Reasoning about the Robustness of Protocols

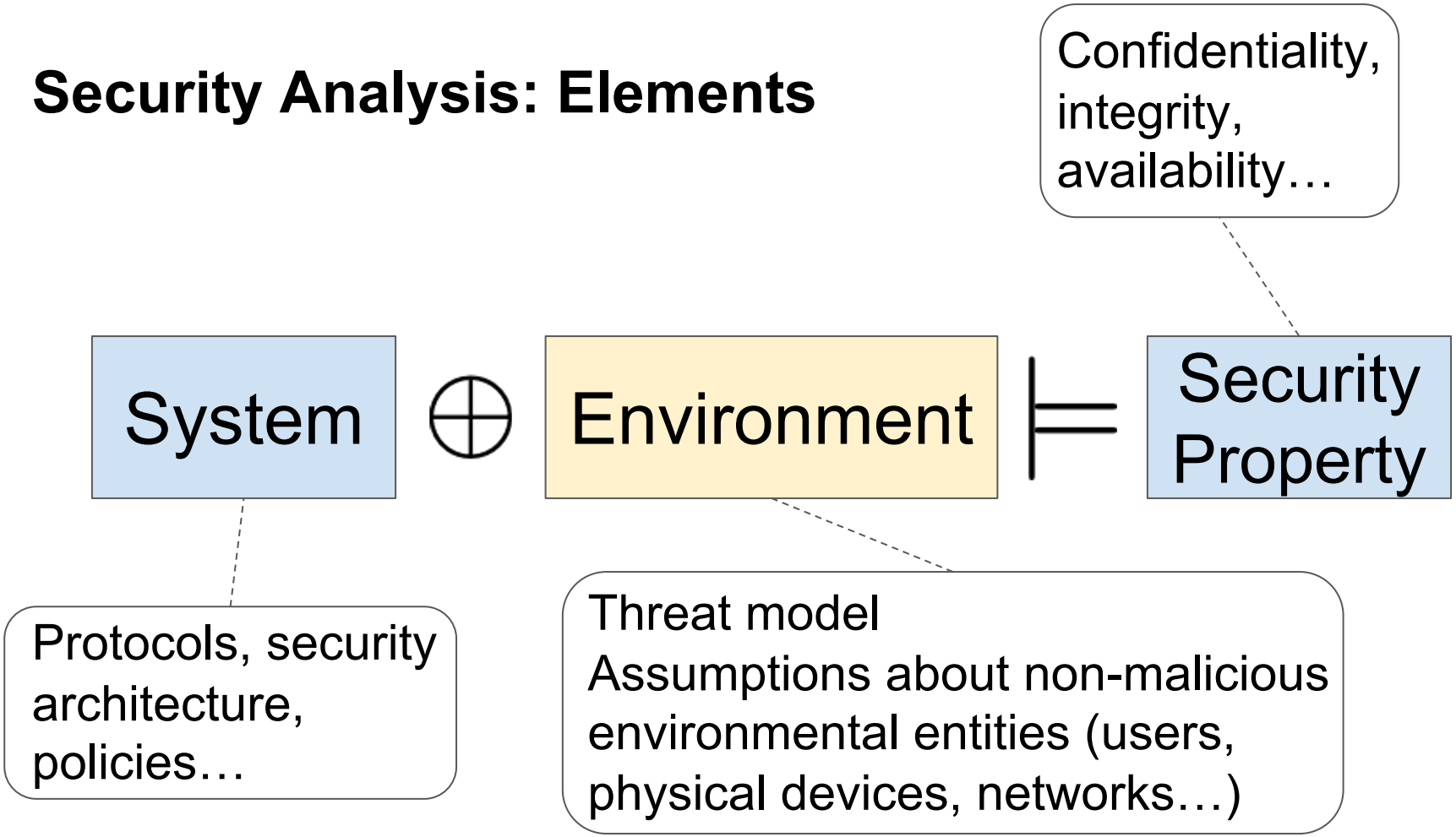
Eunsuk Kang

High Confidence Software & Systems Conference
May 9, 2023

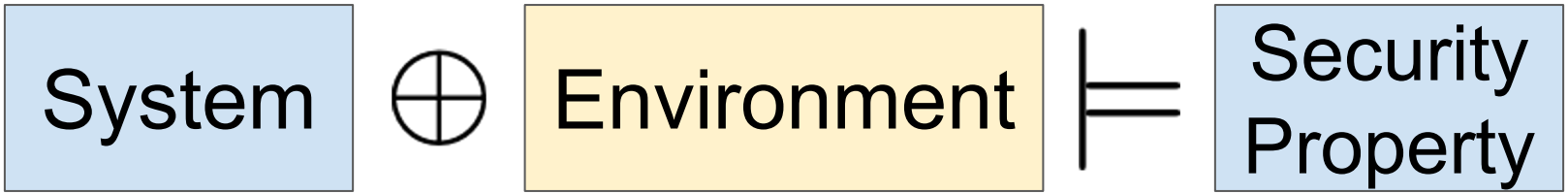
Carnegie Mellon University
CyLab
Security and Privacy Institute

S3D Software and Societal Systems Department

Security Analysis: Elements

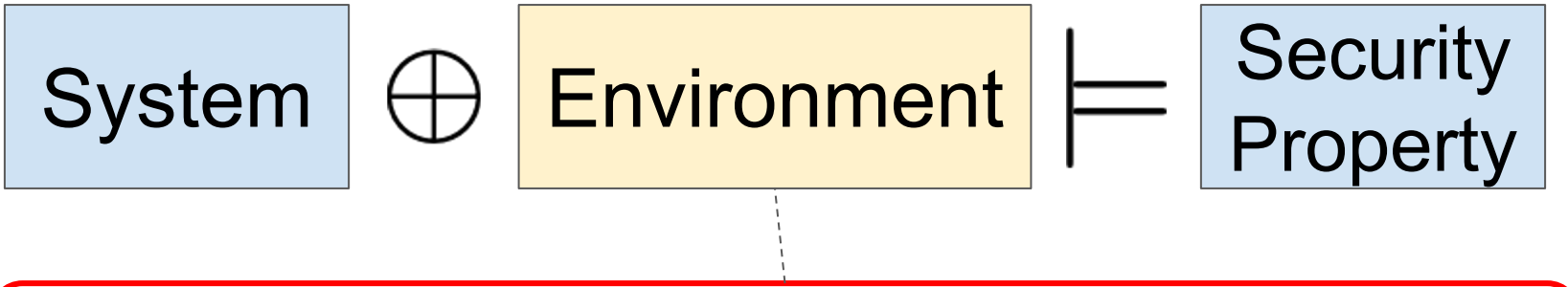


Security Analysis: Goal



Does the system, when deployed under the assumed environment, satisfy the property?

Security Analysis: Challenges



1. How do we know what assumptions we are making?
2. What if our assumptions turn out to be wrong/broken?

Example: E-Voting Attack (ES&S iVotronic)

FRANKFORT — A former Clay County precinct worker testified Friday that top election officers in the county taught her how to change people's choices on voting machines to steal votes in the May 2006 primary.

Voters walk away from the machine before pressing “confirm”
Election officials enter booth, press “back” & modify the vote

Example: E-Voting Attack

Assumed user behavior: Complete the voting process with “confirm”
But in practice, this assumption may sometimes fail to hold!

Alternative designs might mitigate this issue:
e.g., timeout after no response, require PIN after confirm

Problem

Once a system is deployed, its actual environment may deviate from the assumed one, possibly undermining the security property.

Can we design systems that are robust – providing security even under the presence of such deviations?

Can we provide tools to aid developers in this process?

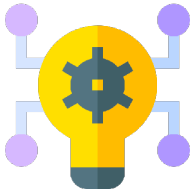
Robust-by-Design Systems

Specification

What does it mean for our system to be robust?

Analysis

How robust is our system?



Robustification

How do we improve its robustness?

