n-Complete Test Suites for IOCO

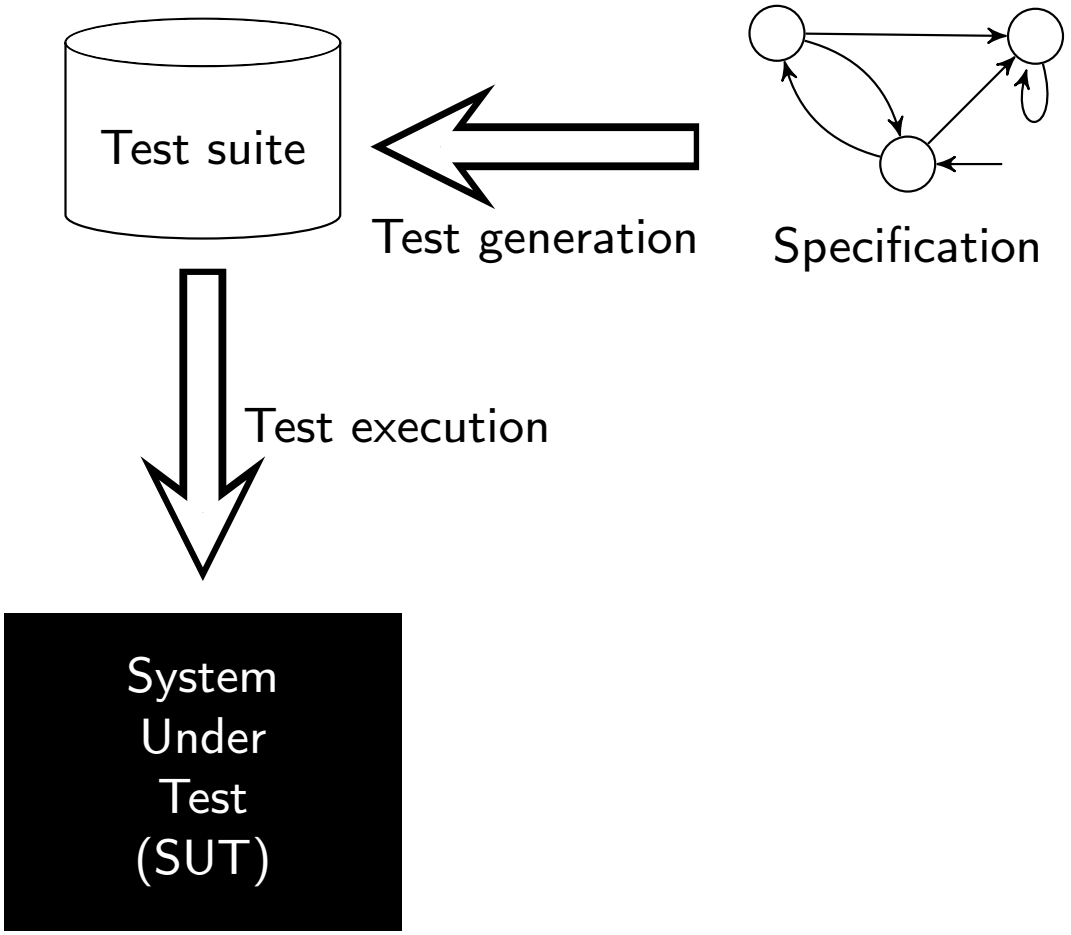
Petra van den Bos
Ramon Janssen
Joshua Moerman

