Revisiting AI and Testing Methods to Infer FSM Models of Black-Box Systems

Roland Groz, Nicolas Bremond, Catherine Oriat, U. Grenoble Alpes, France

Adenilso Simao, U. São Paulo, Brasil

Global context: inferring models thru testing

- Model-based testing is good (systematic)
  - But often NO model available
- Goal: keep benefits of MBT when no model

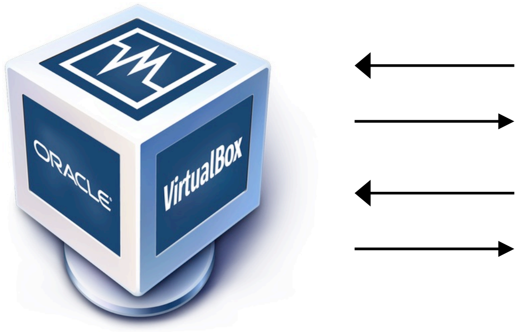
Method: Testing a system is LEARNING the behaviour of a system

➔ Use "ML" techniques to learn model

Problem: learn correct & "complete" behaviour of Black Box systems that cannot be reset

Motivational example

- Reverse-engineer models of Web applications to detect <u>security vulnerabilities</u> using Learning algos (e.g. L*)
- E-Health app provided by Siemens as a Virtual Machine



Learner

- single I/O RTT over LAN: < 1 ms
- reset=reboot VM: ~1 minute

- Timewise: <u>reset is O(10^5)</u> RTT in example
- Many systems CANNOT be reset AT ALL.

Key difficulties when no reset

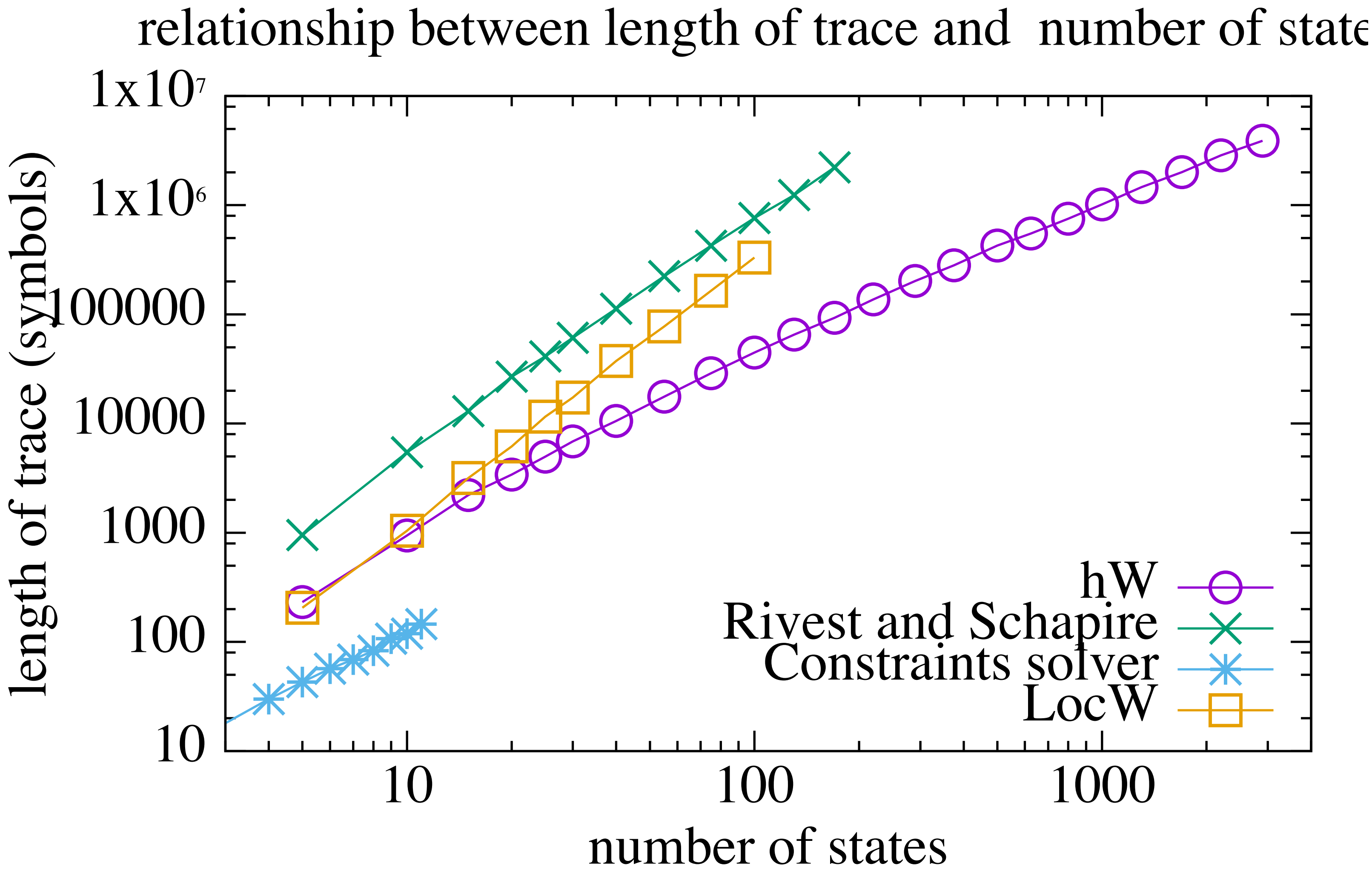
- How can we know in which state seq is applied ?
- No backtrack possible to check other sequence
- Losing track: we no longer know from where we apply an input