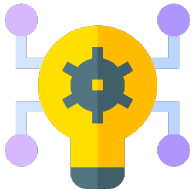
Robust-by-Design Systems

Specification

What does it mean for our system to be robust?

Analysis

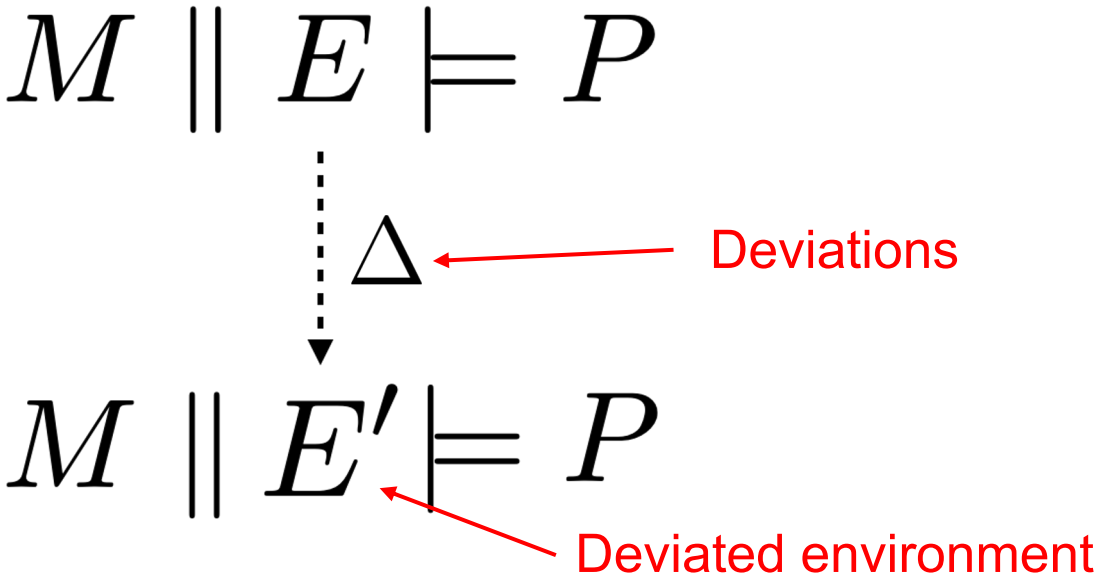
How robust is our system?



Robustification

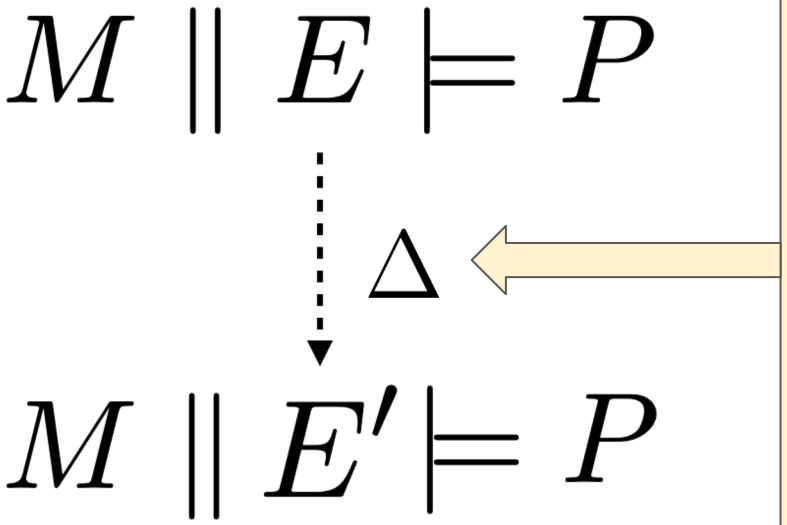
How do we improve its robustness?

Robustness: Formal Definition



System (M) is robust against a set of deviations (Δ) with respect to environment (E) and security property (P)

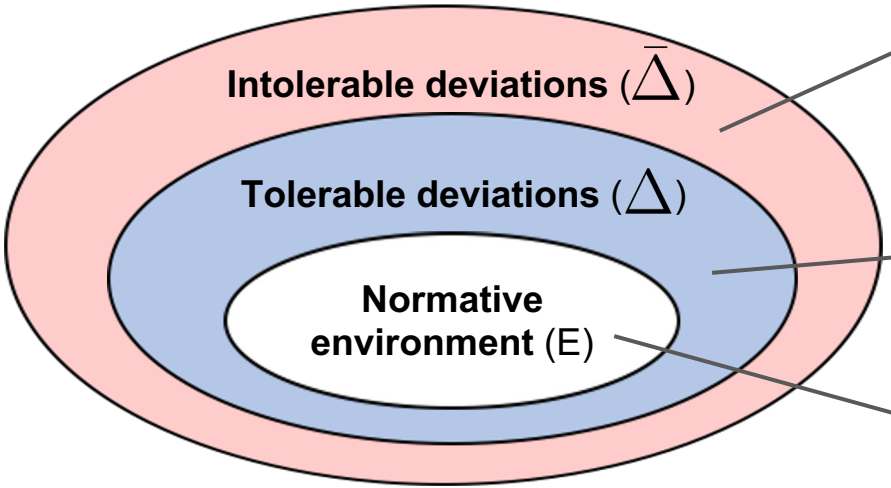
Robustness: Formal Definition



Examples:
User errors (e.g., omit a critical action)
Network failures
Sensor noise
Changes in attacker knowledge/capabilities
…

System (M) is robust against a set of deviations (Δ) with respect to environment (E) and security property (P)

Robustness: Another View



System may violate its security property under these

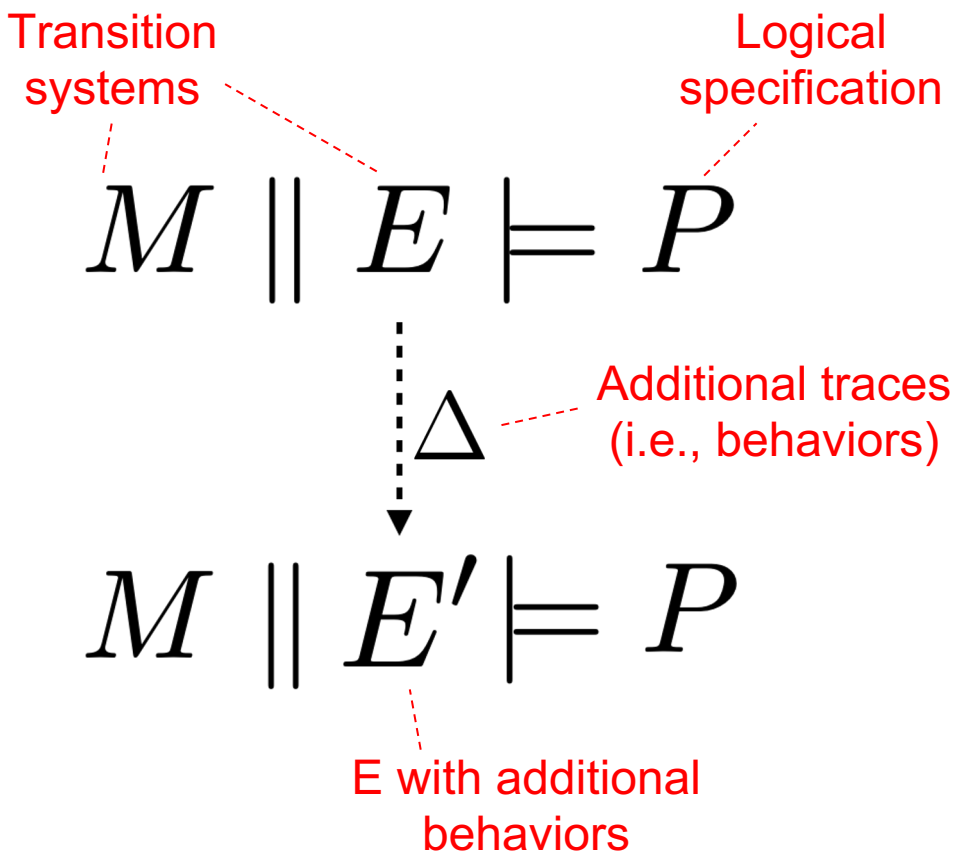
System can be securely deployed in these environments

