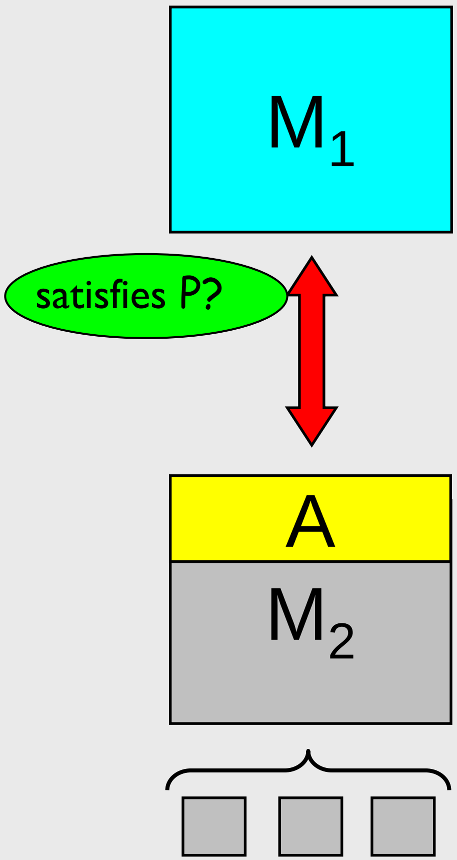
Compositional Verification

Dimitra Giannakopoulou and Corina Păsăreanu
CMU / NASA Ames Research Center

compositional verification

does system made up of M_1 and M_2 satisfy property P?



- check P on entire system: too many states!
- use system's natural decomposition into components to break-up the verification task
- check components in isolation:

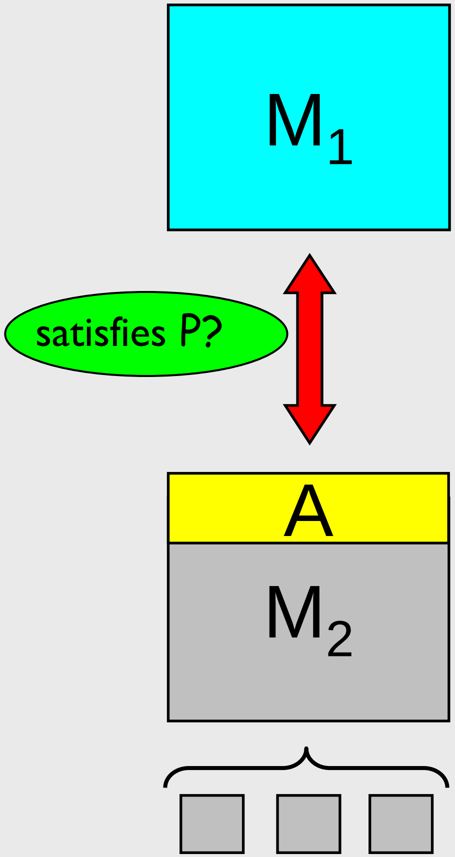
  does M_1 satisfy P?

  – components typically satisfy requirements in specific contexts / environments
- assume-guarantee reasoning
  – introduces assumption A representing M_1's "context"

examples of assumptions

- will not invoke “close” on a file if “open” has not previously been invoked
- accesses to shared variable “X” must be protected by lock “L”
- (rover executive) whenever thread “A” reads variable “V”, no other thread can read “V” before thread “A” clears it first
- (spacecraft flight phases) a docking maneuver can only be invoked if the launch abort system has previously been jettisoned from the spacecraft

assume-guarantee reasoning



reasons about triples:

$\boxed{\langle A \rangle\ M\ \langle P \rangle}$

is *true* if whenever M is part of a system that satisfies A, then the system must also guarantee P

simplest assume-guarantee rule (ASYM):

$$\frac{\begin{array}{l}1.\ \langle A \rangle\ M_1\ \langle P \rangle \\ 2.\ \langle \mathit{true} \rangle\ M_2\ \langle A \rangle\end{array}}{\langle \mathit{true} \rangle\ M_1 \parallel M_2\ \langle P \rangle}$$

"discharge" the assumption

assume-guarantee reasoning

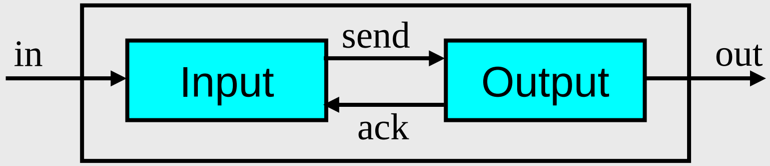
how do we come up
with the assumption?

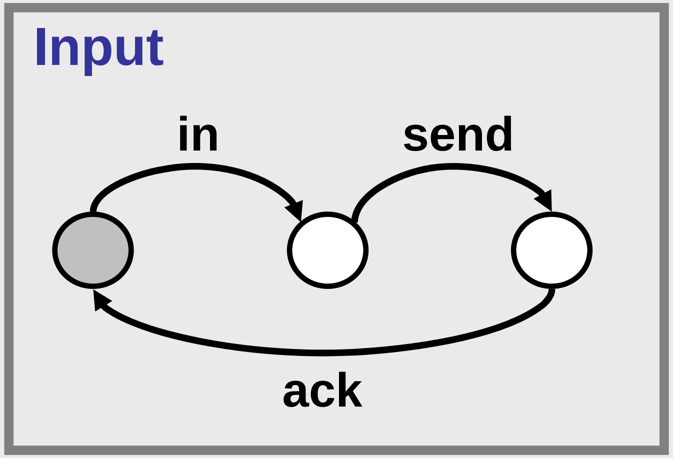
formalisms

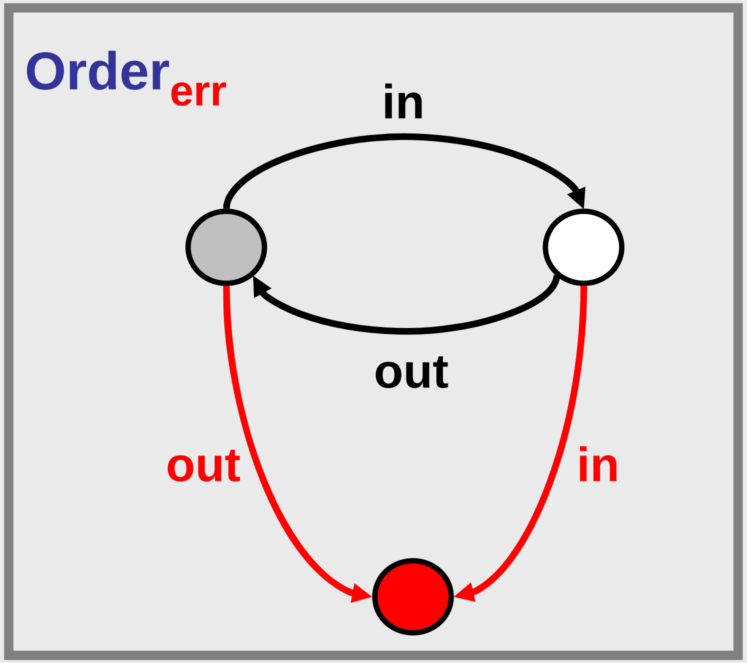
- components modeled as finite state machines (FSM)
  - FSMs assembled with parallel composition operator "||"
    - synchronizes shared actions, interleaves remaining actions
- a safety property P is a FSM
  - P describes all legal behaviors in terms of its alphabet
  - P_{err} – complement of P
    - determinize & complete P with an "error" state;
    - bad behaviors lead to error
  - component M satisfies P iff error state unreachable in ($M \parallel P_{err}$)
- assume-guarantee reasoning
  - assumptions and guarantees are FSMs
  - $\langle A \rangle\ M\ \langle P \rangle$ holds iff error state unreachable in ($A \parallel M \parallel P_{err}$)

