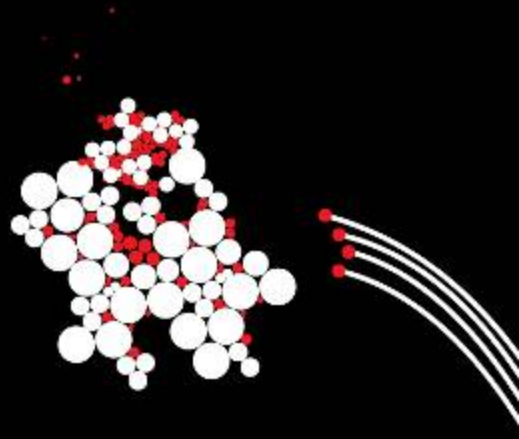
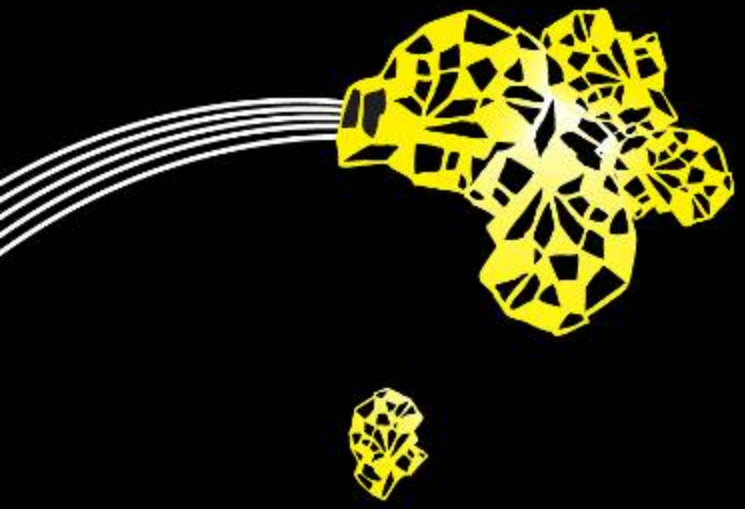
UNIVERSITEIT TWENTE.



Testing Techniques

Lecture: Testing with Finite State Machines



Marielle Stoelinga

Formal Methods & Tools

Today's agenda

1. **What is model-based testing?**
   - and what is it good for?
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Applications**

Practicals: fMBT

QUiZZES

Multiple choice questions

Open questions

How not to test? Ad hoc testing

spec

execute test suite

inputs/stimuli

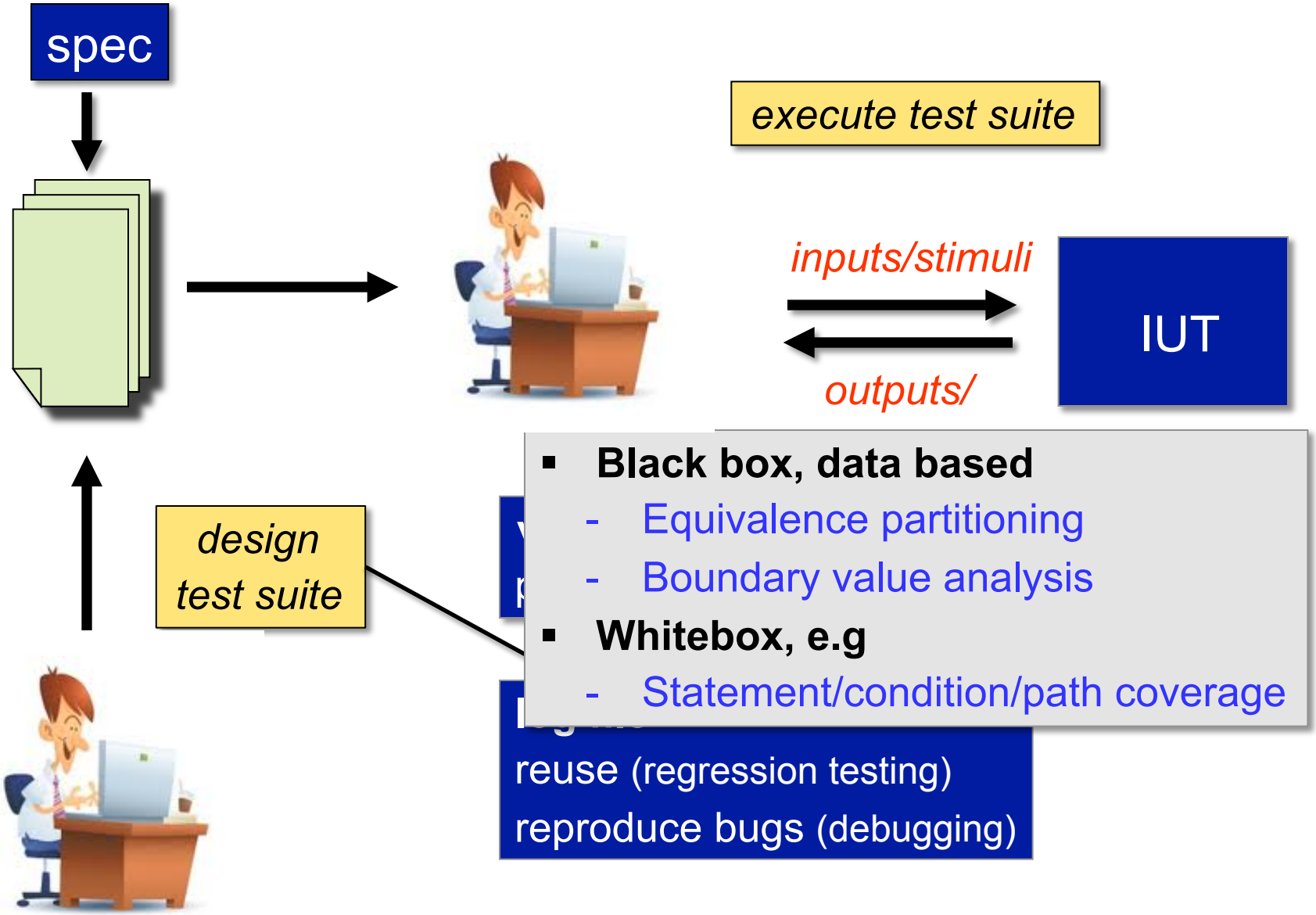
outputs/ responses/ observations

IUT

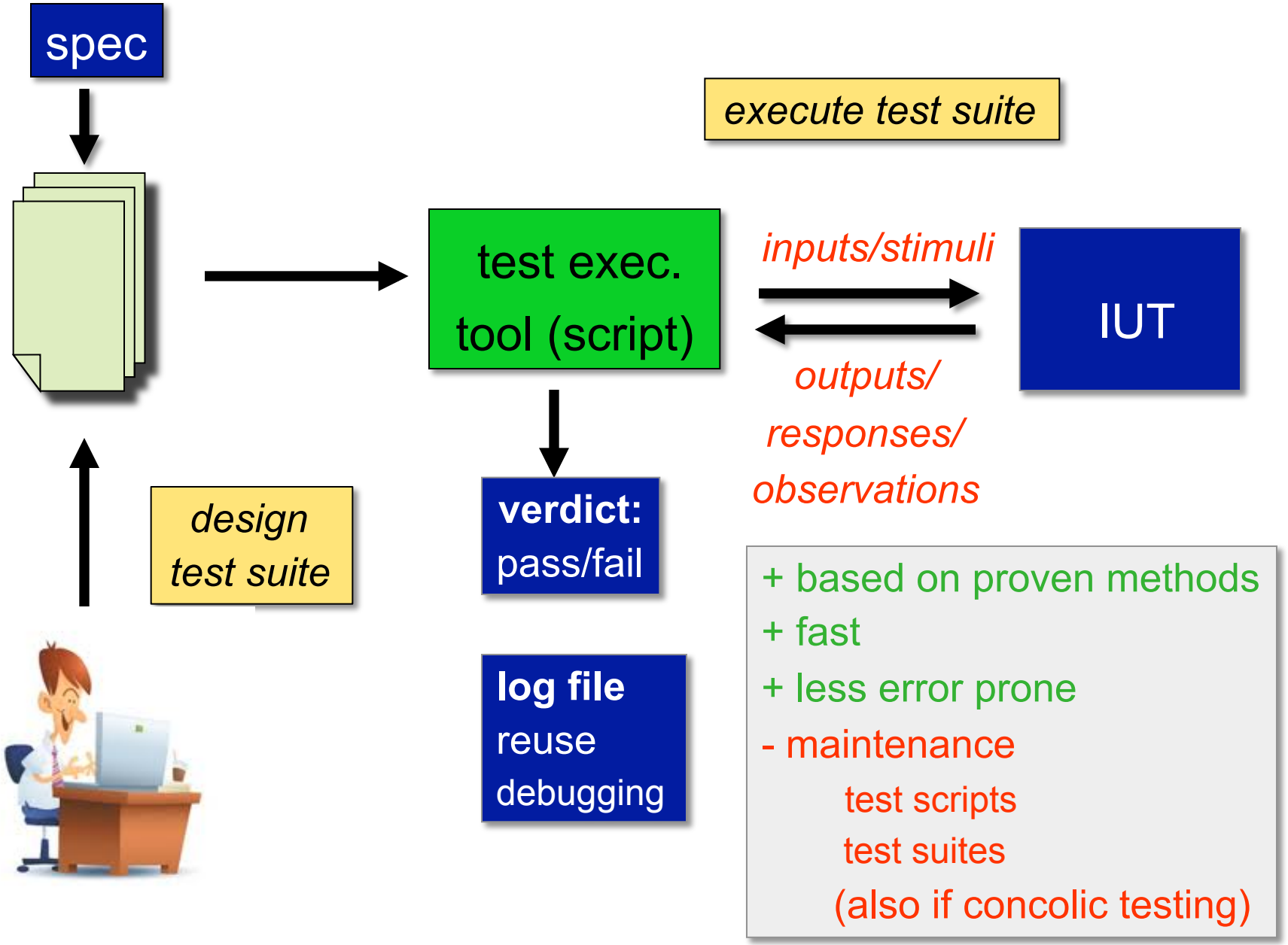
verdict: pass/fail

\+ flexibility
\- ad hoc
\- error prone
\- boring
\- no planning

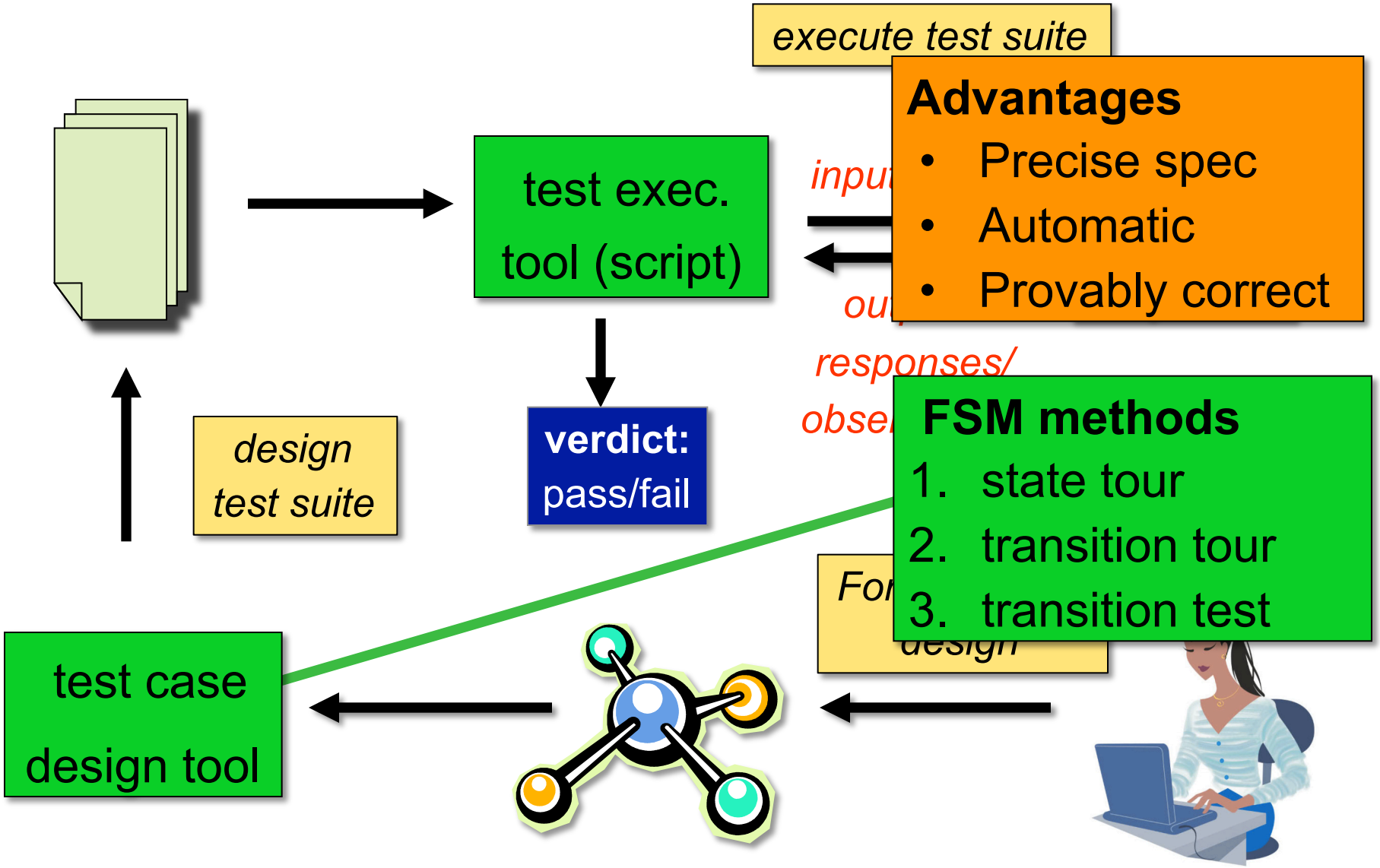
How to test? Structured Testing



How to test? Automatic structured Testing



How to test? Model-based testing (MBT)