Normal, expected environment

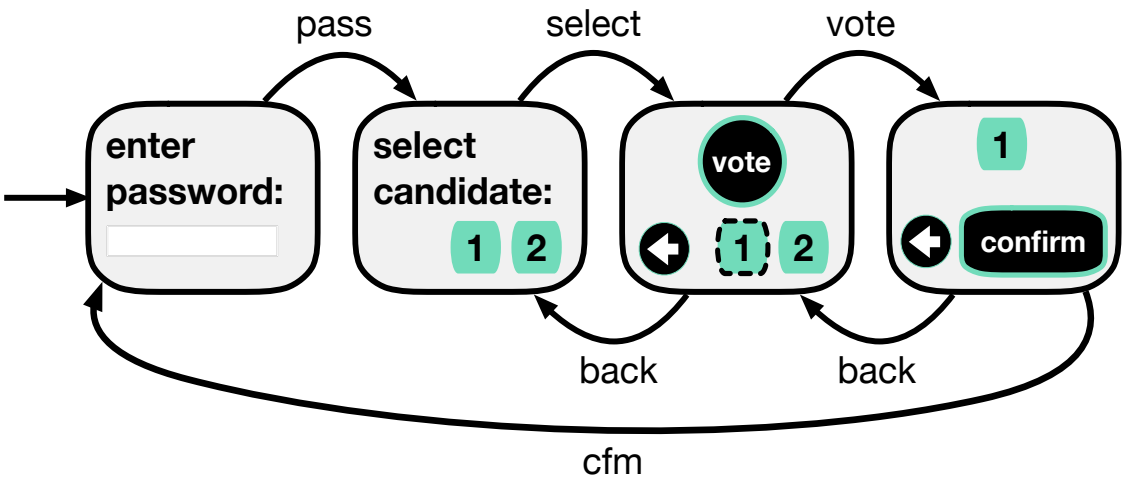
Overall robustness = maximal Δ set

Larger $\Delta \Rightarrow$ more robust system!

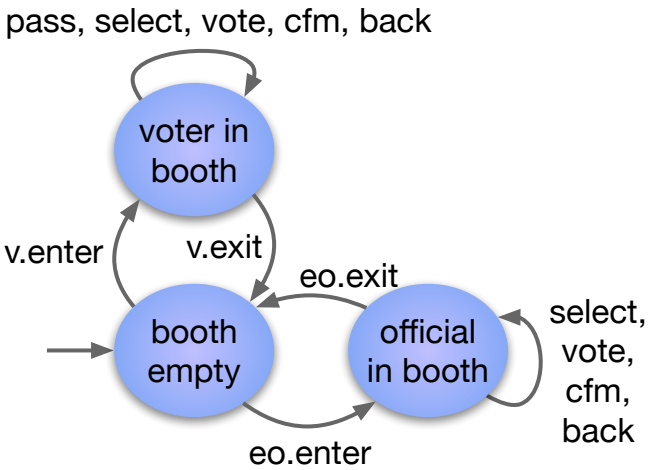
Automata-Theoretic Definition



Example: Voting Machine as Transition Systems



Voting Interface

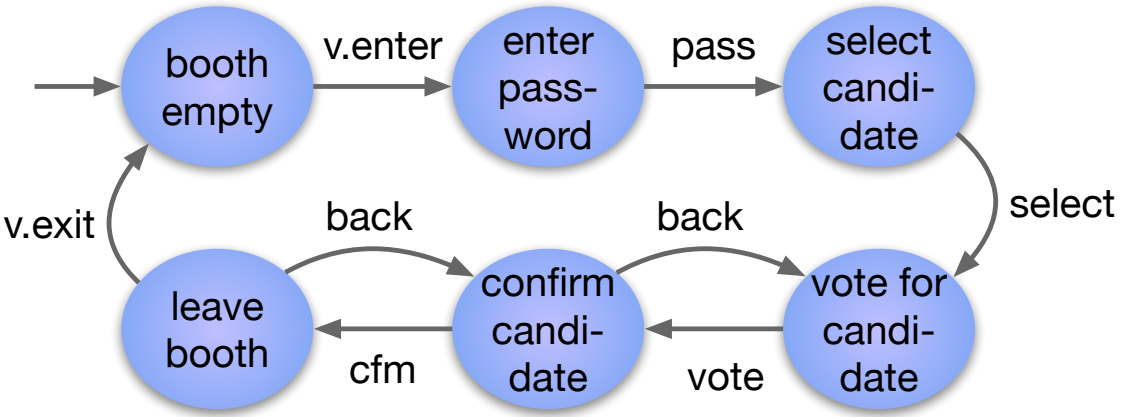


Booth

System (M) = Voting Interface || Booth

Voting Machine as Transition Systems

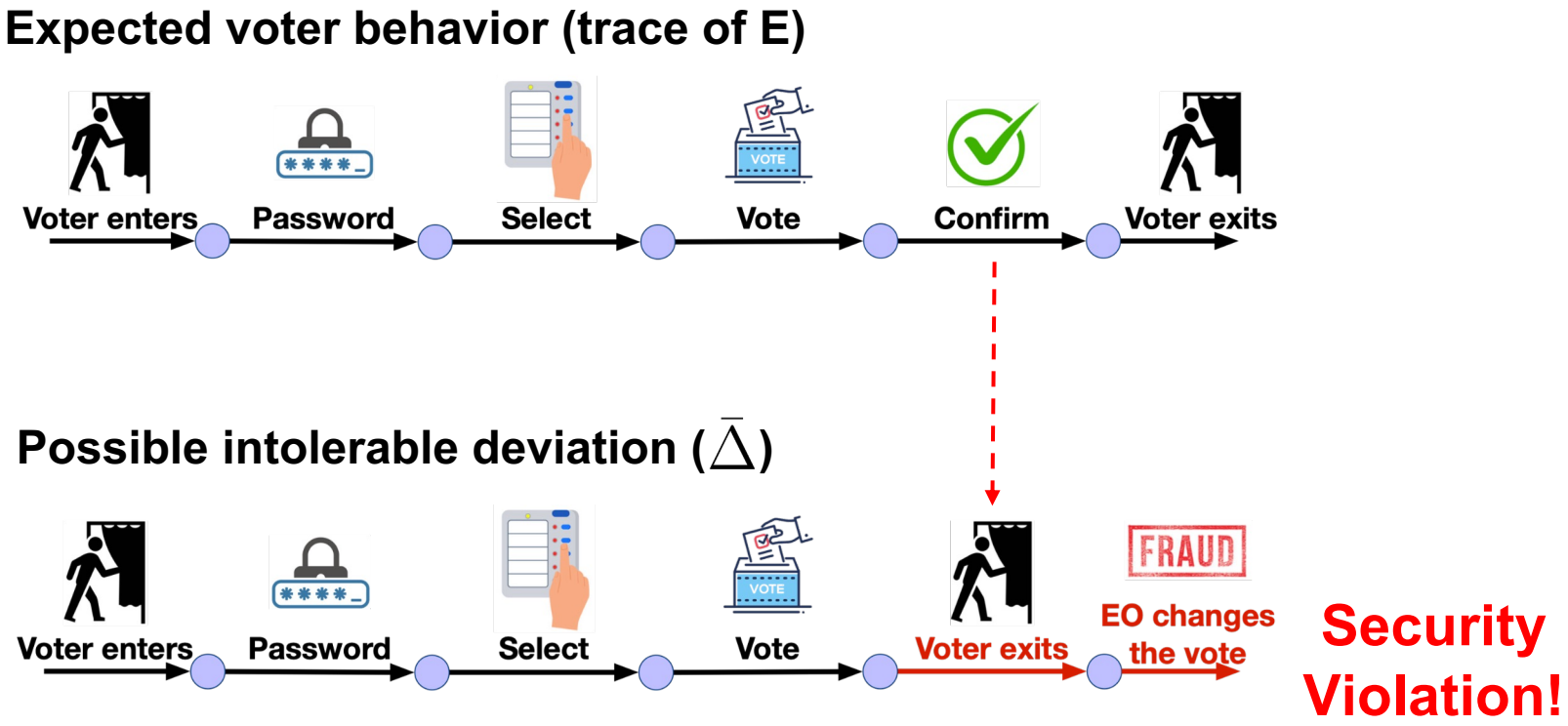
Environment (E)
i.e., expected voter behavior