October 9, 2017

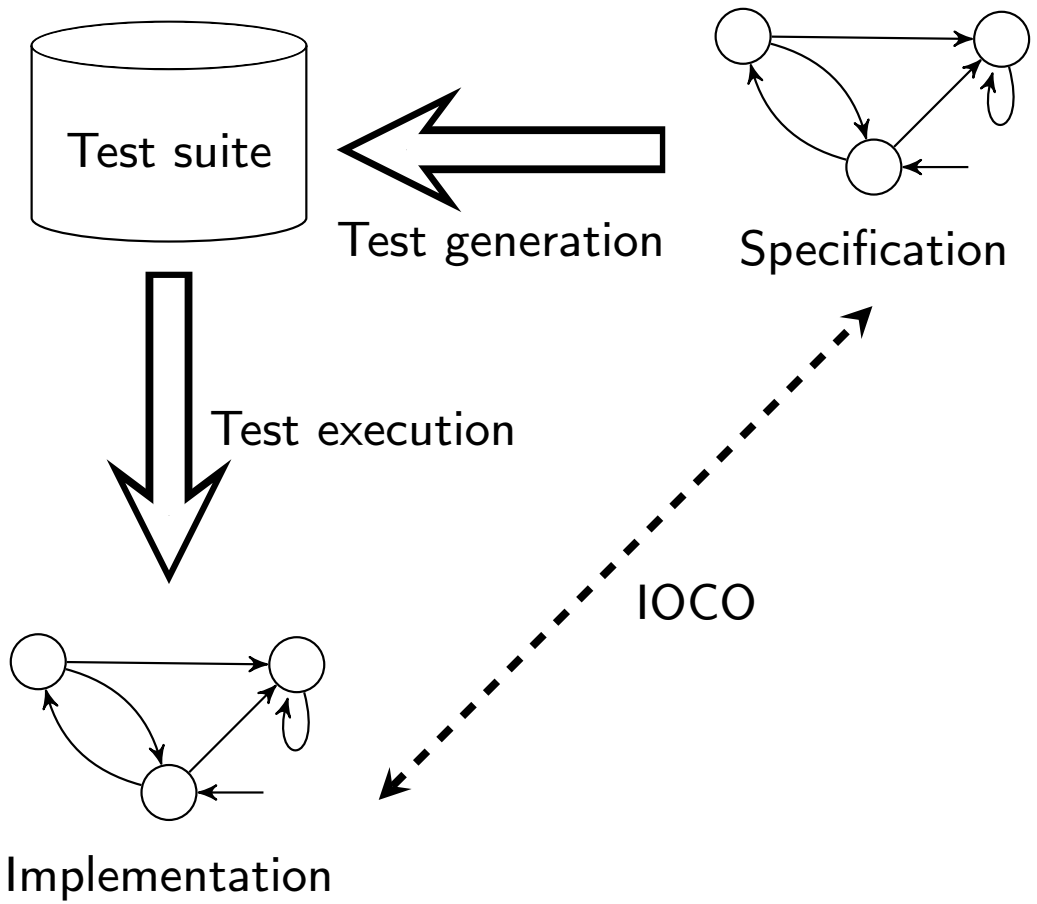
Enabling new technology

iCIS | Software Science
Radboud University

What is model based testing?



What is model based testing?

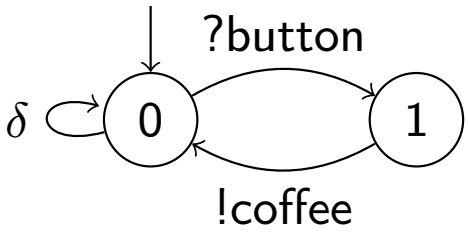


Holy grail: a complete test suite

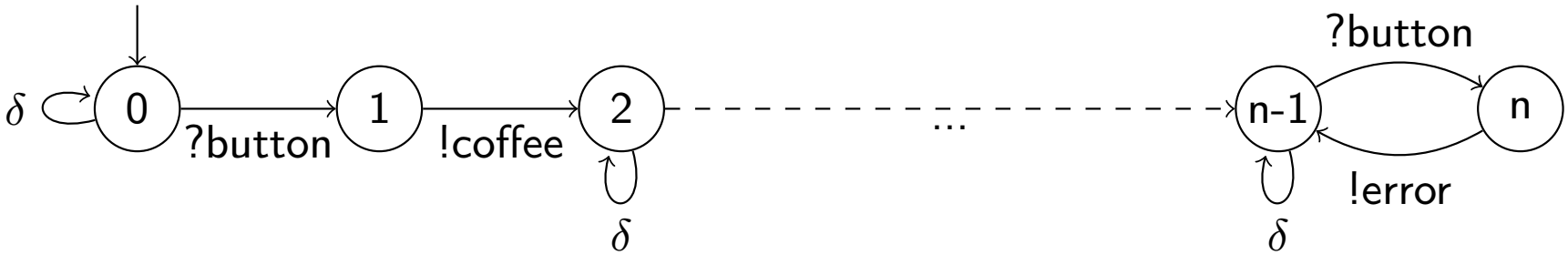
A complete test suite detects any faulty implementation

