

# Can a Model Checker Generate Tests for Non-Deterministic Systems?

Sergiy Boroday, Alexandre Petrenko

CRIM, Montreal, Canada

Roland Groz

INPG, France

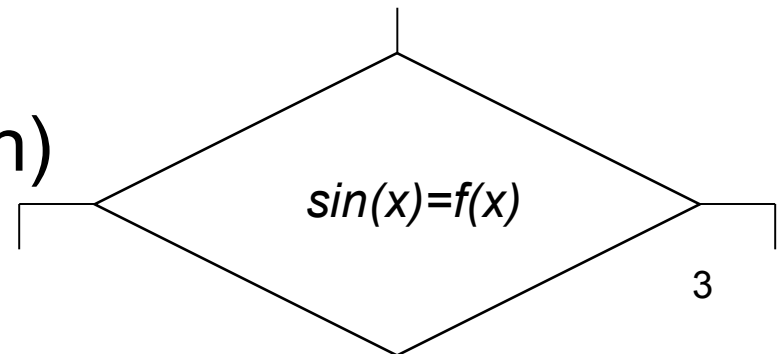
MBT 2007

# Outline

- Motivation
- Weak and Strong Tests
- Test Generation
  - Model Checking
    - Deterministic FSM
    - Weak Tests Non-deterministic FSM
  - Module Checking
    - Strong Tests for Non-deterministic FSM
- Conclusion

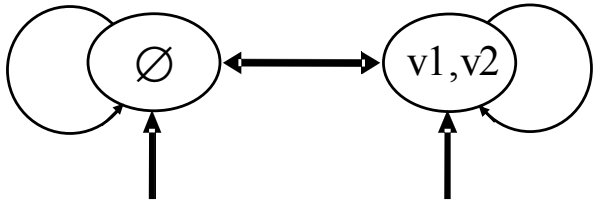
# Sources of ND

- The system under test
  - Concurrency/races
  - Timed
  - Background activities
  - Various configurations
- The model
  - Options or alternatives
  - Imprecise specification
  - Abstraction (simplification)

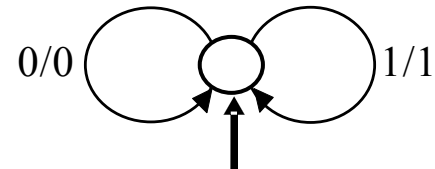


# State Based Formalisms

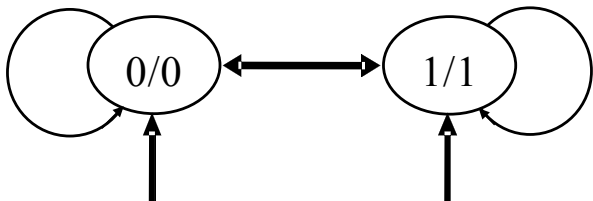
*Kripke Structure*



*Mealy FSM (transducer)*

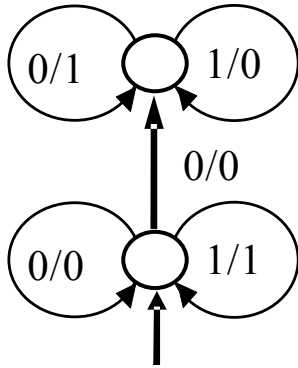


*Module*

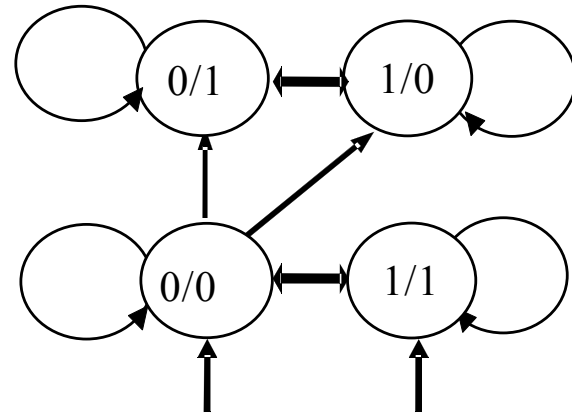


# ND Example

- Mealy FSM



- Module



# Black Box Testing

- Black box means that the full state of the system is not observable, in particular, some variables (actions) are
  - Unessential, or
  - Hidden from tester
    - instrumentation is usually limited
    - code is obfuscated
- White box is a special case when state is completely observable