Model-based testing

In the wild:

- Exist since ‘90s
  - Roots at UT and other places
- Gained indutsrial popularity recently
- Sogeti, Microsoft, Axini, Conformiq,…
  - UML
  - MathLab tool boxes
  - …

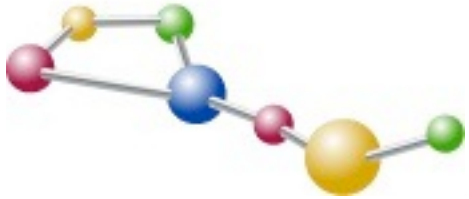
In this course

- Focus on algorithmic basics
- FSMs

MBT Tools

CONFORMIQ
Automated Test Design™

Spec Explorer



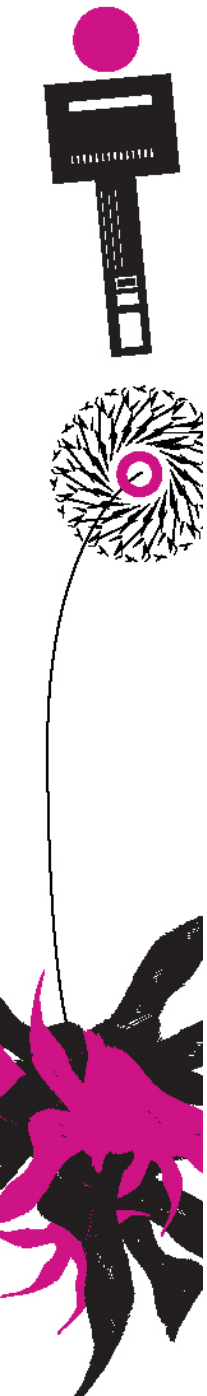
JTorx

MaTeLo

TestOptimal

Differ in:
- Input languages:
- FSMs | MathLab/Simulink | UML …
- Test methods
- Application area

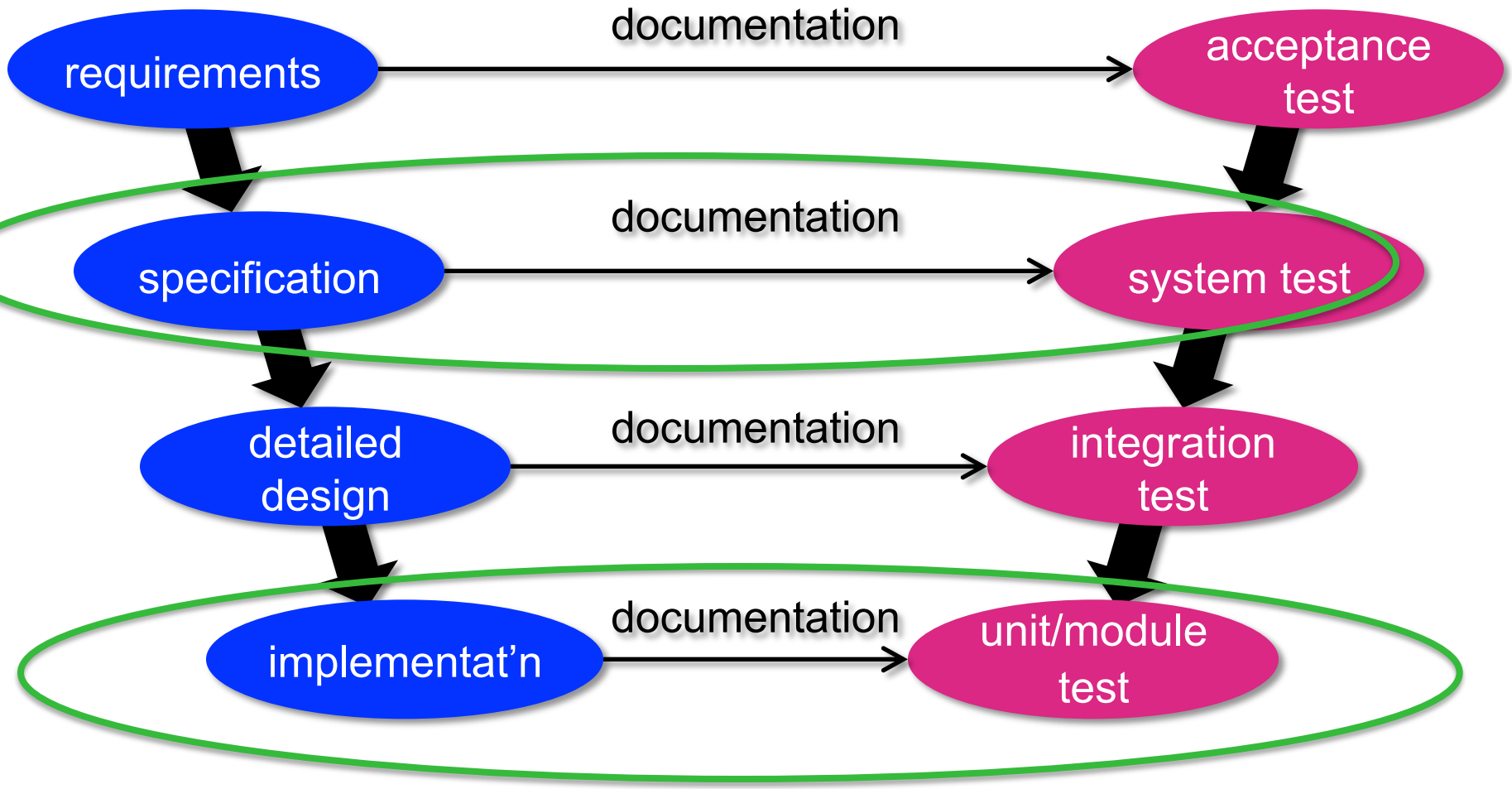
And many others
- Many companies have in-house tools
- http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html

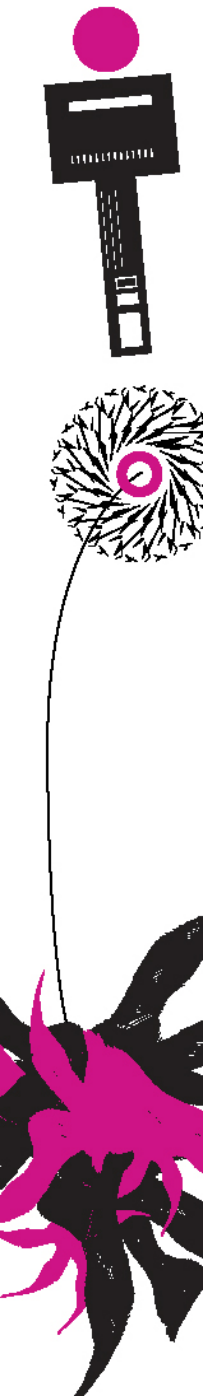


Today's agenda

1. **What is model-based testing?**
   - and what is it good for?
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Variations and Applications**

When to test? The V-model



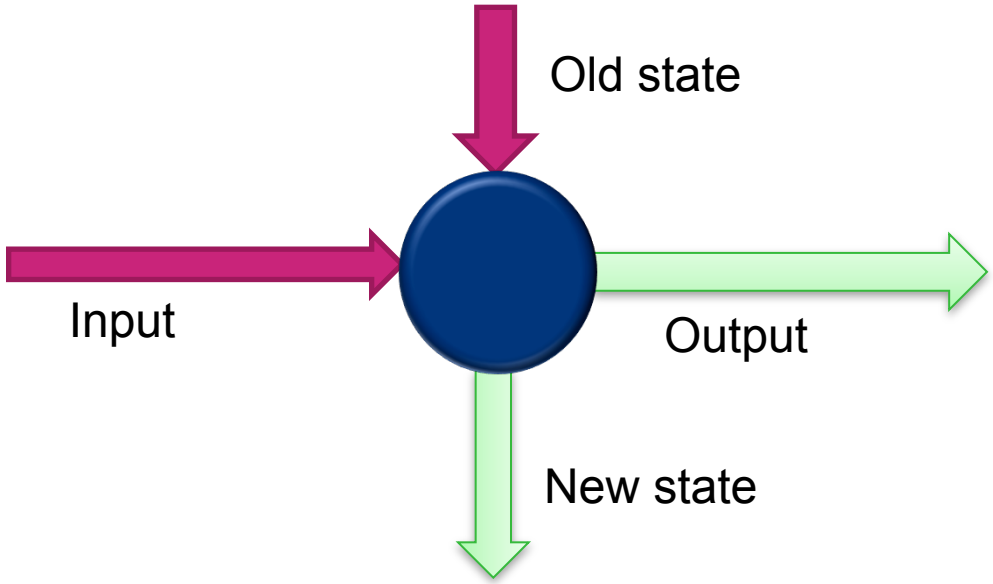


Today's agenda

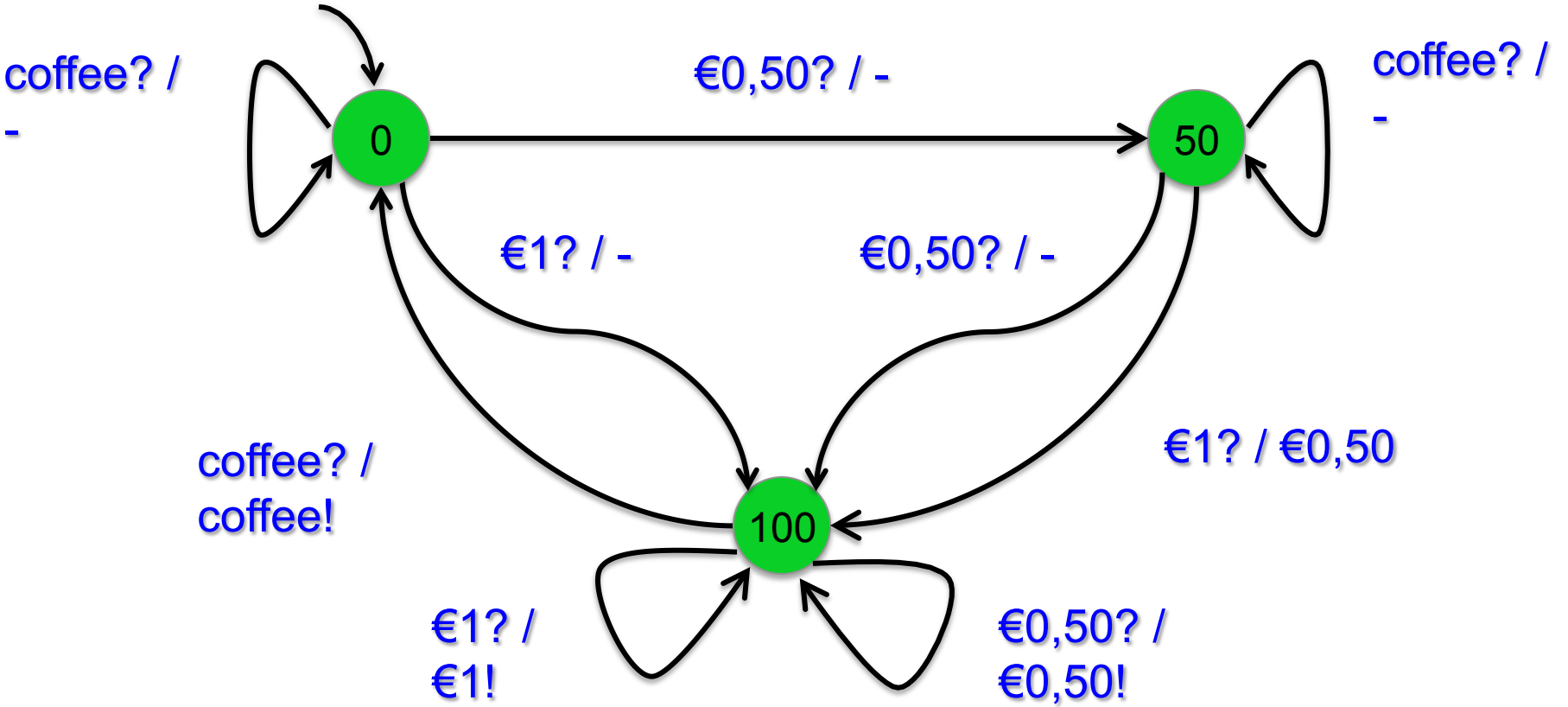
1. **FSM model**
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Variations and Applications**