Testing infinitely

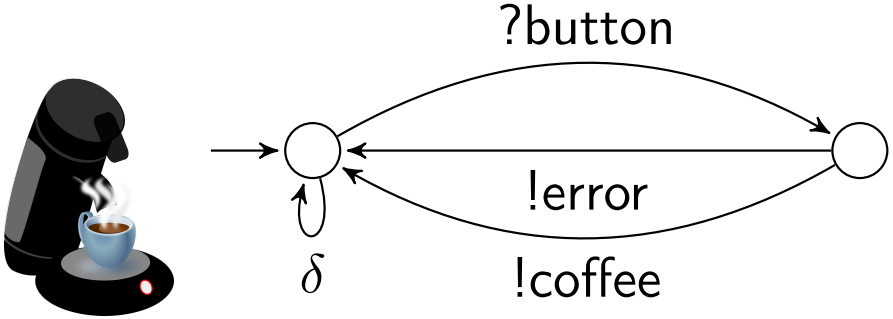
Specification:



Implementation:



Automata



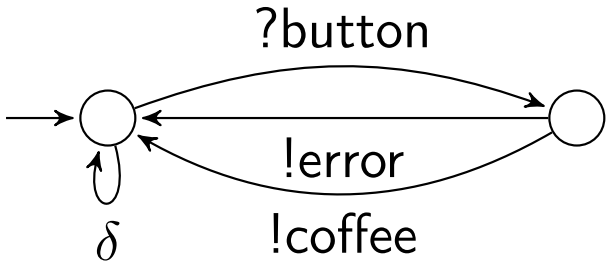
A suspension automaton

- has a *finite* number of *states* and *transitions*
- has transitions labeled with an *input* (?) or *output* (!) and quiescense (δ)
- is *deterministic* (or transformed to a deterministic system)
- though output nondeterminism is allowed!

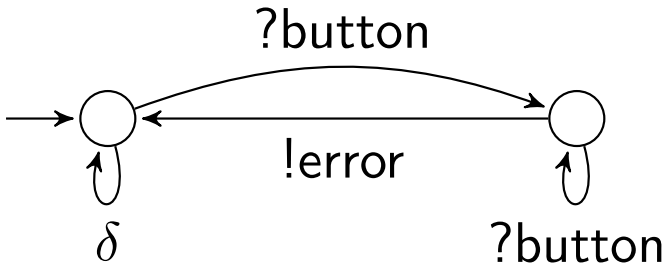
IOCO: Input Output COnformance

Formally: I IOCO S $\Leftrightarrow \forall \sigma \in$ Traces(S): out(I after σ) $\subseteq$ out(S after σ)

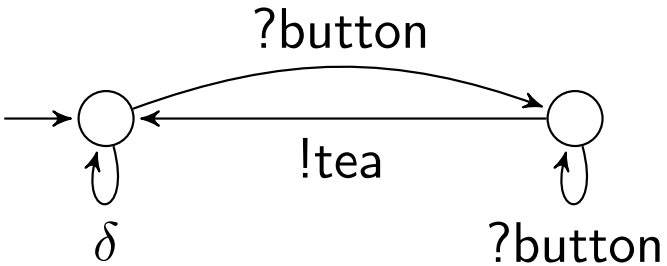
Specification S:



Implementation I_1:



Implementation I_2:

