

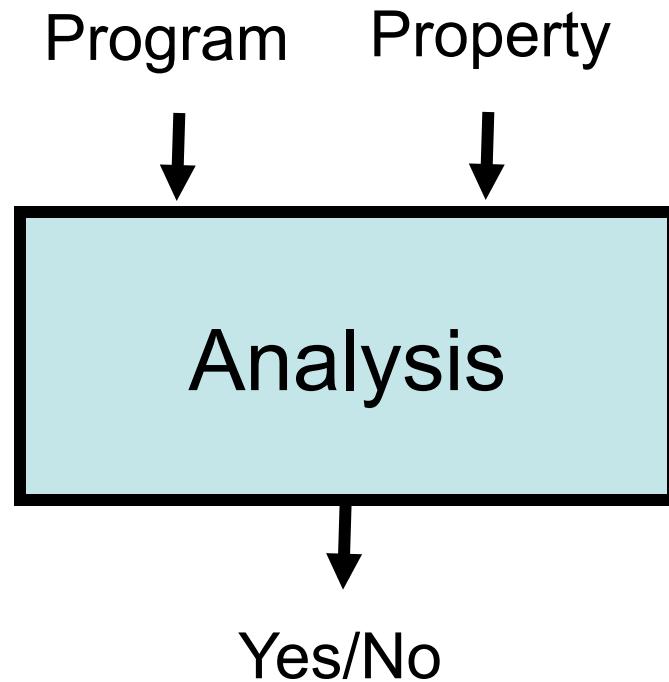
# From Qualitative to Quantitative Theories of Software