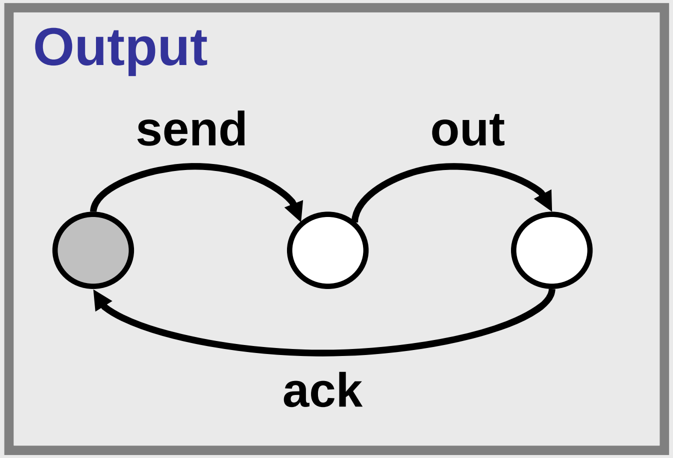
example

Require in and out to alternate (property Order)

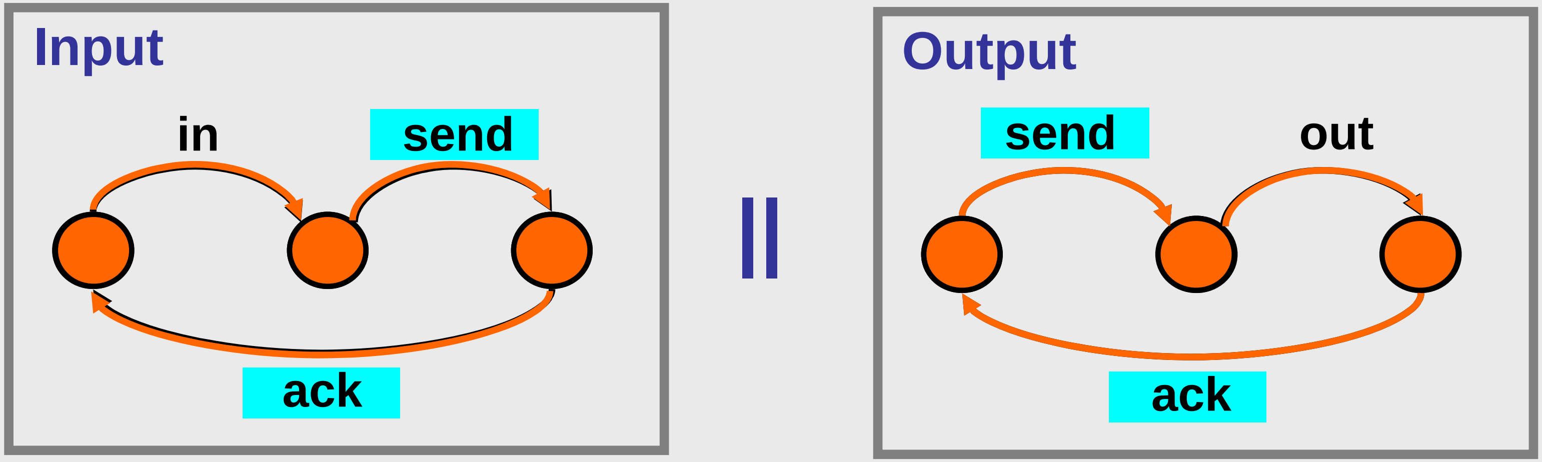




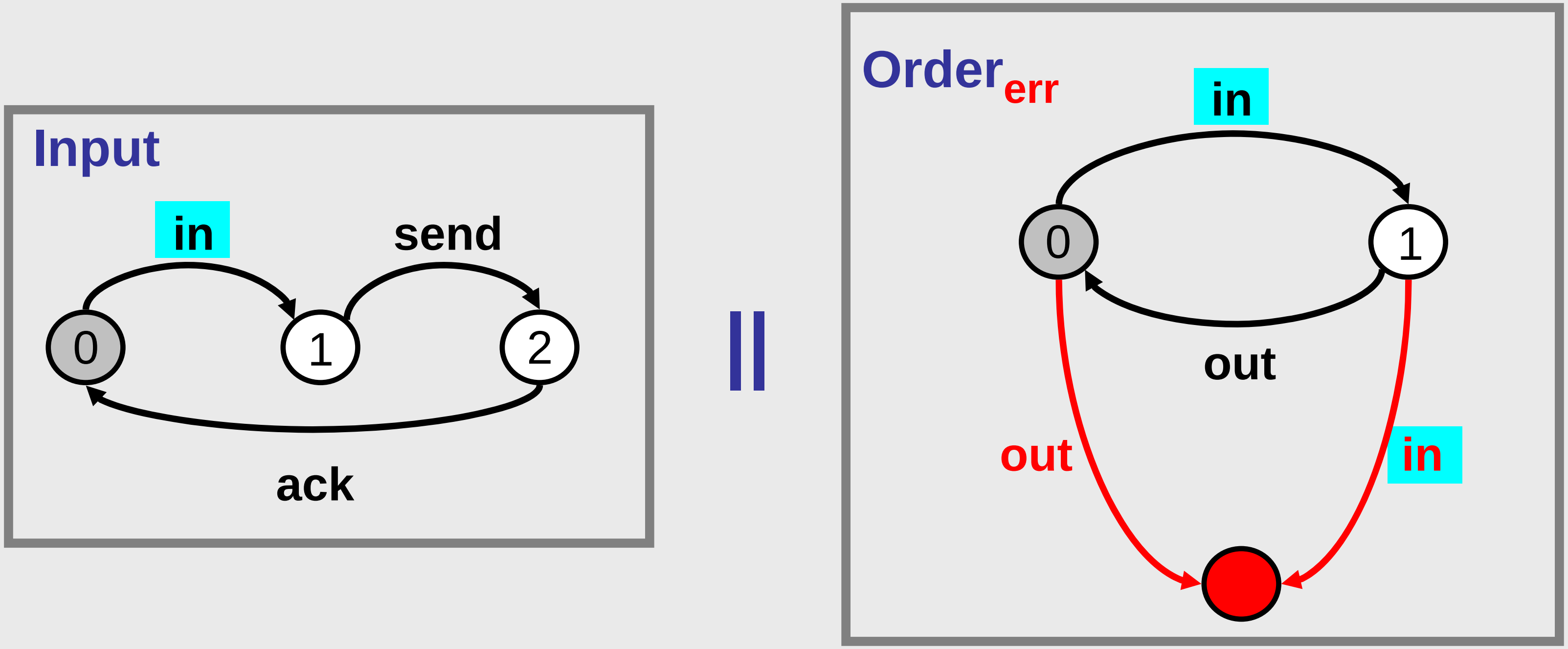




parallel composition



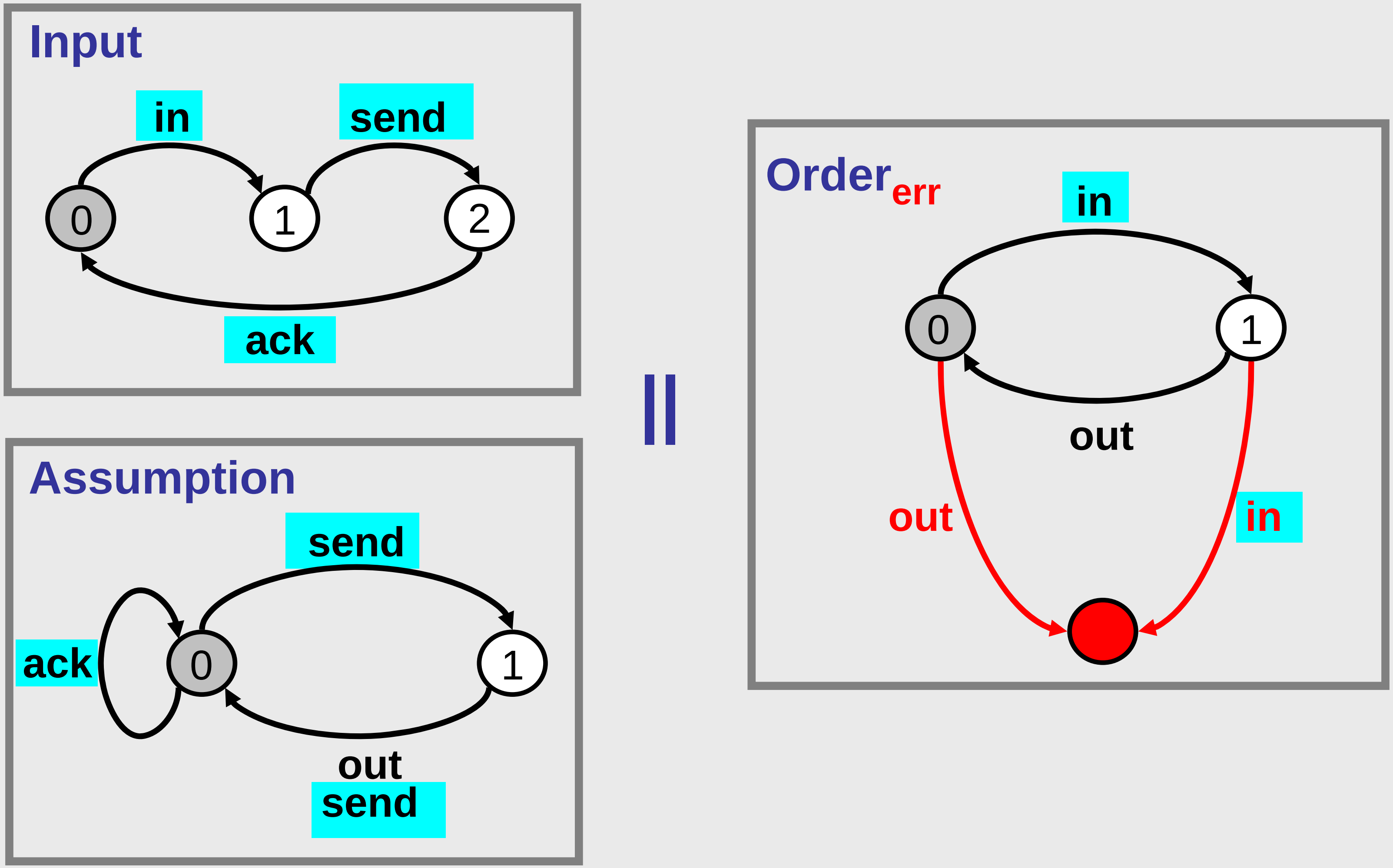
property satisfaction



crex. 1: (I_0, O_0) out (I_0, O_{error})

crex. 2: (I_0, O_0) in (I_1, O_1) send (I_2, O_1) out (I_2, O_0) out (I_2, O_{error})

assume-guarantee reasoning



crex 1: (I_0, A_0, O_0) out ✗

crex 2: (I_0, A_0, O_0) in (I_1, A_0, O_1) send (I_2, A_0, O_1) out ✗

the weakest assumption

- given component M, property P, and the interface of M with its environment, generate the weakest environment assumption WA such that: $\langle WA \rangle$ M $\langle P \rangle$ holds

- weakest means that for all environments E:

$$\langle \mathit{true} \rangle \ M \ || \ E \ \langle P \rangle \ \text{IFF} \ \langle \mathit{true} \rangle \ E \ \langle WA \rangle$$

- in other words, weakest means **safe** and **permissive**

assumption generation [ASE'02]

STEP 1: composition, hiding, minimization

property true!
(all environments)

STEP 2: backward propagation of error along τ transitions

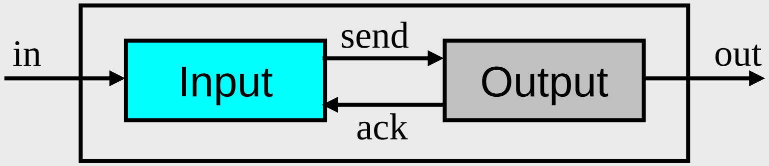
property false!
(all environments)