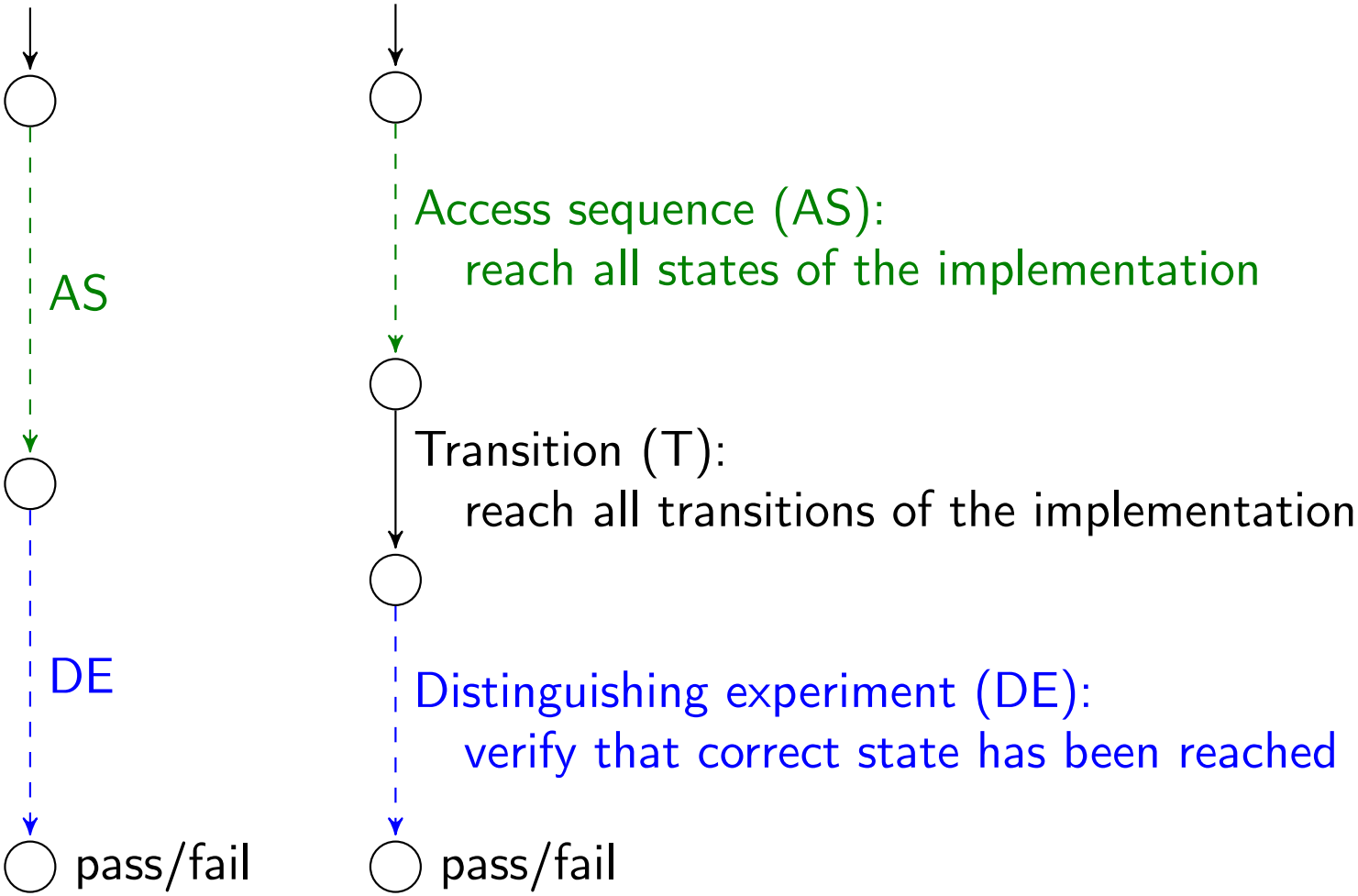


n-Complete test suite

S: a specification
T: a test suite

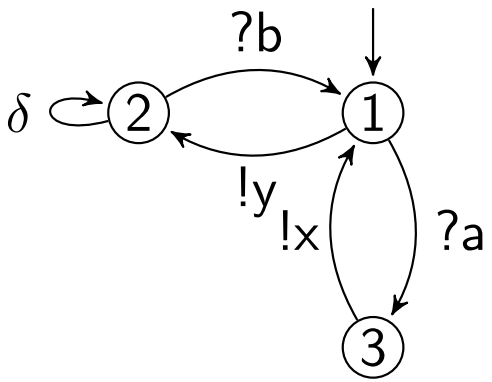
T is n-complete w.r.t. S if:
$\forall I$: I passes T $\wedge$ |I| $\leq n \implies$ I IOCO S
$\approx$ If I has $\leq n$ states, T finds out whether I is faulty

Test case of n-complete test suite



Simple example

Specification S:

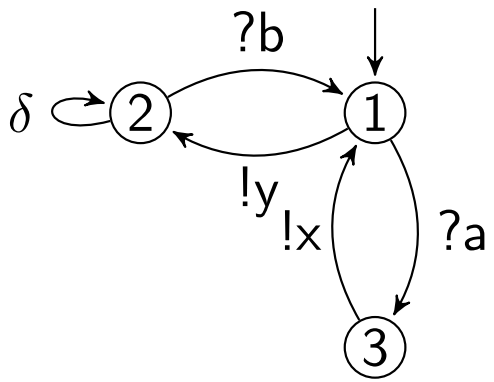


3-complete test suite:

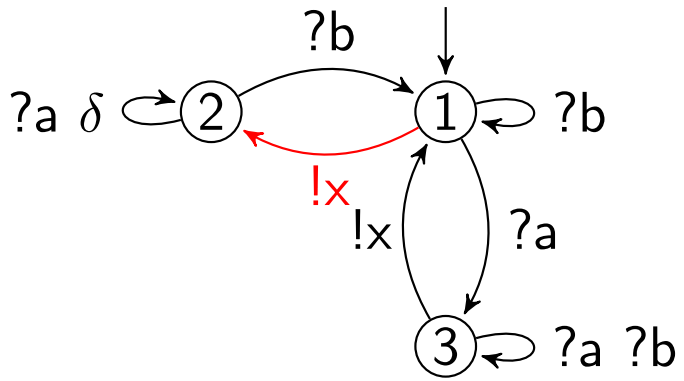
Access sequence	Transition	Distinguishing experiment
		y
y		z
a		x
	a	x
	y	z
y	b	y
y	z	z
a	x	y

Discover non-conforming implementations

Specification:



Implementation:

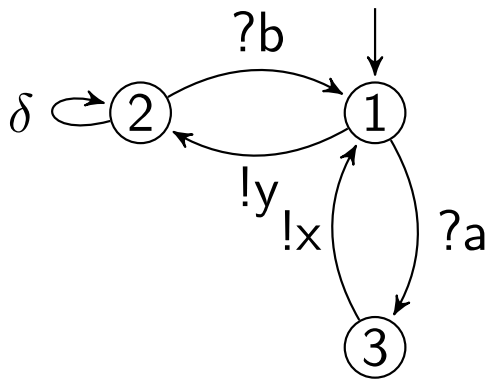


3-complete test suite:

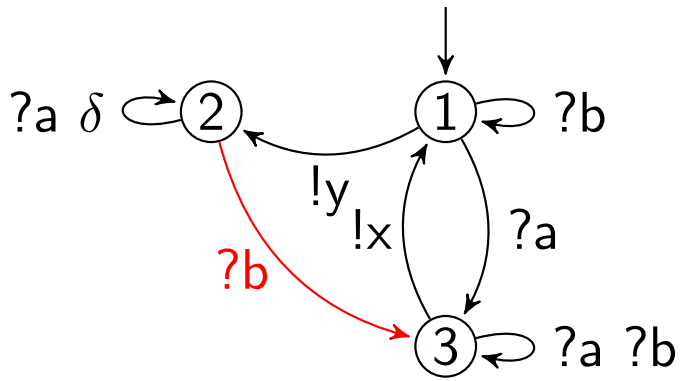
Access sequence	Transition	Distinguishing experiment
		y
y		z
a		x
	a	x
	y	z
y	b	y
y	z	z
a	x	y

Discover non-conforming implementations

Specification:



Implementation:



3-complete test suite:

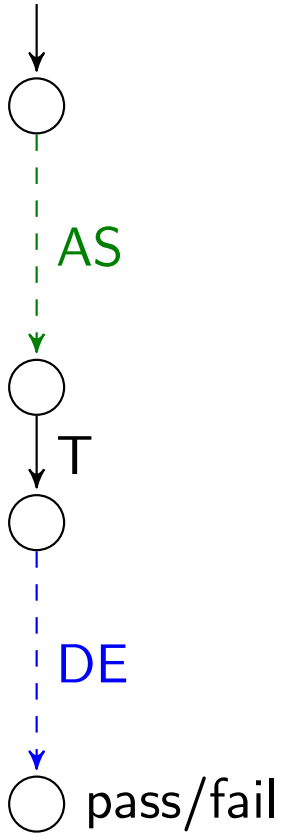
Access sequence	Transition	Distinguishing experiment
		y
y		z
a		x
	a	x
	y	z
y	b	y
y	z	z
a	x	y

Existing methods for n-complete test suites

Well-established FSM theory:

Exponential in n - |S|

Feasible



Restricted IOCO[1]:

- no compatible states
- |S|=n

Efficient

Feasible

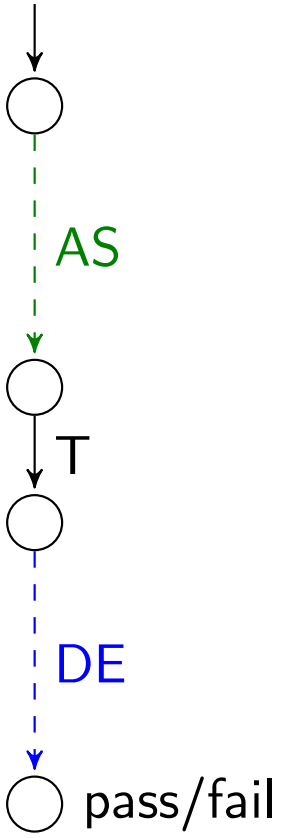
[1] Adenilso Simao, Alexandre Petrenko. Generating Complete and Finite Test Suite for ioco: Is It Possible?. MBT 2014.