---

Tom Henzinger  
IST Austria

# Qualitative Software Theories

---



# Qualitative Software Theories

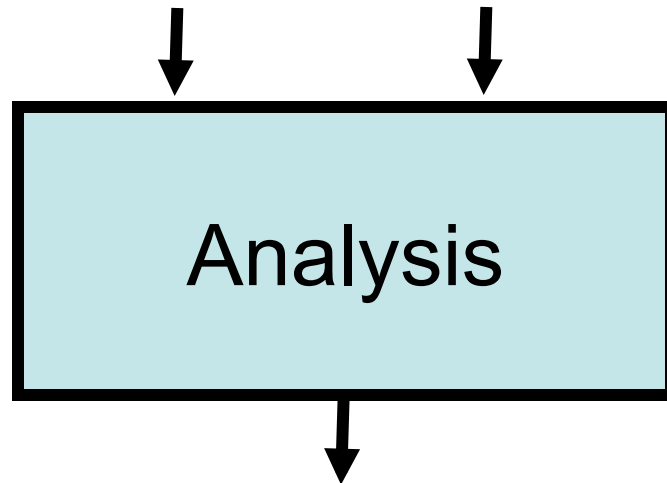
---

Kripke  
Structure

Program

Property

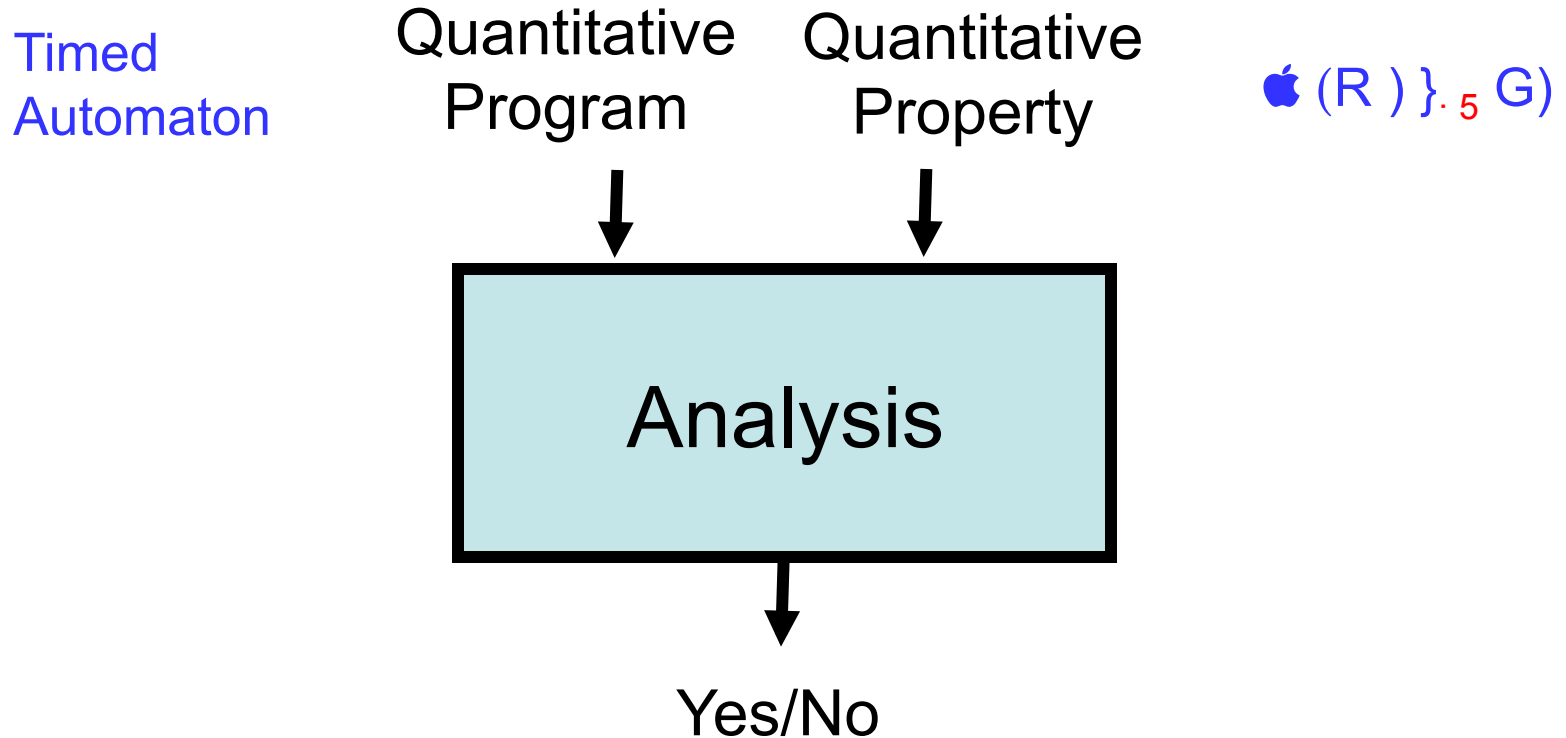
$\{ (R) \} G$



Yes/No

# Qualitative Software Theories

---



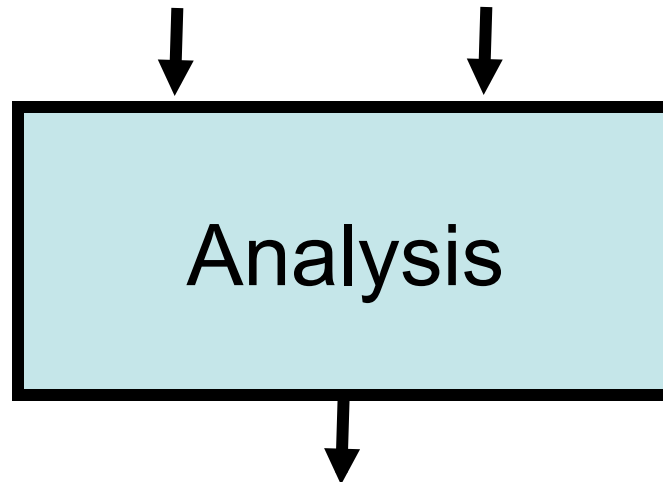
# Qualitative Software Theories

Markov  
Process

Quantitative  
Program

Quantitative  
Property

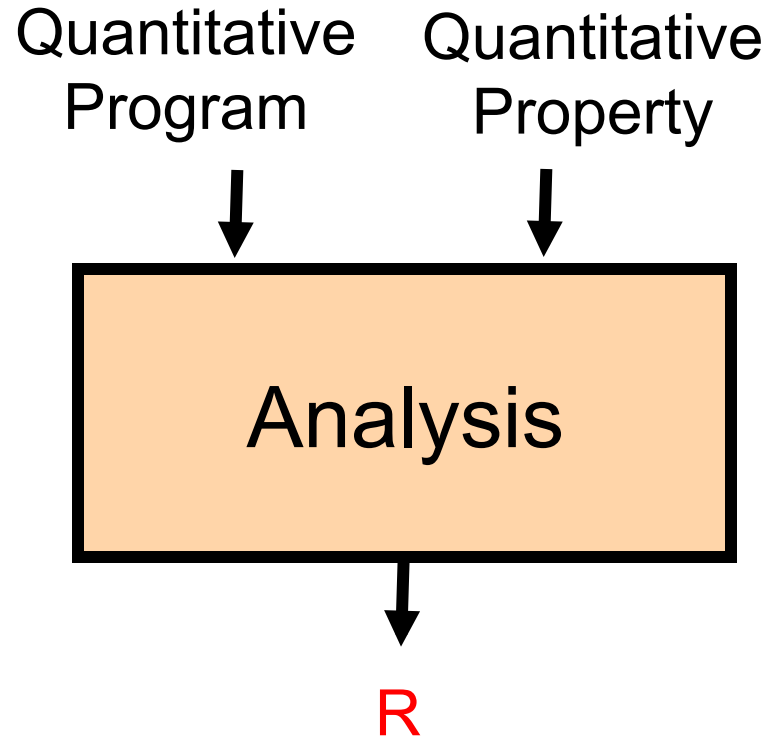
$8 \text{ } \text{Pr}(\{$   
 $G) , 0.5)$



Yes/No

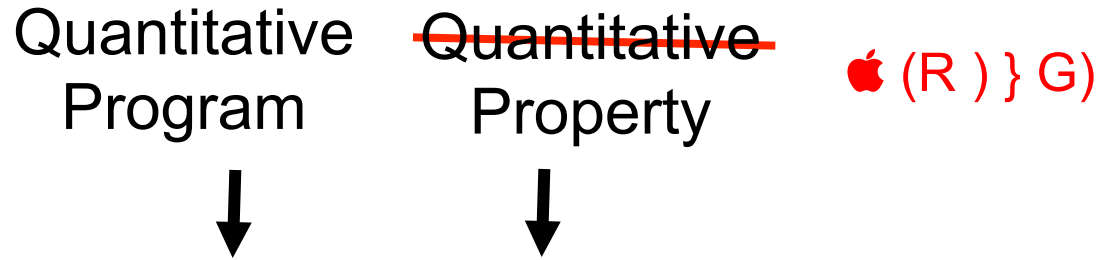
# Quantitative Software Theories

---



- measure of “fit” between program and property
- could involve cost, quality, performance, etc.

# Quantitative Software Theories



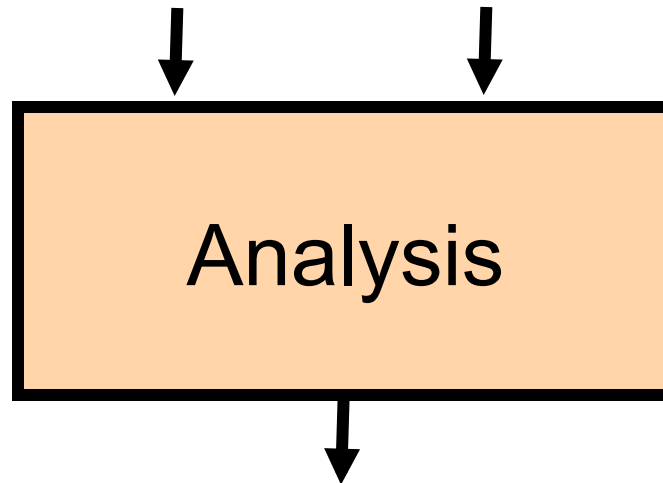
R

The less time between R and G, the better.

- measure of “fit” between program and property
- could involve cost, quality, performance, etc.

# Quantitative Software Theories

~~Quantitative~~ Program    ~~Quantitative~~ Property     $\{ (R) \} G$



The fewer  
“unnecessary” grants  $G$ ,  
the better.

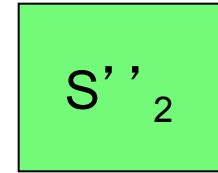
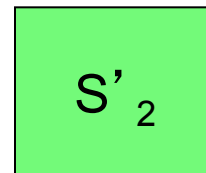
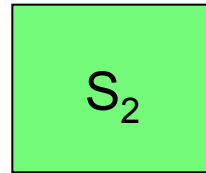
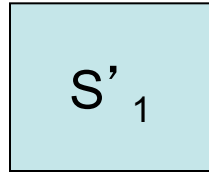
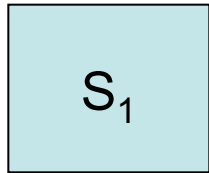
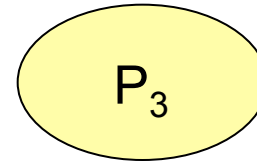
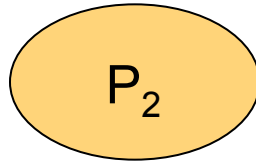
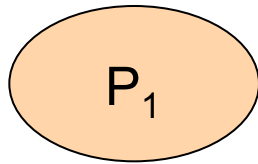
$R$

-measure of “fit” between program and property  
-could involve cost, quality, performance, etc.