Finite State Machines (FSMs)

- States
- Transitions
- Inputs (“triggers”)
- Outputs



Example: Coffee Machine



- Coffee costs €1
- Machine accepts €1, €0,50
- Machine returns money

Conventions

- Plain name: state
- Name?: input
- Name!: output

FSMs formally

FSM (a.k.a. Mealy Machine) is 5-tuple M = (S, I, O, δ, λ)

- S — finite set of states
- I — finite set of inputs
- O — finite set of outputs
- **δ: S x I → S** — transfer function
- **λ: S x I → O** — output function

- Usually initial state
- Extension to sequences
  - **δ: S x I* → S**
  - **λ: S x I* → O***

FSMs formally

By definition, FSM restrictions

- *Deterministic*

  δ: S x I → S and λ: S x I → O are functions

- *Completely specified*

  δ: S x I → S and λ: S x I → O are complete functions

  (empty output is allowed; sometimes **implicit completeness**)

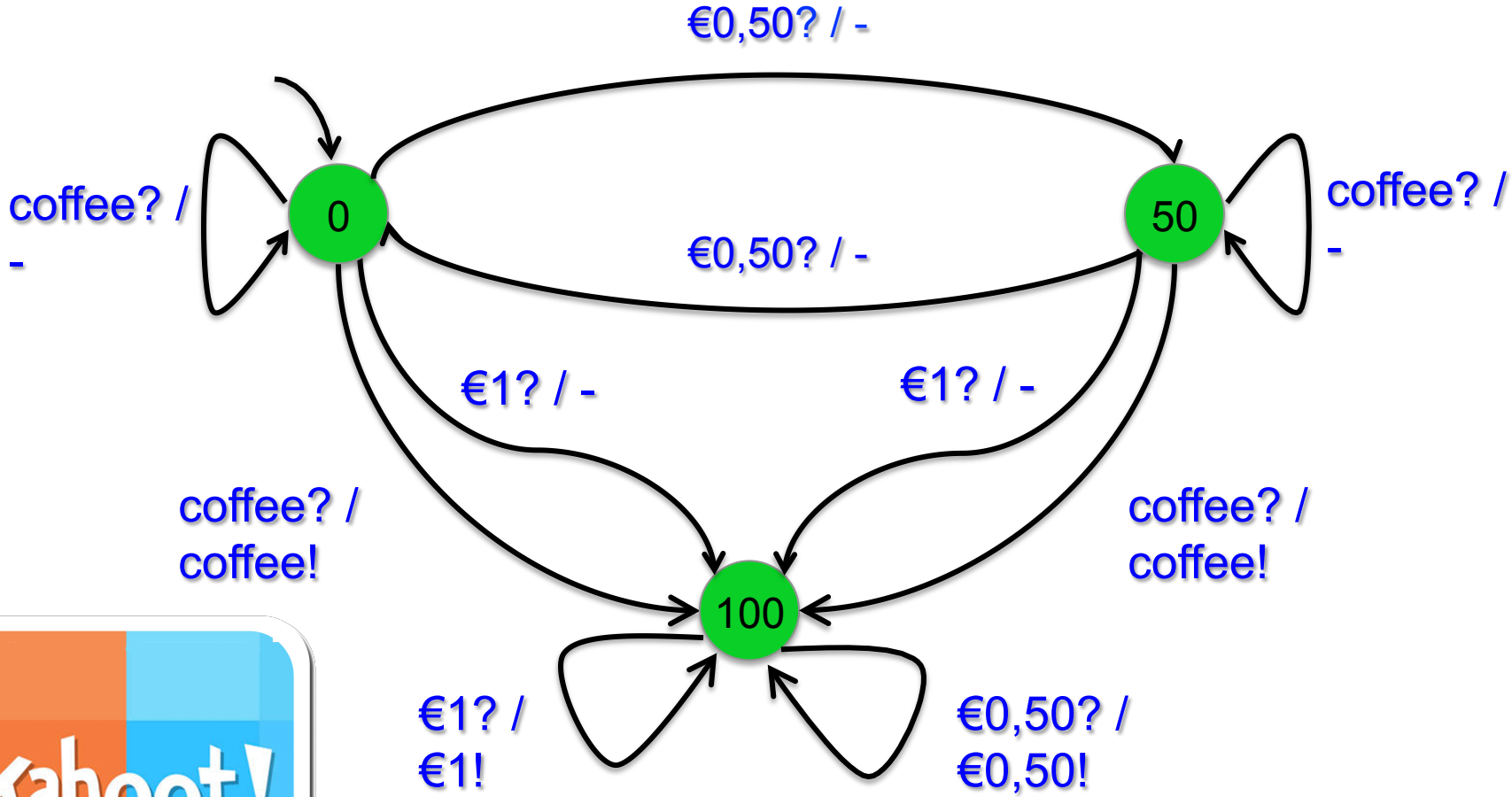
In addition, we require

- *Strongly connected:* from any state, any other state can be reached;
- *Reduced:* there are no equivalent states

FSMs formally

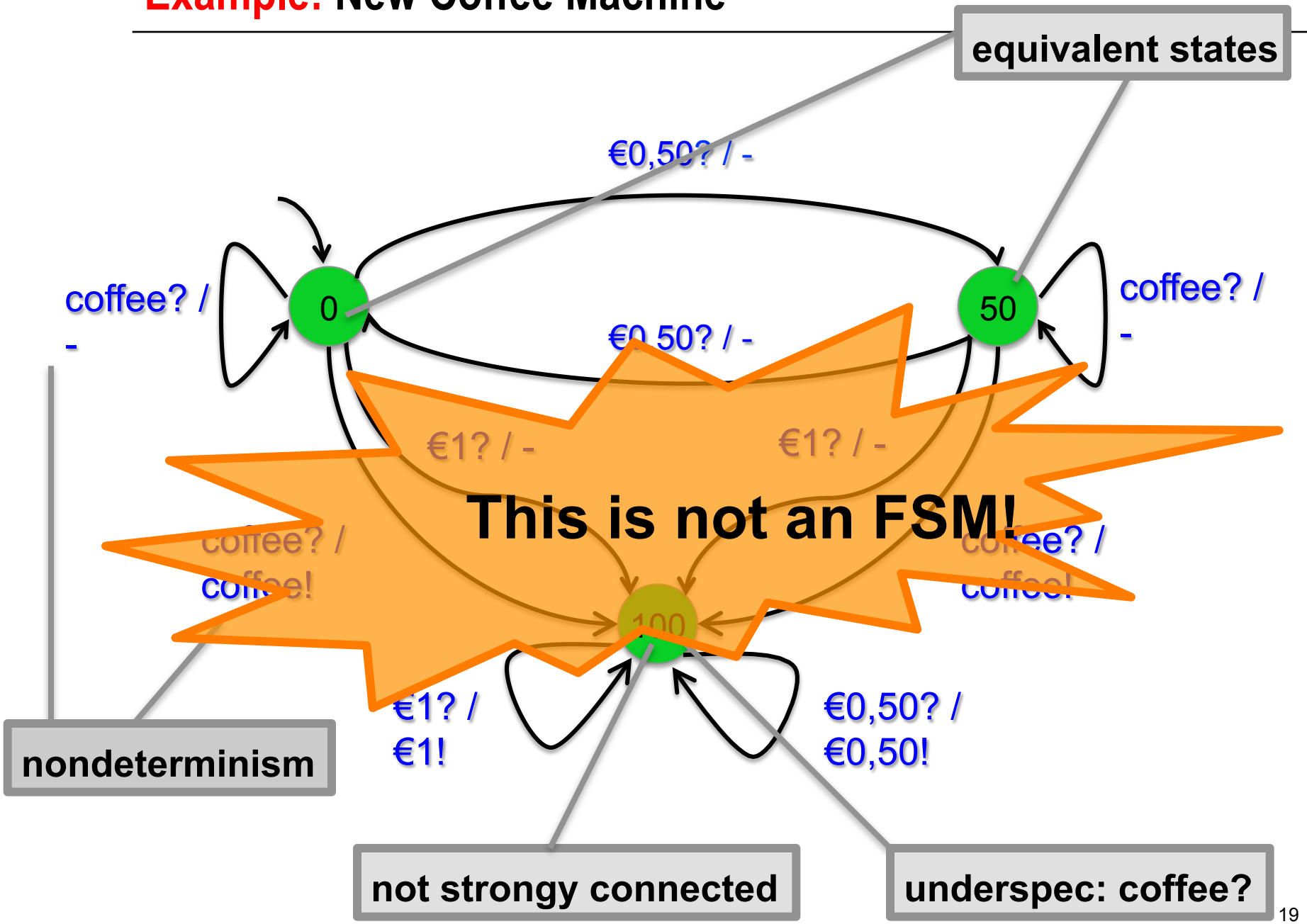
- **State equivalence**
  - States s, t are equivalent iff
    for all sequences x: $\lambda(s,x) = \lambda(t,x)$
- **FSM equivalence**
  - FSMs A, B are equivalent iff:
    - For all states s in A there is an equivalent state t in B **and**
    - For all states t in B there is an equivalent state s in A
- **There are many equivalent FSMs**
  - Reduced FSMs are unique (up to state naming)
- In this course: all FSMs are reduced and strongly connected

Example: New Coffee Machine



Go to **kahoot.it** via mobile phone or web browser

Example: New Coffee Machine



What is a test case in an FSM?

