Designing Software Systems to be Robust

Eunsuk Kang

Software Engineering Seminar
University of Waterloo
June 20, 2023

S3D

Carnegie Mellon University


70


670 Maple
FRONT DOOR


200.0

Environment Assumptions

To establish its requirements, a system relies on various assumptions about the environment

Nurse performs actions in correct order
Network latency is at most 50 ms
Attacker doesn't have access to secret key

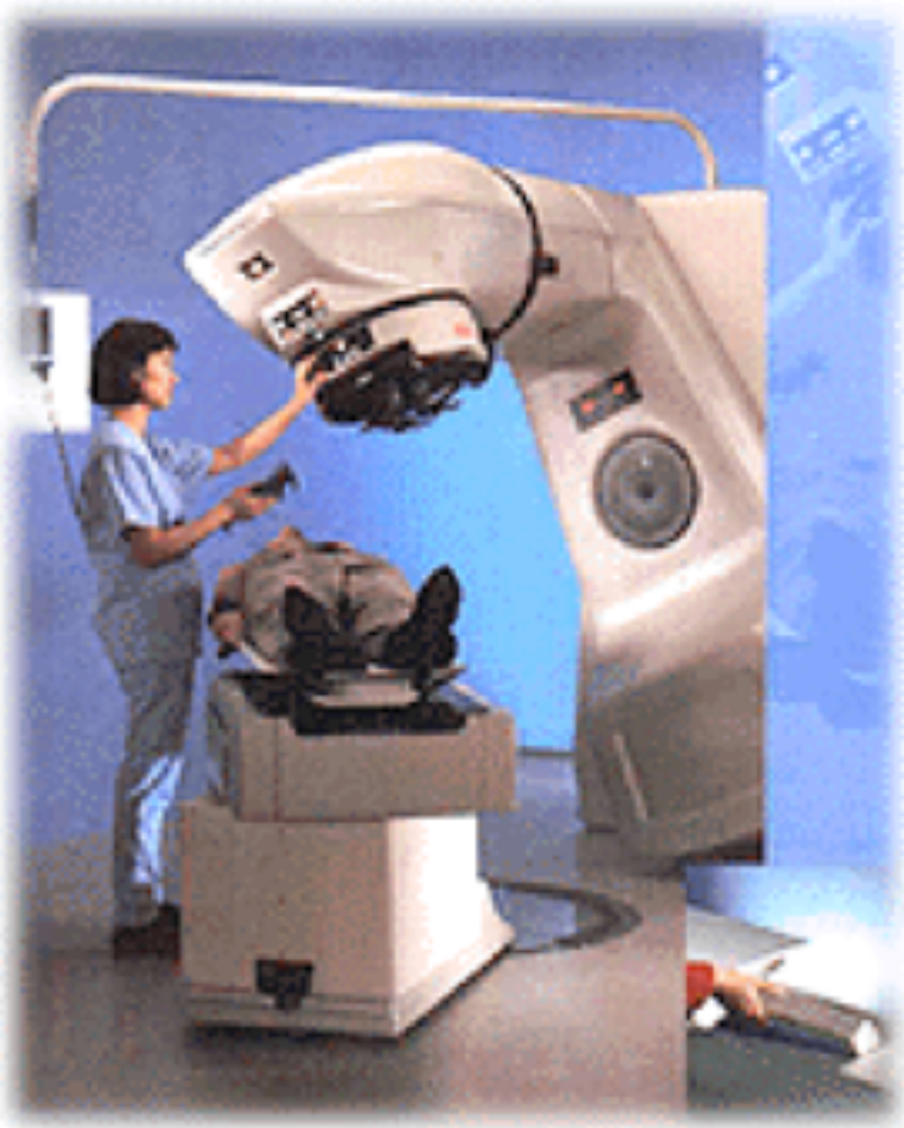
Environment Deviations

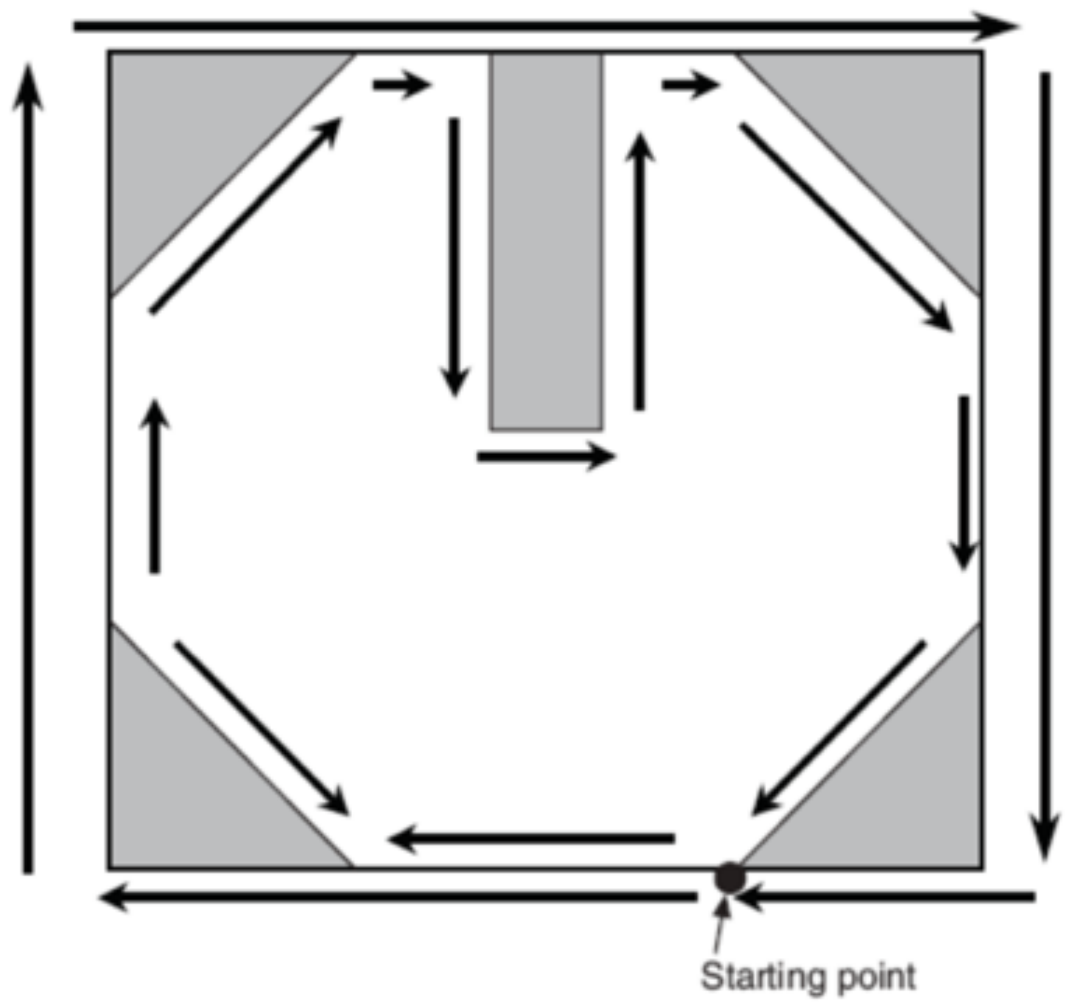
What happens if the environment deviates from these assumptions?

Nurse inadvertently omits a critical action
Network experiences an unexpected disruption
Attacker obtains a secret through a side channel

Does the system still provide any guarantees?

Panama City Public Hospital (2001)