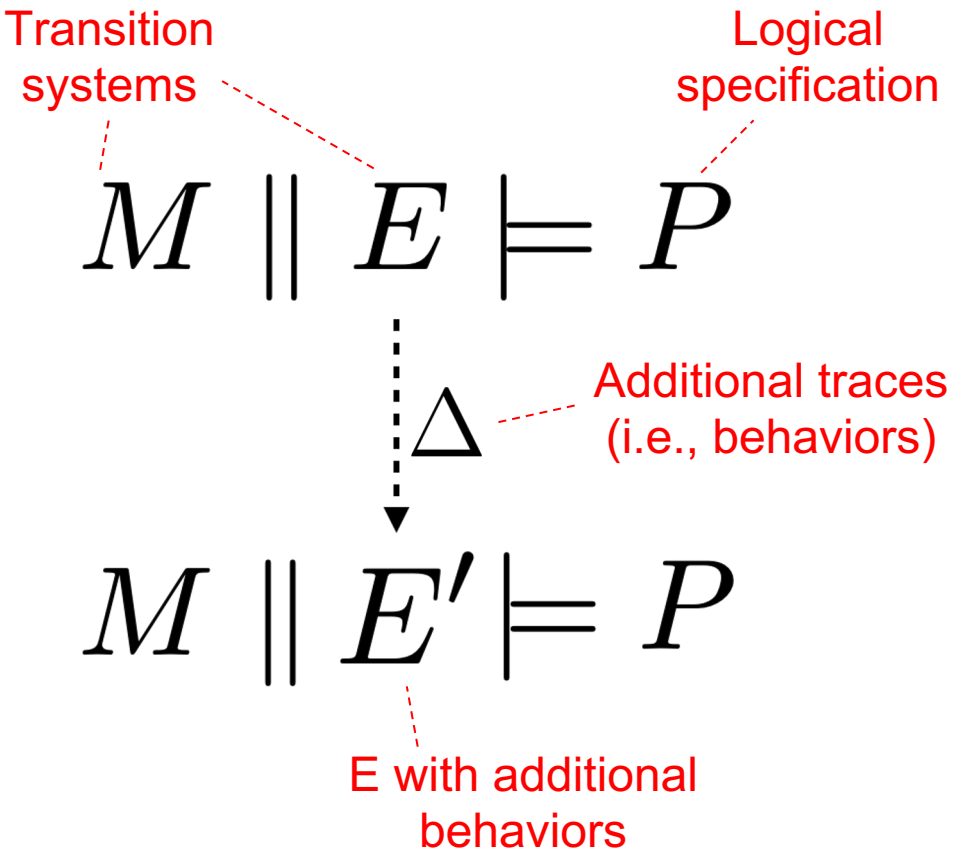
Security Property (P)

"No user can change the vote made by another person"
(in a logical specification)

Deviation as an Additional Trace



Automata-Theoretic Definition



Key idea

Given M & E as state machines, we can compute deviations (i.e., Δ & $\bar{\Delta}$) as a measure of robustness

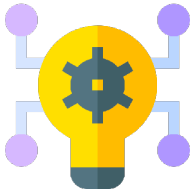
Robust-by-Design Systems

Specification

What does it mean for our system to be robust?

Analysis

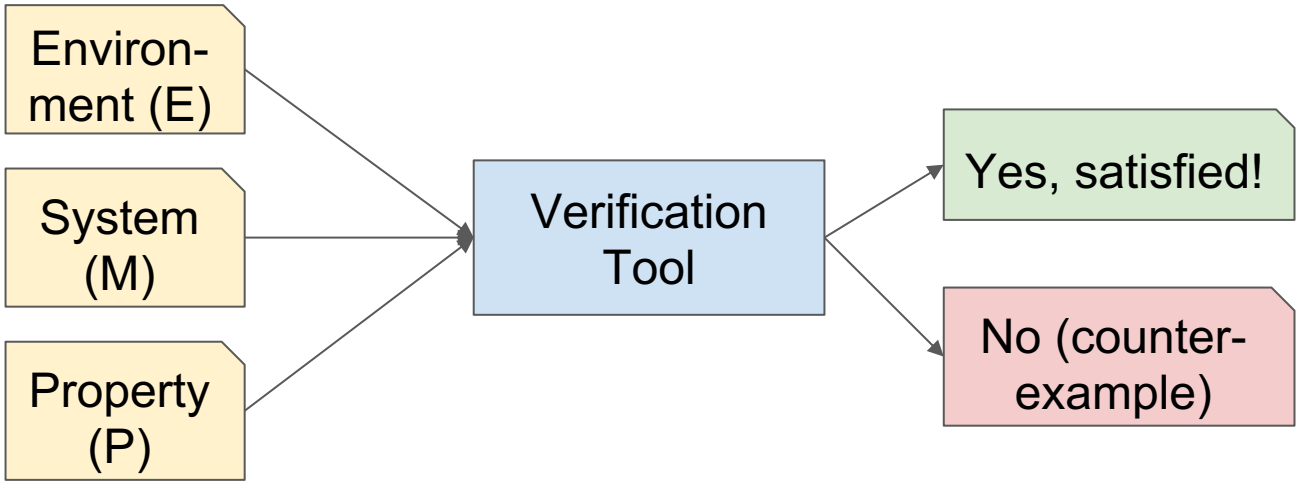
How robust is our system?



Robustification

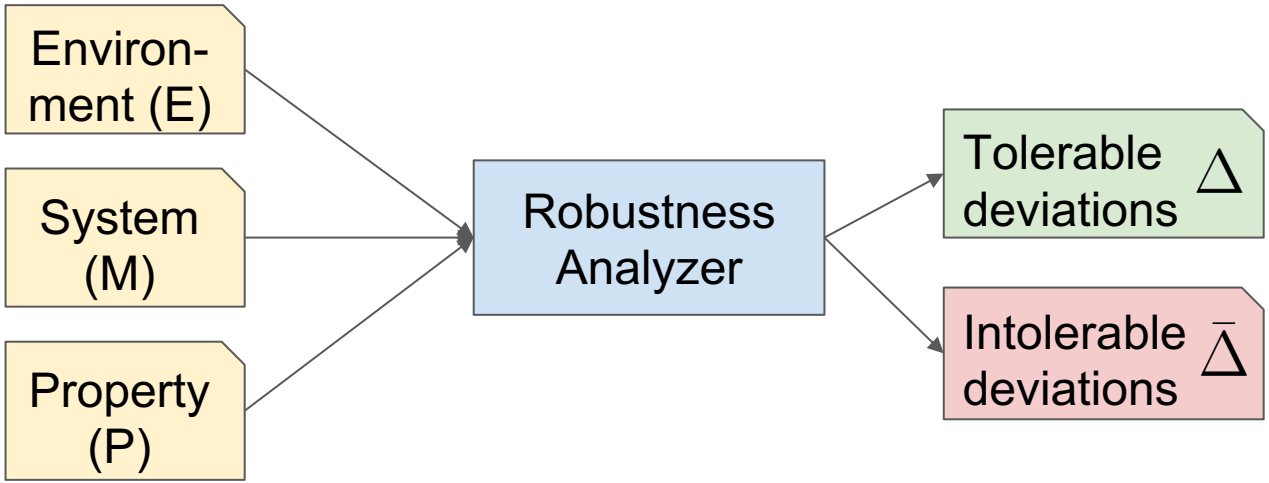
How do we improve its robustness?