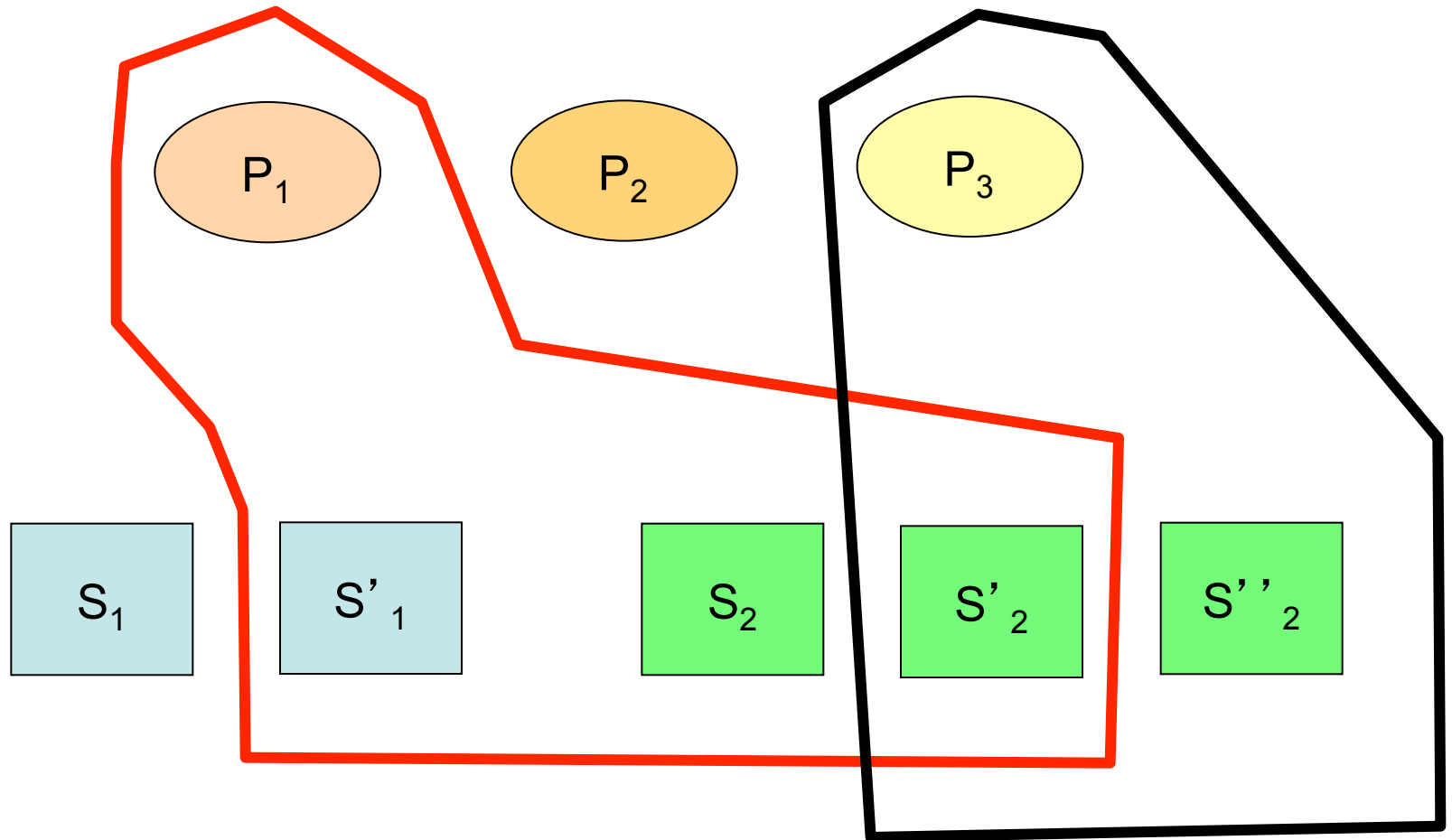


# Qualitative Software Theories

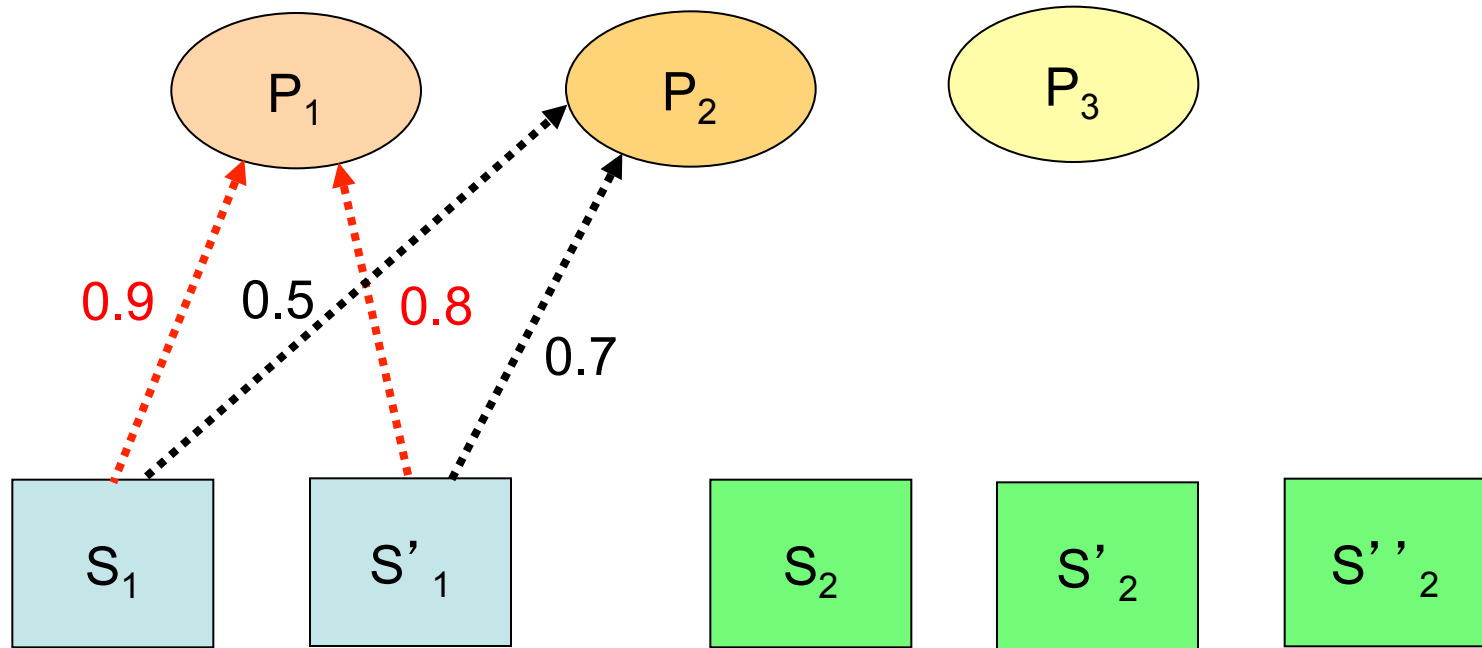
---



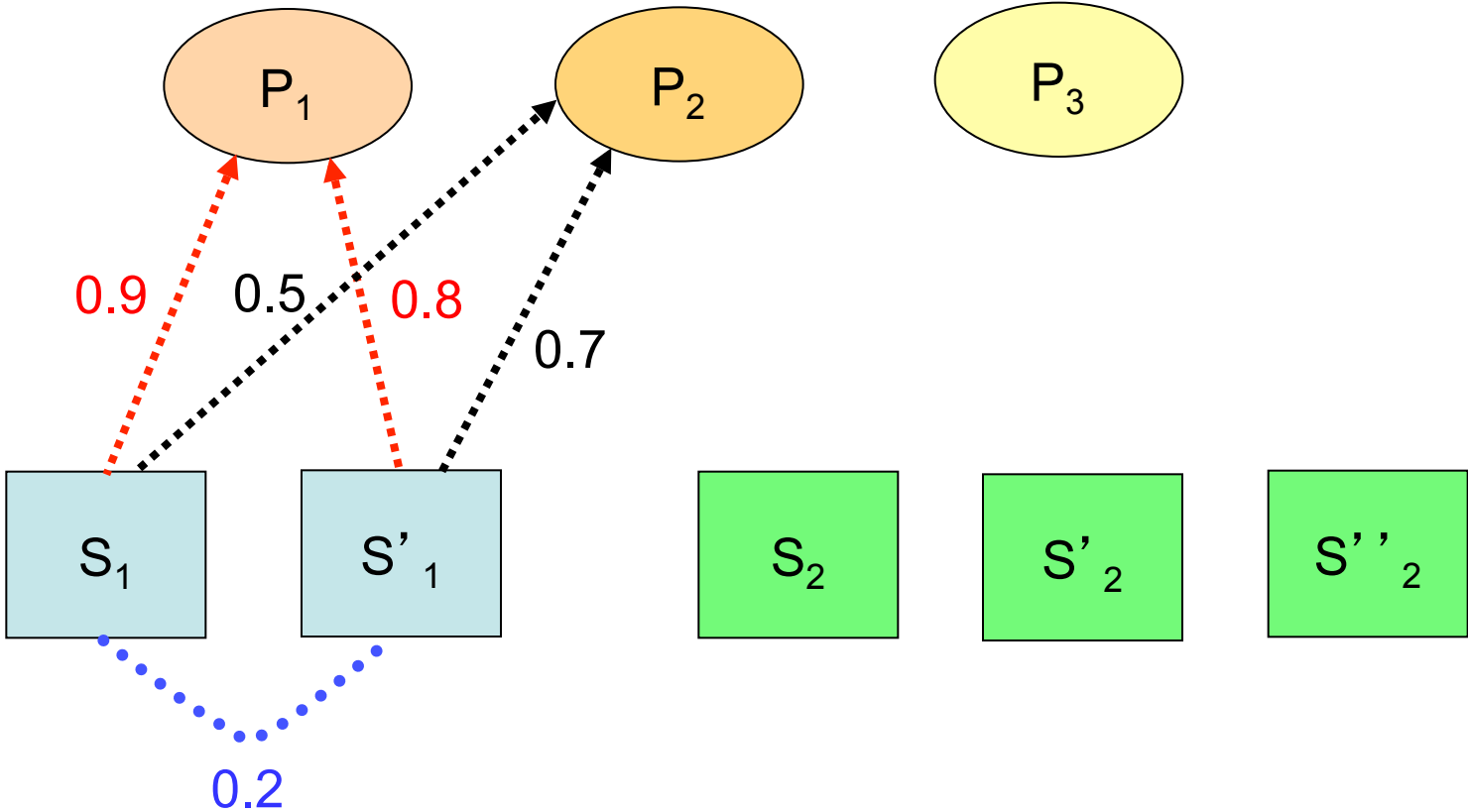
# Qualitative Software Theories



# Quantitative Software Theories



# Quantitative Software Theories



# Quantitative Software Models

---

- Q1     Assign values to program behaviors
- Q2     Assign values to programs/properties
- Q3     Assign values to pairs of programs/properties

# Q1 Assign Values To Program Behaviors

---

## a. Probabilities

# Q1 Assign Values To Program Behaviors

---

a. Probabilities

b. Resource use

worst case (sup) vs. average case (limavg) vs. accumulative (sum)  
(e.g., response time, power consumption)

# Q1 Assign Values To Program Behaviors

---

a. Probabilities

b. Resource use

worst case (sup) vs. average case (limavg) vs. accumulative (sum)  
(e.g., response time, power consumption)

c. Quality measures

discounting vs. long-run averaging




# Q1 Example: Reliability Values

---

a: ok

b: fail

Discounted value ( $0 < d < 1$ ):  a

aaaaaaaaaaa...