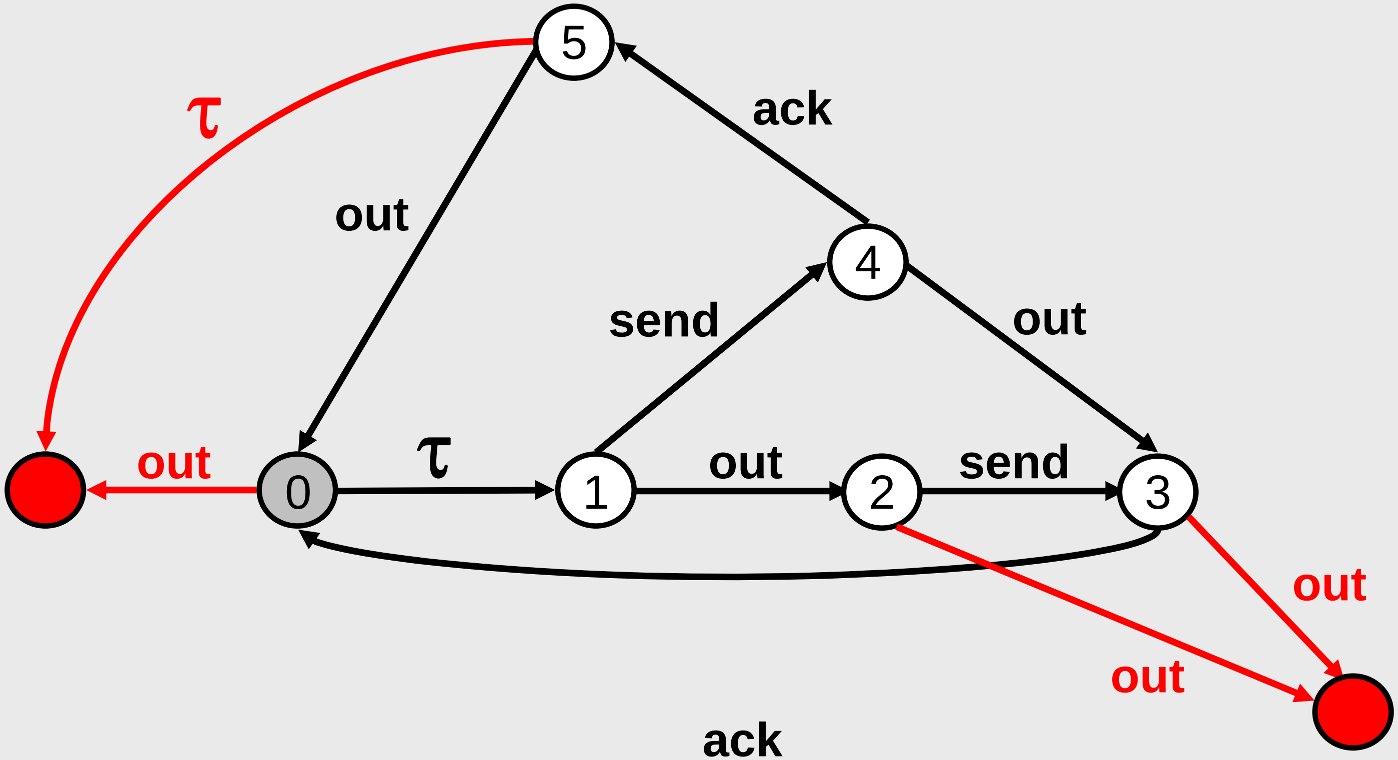
STEP 3: property extraction (subset construction & completion)