Standard Verification Problem



Given M, E, P, does the system satisfy the property?

Robustness Analysis

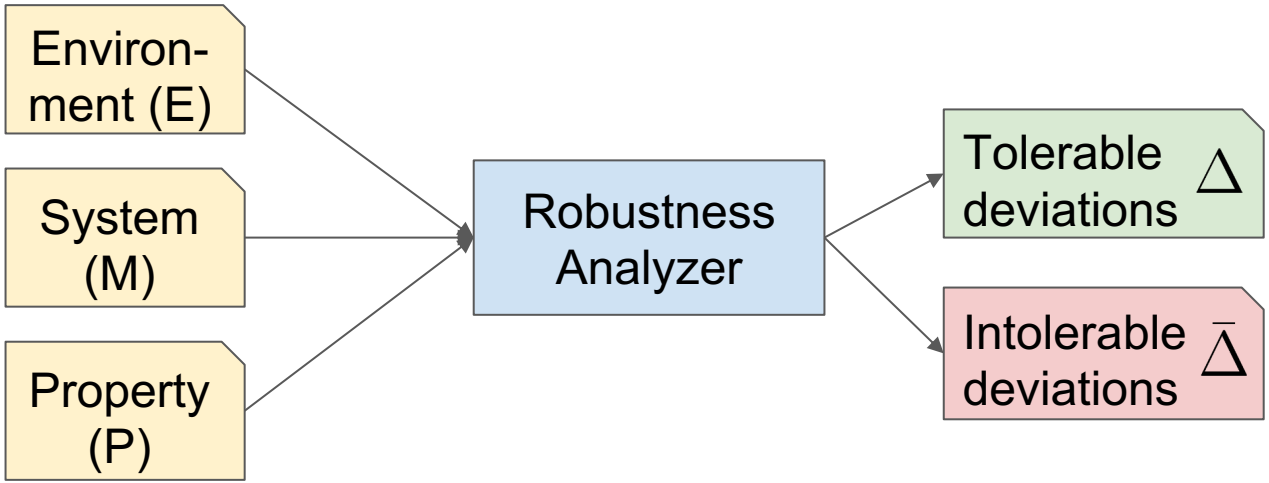


Robustness Analysis

Given M, E, P, how robust is the system (Δ)?

What are deviations that it cannot tolerate ($\bar{\Delta}$)?

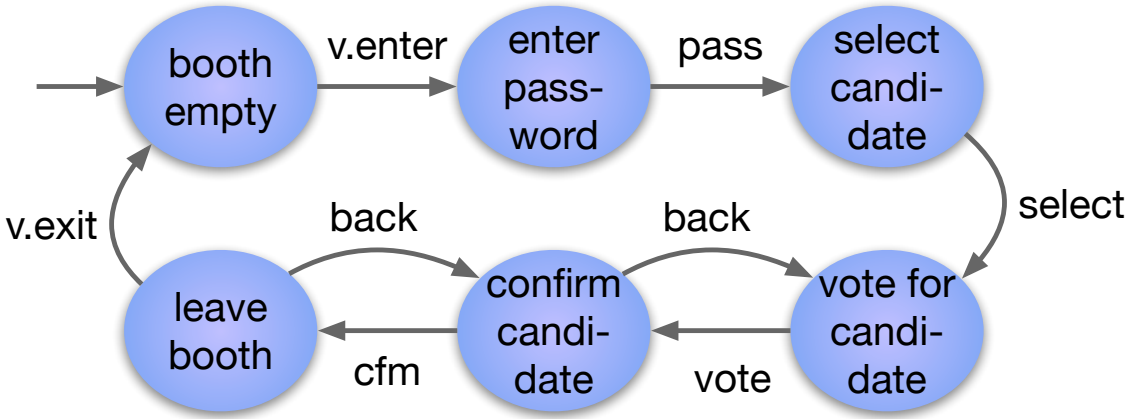
Robustness Analysis



Technical challenges:

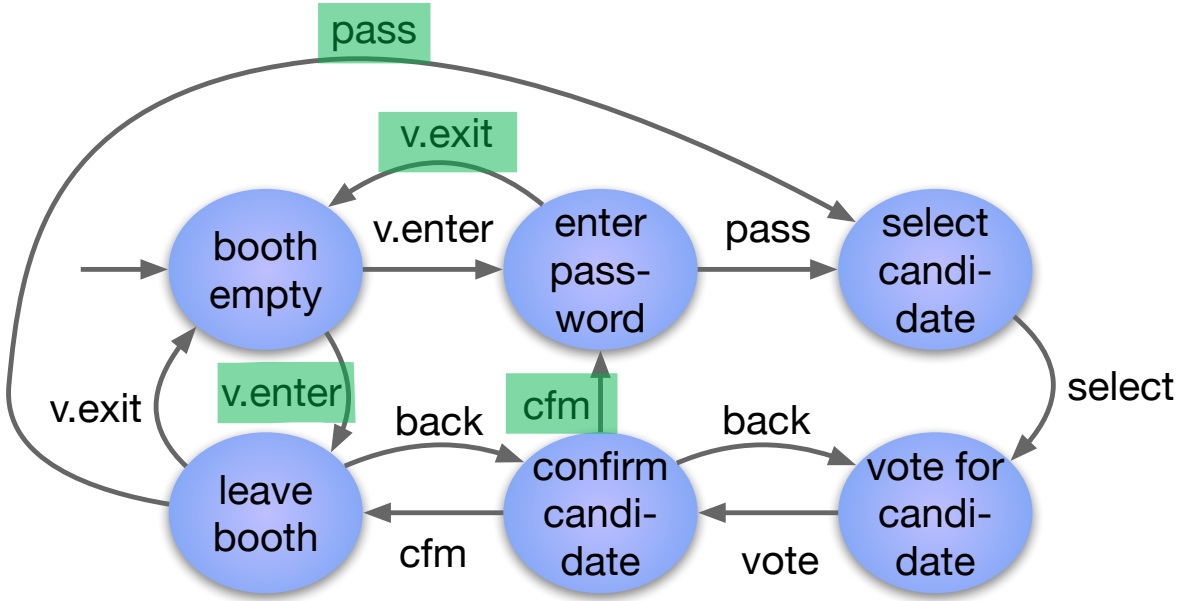
1. Computing the set of all deviations efficiently
2. Representing Δ concisely for comprehension

Example: Computing Deviations in Voter Behavior



Environment (E): Assumed voter behavior

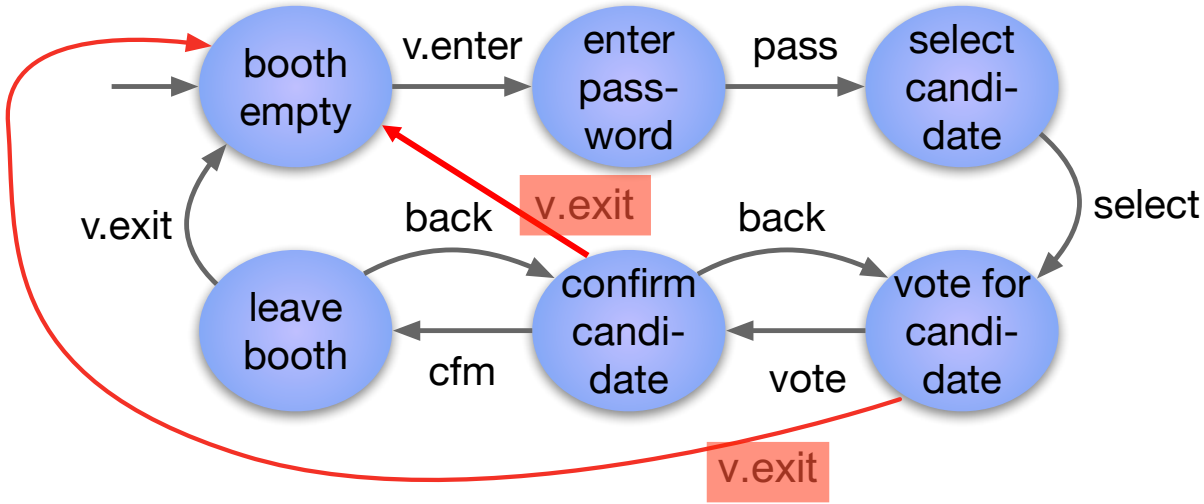
Example: Tolerable Deviations (Δ)