1

aaaaaaaaab...

$1 - d^8$

aaab...

$1 - d^3$

b...


0

# Q1 Example: Reliability Values

---

a: ok

b: fail

Discounted value ( $0 < d < 1$ ):  a

aaaaaaaaaaa... 1

aaaaaaaaaab...  $1 - d^8$

aaab...  $1 - d^3$

b... 0

Long-run average value:  $\limavg$  a

aaaaaaaaaaa... 1

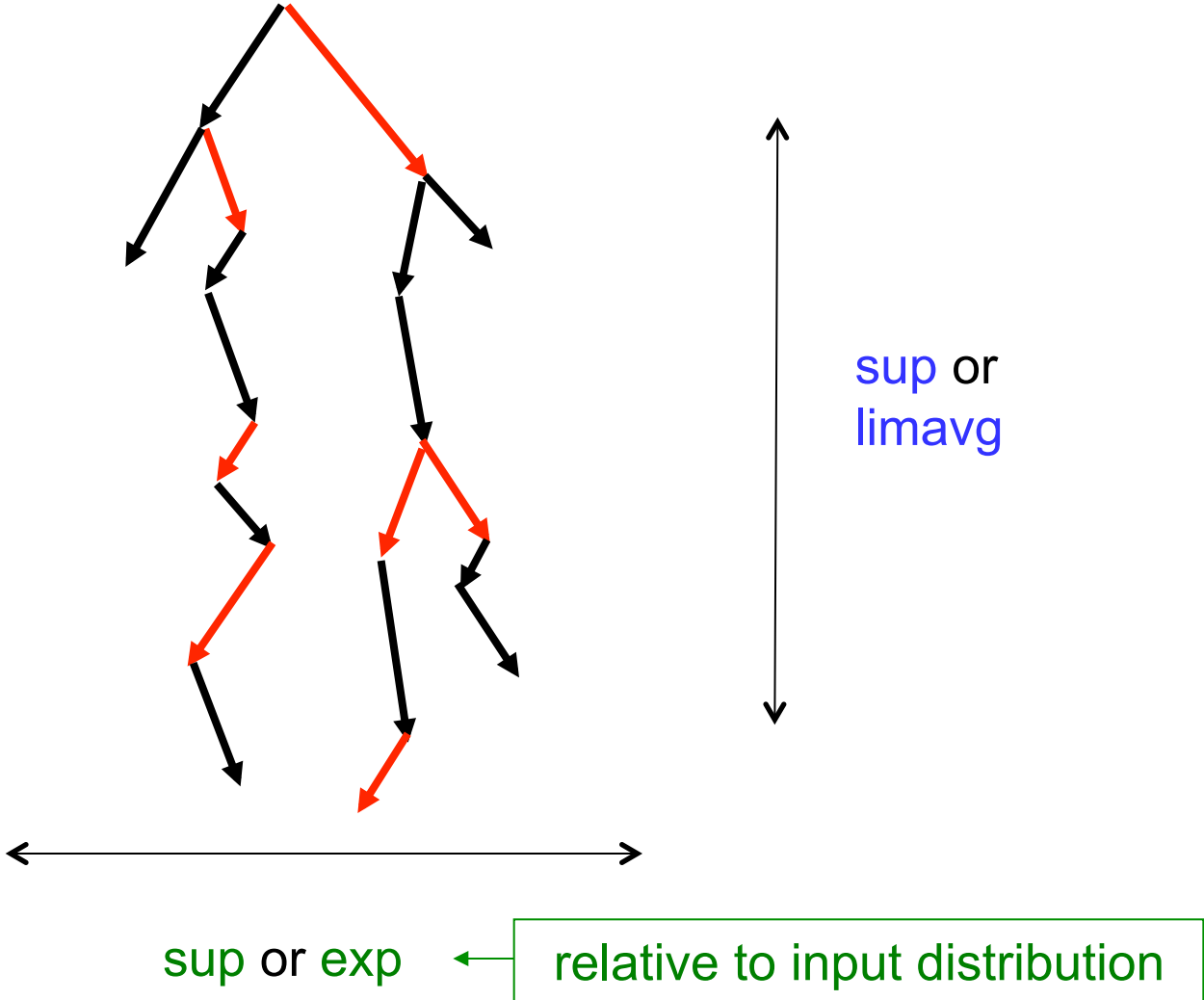
abaabaaab... 1

aaabaaabaaab...  $3/4$

babbabbba... 0