assumption

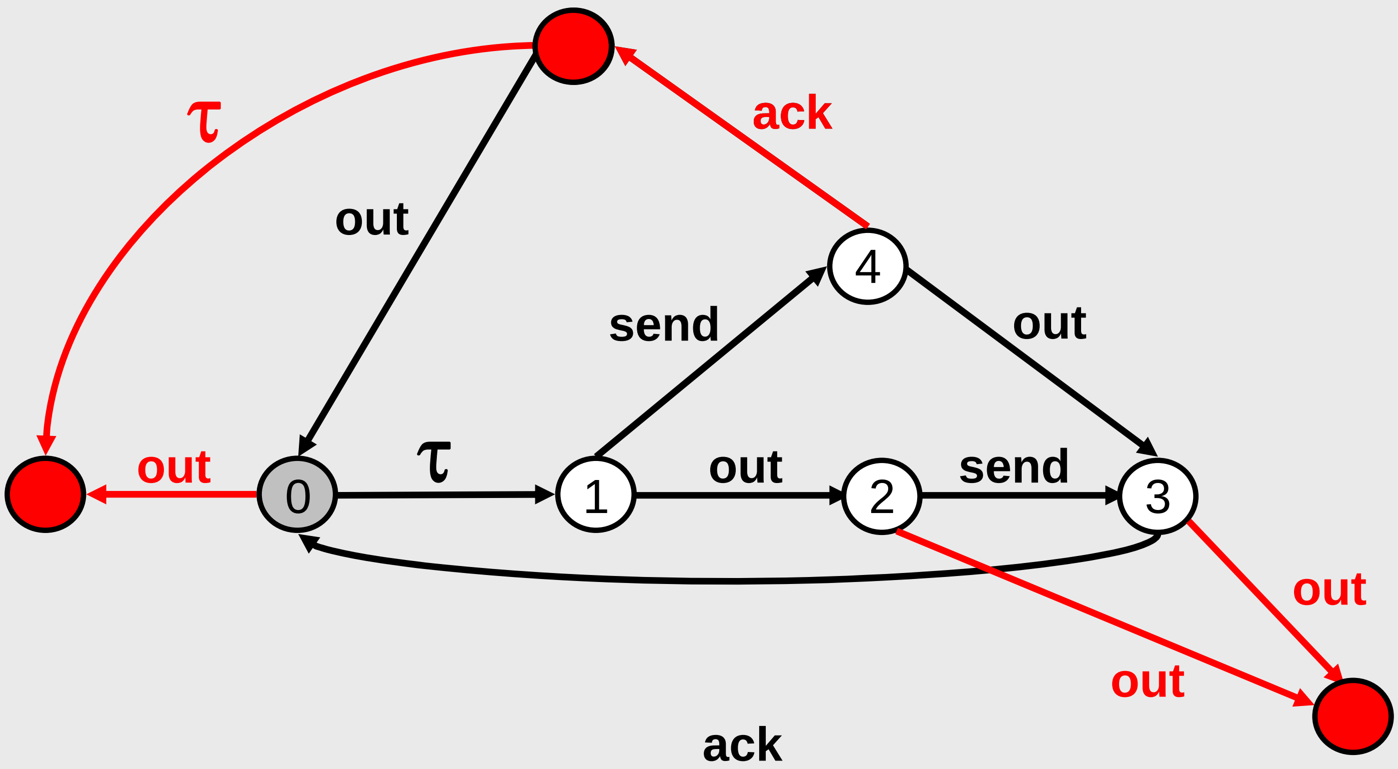
step 1: composition & hiding

Input || Order$_{err}$ \ {in}

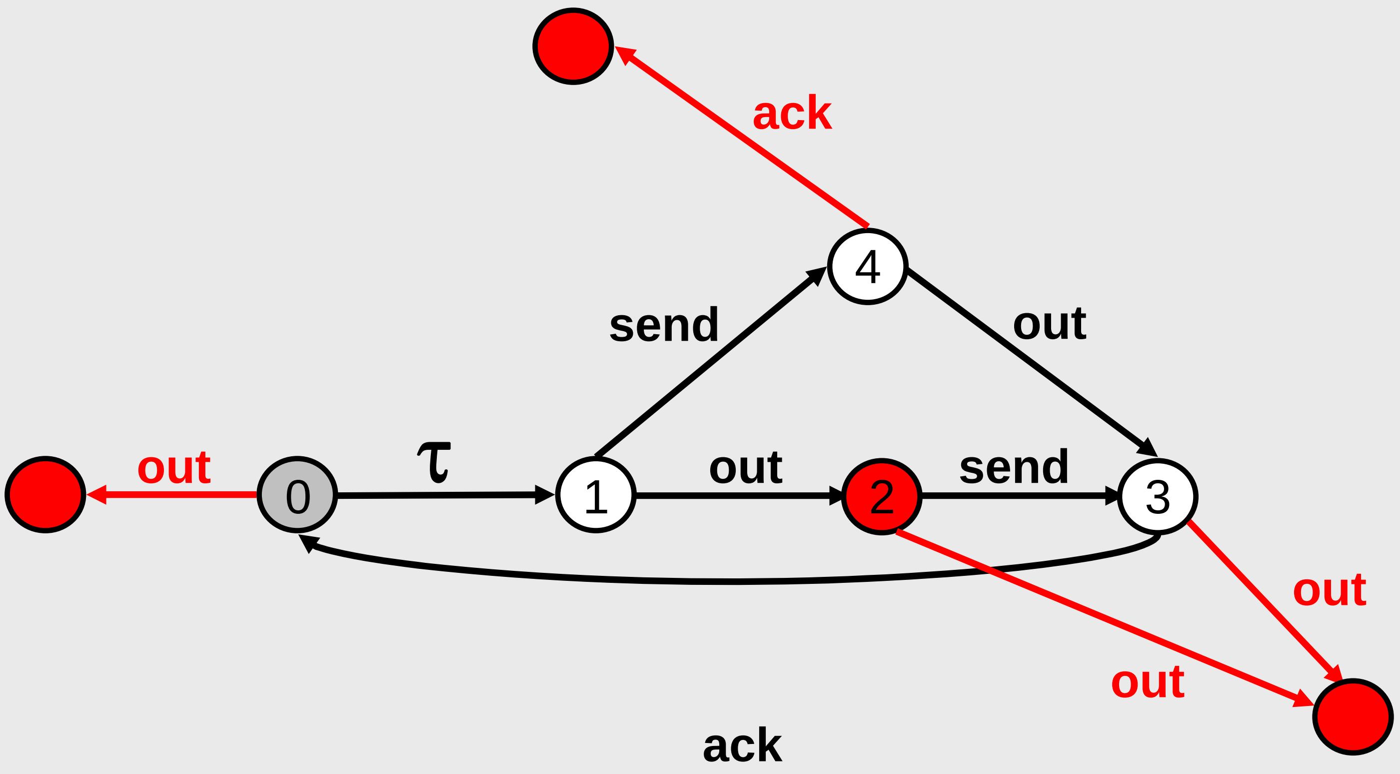




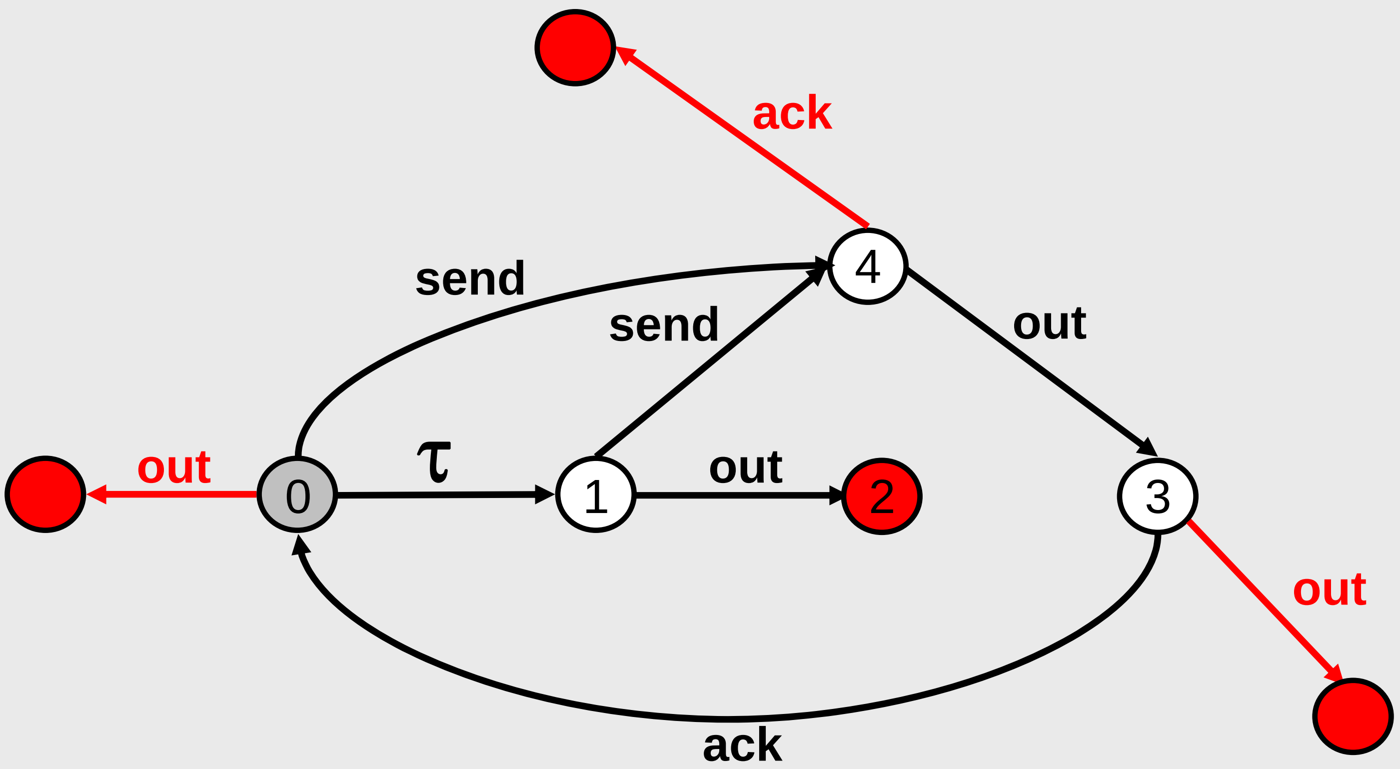
step 2: error propagation



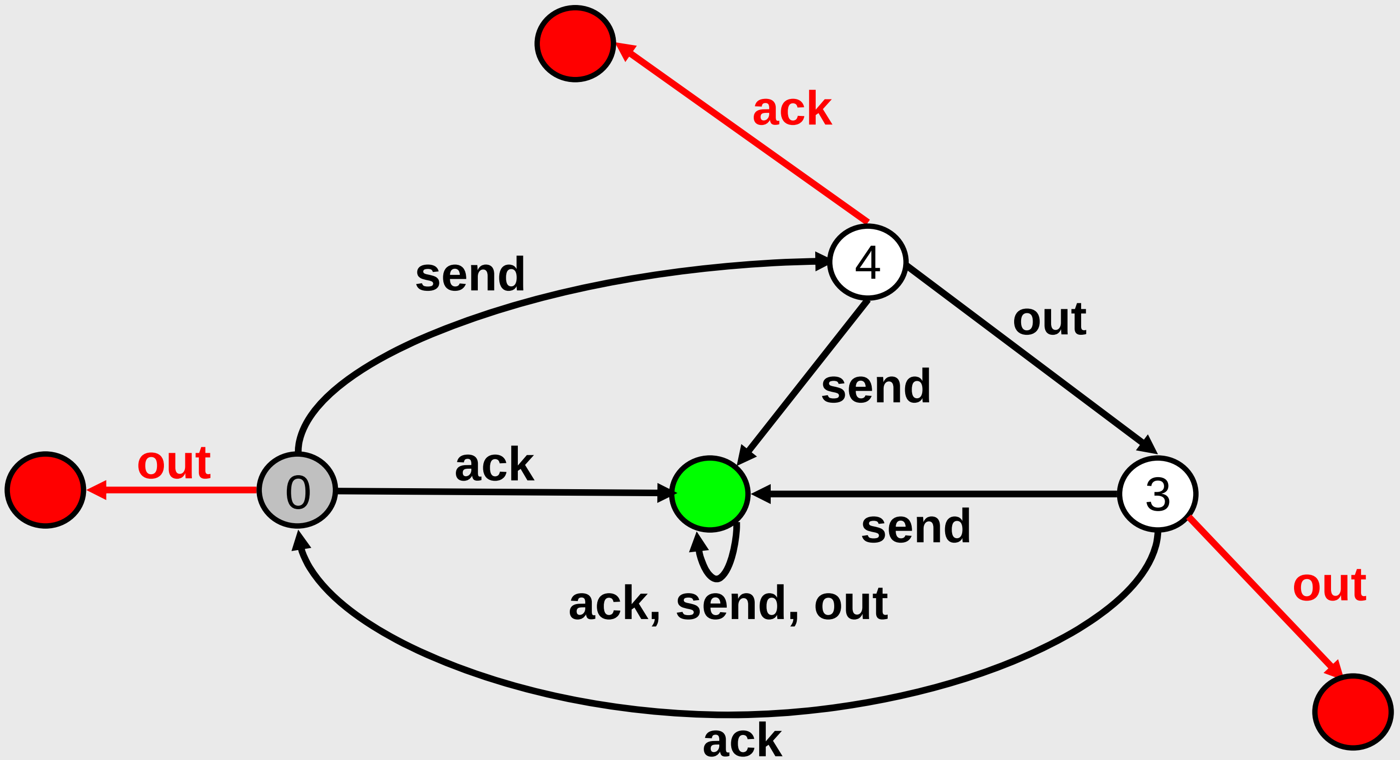
step 3: subset construction



step 3: subset construction



step 3: property construction



weakest assumption in AG reasoning

$$\frac{\begin{array}{l}1.\quad \langle A\rangle\ M_1\ \langle P\rangle \\ 2.\ \langle \mathit{true}\rangle\ M_2\ \langle A\rangle\end{array}}{\langle \mathit{true}\rangle\ M_1 \,||\, M_2\ \langle P\rangle}$$