Test cases for FSMs

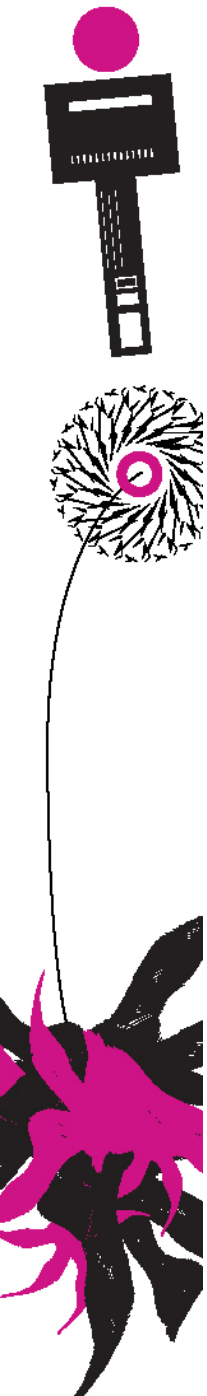
- a1?/b1! a2?/b2! a3?/b3! … an?/bn! **pass**
- **Fail** if outcome differs from above

Test case = path in FSM

- Inf. many paths → Inf. many test cases
- → test selection strategies needed

3 methods

1. State tour
   - visit every state in specification
2. Transition tour
   - visit every transition in specification
3. Transition testing
   - Make one test case for each transition separately



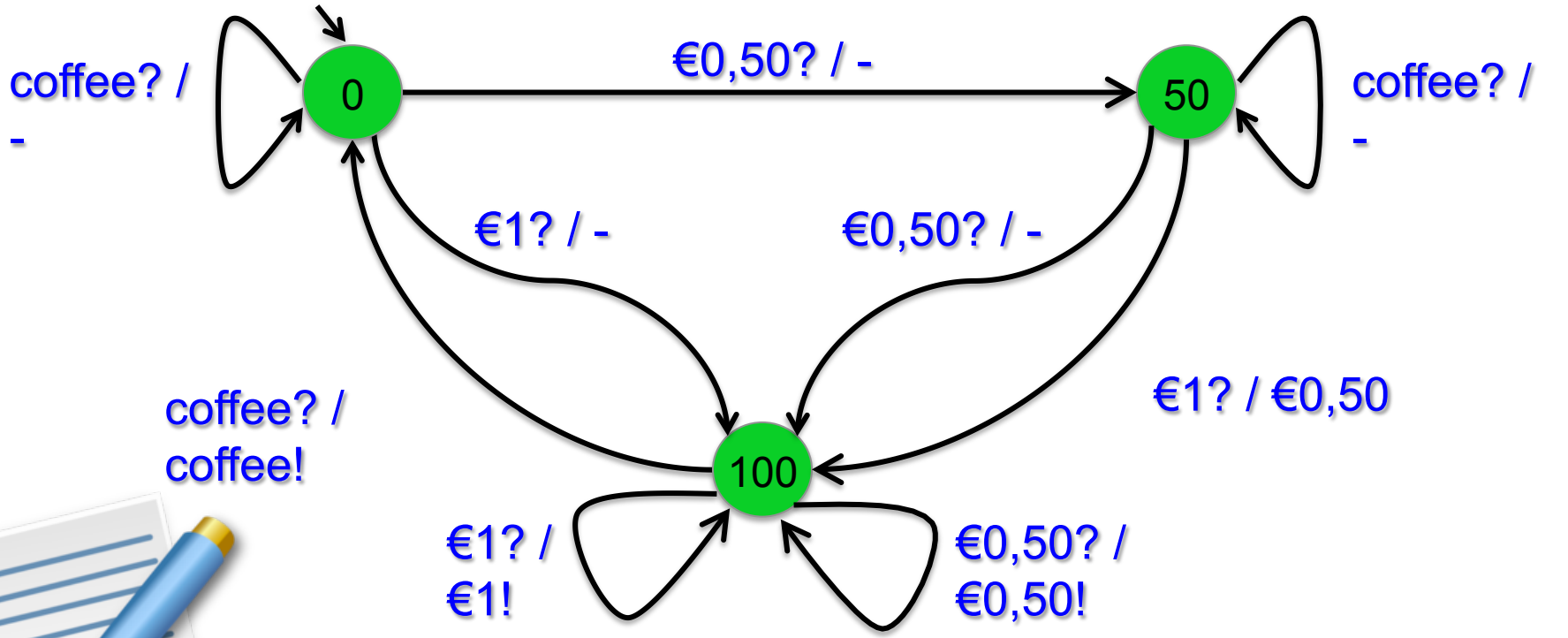
Today's agenda

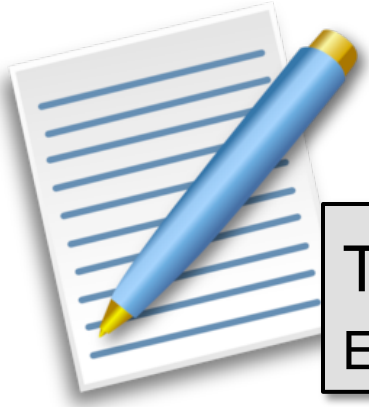
1. **FSM model**
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Variations and Applications**

State Tour

Procedure

- Make tour that covers every state at least once
- NOTE: start state = final state





Test sequence: €0,50? 1€? coffee?

Expected output: - €0,50! coffee!

Verdicts

Pass if as expected

Fail otherwise

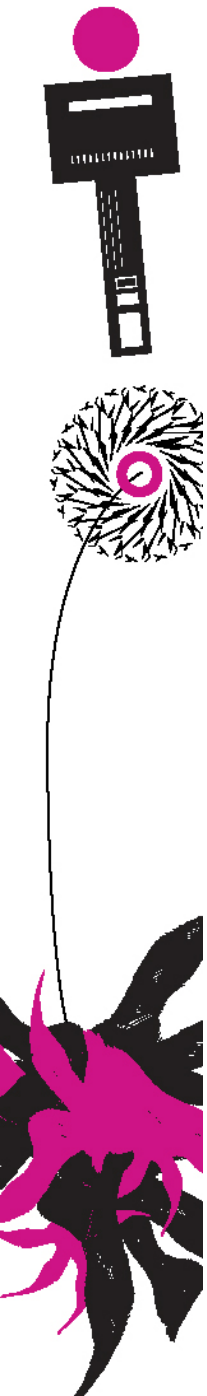
State Tour: efficiency

- Make State Tour that covers every state as cheaply (minimal # edges) as possible!

State Tour: efficiency

- Make State Tour that covers every state as cheaply (minimal # edges) as possible!

- This is NP-Complete:
  - Traveling Salesman Problem
  - Use approximation algorithms, not exact ones
  - Balance test generation effort and test execution effort



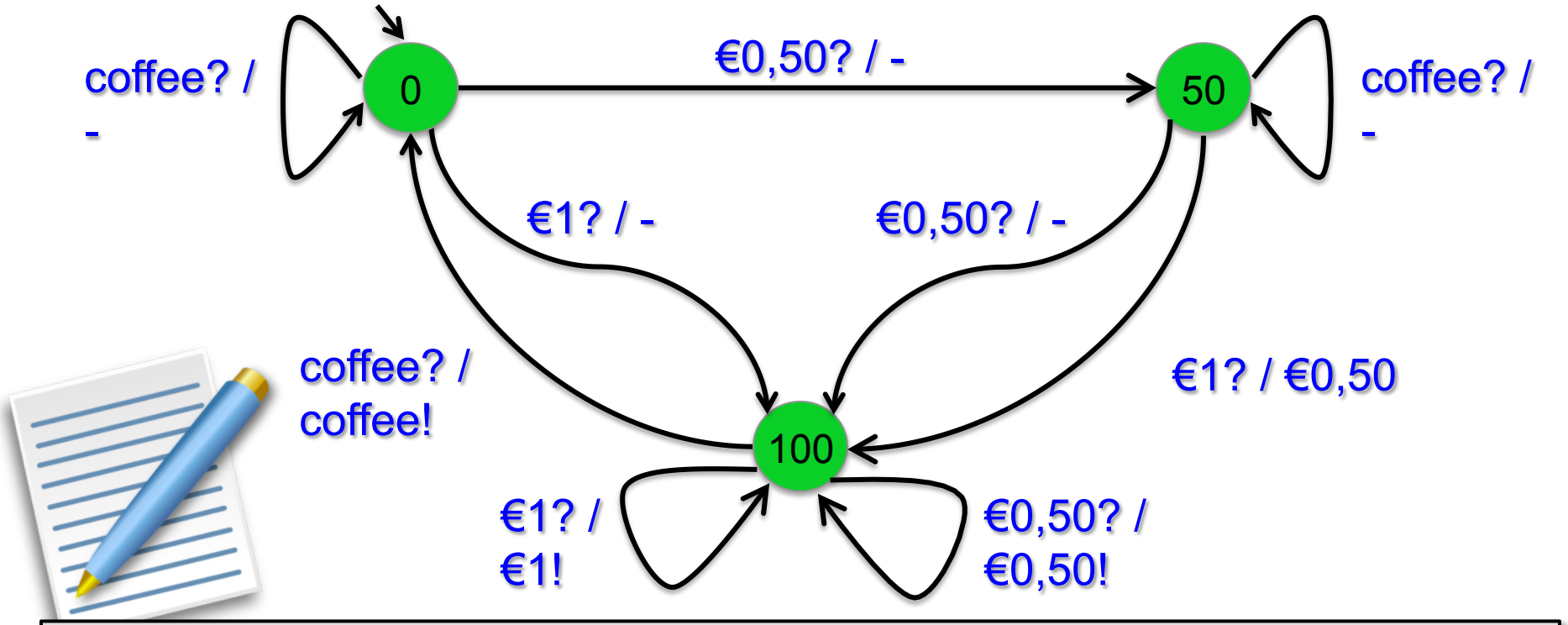
Today's agenda

1. **FSM model**
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Variations and Applications**

Transition Tour

Procedure

- Make tour that covers every edge at least once
- NOTE again: start state = final state



Test sequence

coffee?	€0,50?	coffee?	€0,50?	€0,50?	1€?	coffee?	1€?	coffee?	€0,50?	coffee?	1€?
-	-	-	-	€0,50!	1€!	coffee!	-	coffee!	-	-	€0,50!

Transition Tour: efficiency

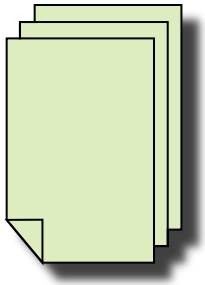
- Make transition tour that covers every edge as cheaply (minimal # edges) as possible!

Transition Tour: efficiency

- Make transition tour that covers every edge as cheaply (minimal # edges) as possible!
- This is polynomial!
  - This is polynomial: Chinese Postman Tour
    - Deliver mail to all streets
    - Minimize the walking distance
  - For Euler graphs: Euler tour
  - For other graphs: solve LP problem

State/ transition tours: Correctness (aka soundness, aka tricky stuff)

€0,50?/- €1?/€0,50! coffee?/coffee!



execute test suite

test exec. tool (script)

inputs/stimuli

outputs

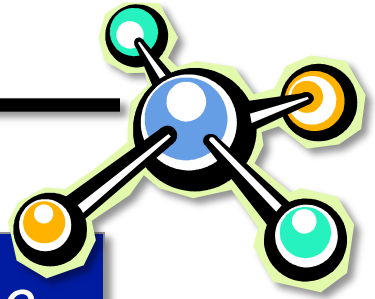
IUT

Impl

design test suite

verdict:
pass:
output as specified
fail:
otherwise

test case design tool

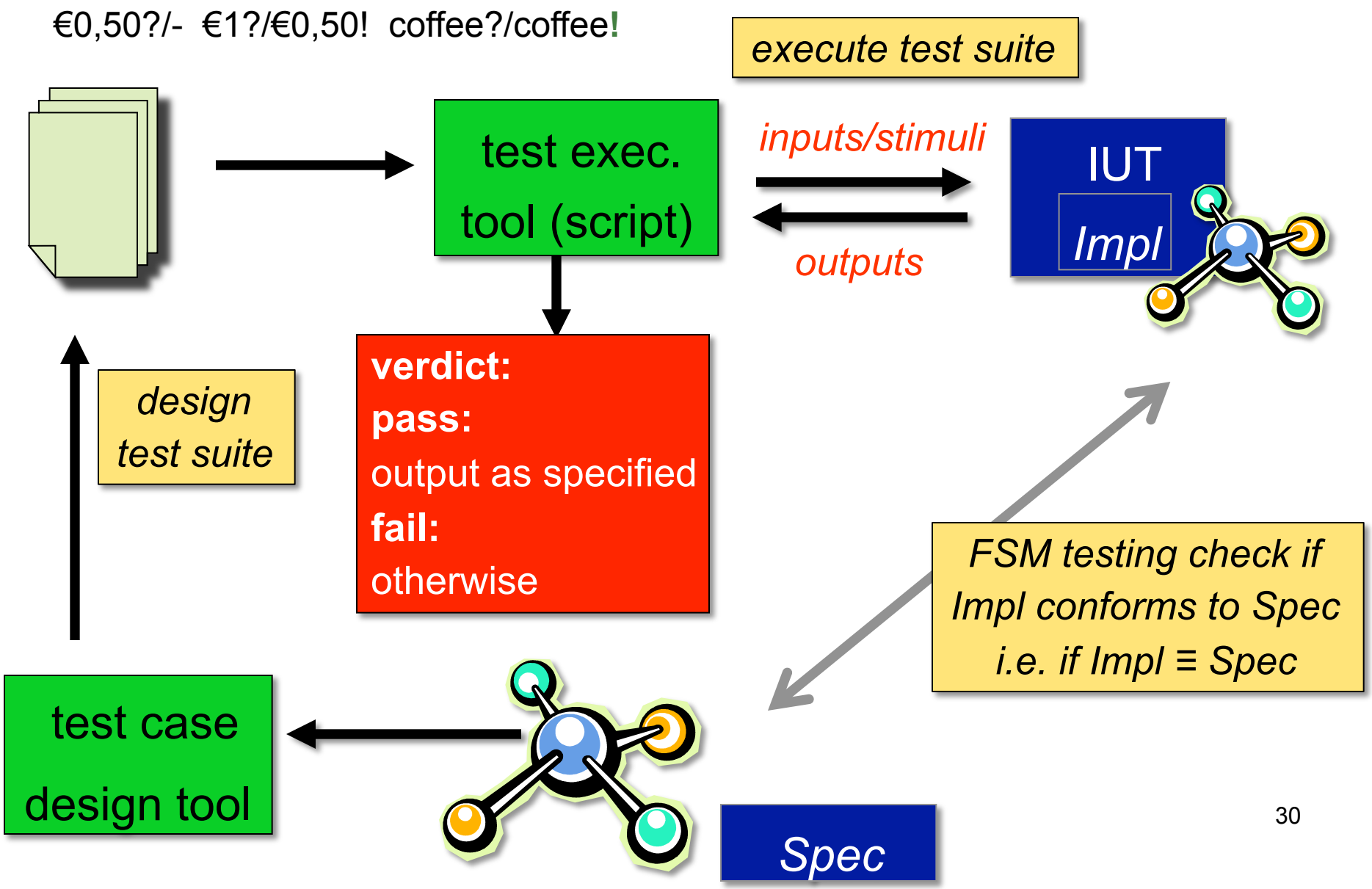


Spec

Why is this correct?