Therapy planning software by Multidata Systems

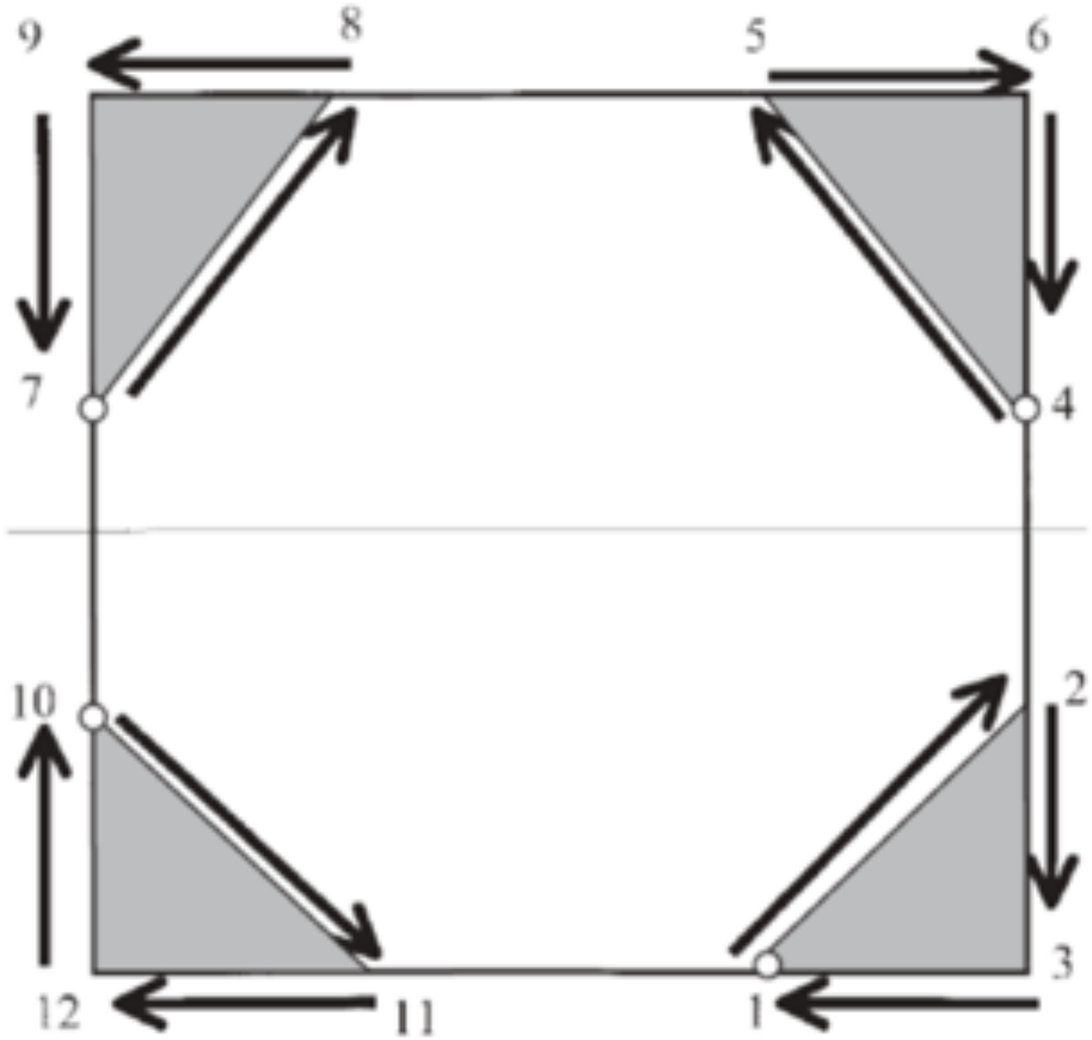
Theratron-780 by Theratronics (maker of Therac-25)