Existing algorithms without reset

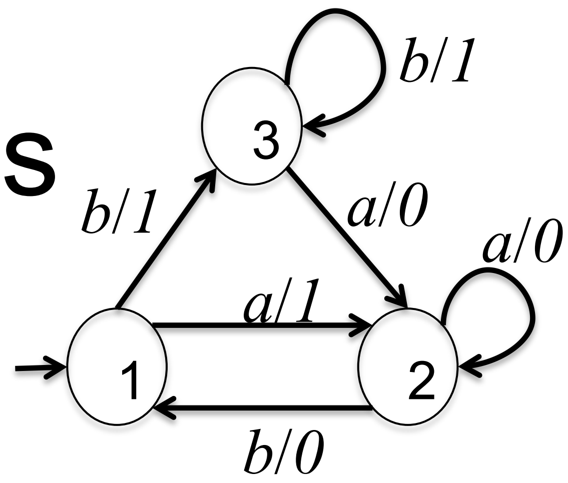
- Rivest & Schapire 1993
  - *Homing sequence*: ersatz for resetting in one of several states
  - Then use a copy of L* for each homed state
- LocW (Groz & al. 2015)
  - *Assume W-set known (identifying sequences)*
  - Localize in an identifiable state with nested W
- Constraint-solving (Petrenko & al. 2017)
  - *Assume bound n* on #states.

- NEW (this paper): hW inference
  - *No assumption* ! Discovers h(oming) and W (characterizing)

Results on random machines
(log-log)



Homing seq and W-sets



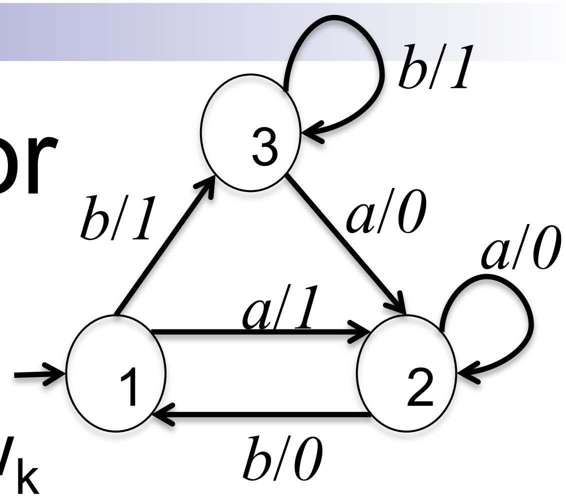
- h=a is homing sequence:
  - After a/0 or a/1, final state=2,

    (in this case h is a reset because single final state)
- W={a,b} is a characterizing set
  - a/1, b/1 : characterize state 1
  - a/0, b/0 : characterize state 2
  - a/0, b/1 : characterize state 3