weakest assumption makes rule complete

$\langle WA\rangle\ M_1\ \langle P\rangle$ holds (WA could be *false*)
$\langle \mathit{true}\rangle\ M_2\ \langle WA\rangle$ holds implies $\langle \mathit{true}\rangle\ M_1 \,||\, M_2\ \langle P\rangle$ holds
$\langle \mathit{true}\rangle\ M_2\ \langle WA\rangle$ not holds implies $\langle \mathit{true}\rangle\ M_1 \,||\, M_2\ \langle P\rangle$ not holds

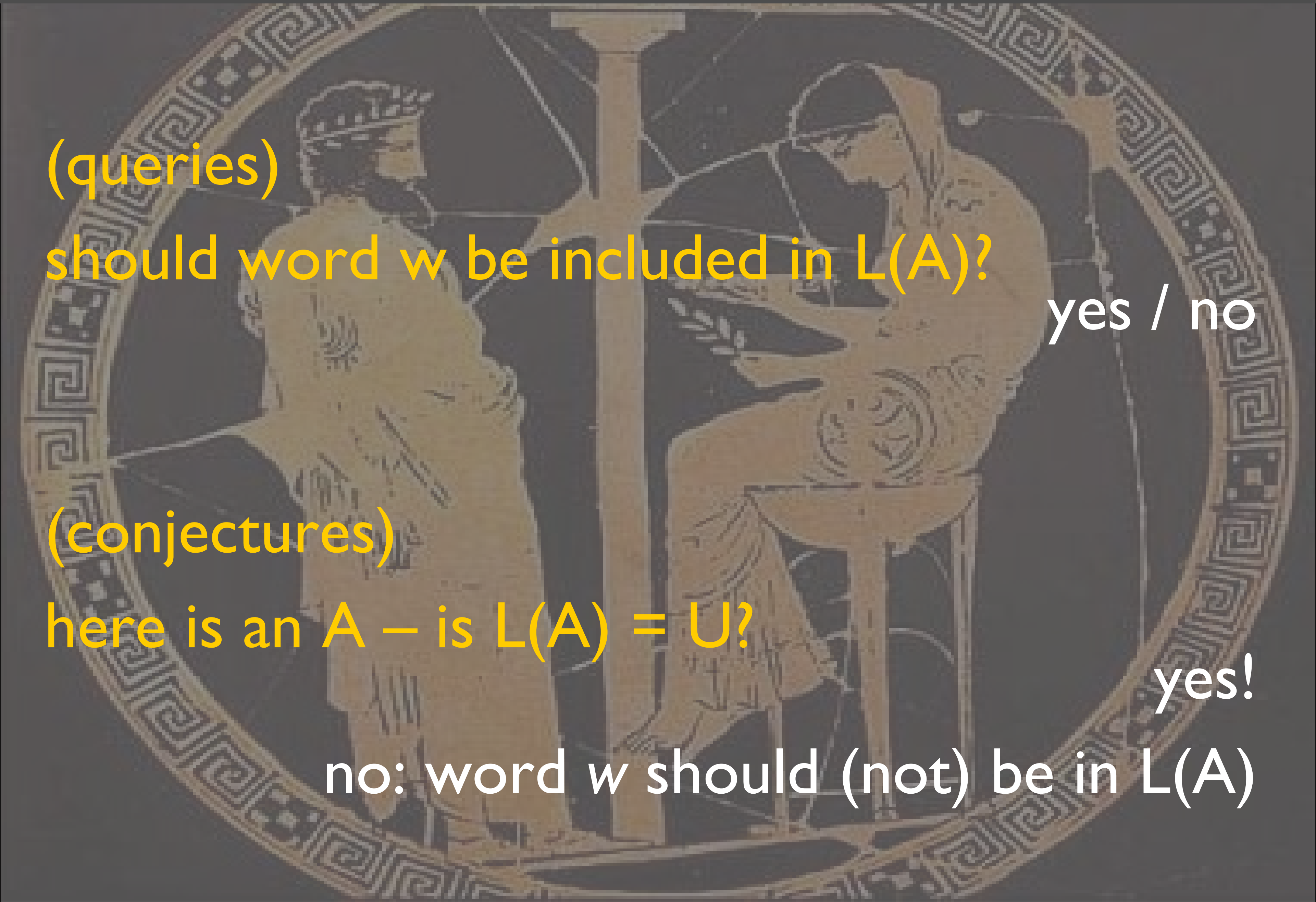
learning assumptions

iterative solution +
intermediate results

L* learns unknown regular language U (over alphabet Σ) and produces minimal DFA A such that L(A) = U