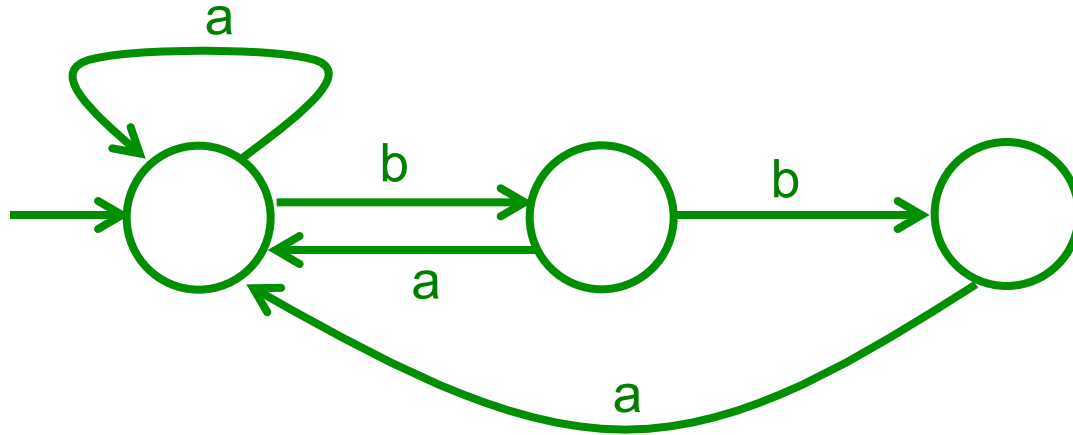
aaaaaabb... 0

# Q2 Assign Values To Programs

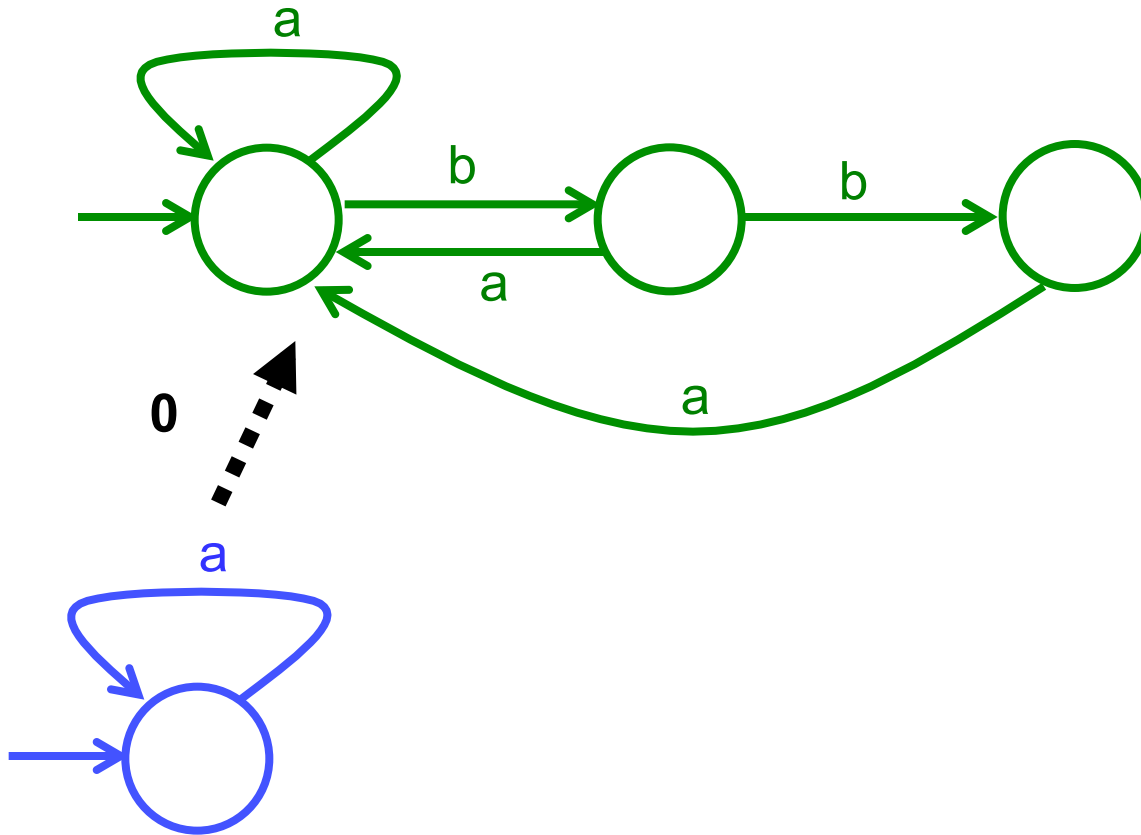


# Q3 Assign Distances To Programs

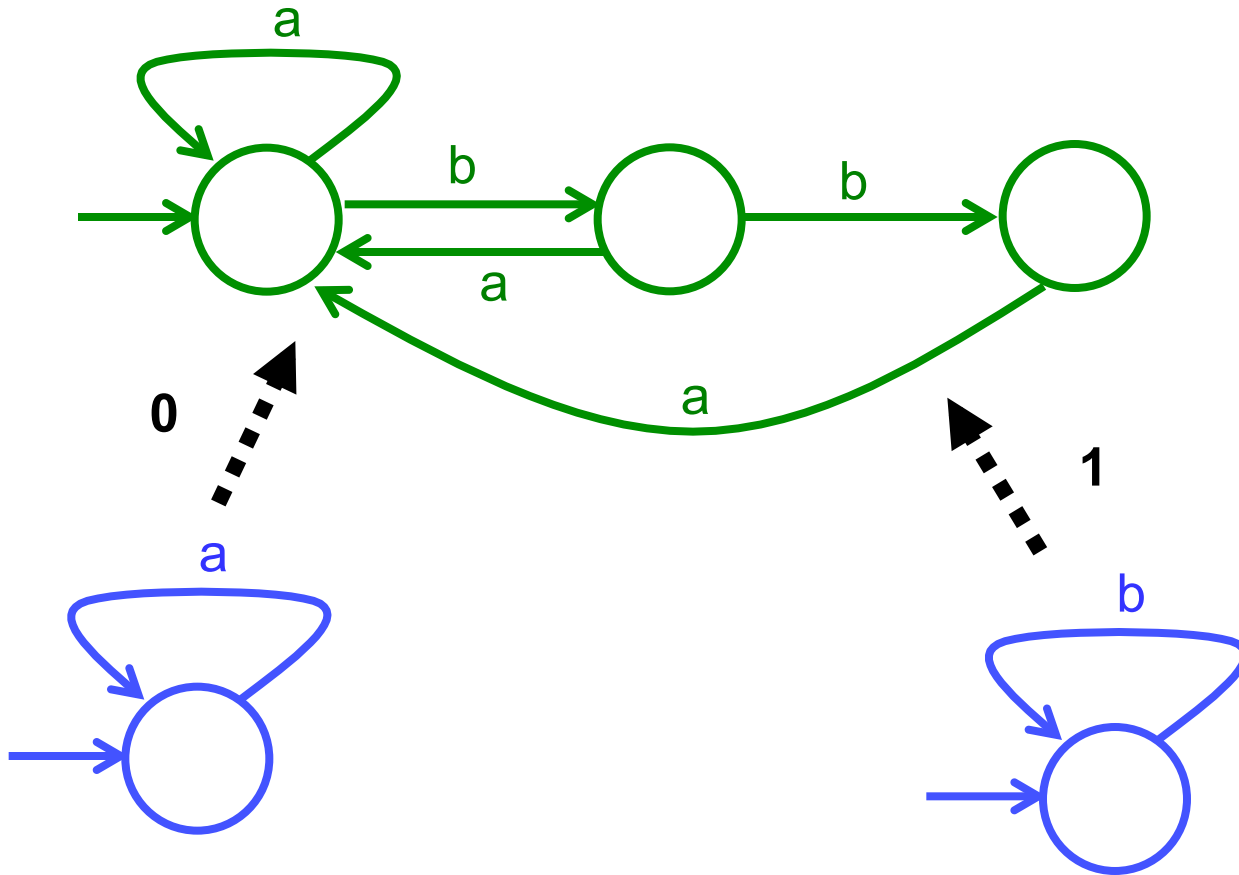
---



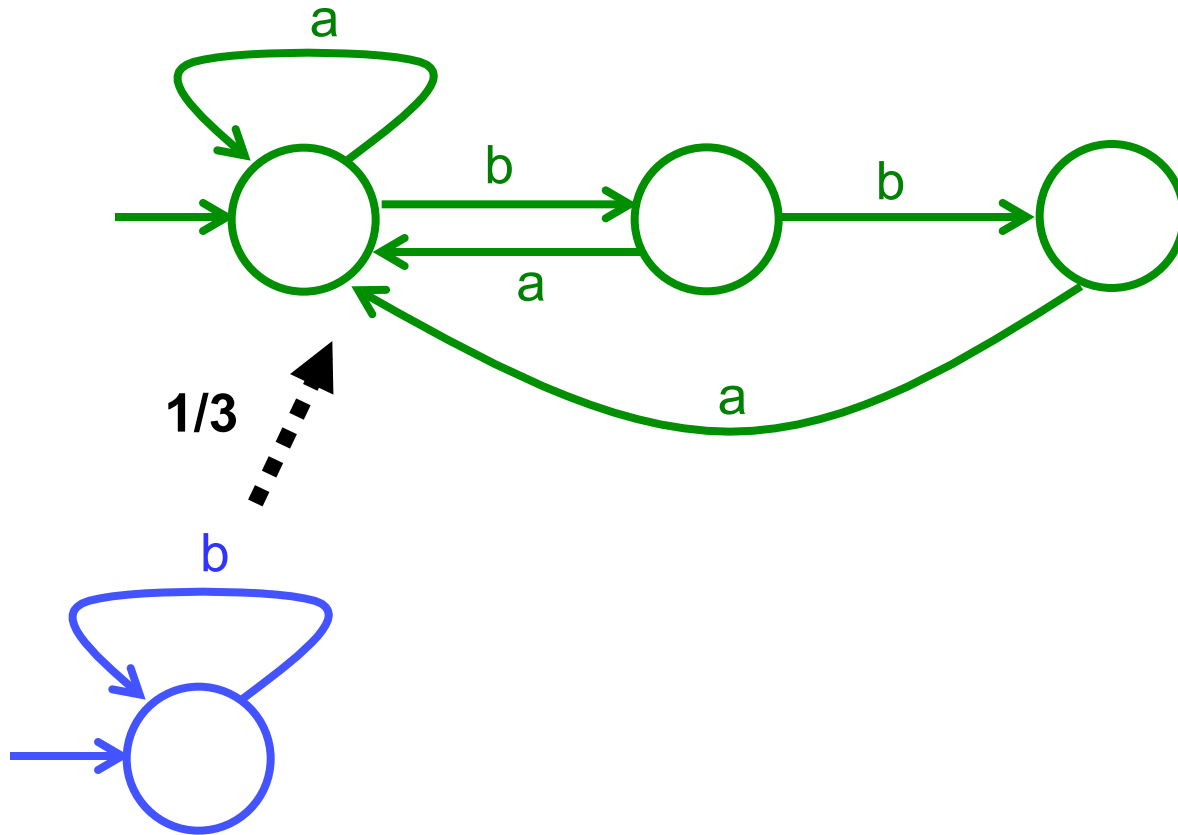
# Q3 Example: Correctness Distance



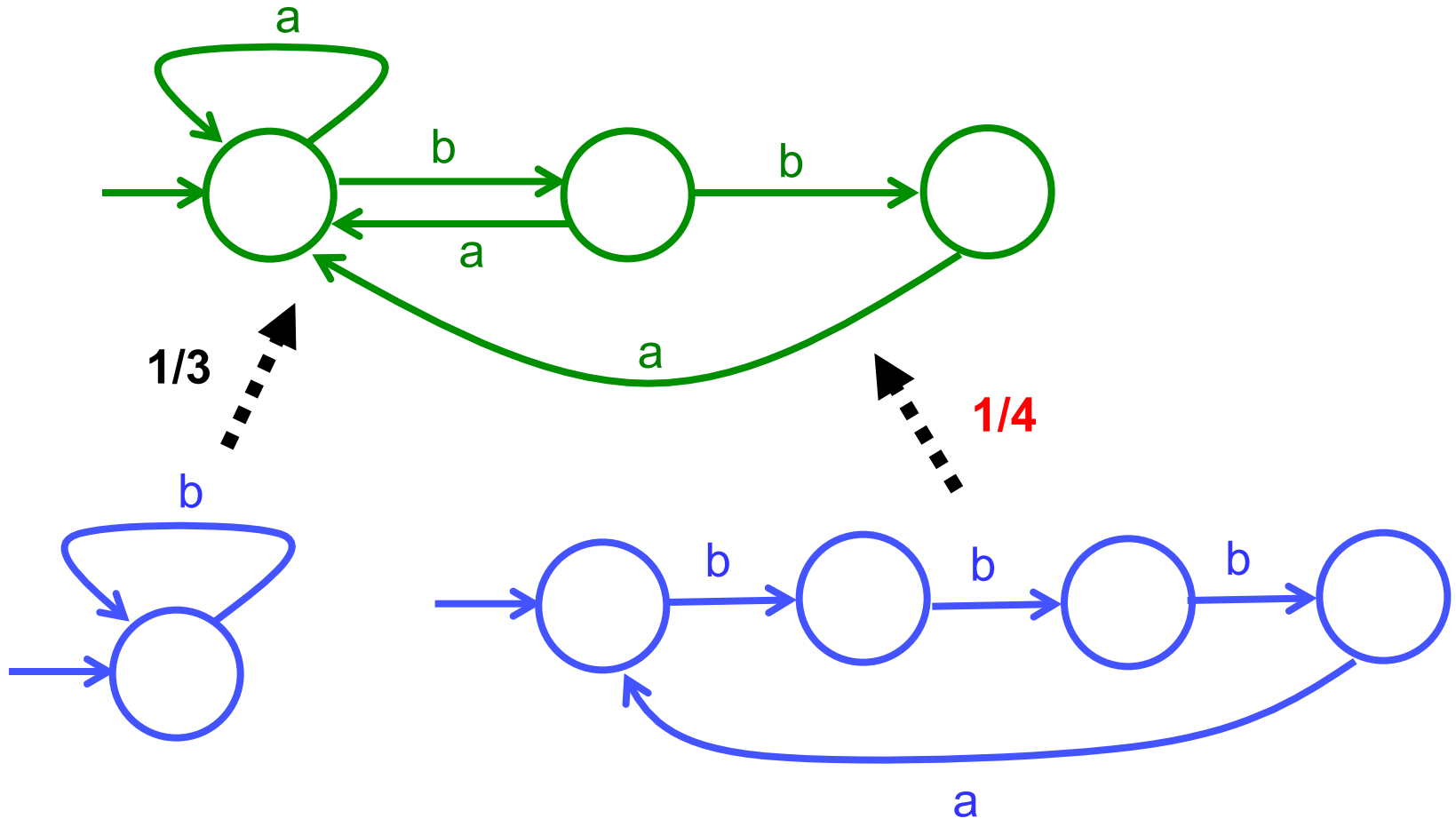
# Q3 Example: Correctness Distance



# Q3 Example: Correctness Distance

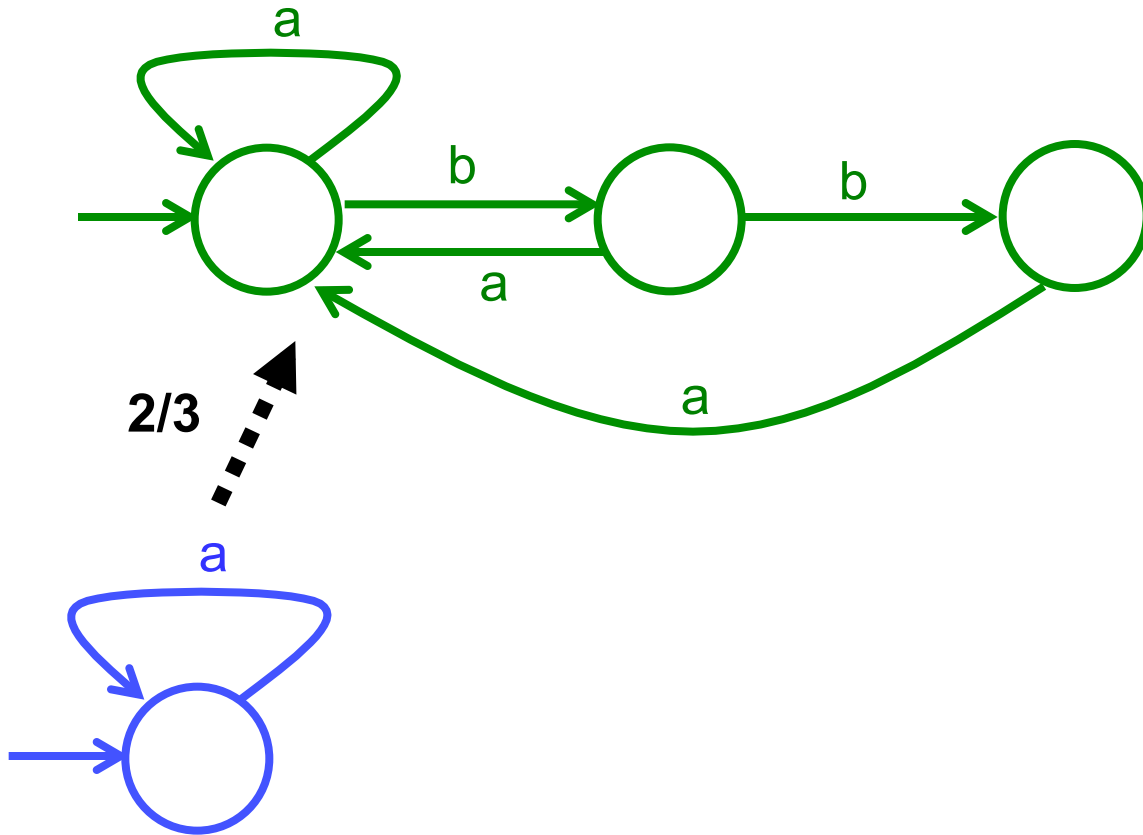