Note: single homing sequence, but most machines require |W|> 1

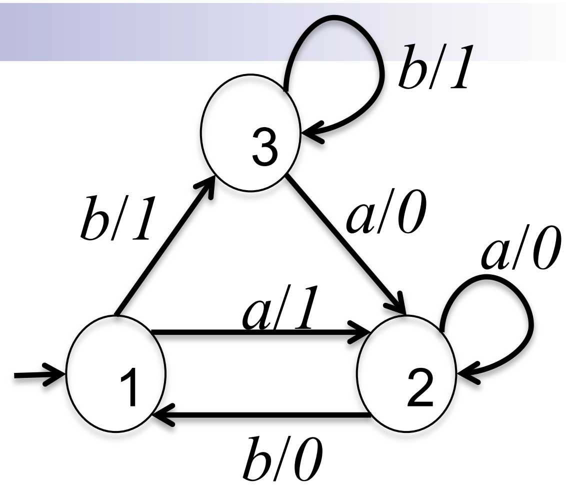
hW inference: core loop for

h=a W={a, b}

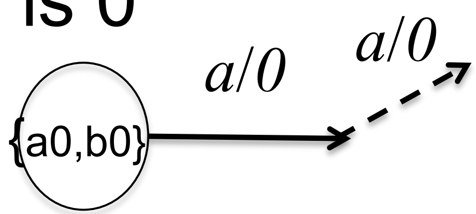


- Repeatedly apply h, an input and w_k to progressively learn transitions
  - More generally $h\alpha x w_k$, α transfer seq., x input
- h/1.w_1/0 h/0. w_1/0 h/0. w_2/0
  - At this point we know that tail state of h/0 is state characterized by {a/0,b/0} (and we are now in state 1)
- h/1: we are again in tail state h/1, apply w_2
- b/0: now we know tail state h/1 is {a/0,b/0}

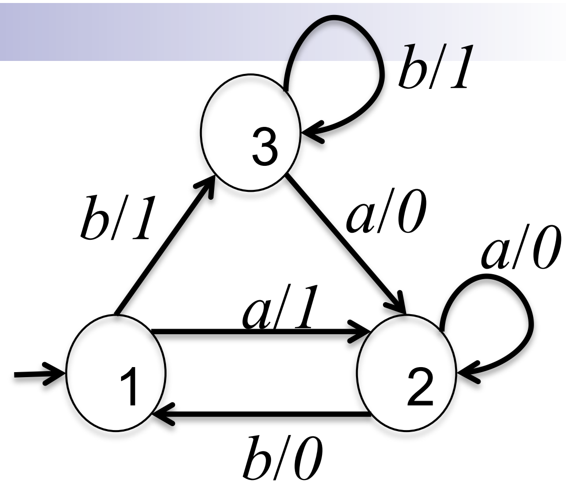
hW inference: cont'd
h=a W={a, b}



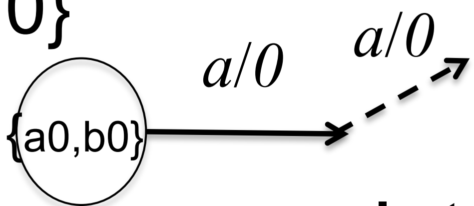
- Known
  - h/0 -> {a/0,b/0}
  - h/1 -> {a/0,b/0}
- (and we are in 1). Apply h: a/1. We are now in a known state {a/0,b/0}
- So we learn a transition from it:
  - a/0 so we know the output on a is 0
  - And tail state answers w_1/0.



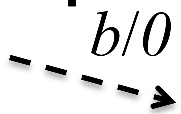
hW inference: cont’d
h=a W={a, b}



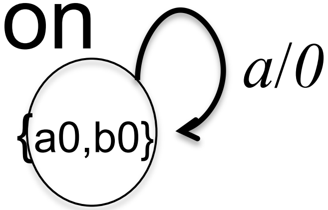
- Known
  - h/0 -> {a/0,b/0} ; h/1 -> {a/0,b/0}
  - Partial transition



- We reapply h/0. So now we can complete knowledge of transition: a/0 b/0



- So we have completely learnt transition
- Going on, we learn the full FSM



Learning with unknown h, W
Key idea: use putative h, W

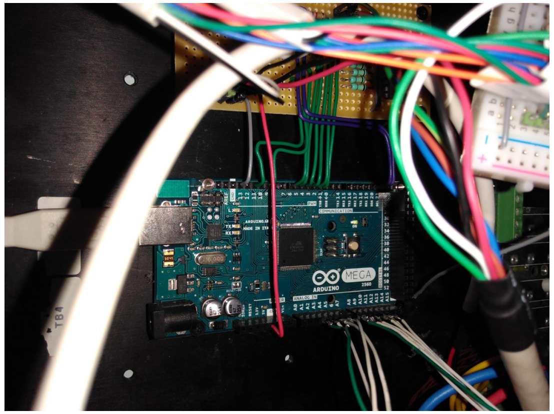
- Start with any (incorrect) h and W
  - E.g. empty sequence and set
  - Different states will be confused (merged)
  - So this will lead to apparent NonDeterminism (ND)
- ND: reapplying a transition x/0, we see x/1
  - Depending on context, we can either *extend* h to hx or W to W $\cup$ {x}
- Progressively *extending* h and W until they are homing & characterizing for the BB