(L originally proposed by Angluin)*

L* learner the oracle



(queries)

should word w be included in L(A)?

yes / no

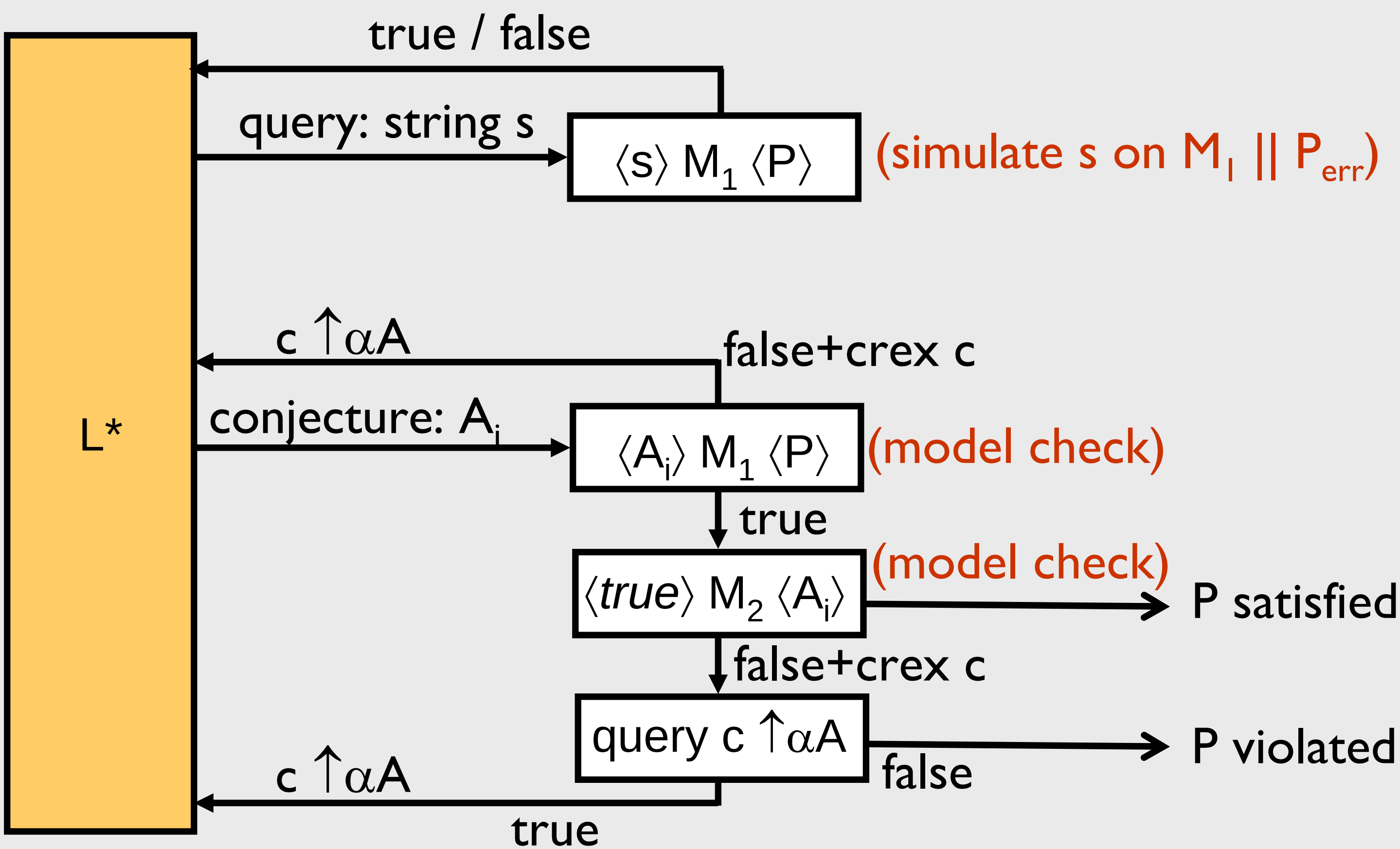
(conjectures)

here is an A – is L(A) = U?

yes!

no: word *w* should (not) be in L(A)

oracle for WA in assume-guarantee reasoning



⟨WA⟩ M_1 ⟨P⟩ holds (WA could be *false*)

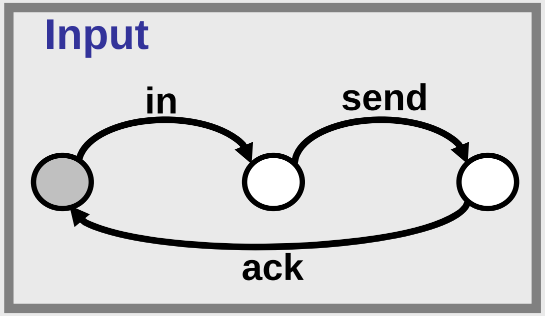
⟨*true*⟩ M_2 ⟨WA⟩ holds implies ⟨*true*⟩ M_1 || M_2 ⟨P⟩ holds

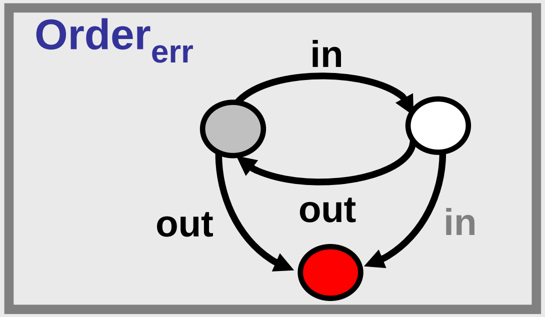
⟨*true*⟩ M_2 ⟨WA⟩ does not hold implies ⟨*true*⟩ M_1 || M_2 ⟨P⟩ does not hold

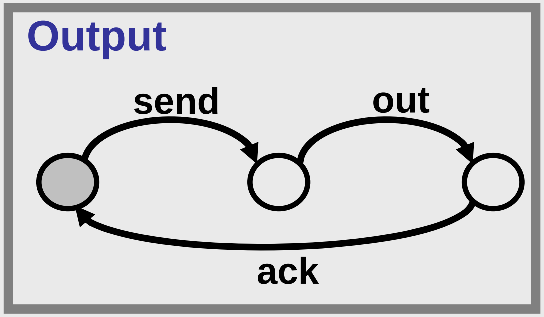
characteristics

- terminates with *minimal* automaton A for U
- generates DFA candidates A_i: $|A_1| < |A_2| < \ldots < |A|$
- produces at most n candidates, where $n = |A|$
- # queries: $O(kn^2 + n \log m)$,
  - m is size of largest counterexample, k is size of alphabet
- for assume-guarantee reasoning, may terminate early with a smaller assumption than the weakest