- If test case fails, IUT does not conform to Spec
- IUT conforms to Spec??
- Assumptions:
  - IUT has FSM model *Impl*
  - Conformance: FSM equivalence
  - Fail: Impl ≠ Spec
- What if verdict is pass?
  - No fault uncovered
  - But IUT can be nonconforming

State and transition tours: Correctness (aka soundness)



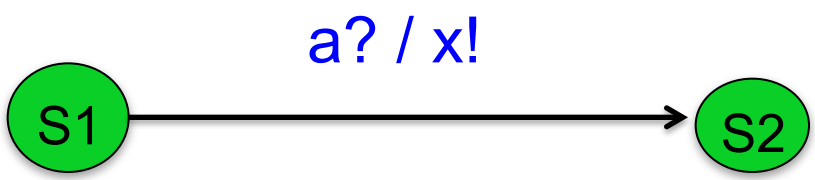
Today's agenda

1. **What is model-based testing?**
   - and what is it good for?
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Applications**

Practicals: fMBT

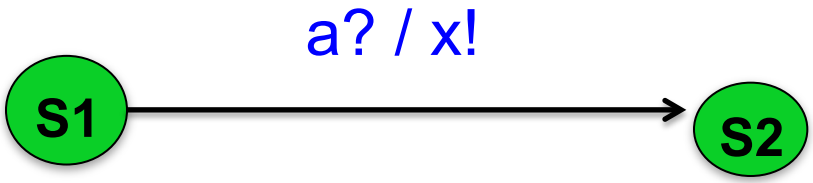
Transition Testing

- One big tour as test case not always desirable
  - Too long, too complex, difficult to analyze, not specific
- *Hence:* Make test case for each transition separately
  - Test if the system in state S1, produces output x! on input a? and goes to state S2
- *Problem:* state in implementation is unknown
  - *Solution:* make sure you move to S1 before testing transition



Problem: state In impl is unknown

Transition Testing: Make test case for every transition separately

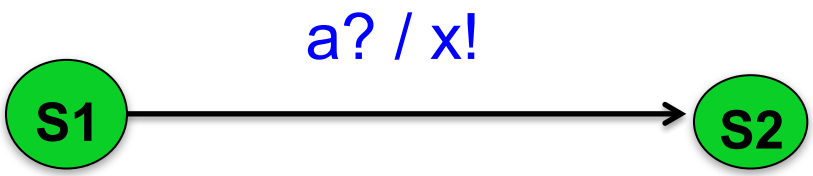


Problem: state In impl is unknown

- **Procedure**
  - step 1: Go to state **S1** (synchronizing + transfer seq)
  - step 2: Apply input a?
  - step 3: Check output x!
  - step 4: Verify state **S2** (UIO/DS/W-set)

- **Test purpose:** "Test whether the system, when in state S1, produces output x! on input a? and goes to state S2"

Transition Testing



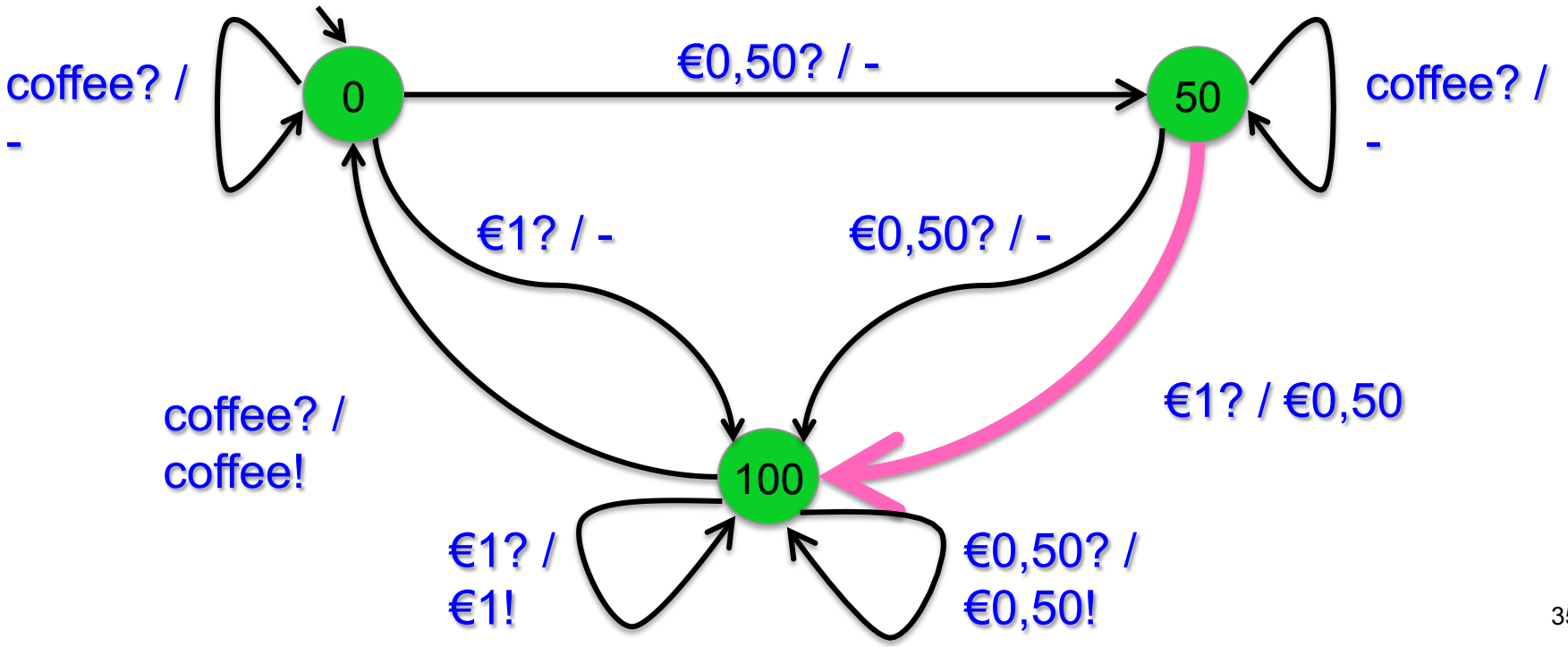
Step 1: Go to state S1

- step 1a:
  - synchronizing sequence brings machine to particular state, say S0, from any state
  - but synchronizing sequence may not exist
  - or: use reset transition if available
- step 1b
  - go from S0 to S1
  - always possible because of determinism and completeness

Example: Transition Testing

Test pink transition

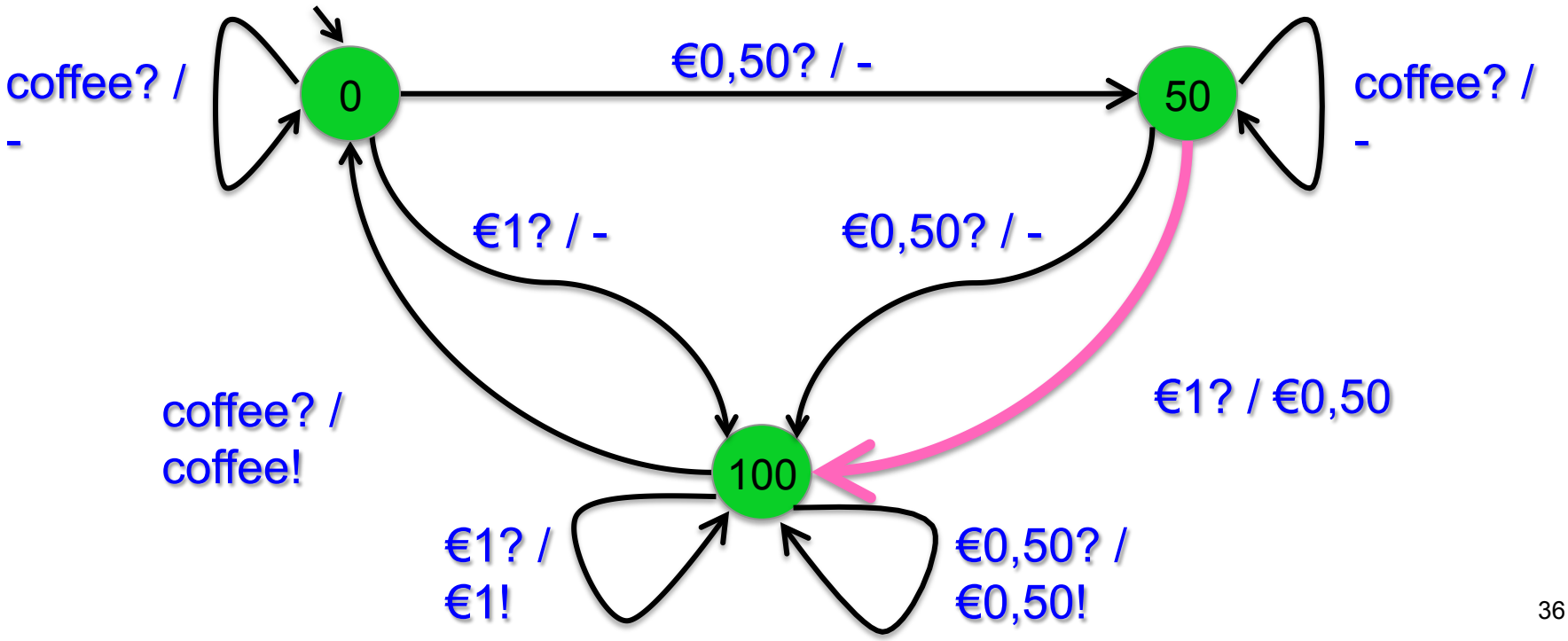
- implemented correctly?