# Mutation Based Testing

- Faults are modeled by mutant modules
- Mutation operators
  - Transitions redirected, added, removed, permuted...
  - Variables/labels changed, permuted...
  - Many are defined for SDL, EFSM...
- Here we allow any mutation preserving input and output variables
- A test should expose an unexpected behavior of a mutant w.r.t. a specification
- Mutant explosion could be handled by merging mutants (into a “meta-mutant”) and abstraction

# Strong and Weak Tests

## Strong test

(separating sequence)

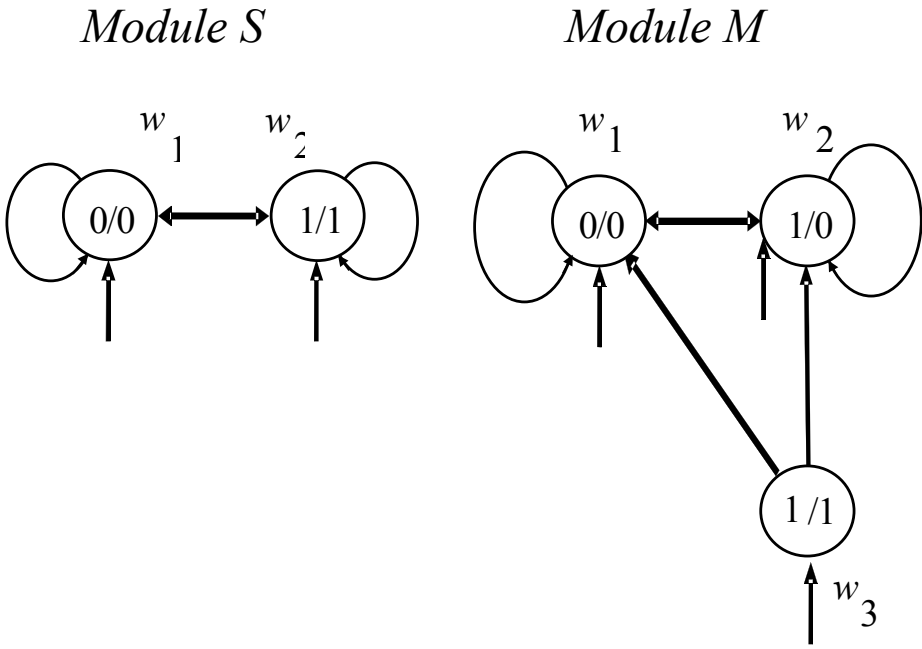
- (Finite) input sequence, such that sets of specification and mutant output sequences are disjoint
- Mutant is killed by a single shot, fault is detected

## Weak test

- (Finite) input sequence, such that at least one output sequence of the mutant is not allowed by specification
- May detect fault
  - with machine gun
  - completeness/Milner weather assumption
- May exist, even when strong test does not