Shielding blocks

Inserted into beam path, protect healthy tissue

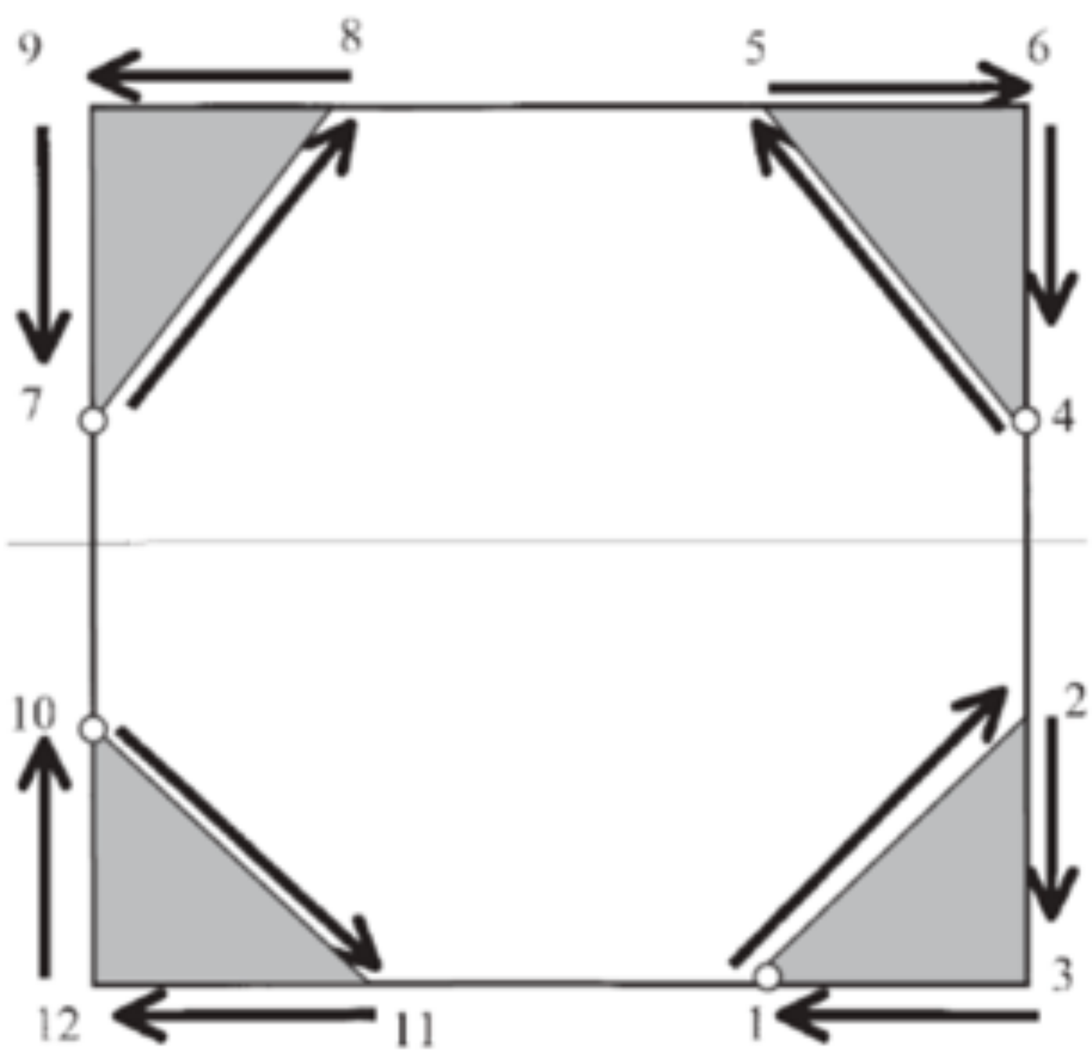
Therapist draws block shapes; SW computes dose