# Q3 Example: Correctness Distance

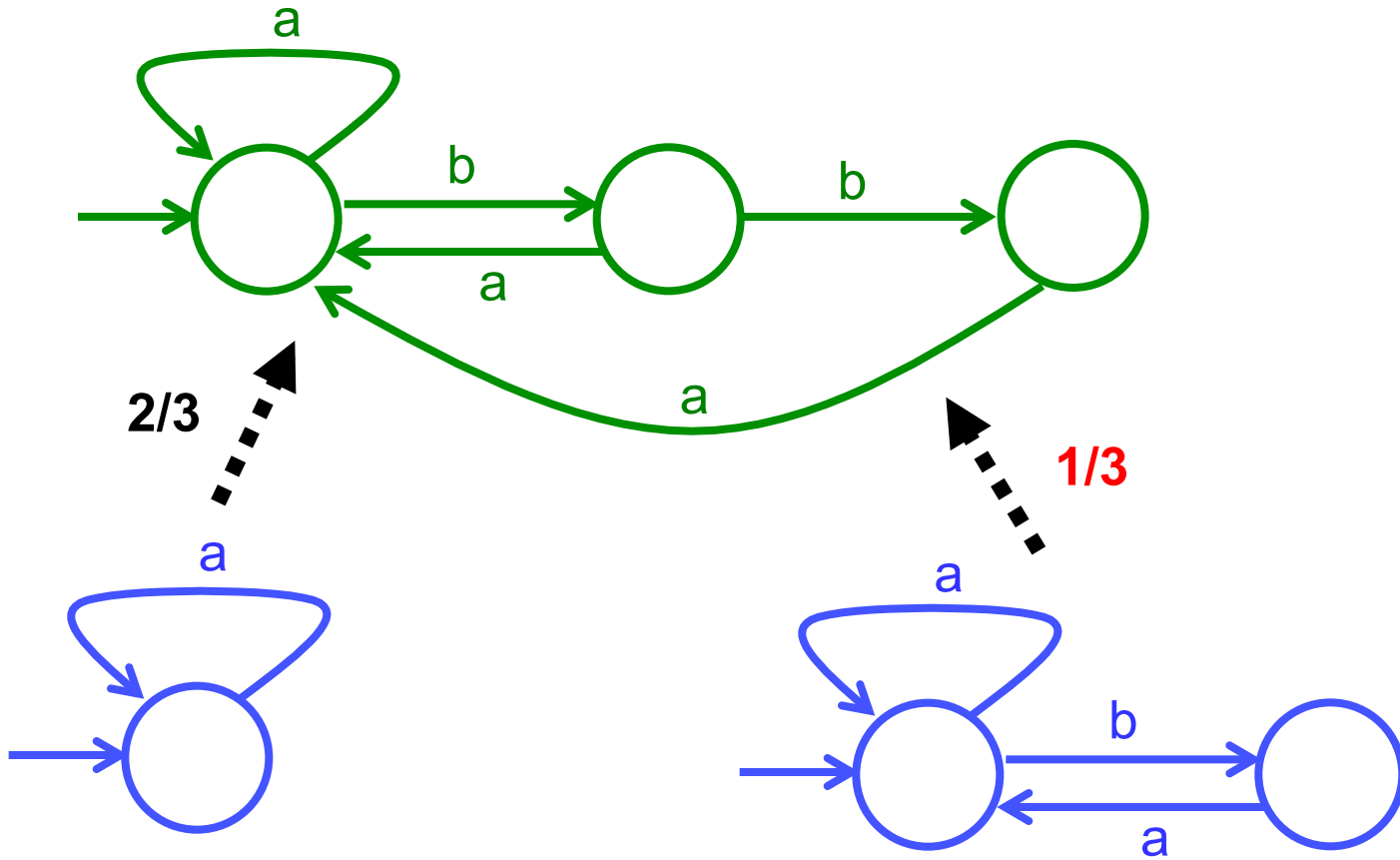




# Q3 Example: Robustness Distance



# Q3 Example: Robustness Distance



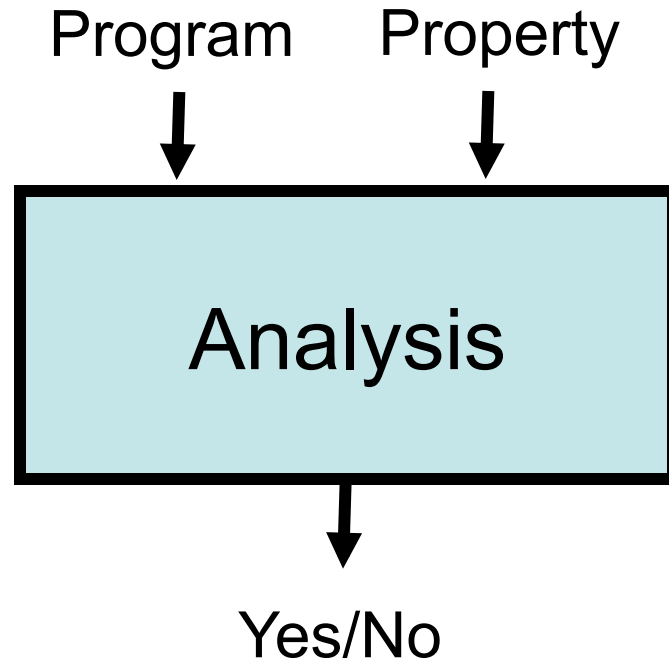
# References

---

- 1 Simulation and bisimulation distances  
[CONCUR 2010 Cerny et al.]
- 2 Quantitative languages  