# Strong and Weak Tests: Examples

For modules  $S$  and  $M$   
 input 1 is a weak test  
 11 is a strong test



	Input sequence	
	1	11
S	1	11
M	0 1	00 10

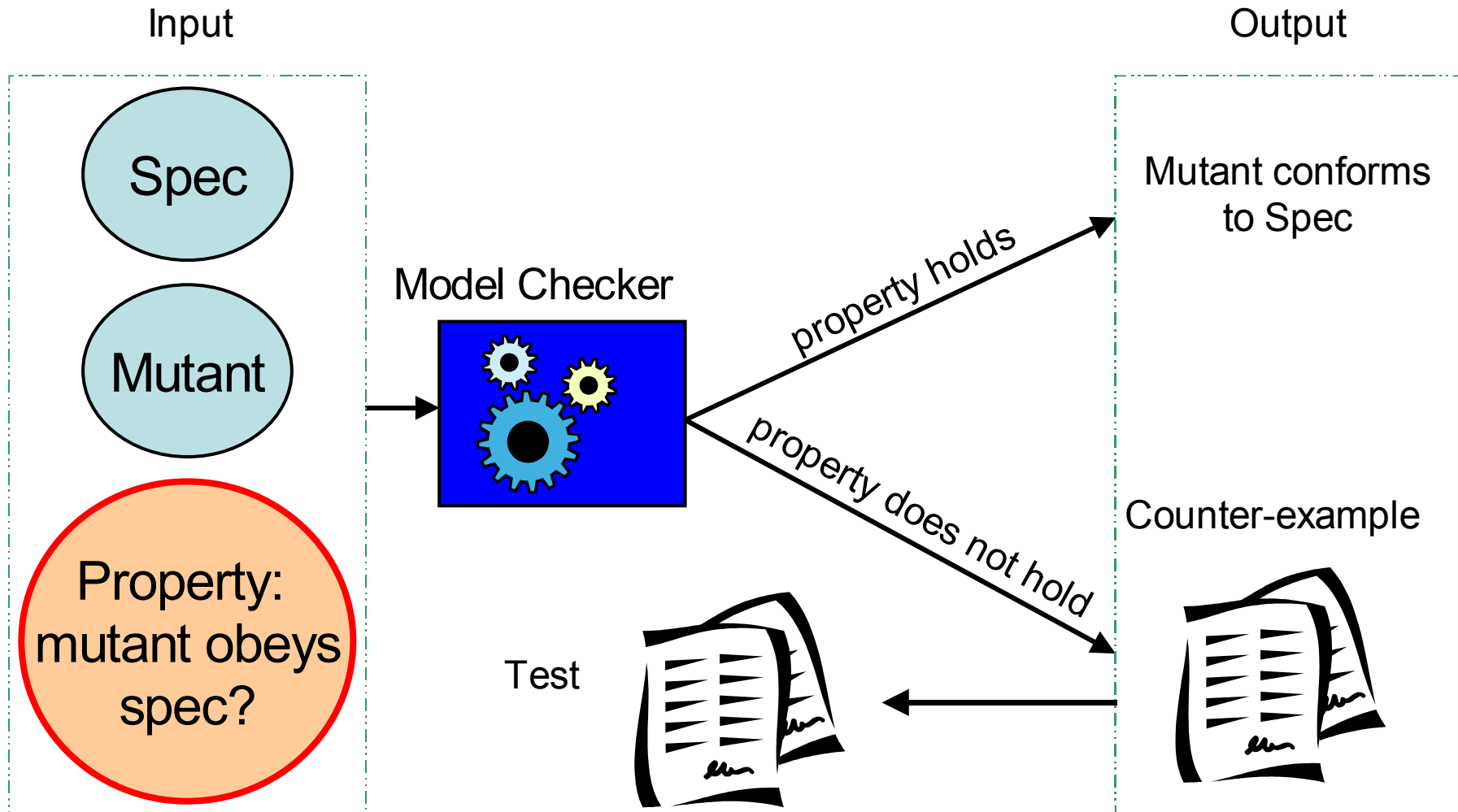
# Weak Tests and Fairness

- Fairness: if for each state occurring infinitely often in the path each outgoing transition is taken infinitely often
- Reset input is required to repeat a test
- Intuitively, a finite weak test, repeated infinitely often (with resets), is an infinite strong test under fairness assumption

# Is MBT Fair?

- Strong test for conservative abstract systems (models) is also strong for concrete systems
- Not so for weak tests, as fairness is not guaranteed (do not expect fairness from a conservative abstraction)

# Building Test by Model Checking



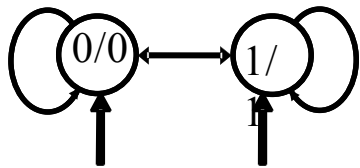
# Deterministic Spec and Mutant

Strong and weak tests coincide

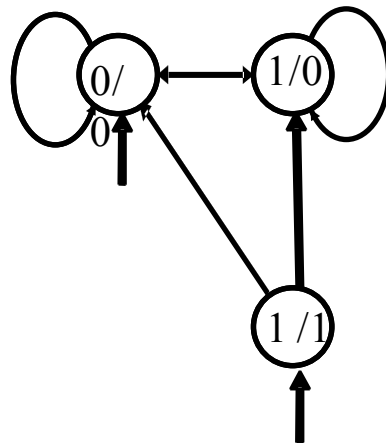
Test could be built from counterexample to