Therapist Interaction

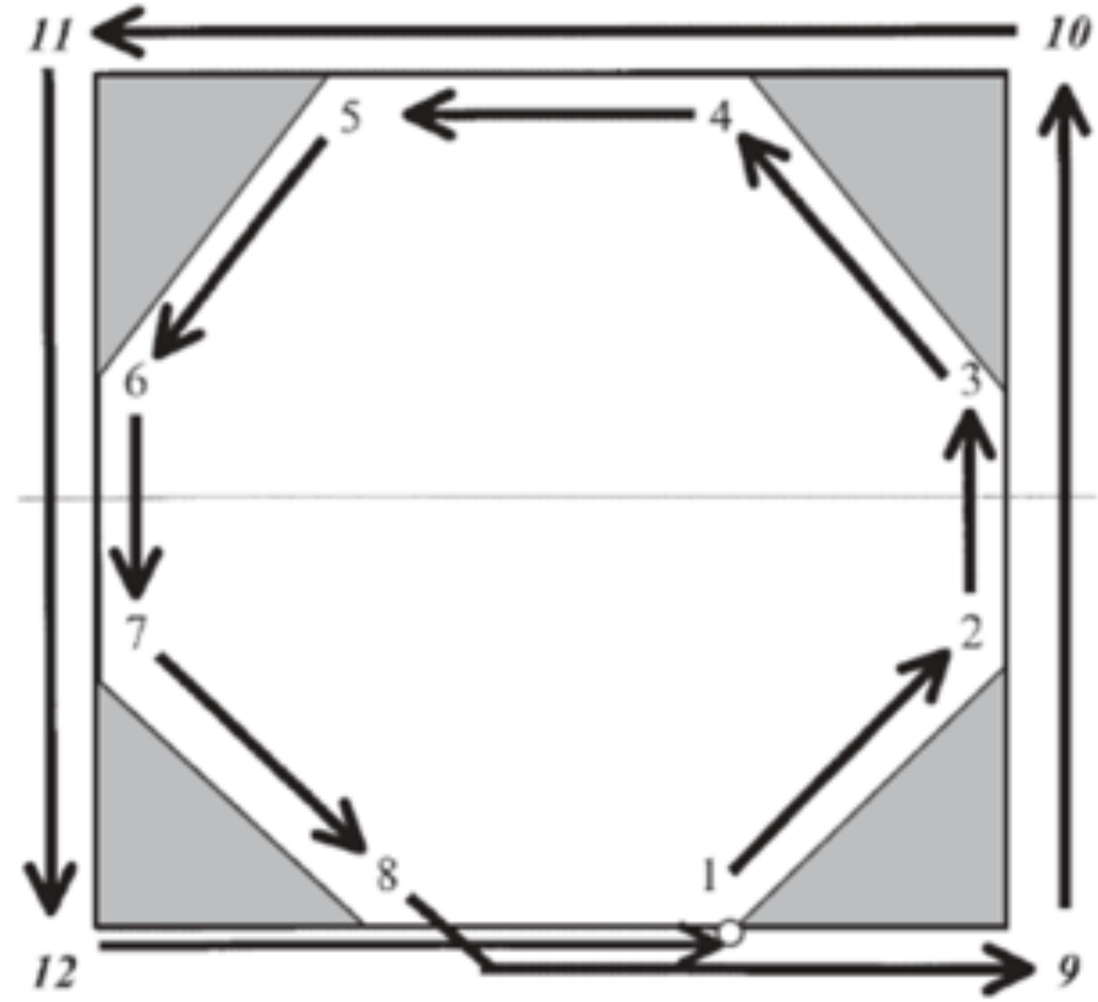


dose = D

Accidents



dose = D



dose = $2D$

28 patients overdosed; 21 deaths

Blame User or Software?