[CSL 2008, LICS 2009, CSL 2011 Boker et al.]
- 3 Quantitative synthesis  
[CAV 2009, CAV 2010, CAV 2011 Cerny et al.]

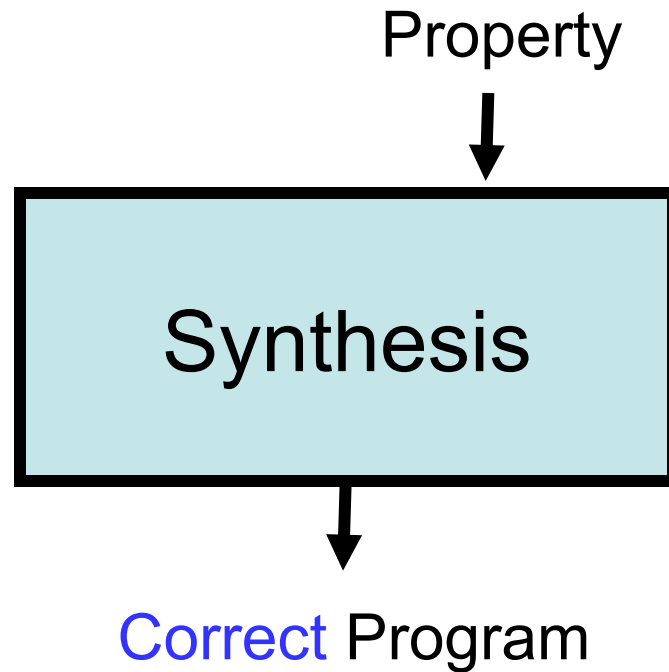
# Qualitative Software Theories

---



# Qualitative Software Theories

---



# Qualitative Software Theories

---

$\omega$ -Regular Automaton



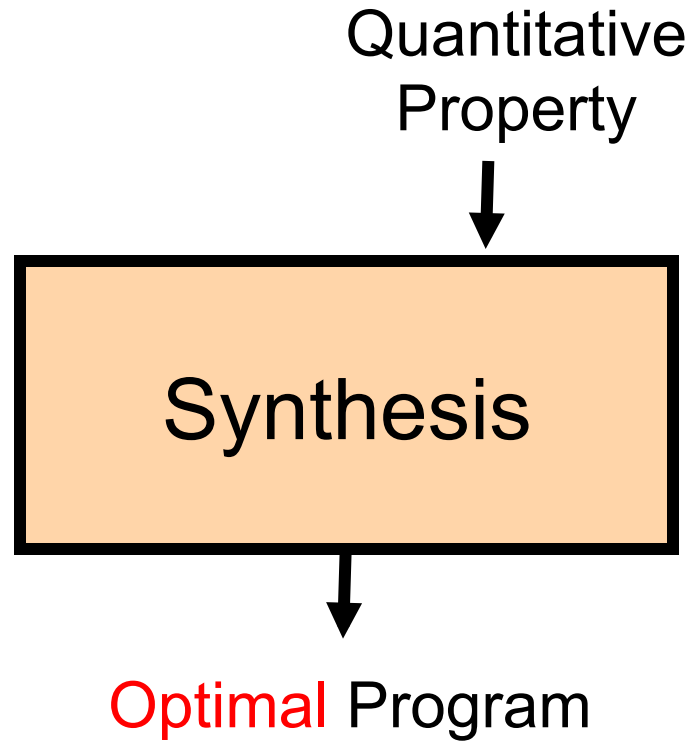
Graph Game with  
 $\omega$ -Regular Objective