Represented as added transitions to E

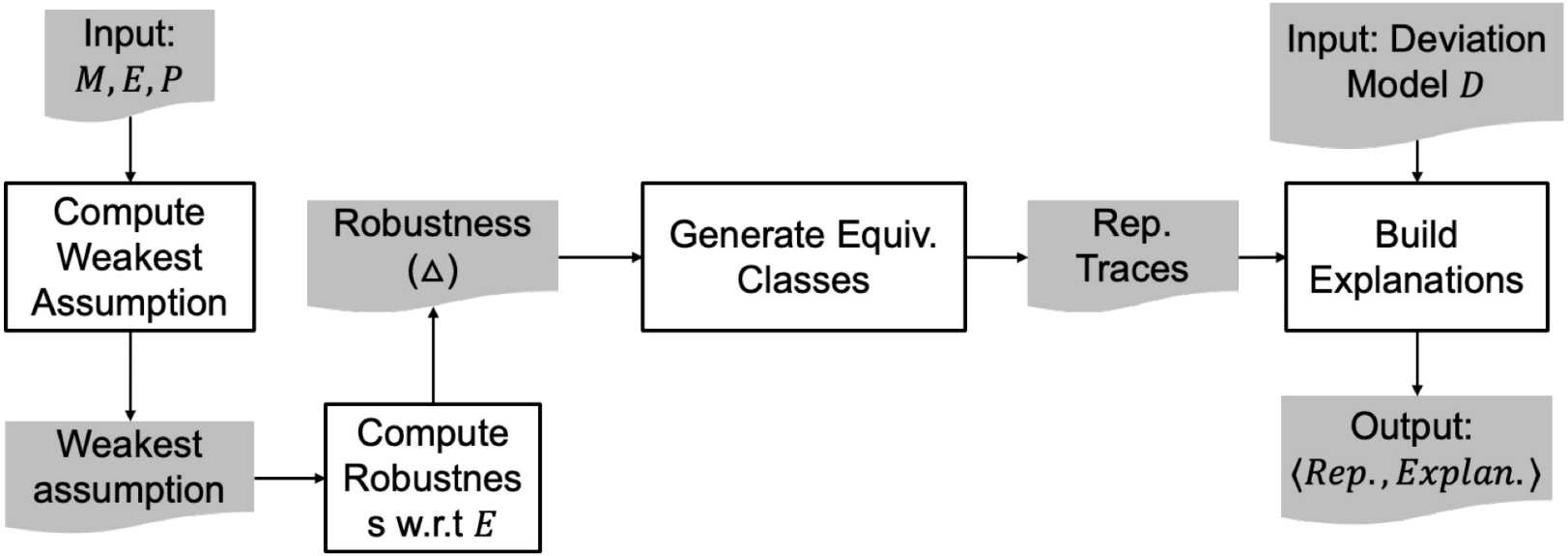
System preserves its property under these deviations

Example: Intolerable Deviations ($\bar{\Delta}$)



System may violate its property under these deviations!

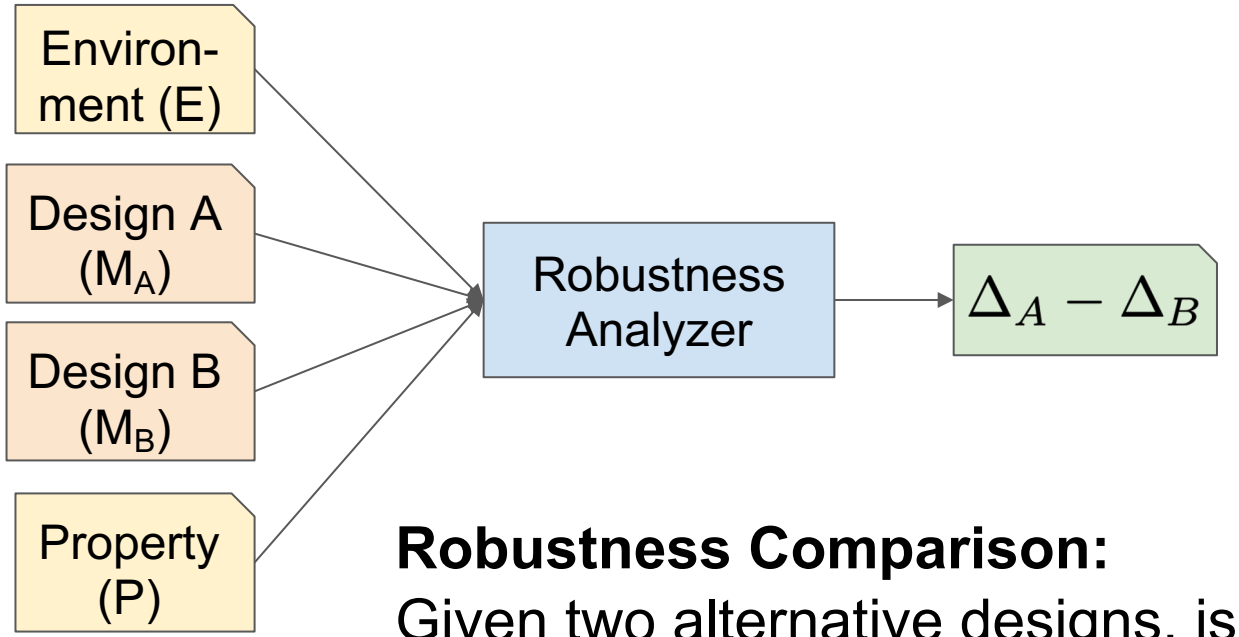
Analysis Process



More details in our paper!

A behavioral notion of robustness for software systems.
Zhang, Garlan, and Kang. ESEC/FSE 2020.

Comparing Designs w.r.t. Robustness



Robustness Comparison:
Given two alternative designs, is one more robust than the other (and if so, under what deviations)?

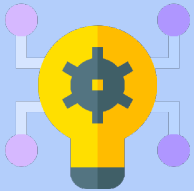
Robust-by-Design Systems

Specification

What does it mean for our system to be robust?