$$S \parallel M' \models AG \text{ out} = \text{out}'$$

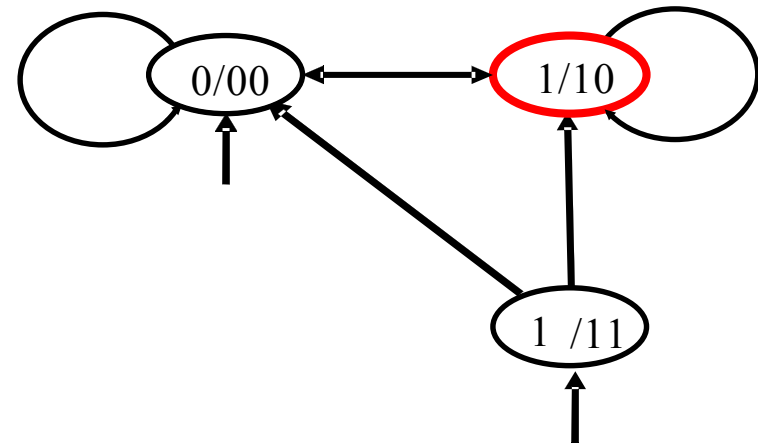
*Module S*



*Module M*



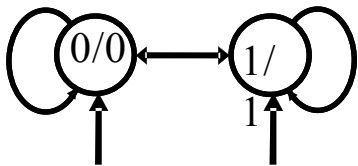
*Module S || M'*



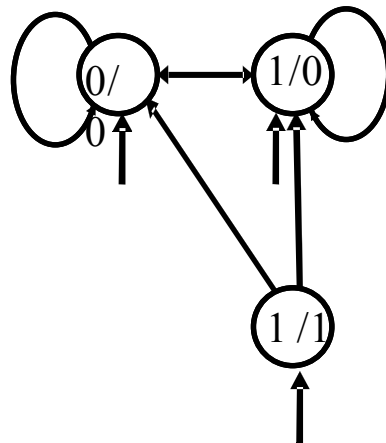
# Tests for Deterministic Spec and Non-Deterministic Mutant

Weak test could be built from counterexample to  
 $S \parallel M' \models AG \text{ out} = \text{out}'$

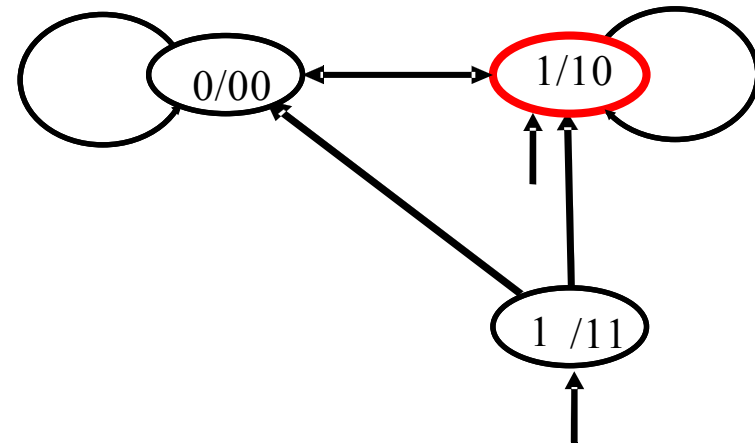
*Module S*



*Module M*



*Module S || M'*



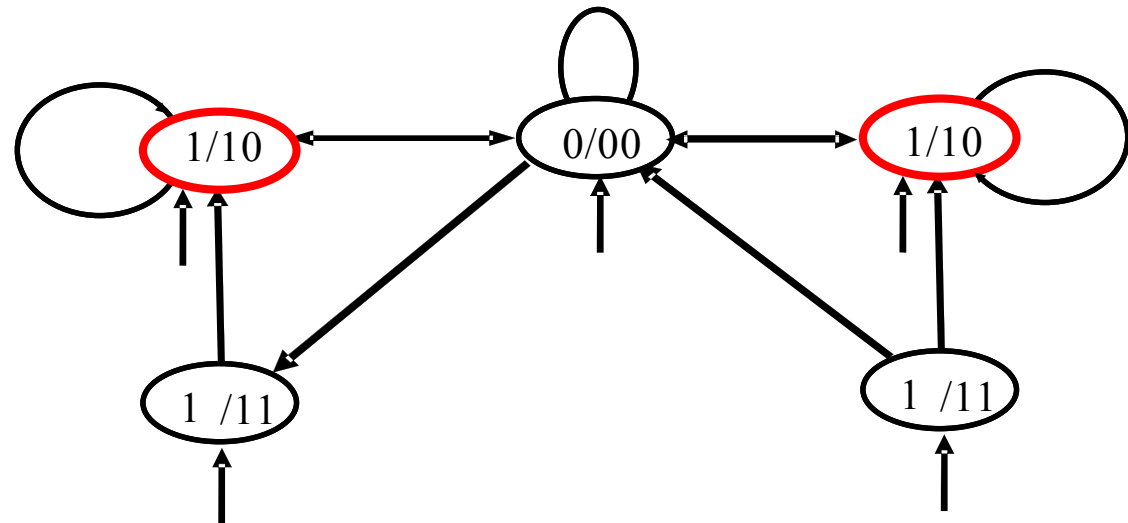
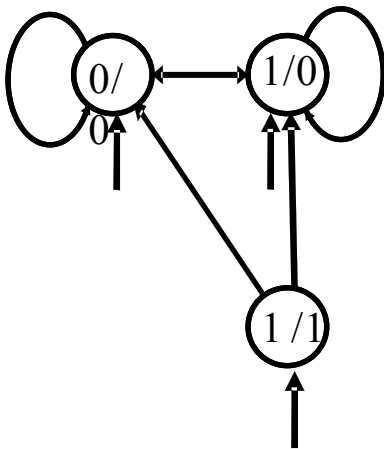
Weak tests are not necessarily strong