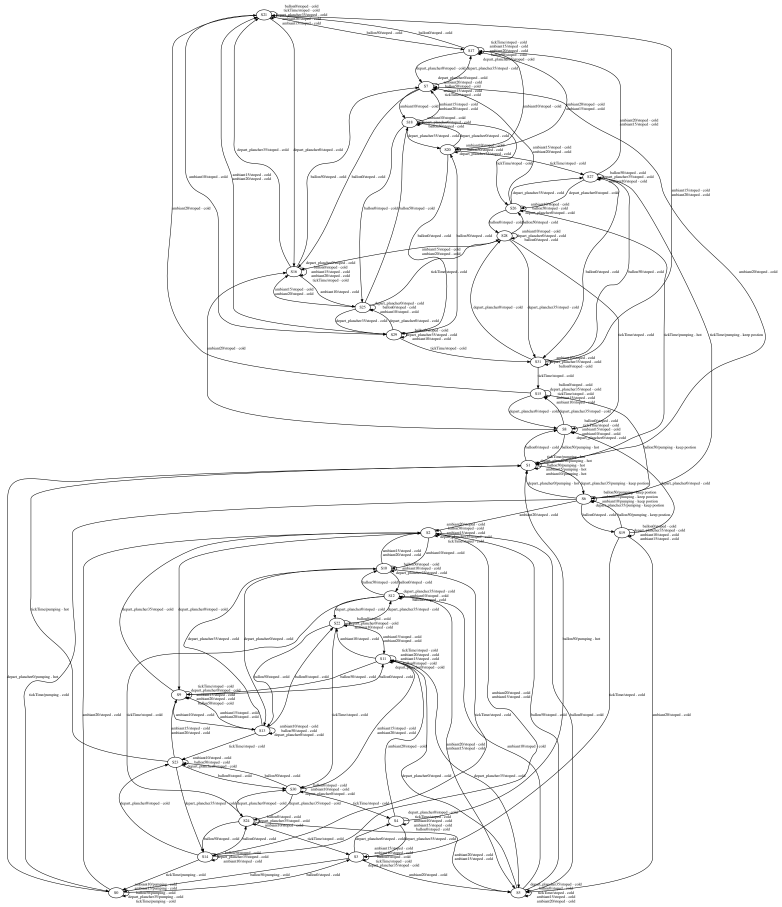
Does it work ?

- Yes !
  - Naive, but turns out to converge fast
    - Actually, enhanced with a number of heuristics not detailed here
  - Outperforms previous algorithms
    - And even algorithms with reset, such as L*
  - No initial knowledge needed (apart input set)
  - Still needs an oracle to check equivalence in the end (or get counterexample to refine)
    - Oracle can just be random walk

Does it help with s/w testing ?

- Example: a Heating Mngmt System





- C++ controller
- 3 temperature inputs + timer -> 9 inputs
- Inferred 36 states, in a few minutes

Results on HMS controller

- RQ1: does hW yield usable models on real CPS ? Yes
- RQ2: testing efficiency / random testing
  - 54 mutations
    - 10 crashes without inputs (hW = RT)
    - 4 killed during inference – also by RT
      but RT requires many more inputs to kill
    - 35 model inferred: exposes mutation
    - 5 equivalent models (w.r.t. input abstraction)

Conclusion

- New approach to learn FSM models of s/w components without reset
- Full black box, no assumption
- Works surprisingly well, scales up to 1000s states
- Also provides very systematic way of testing reactive software

Perspectives

- Potential breakthrough in Learning Based Testing
  - Resetting a system is a superfluous luxury
  - hW is fast, scaling, does not require any knowledge
- Check applicability on other types of s/w
- Extension to EFSM (data inference)

Thank you !

- Following: backup slides

Inferring model of Black Box

Testing as a means of reverse-engineering a model of a BB

- Classical active inference algorithms assume BB machine can be reset
  - Essential to merge traces (scenarios) on a common basis
- Assume an oracle can provide counterexamples (CE)
  - Essential to bring complexity down to polynomial in #states
  - Example: L* (Angluin). Complexity is $O(\#inputs\ \ CE\_length\ \#states^2) = O(fmn^2)$ queries (test seq.)
  - So $O(fmn^2)$ <u>resets</u>