Analysis

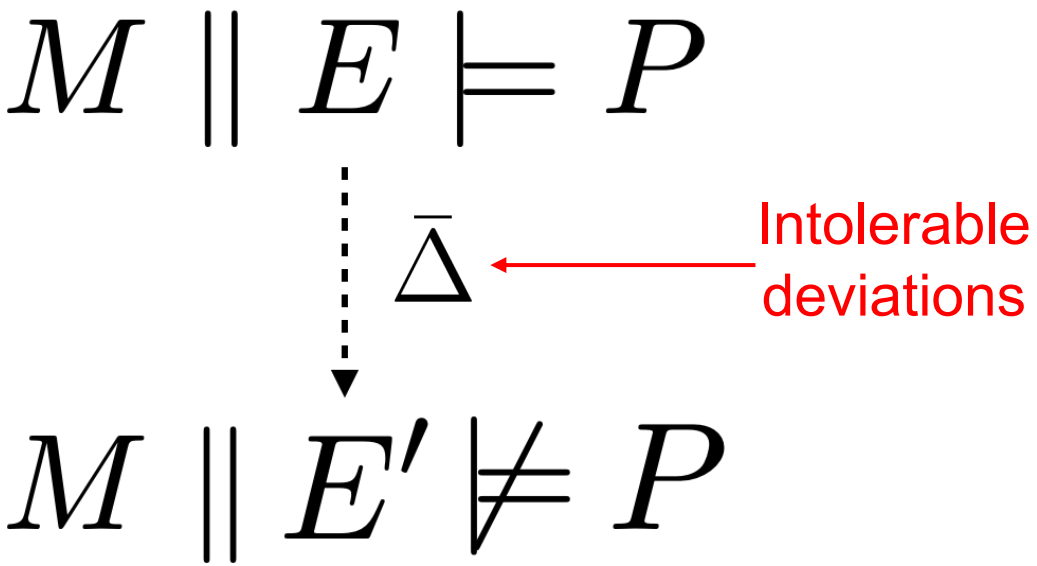
How robust is our system?



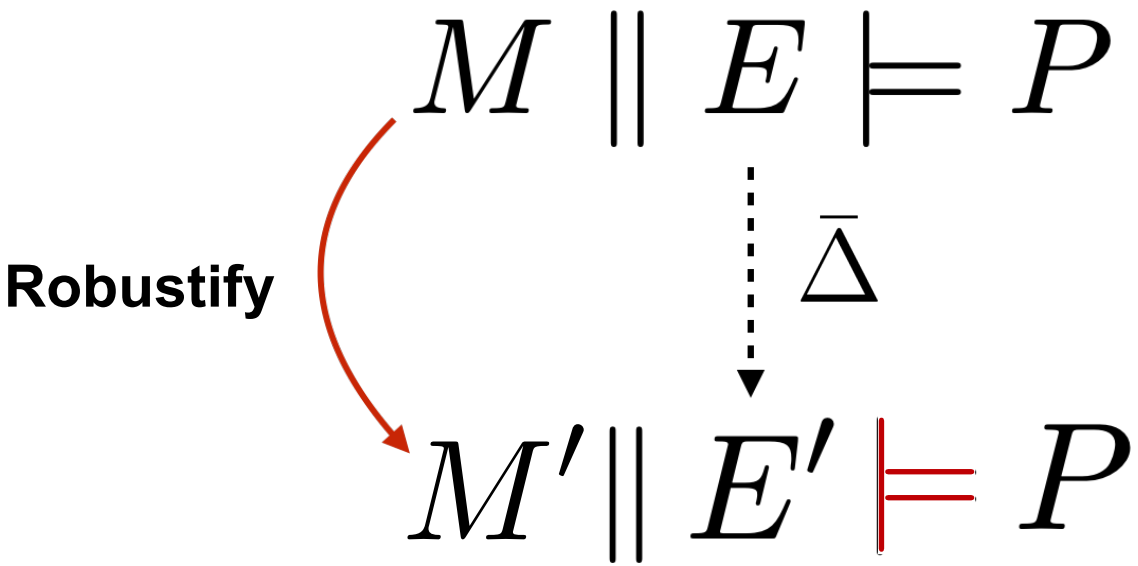
Robustification

How do we improve its robustness?

Robustification

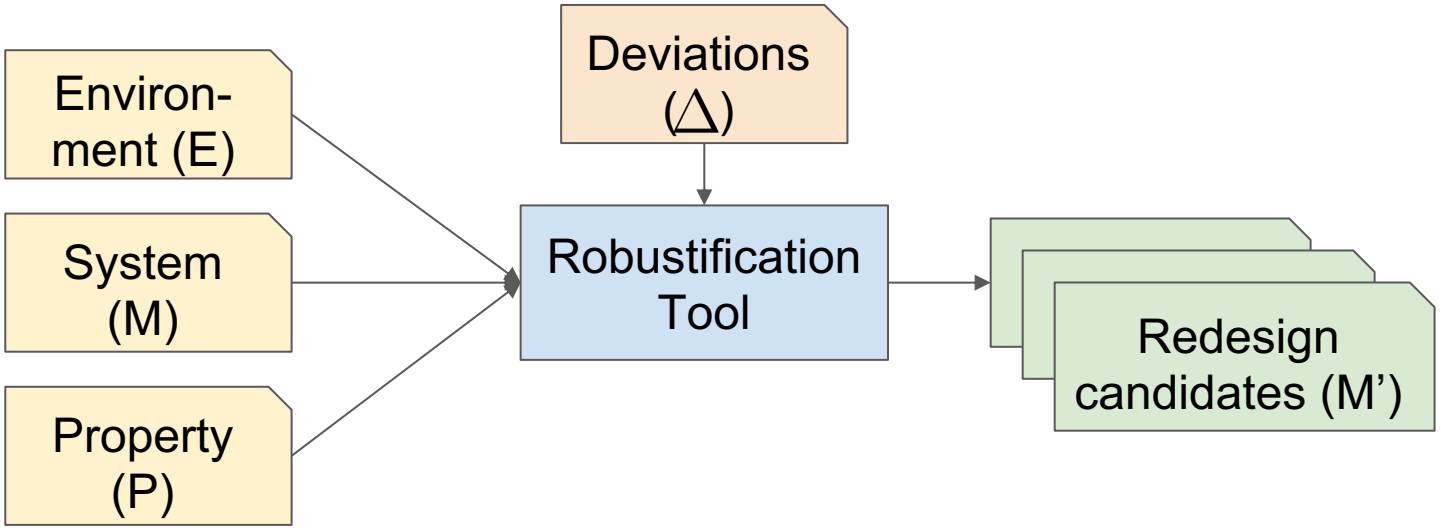


Robustification



Can we generate suggestions for enhancing the original design to tolerate additional deviations?

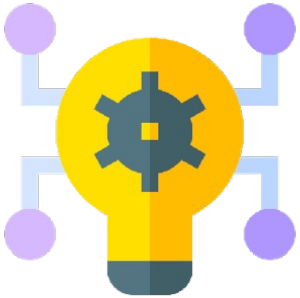
Robustification



Technical challenges:

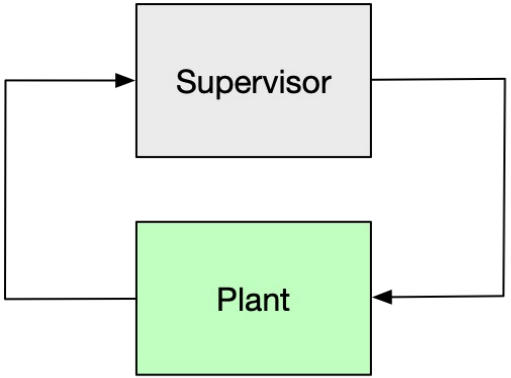
1. Searching a large space of candidate solutions
2. Trade-offs between permissiveness vs. complexity