Our contributions

Well-established FSM theory:

Exponential in n - |S|

Feasible

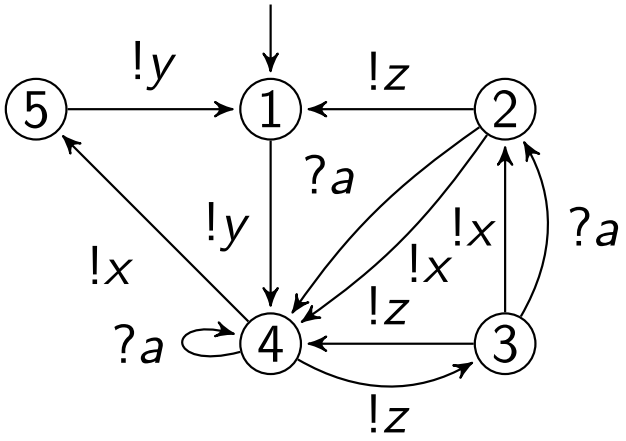


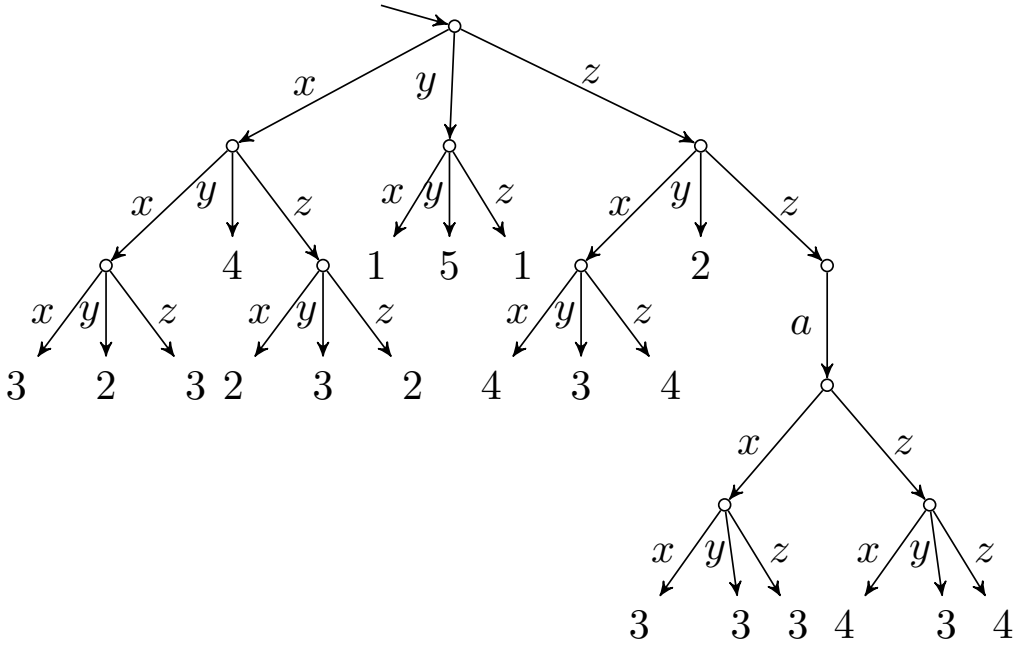
IOCO: General approach

Exponential in n

? ⇐ Focus

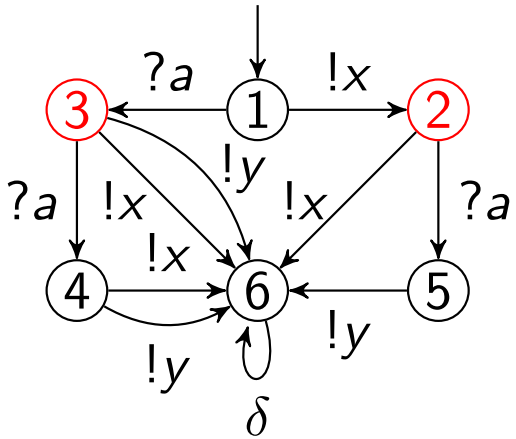
Distinguishing tree



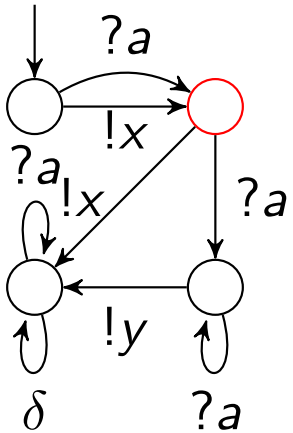


What if states cannot be distinguished?