Correct Program =  
Winning Strategy

# Quantitative Synthesis

---



# Quantitative Synthesis

---

Weighted  
Automaton



Graph Game with  
Quantitative Objective

worst case



Optimal Program =  
Optimal Strategy



# Quantitative Synthesis

---

Weighted  
Automaton



Stochastic Graph Game  
with Quantitative Objective

avg case



Optimal Program =  
Optimal Strategy

# Games for Quantitative Synthesis

---

## 1 Optimizing Resource Use / Performance

- costs refer to resource use  
(e.g., power consumption, context switch)
- optimize peak or accumulative or average resource use
- formalized using **sup** or **sum** or **limavg objectives**
- synthesize schedules, routes, lock placement

## Fine grained vs. coarse grained locks:

-fine grained locks allow more interleavings,  
and therefore cause less waiting of threads

-coarse grained locks cause fewer context switches,  
which are expensive

Process 1:

loop

access x;

access y

end.

Process 2:

loop

access x;

access y

end.

## Fine grained vs. coarse grained locks:

-fine grained locks allow more interleavings,  
and therefore cause less waiting of threads

-coarse grained locks cause fewer context switches,  
which are expensive

Process 1:

loop

access x;

access y

end.

Process 2:

loop

access x;

access y

end.

## Fine grained vs. coarse grained locks:

-fine grained locks allow more interleavings,  
and therefore cause less waiting of threads

-coarse grained locks cause fewer context switches,  
which are expensive

Process 1:

loop

access x;

access y

end.

Process 2:

loop

access x;

access y

end.

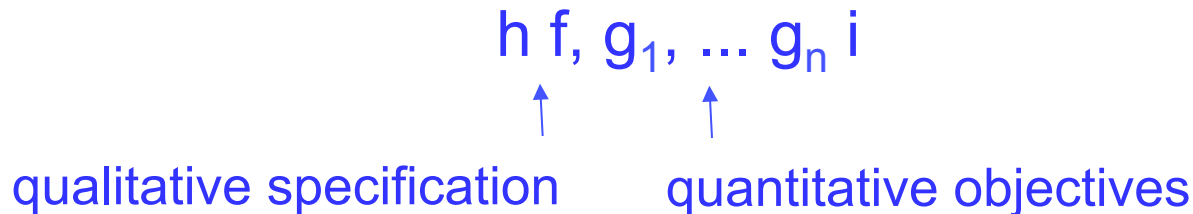
# Games for Quantitative Synthesis

---

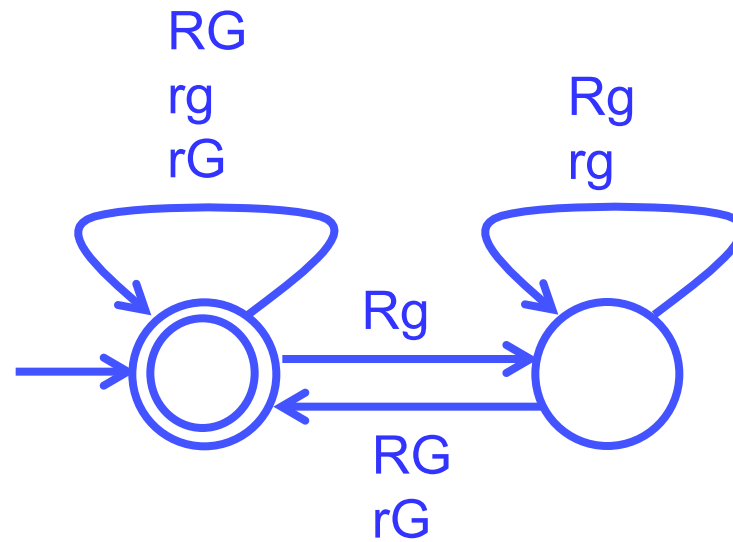
1 Optimizing Resource Use / Performance

2 Preference between Different Programs

- qualitative property, but some programs preferred over others
- can be formalized using **lexicographic objectives**

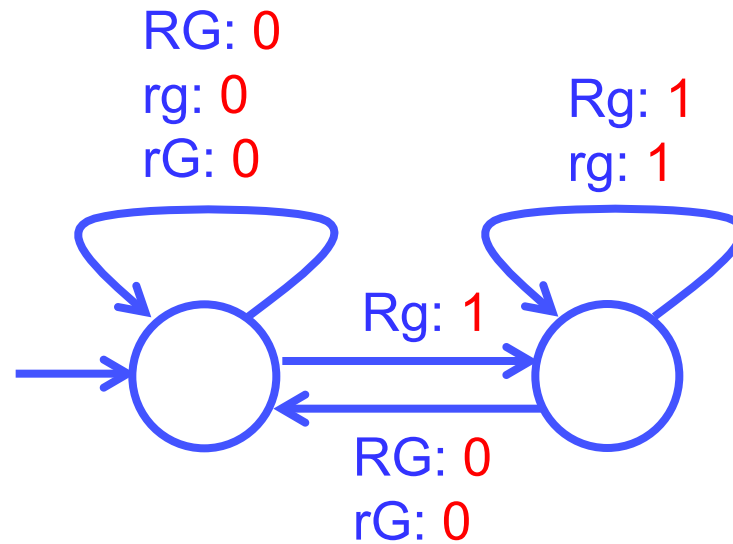


# Request-Grant Buchi Automaton



Every request is followed by a grant.

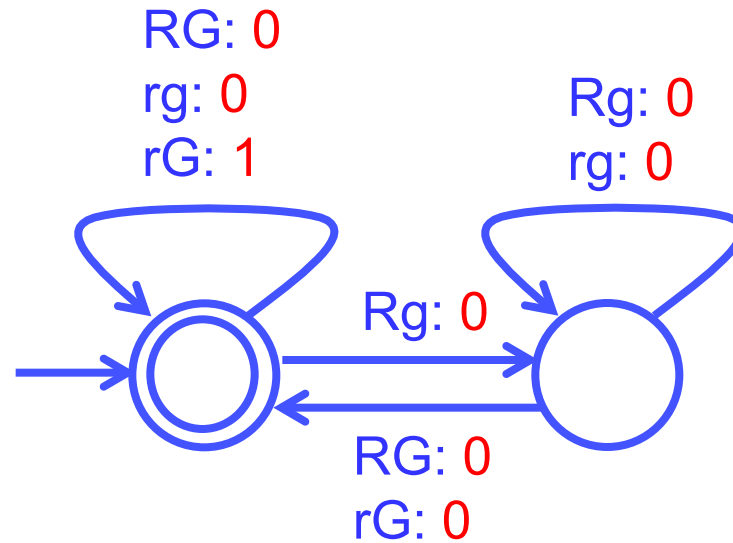
# Request-Grant limavg Automaton 1



Following a request, all steps until the next grant are penalized.



# Request-Grant limavg Automaton 2



All unnecessary grants are penalized.

# Conclusions

---

- We need to move from boolean program correctness criteria to quantitative program preference metrics.

# Conclusions

---

- We need to move from boolean program correctness criteria to quantitative program preference metrics.
- “Quantitative” is more than “timed” and “probabilistic.”

# Conclusions

---

- We need to move from boolean program correctness criteria to quantitative program preference metrics.
- “Quantitative” is more than “timed” and “probabilistic.”
- Weighted automata over infinite words offer a quantitative specification language:

Limit average

Sum/

energy

Discounting

# Conclusions

---

- We need to move from boolean program correctness criteria to quantitative program preference metrics.

- “Quantitative” is more than “timed” and “probabilistic.”

- Weighted automata over infinite words offer a quantitative specification language:

Limit average

Sum/

energy

Discounting

- Games with quantitative objectives offer algorithmic solutions:

Quantitative synthesis

Simulation distances