# Non-Deterministic Spec and Mutant

Test could not be built from counterexample to  
 $S \parallel M' \models AG \text{ out} = \text{out}'$

*Module S = Module M*

*Module S || M'*



Due to lack of output synchronization

# Weak Tests for Non-Deterministic Spec and Mutant

Build an observer from the spec by renaming outputs into inputs, determinizing, and completing with sink states

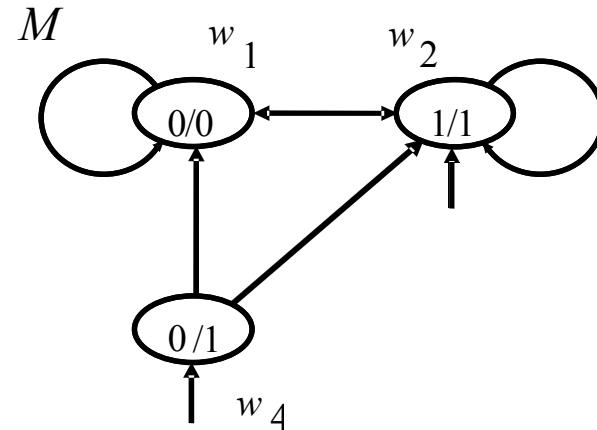
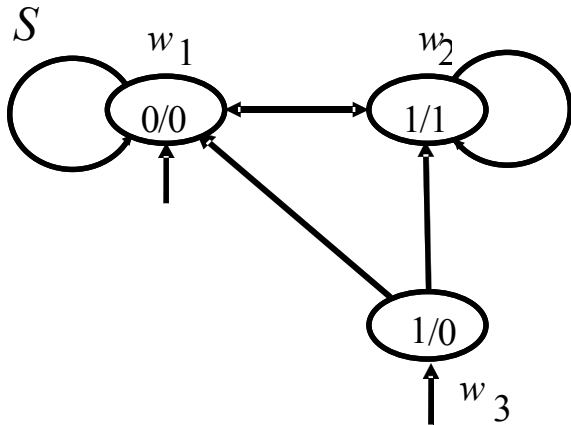
*Weak test could be built from counterexample to*

$M \parallel \text{Obs}(S) \models AG \text{ sink}$

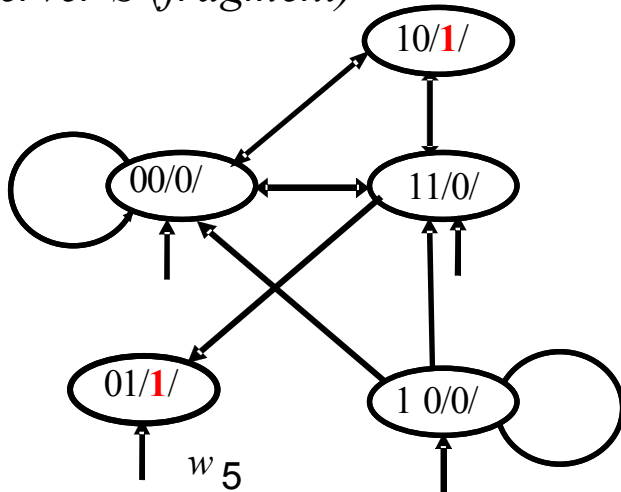
But not each weak test is strong

Apparently, model checkers are not fit to derive strong tests

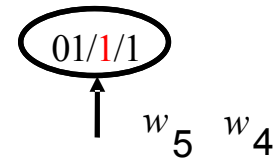
# Example



*Observer S (fragment)*



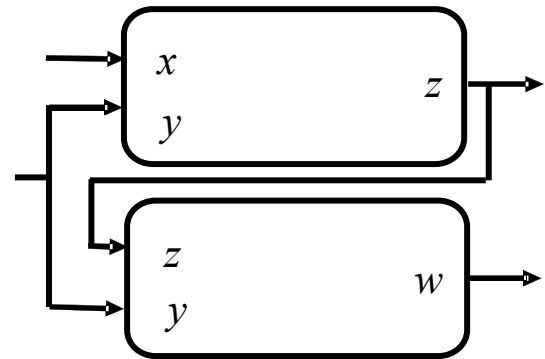
Counterexample to  $Obs(S) \parallel M \models AG \text{ sink}$   
 (fragment of  $Obs(S) \parallel M$ )