Specification



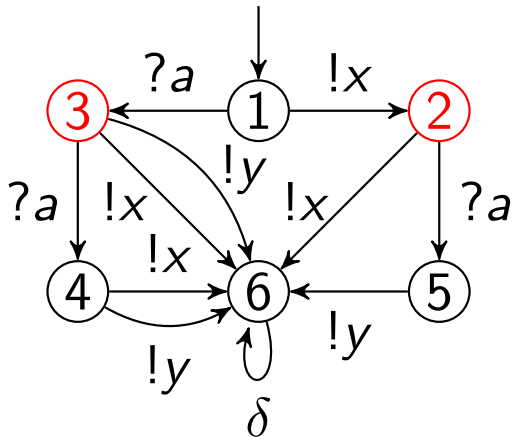
Implementation



- States 2 and 3 are implemented with a single state!
  - They are *compatible*

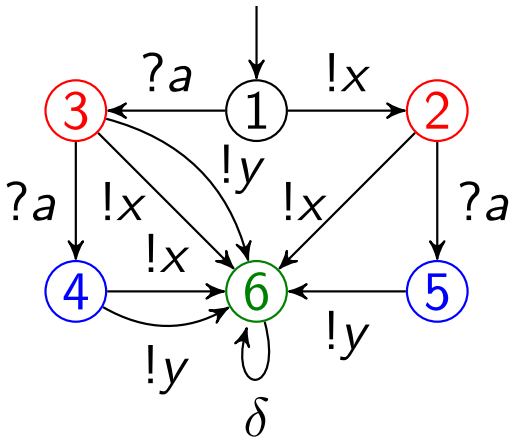
Compatible

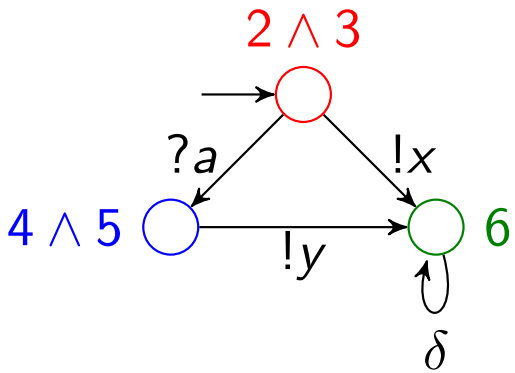
- Two states q and q' are compatible ($\Diamond$) if:
  - $\exists x \in out(q) \cap out(q') : q \text{ after } x \Diamond q' \text{ after } x$, and
  - $\forall a \in in(q) \cap in(q') : q \text{ after } a \Diamond q' \text{ after } a$



Testing with compatible states

- A distinguishing experiment for q and q':
  - Shows non-conformance to q or q' (as usual)
  - Or proves conformance to both (new)
- Calculate the merge[1]: 'the compatible part'





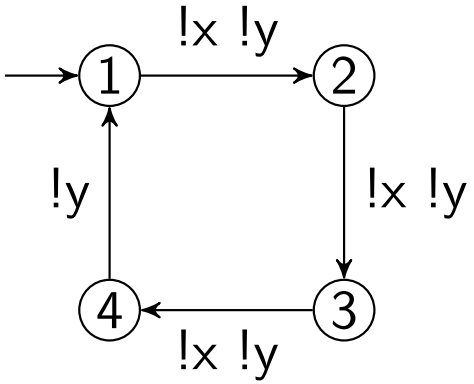
- Use *n*-complete test suite for proving conformance to both (recursive definition)

[1] Nikola Beneš, Przemysław Daca, Thomas A. Henzinger, Jan Křetínskỳ, and Dejan Ničković. Complete Composition Operators for ioco-Testing Theory. CBSE 2015.

