A. Seshia

24

## Pros of Early Error Detection

- Computing $C_A$ and $C_B$ does not require composing A and B together
  - Avoids state space explosion
- Model checking for $G(C_A \wedge C_B)$ can find bugs faster
  - Reach uncontrollable states earlier
- Note: uncontrollable states are like the "root cause" of the bug
  - Useful for error localization



## Complexity

- Synthesis is (not surprisingly) harder than verification
- Verification of LTL properties of finite-state systems
  - PSPACE
- Synthesis of finite-state systems to satisfy an LTL objective
  - 2EXPTIME-complete
  - For Gp it is EXPTIME-complete

# Next class

- Model generation