0 is a weak test, but not strong

# Module Checking

- Module is Kripke structure + partition of variables onto input, output, and internal
- Module composition  
(internal variables are hidden)



- Model checking problem: satisfaction of a formula in a module (underlying Kripke structure)
- Module checking problem: reactive satisfaction  
satisfaction of a formula in each deadlock free  
composition of the module with any other module (called environment)

# Strong Tests for Non-Deterministic Specification and Mutant

There is no strong test iff  
 $\text{HideOut}(S \parallel M')$  satisfies reactively

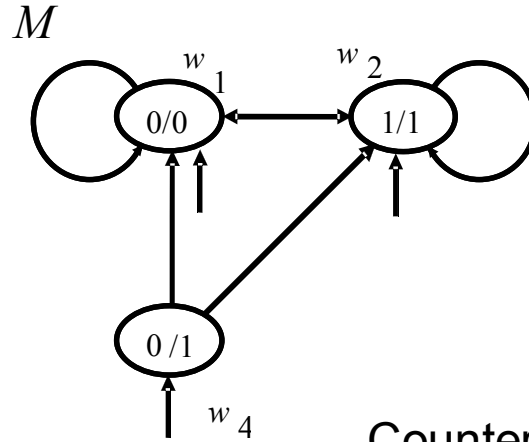
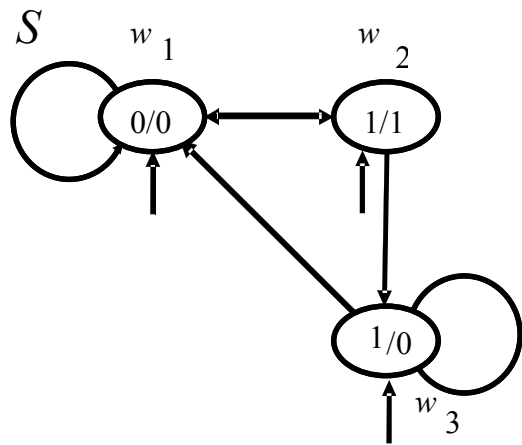
$$\text{EG out} = \text{out}'$$

i.e., for all non-blocking  $Env$

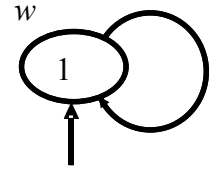
$$Env \parallel \text{HideOut}(S \parallel M') \models \text{EG out} = \text{out}'$$

HideOut operation converts all the output variables into internal

# Example

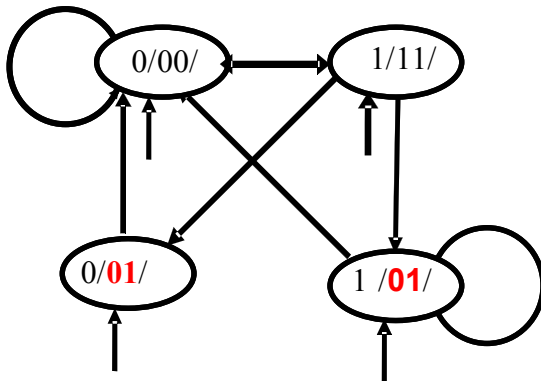


Counterexample Environment  $Env$

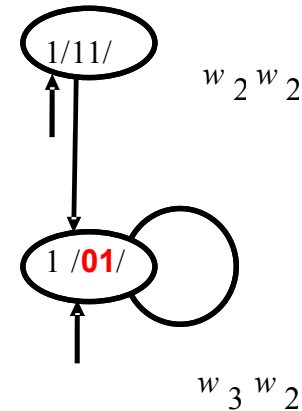


Counterexample to EG  $out = out'$   
(fragment of  $Env \parallel HideOut(S \parallel M)$ )

HideOut ( $S \parallel M$ )



11 is a strong test



# Conclusion

- “Can a Model Checker Generate Tests for Non-Deterministic Systems?”
- Yes, for weak tests
- But with certain transformations that may explode size
- Yes, with a module checker
- Do you know one?

Thank you