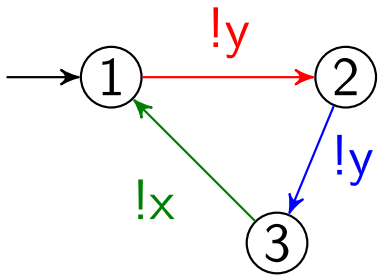
Termination of recursive test suite

- Often the recursion stops
- Otherwise fallback:
- Counterexamples have at most length $|S| \cdot |I|$

Specification:



Implementation:



- Shortest couterexample: yyxyyxyyxyyx, of maximal length $4 \cdot 3 = 12$

Summary

- An n-complete test suites detects all faults in implementations with at most n states
- Straightforward translation to IOCO is not possible
  - Cannot always distinguish states due to compatibility
- Use a recursive test suite or state counting

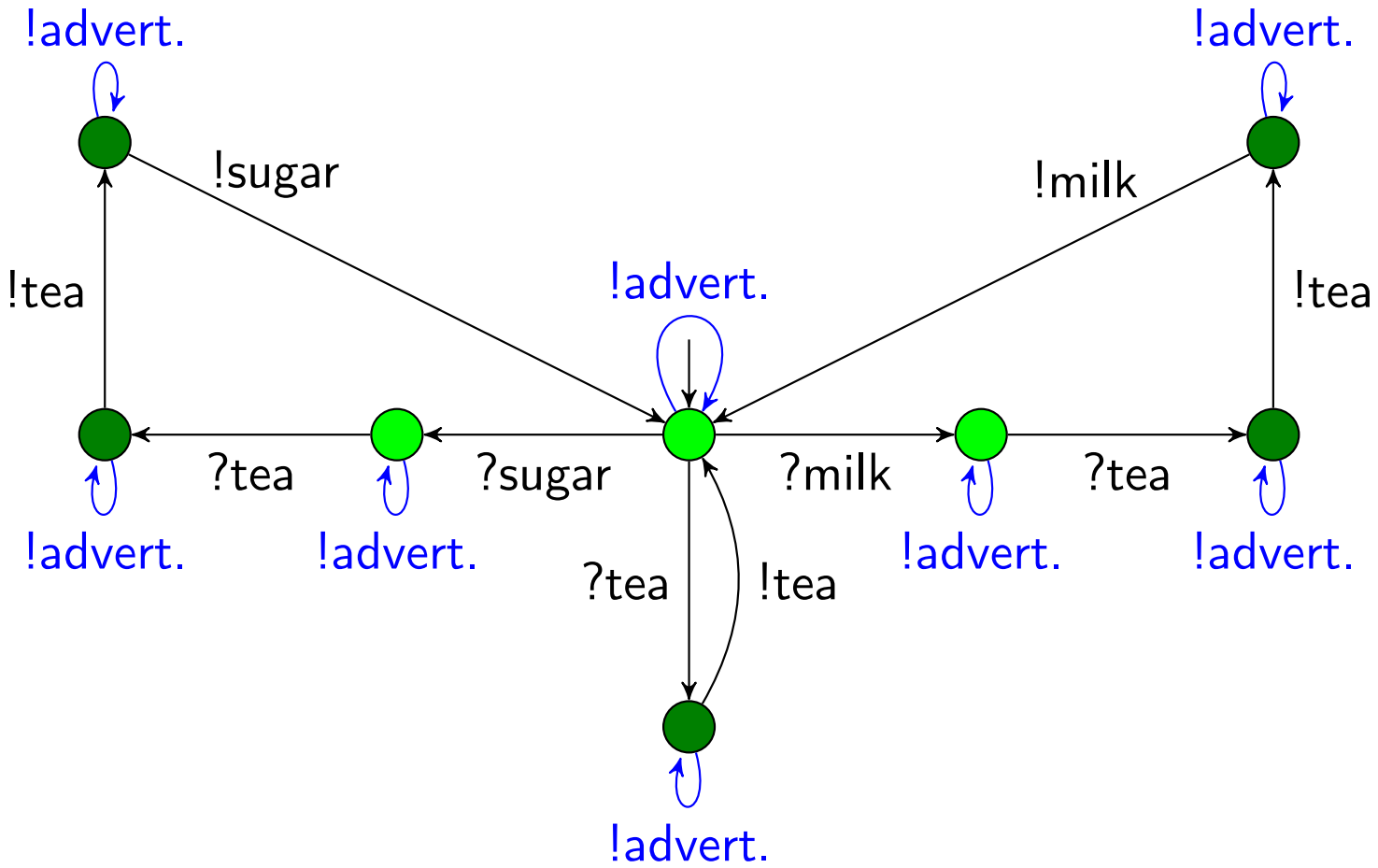
Future work

- Write journal version of this paper
- More efficient access sequences
- Case studies for a quantative comparison
- Extension to setting with data

Thank you for listening

Questions?

Compatible states example: tea machine



Mixed states example: printing two documents