Robustification as Supervisory Control Synthesis



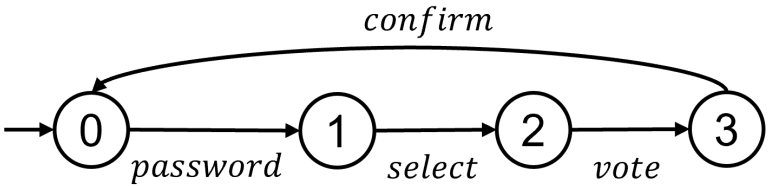
Robustification problem

reduce



Supervisory control problem

Candidate Solutions

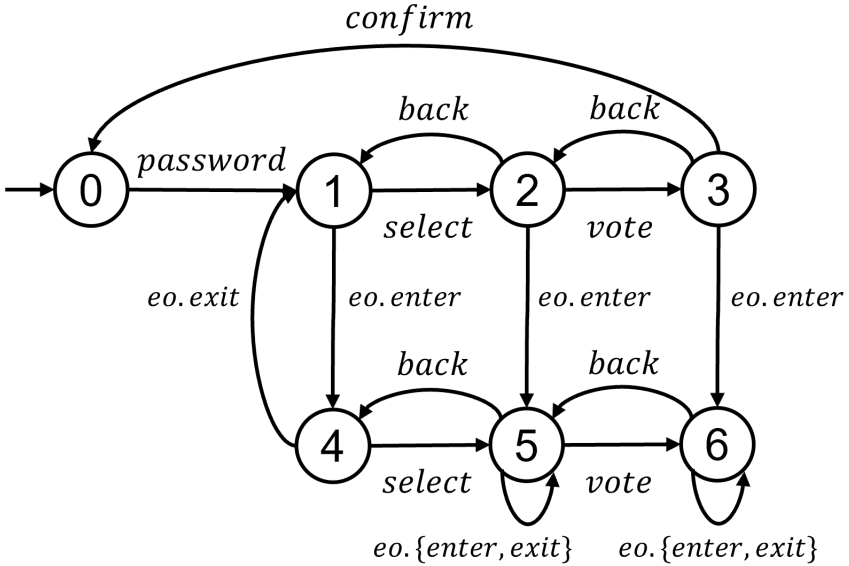


Redesign #1

Disables “back” action

Simple, but not permissive

Does not allow voter to modify selection



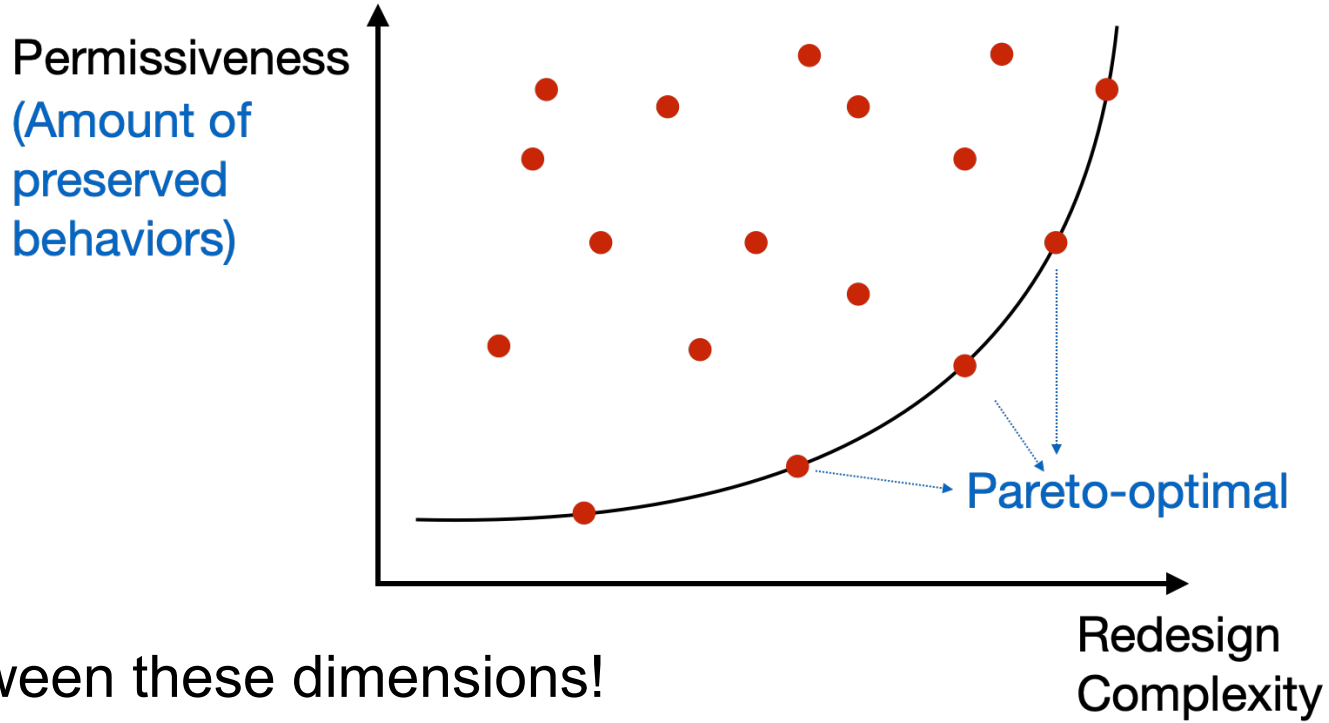
Redesign #2

Disables confirm while the official is in the booth

More permissive: Allows vote change

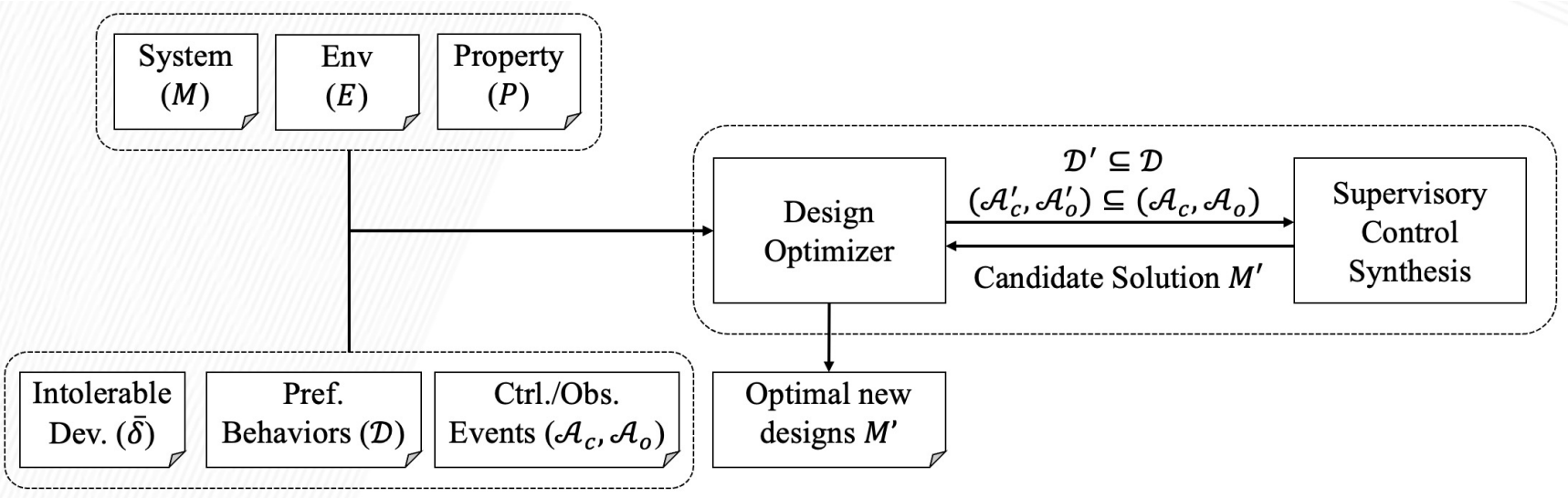
But more complex: Requires keeping track of booth occupant

Optimal Robustification



Trade-offs between these dimensions!
Generate multiple possible Pareto-optimal solutions

Robustification Process



More details in our paper!

Robustification of Behavioral Designs against Environmental Deviations. Zhang et al. ICSE 2023.

Case Studies

OAuth authorization protocols

Oyster Card protocol

Medical device interfaces (radiation therapy, infusion pumps)

Largest model size: ~19k states
Robustness analysis: < 2.0 seconds
Robustification: ~8 minutes

Takeaway

Robustness: What potential deviations can my system tolerate & achieve a desired security goal?

With robustness as a first-class property of systems, we can:
-> Reason about the impact of deviations on security
-> Compare alternative designs w.r.t. robustness
-> Design systems to achieve a desired level of robustness

Try our tool!

https://github.com/cmu-soda/Fortis