Example: Transition Testing

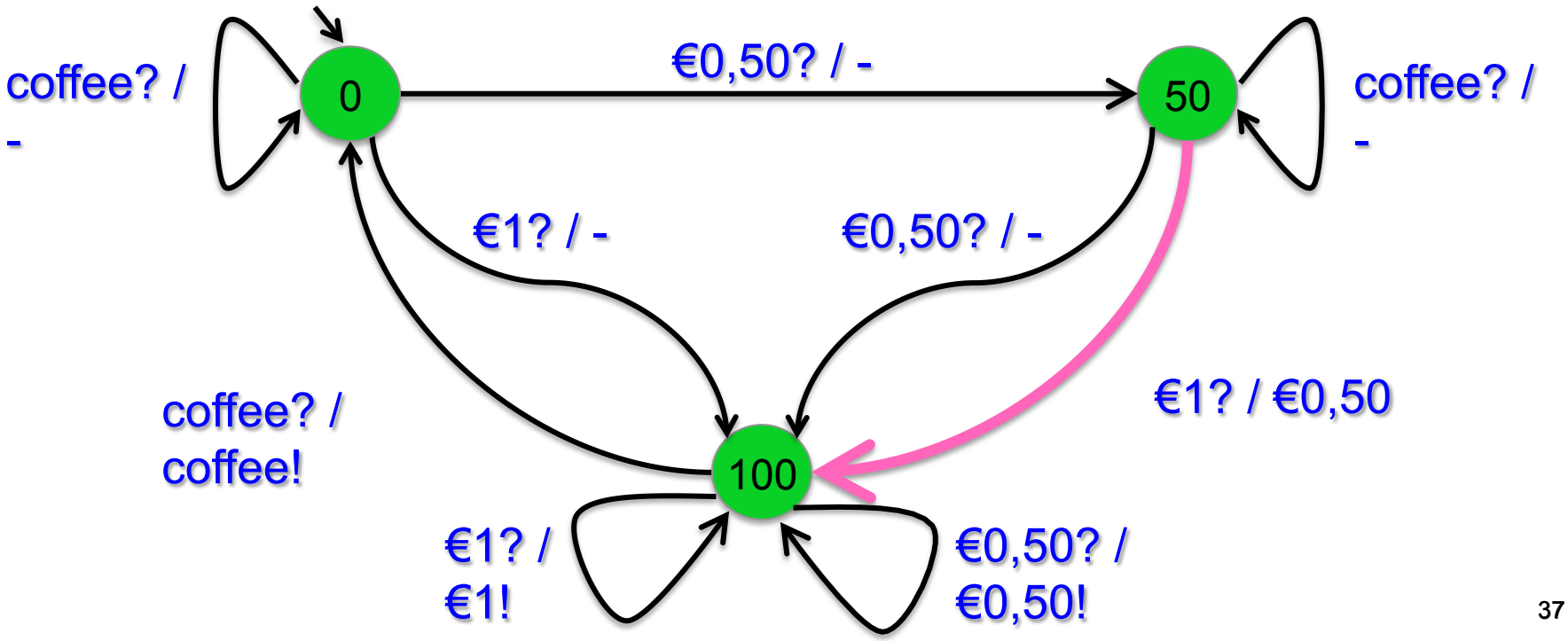
Step 1

- **1a:** find a synchronizing sequence to state 0
- €1? coffee?
- **1b:** find transfer sequence to go from 0 to state 50
- €0,50?



Example: Transition Testing

- **step 2:** apply input 1€?
- **step 3:** check output 0,50
- **step 4:** verify that machine is in state 100 indeed



FSM Transition Testing

Step 4: State identification and verification

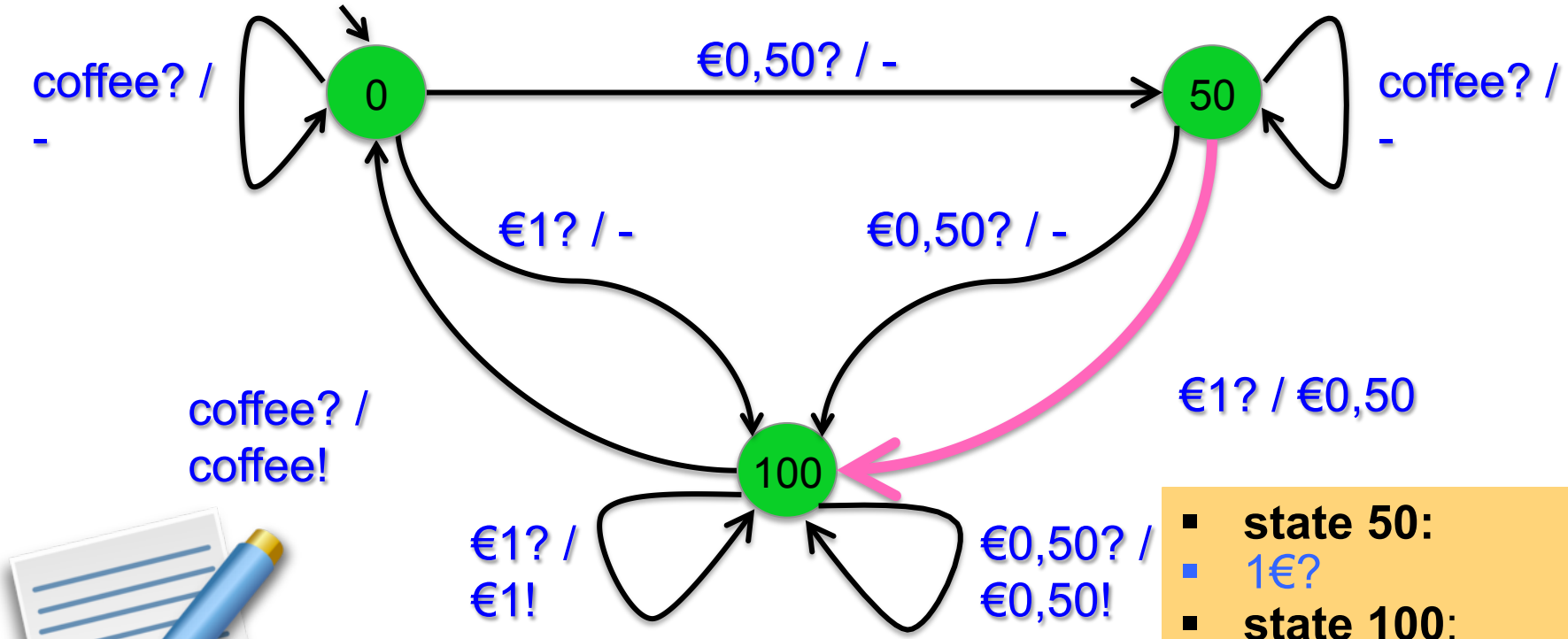
- Did we end up in right state?
- Apply sequence of inputs in current state
  s.t. from the outputs we can
  - identify the state where we started; or
  - verify that we were in a particular start state
- Different kinds of sequences
  - UIO sequences (Unique Input Output sequence, SIOS)
  - Distinguishing sequence (DS)
  - W - set (characterizing set of sequences)
  - UIOv
  - SUIO Single UIO
  - MUIO Multiple UIO
  - Overlapping UIO

State verification

1. **UIO sequence (for state s)**
   1. sequence x that distinguishes state s from all other states: for all $t \neq s$: $\lambda(s, x) \neq \lambda(t, x)$
   2. each state has its own UIO sequence
   3. UIO sequences need not exist
2. **Distinguishing sequence (for entire FSM)**
   1. sequence x that produces different output for each state: for all pairs t, s with $t \neq s$: $\lambda(s, x) \neq \lambda(t, x)$
   2. a distinguishing sequence need not exist
3. **W - set of sequences (for entire FMS)**
   1. set of sequences W; can distinguish any pair of states: for all pairs $t \neq s$ there is $x \in W$: $\lambda(s, x) \neq \lambda(t, x)$
   2. W - set always exists for reduced FSM

Transition Testing

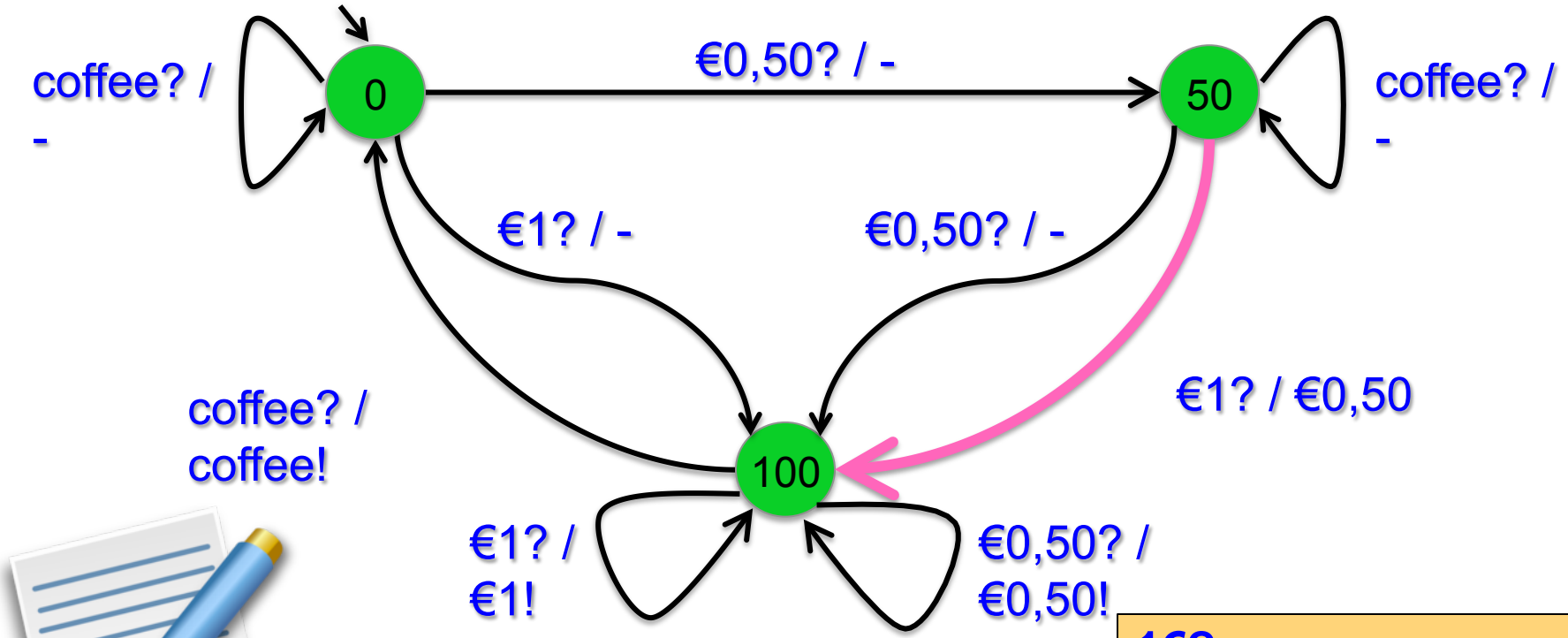
UIO sequence in state s:
sequence x that distinguishes state s from all other states:
for all $t \neq s$: $\lambda(s, x) \neq \lambda(t, x)$



- **state 50:**
- 1€?
- **state 100**:
- coffee?
- **state 0**:
- 50¢? coffee?

Transition Testing

Distinguishing sequence
produces different output for each state
for all pairs t, s with t ≠ s: $\lambda(s, x) \neq \lambda(t, x)$

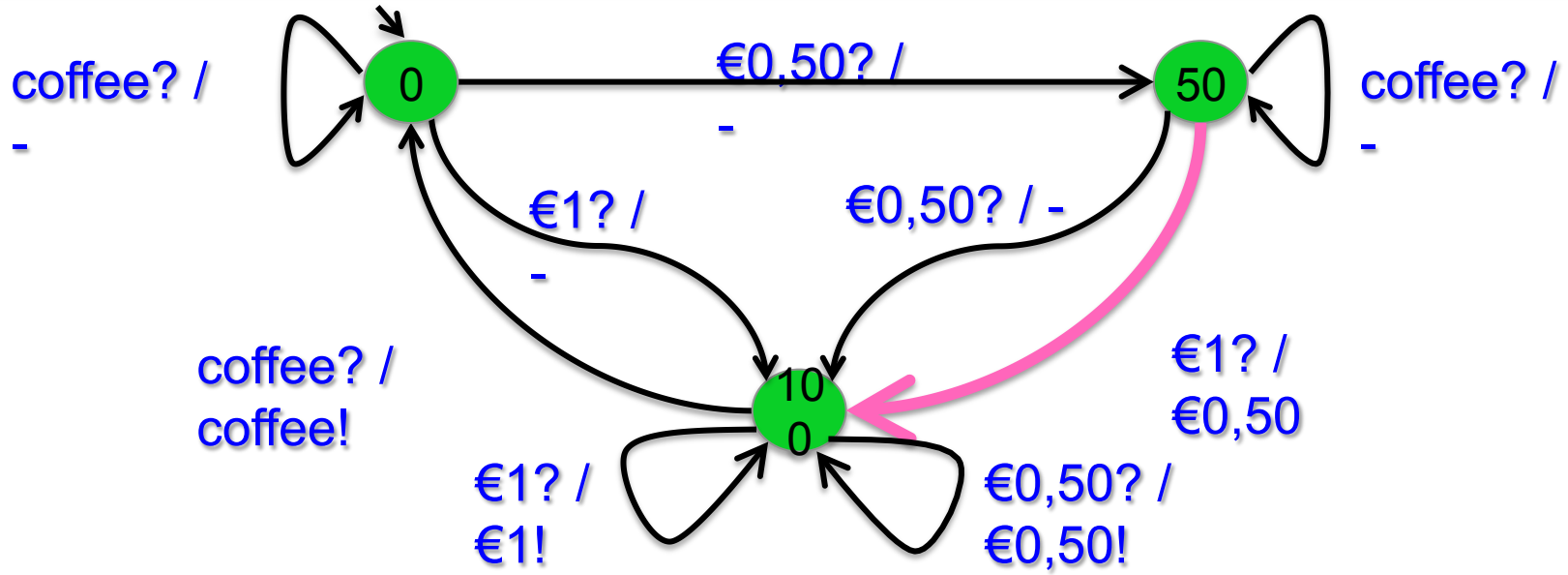


1€?
- output state 0: -
- output state 50: 50¢!
- output state 100: €1!