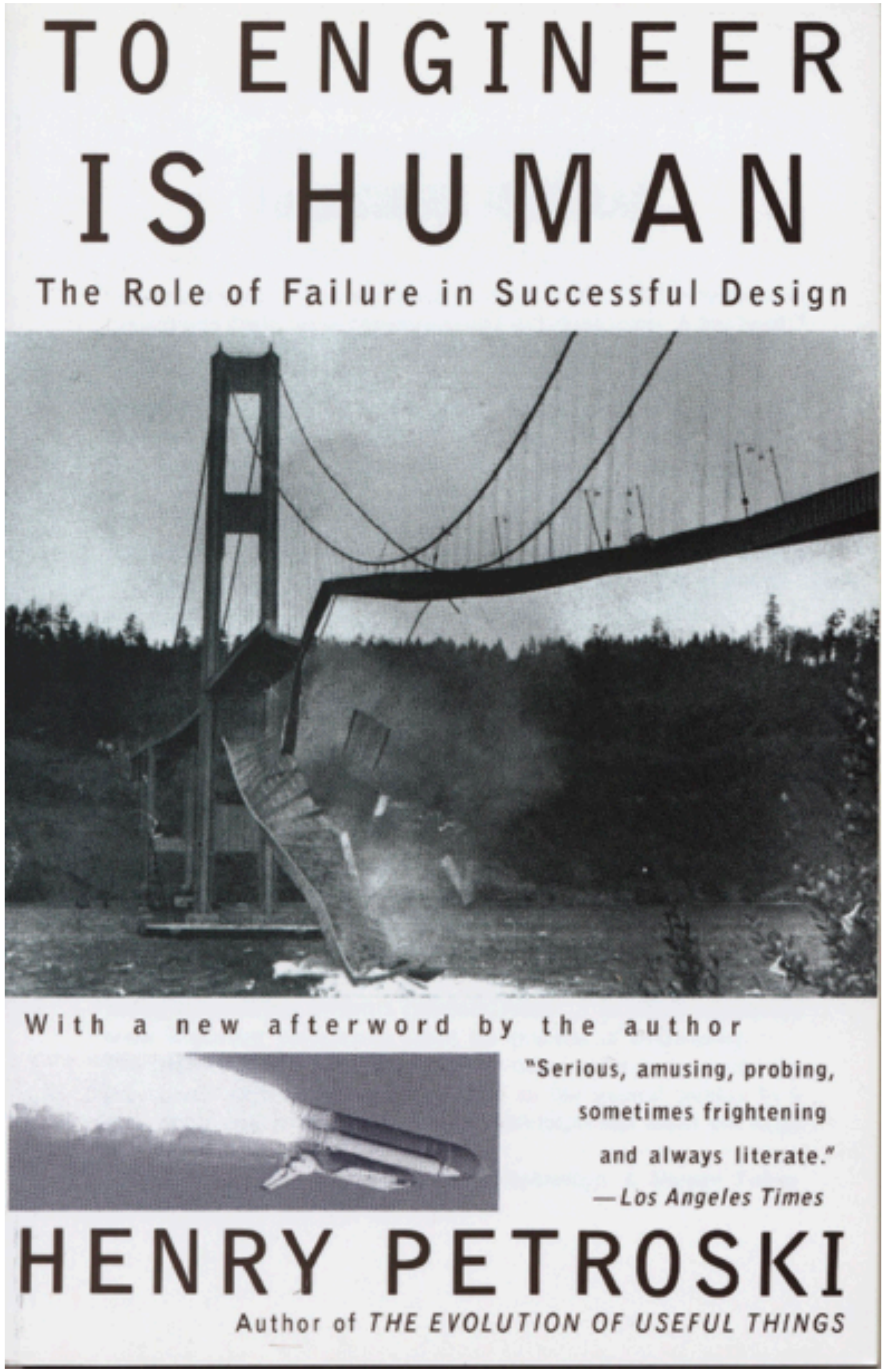
Multidata Systems

“Given [the input] that was given, our system calculated the correct amount, the correct dose. And, if [the staff in Panama] had checked, they would have found an unexpected result.”

Three therapists charged & found guilty for involuntary manslaughter; barred from practice

The environment will occasionally deviate from its expected behavior

A robust system should ensure its critical properties even under such deviations



Successful engineering products are designed with a margin of safety that provides layers of protection against abnormal events

Robust Systems by Design

1) Devise an initial system design
2) Identify types of deviations in the environment
3) Analyze the design to check whether it's robust against those deviations
4) If not, redesign the system to improve its robustness

But in software:

What exactly do we mean by "robust"?

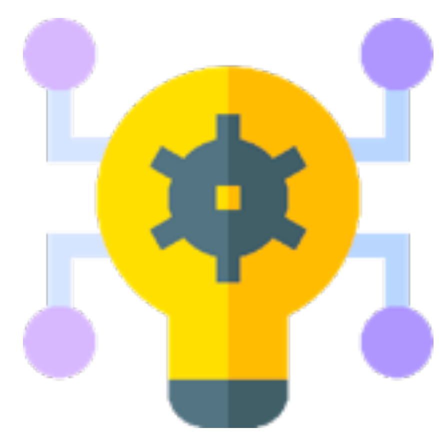
How do we verify that a system is sufficiently robust?

How do we systematically improve its robustness?