example





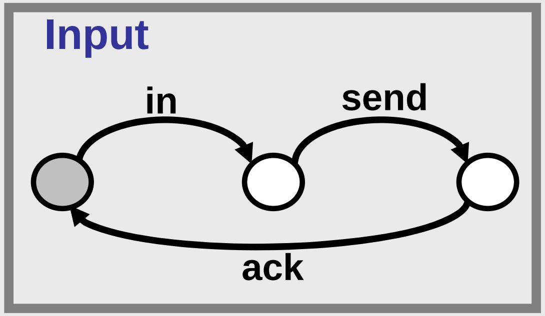


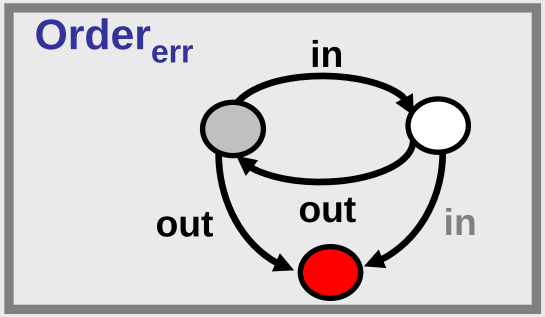
we check: $\langle true \rangle$ Input || Output $\langle Order \rangle$

M_1 = Input, M_2 = Output, P = Order

assumption alphabet: {send, out, ack}

queries





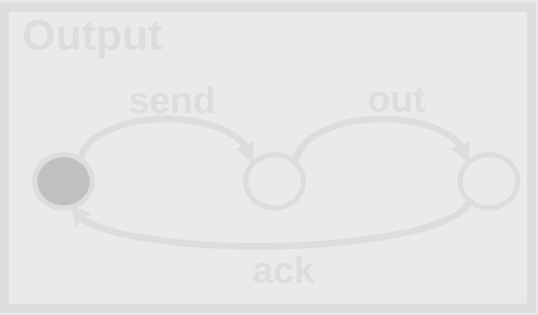


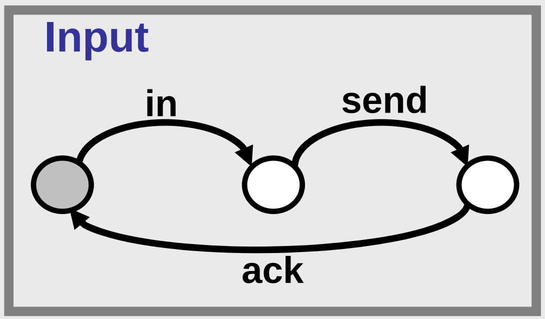
	Table T	***E*** λ
S	λ	true
	out	false
S $\cdot \Sigma$	ack	true
	out	false
	send	true
	out, ack	false
	out, out	false
	out, send	false

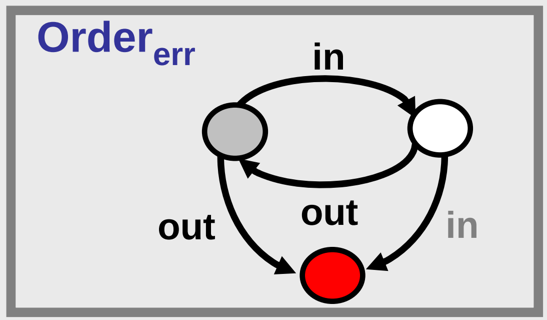
closed (adds to *S*)

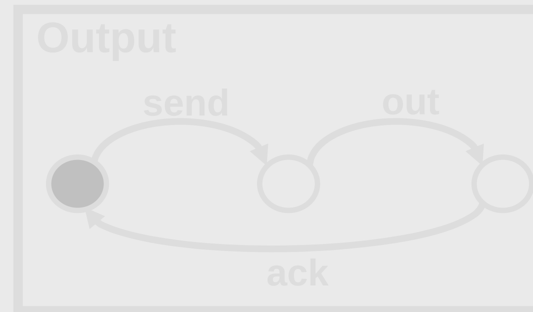
consistent (adds to *E*)

S = set of prefixes
E = set of suffixes

candidate construction





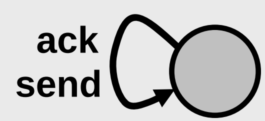


		E
	Table T	λ
S	λ	true
	out	false
S · Σ	ack	true
	out	false
	send	true
	out, ack	false
	out, out	false
	out, send	false

2 states – error state omitted

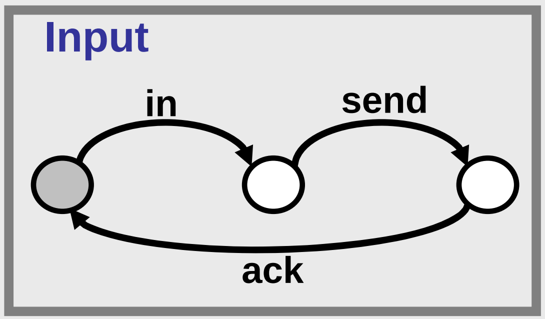
S = set of prefixes
E = set of suffixes

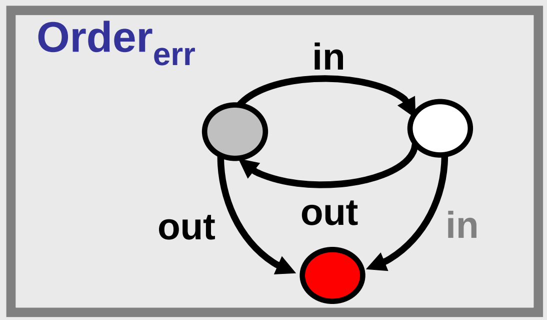
Assumption A_1

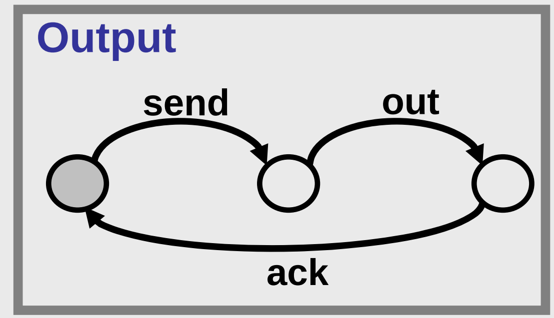


counterexamples add to S

conjectures







A_1:

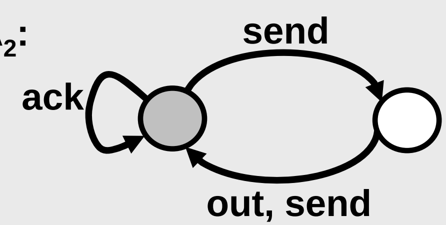
Oracle 1:
⟨A_1⟩ Input ⟨Order⟩

Counterexample:
c = ⟨in,send,ack,in⟩

Return to L*:
c↑ Σ = ⟨send,ack⟩

Queries

A_2:



Oracle 1:
⟨A_2⟩ Input ⟨Order⟩
True

Oracle 2:
⟨true⟩ Output ⟨A_2⟩
True

property Order holds
on Input || Output

end of part 1

please ask LOTS of questions!