Transition Testing: all steps together

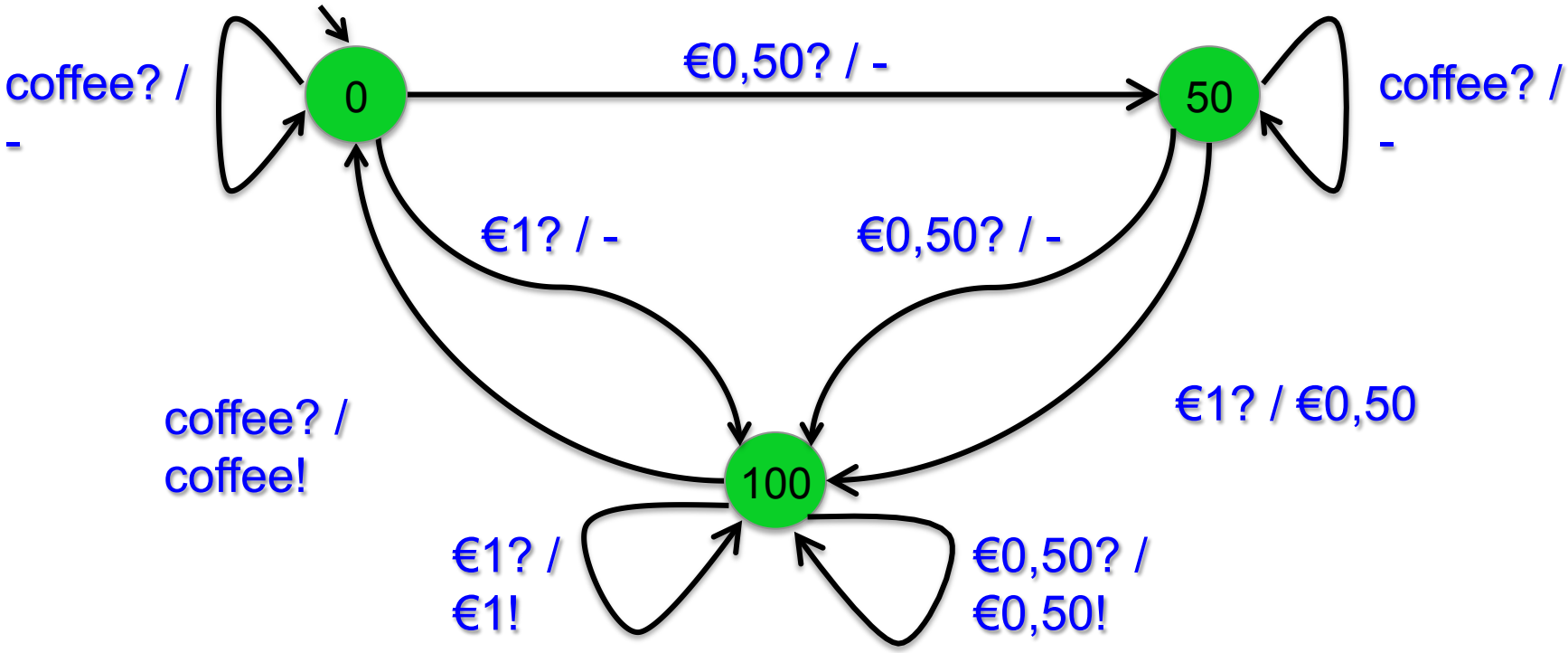
To test 1€? / €0,50!:

1. Go to state **50**: **1€? coffee? €0,50**
2. Give input **1€?**
3. Check output **€0,50!**
4. Apply UIO of state **100**: **coffee? / coffee!**



Test case: €1? / * coffee? / * €0,50? / - €1? / €0,50! coffee? / coffee!

Transition Testing: testing all transitions



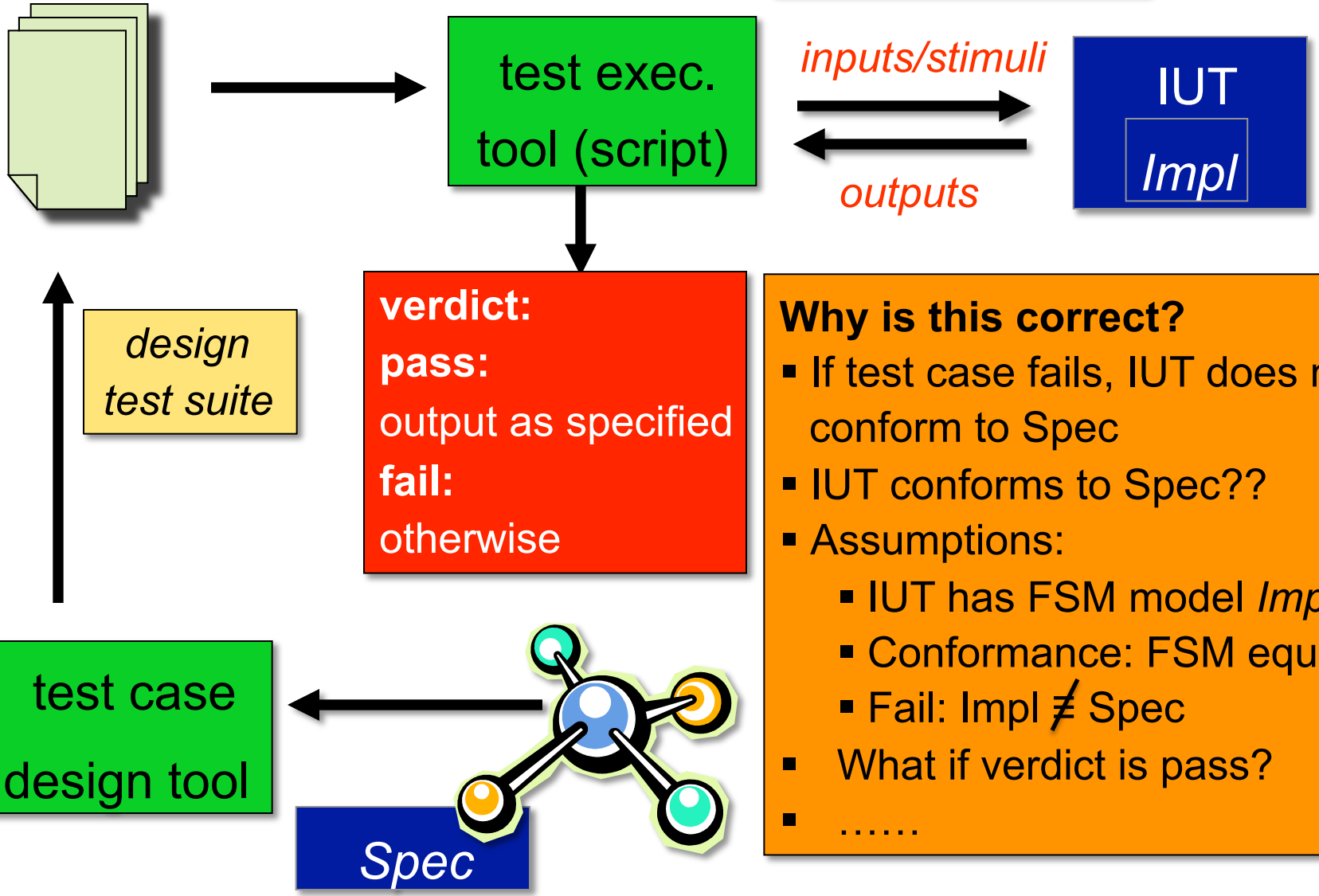
- 9 transitions / test cases for coffee machine
- if end-state of one corresponds with start-state of next then concatenate
- different ways to optimize and remove overlapping / redundant parts
- there are tools to support this

Today's agenda

1. **FSM model**
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Variations and Applications**

Transition testing: soundness is as before

€0,50?/- €1?/€0,50! coffee?/coffee!

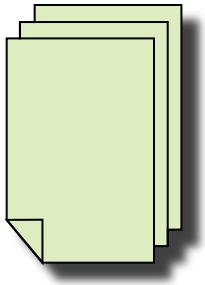


Why is this correct?

- If test case fails, IUT does not conform to Spec
- IUT conforms to Spec??
- Assumptions:
  - IUT has FSM model *Impl*
  - Conformance: FSM equivalence
  - Fail: Impl ≠ Spec
- What if verdict is pass?
- ……

Transition testing: Completeness (a.k.a exhaustiveness)

€0,50?/- €1?/€0,50! coffee?/coffee!



execute test suite

test exec. tool (script)

inputs/stimuli

outputs

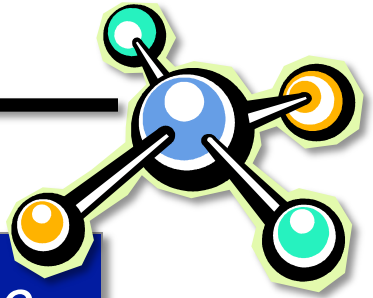
IUT

Impl

verdict:
pass:
output as specified
fail:
otherwise

design test suite

test case design tool



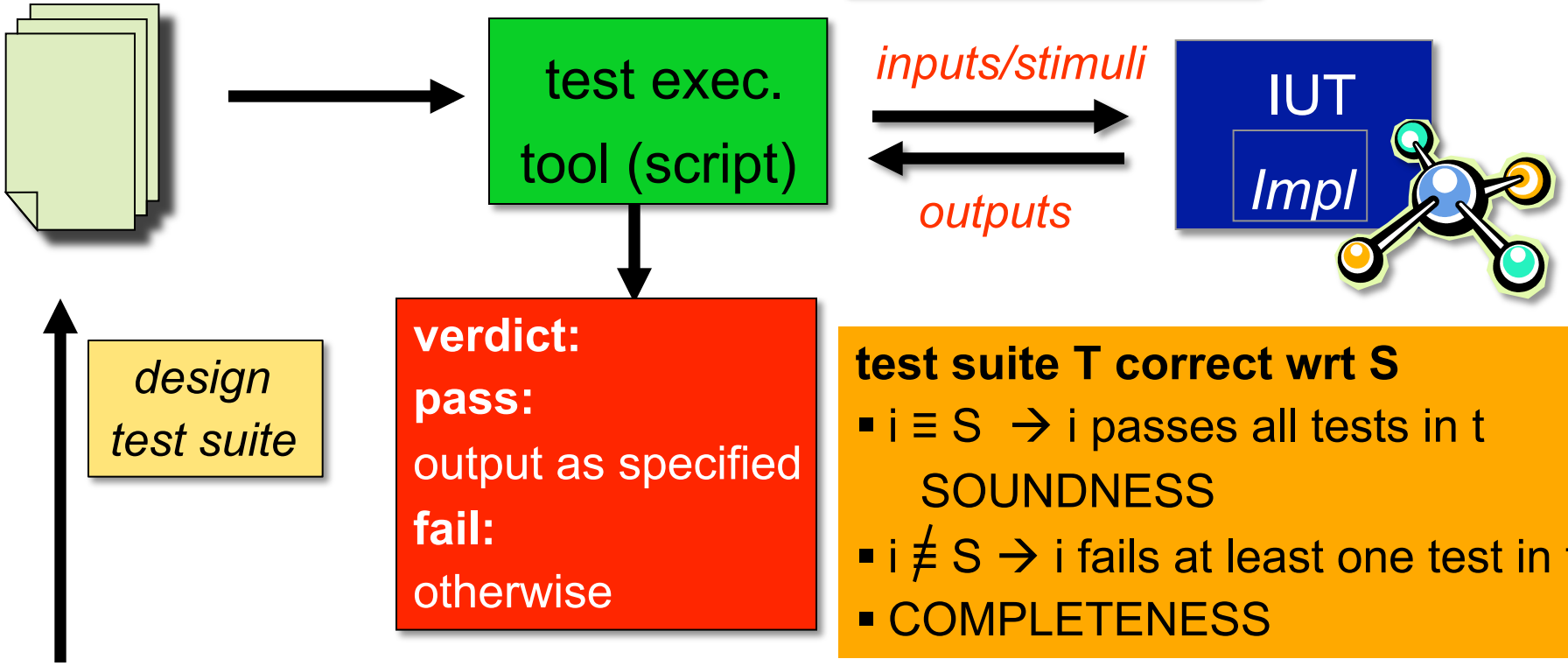
Spec

Why is this correct?

- What if verdict is pass?
- Then IUT is correct !
  - i.e. Impl ≡ Spec
- Provided that
  - Test case for each transition
  - #states Impl ≤ #states Spec
- Due to special FSM restriction
  - Determinism, strongly connectedness, …

Transition testing: soundness & completeness

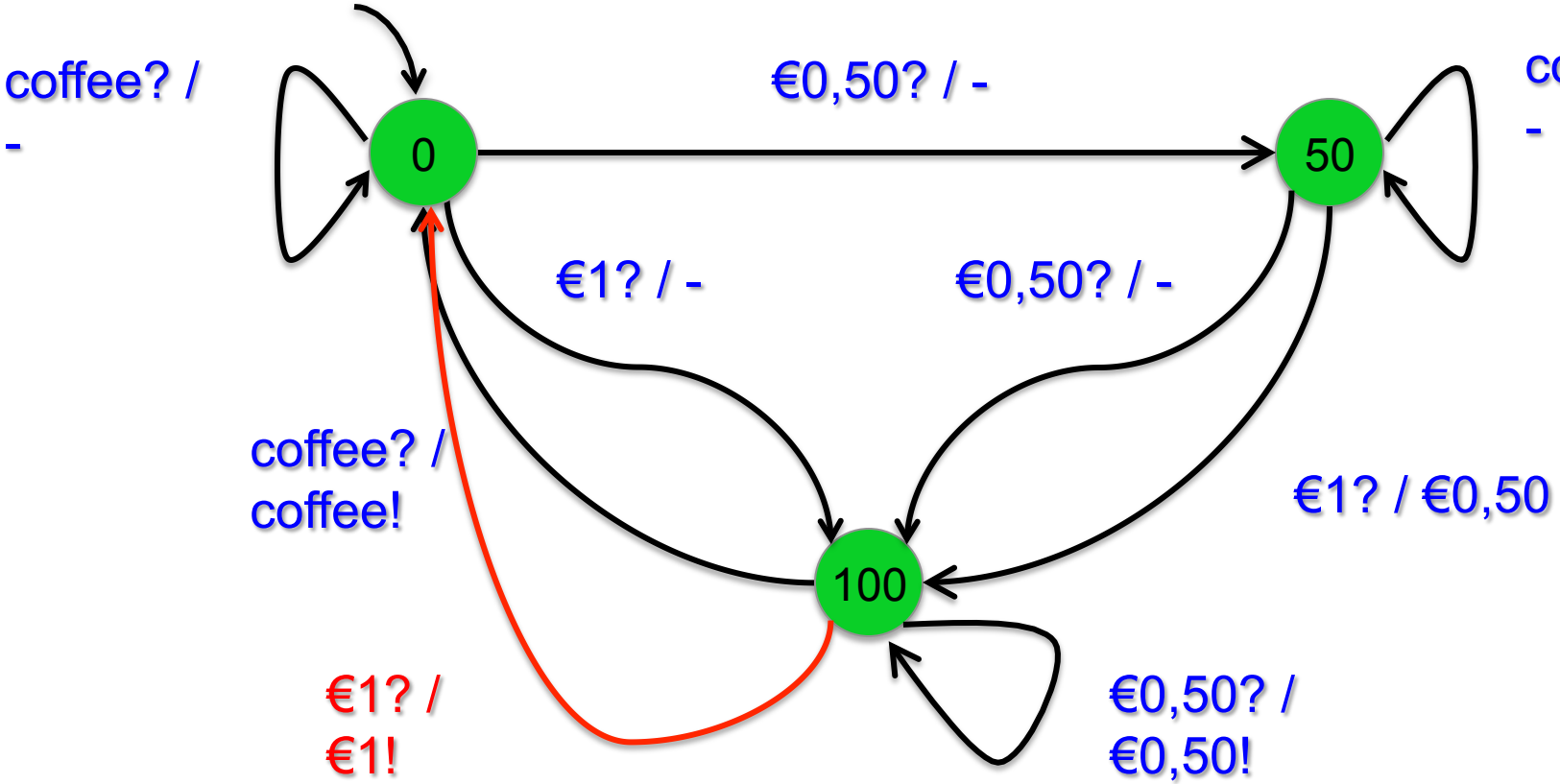
€0,50?/- €1?/€0,50! coffee?/coffee!



test suite T correct wrt S

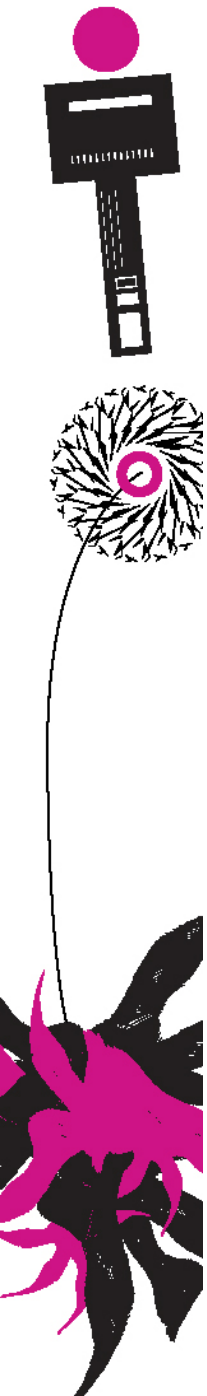
- i ≡ S → i passes all tests in t
  SOUNDNESS
- i ≢ S → i fails at least one test in t
- COMPLETENESS

Example: Coffee Machine



Transition Testing

- **Test transition:**
  1. Go to state **S1**
  2. Apply input **a?**
  3. Check output **x!**
  4. Verify state **S2**
- Checks every output fault and transfer fault (to existing state)
- If we assume that
  # states of implementation machine **MI ≤**
  # states of specification machine to **Spec.**
  then testing all transitions leads to equivalence of reduced machines,
  i.e., complete conformance
- **I.e. MI conforms to Spec iff**
  Imp passes all transition test generated from Spec



Today's agenda

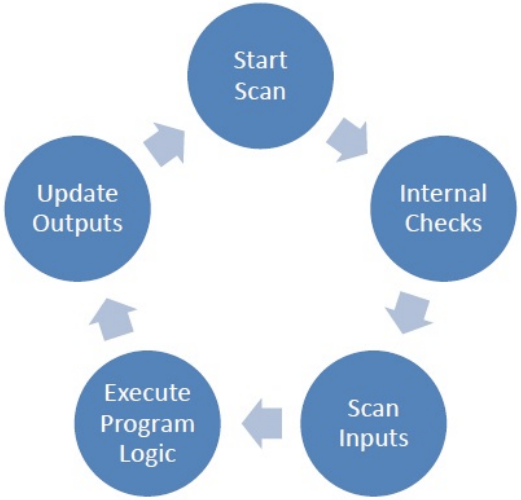
1. **FSM model**
2. **Test derivation methods**
   1. State tour method
   2. Transition tour method
   3. Transition testing method
3. **Correctness of test derivation methods**
   - Are all verdicts (pass / fail) correct?
   - Soundness & completeness
4. **Variations and Applications**

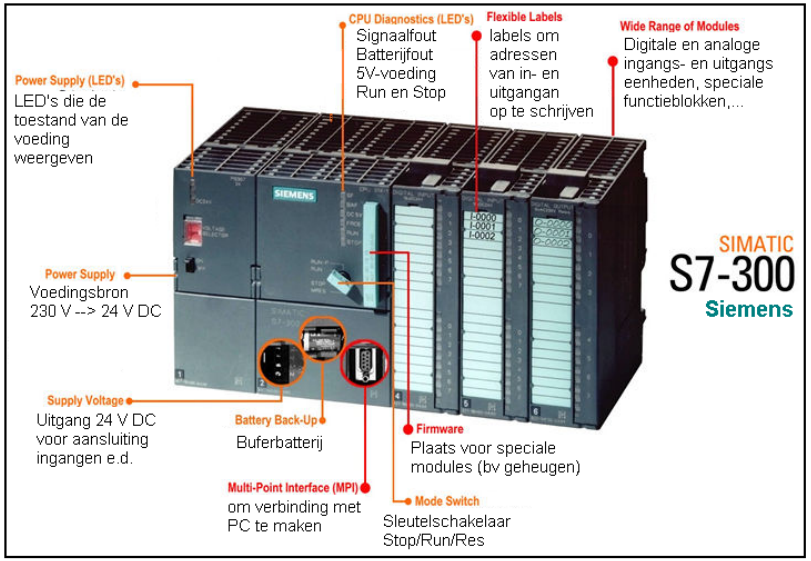
Applications

1. PLCs
2. Synchronous programming languages
3. MatLab

Programmable Logic Controllers (PLCs)

- Heavily used in industrial application
  - Railroads
  - Processing industry (food, oil,…)
- Work in fixed-timed cycles
  - Read input
  - Compute result
  - Produce output
- Modeled as FSMs

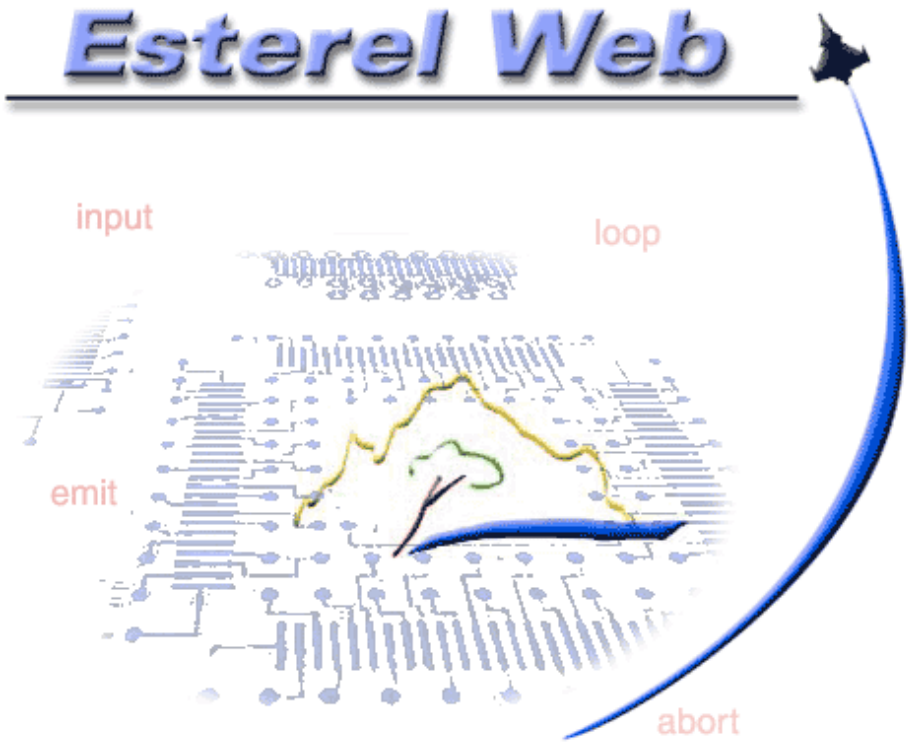




Synchronous programming languages

Programming languages paradigms:

- Functional
- Object-oriented
- Logical
- Aspect-oriented
- Synchronous
  - Lustre
  - Esterel



FSMs in MatLab

- see
- http://nl.mathworks.com/videos/stateflow-overview-61210.html
- Note:
  - Extended Finite State Machines
- Application to testing
  - http://www.mathworks.com/tagteam/28434_AIAA_2005_Mosterman_Ghidella.pdf


I wish you a nice & productive week