Robust Software Design: Roadmap

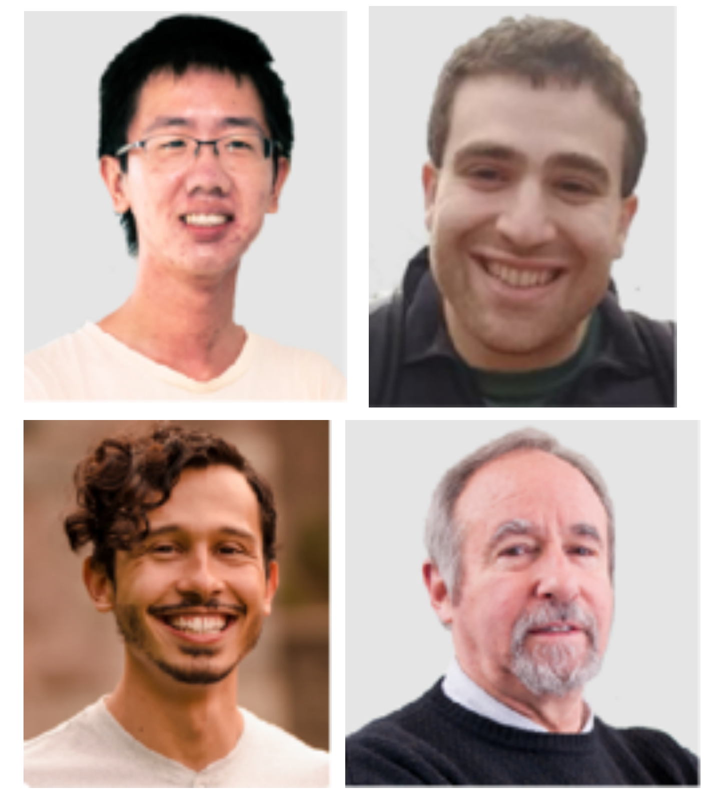
Specification
What does it mean for
our system to be robust?

Analysis
How robust is our
system?



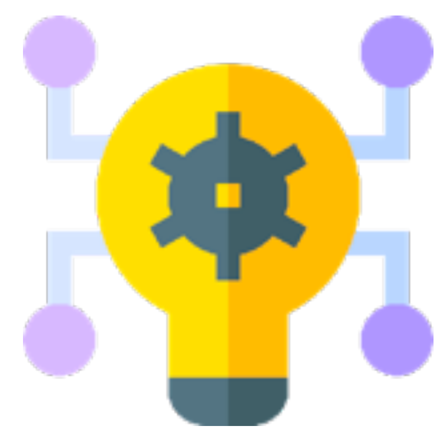
Robustification
How do we improve
its robustness?

Robust Software Design: Roadmap



Specification
What does it mean for
our system to be robust?

Analysis
How robust is our
system?

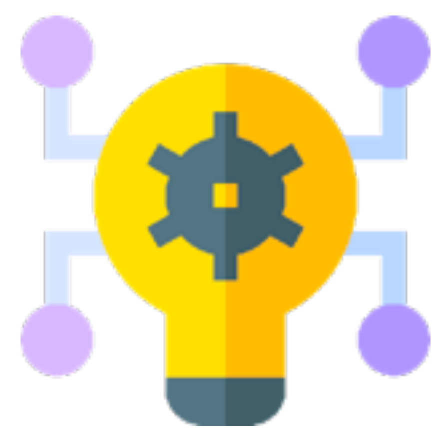


Robustification
How do we improve
its robustness?

Robust Software Design: Roadmap

Specification
What does it mean for our system to be robust?

Analysis
How robust is our system?



Robustification
How do we improve its robustness?

What exactly does it mean for software to be robust?

Robustness: High-Level Idea

The maximum amount of environmental deviations under which the system is capable of satisfying a desired property

Robustness: High-Level Idea

Software Specification

System Property

$$M \parallel E \models P$$

Assumptions

Robustness: High-Level Idea

$$M \parallel E \models P$$

$$\Big\downarrow \delta$$

Deviations

$$E'$$

Robustness: High-Level Idea

$$M \parallel E \models P$$

$$\Big\downarrow \delta$$

$$M \parallel E' \models P$$

Preserves P even under deviated environment

System M is *robust* *against a set of deviations δ with respect to environment E and property P*

Robustness: Behavioral View

LTS

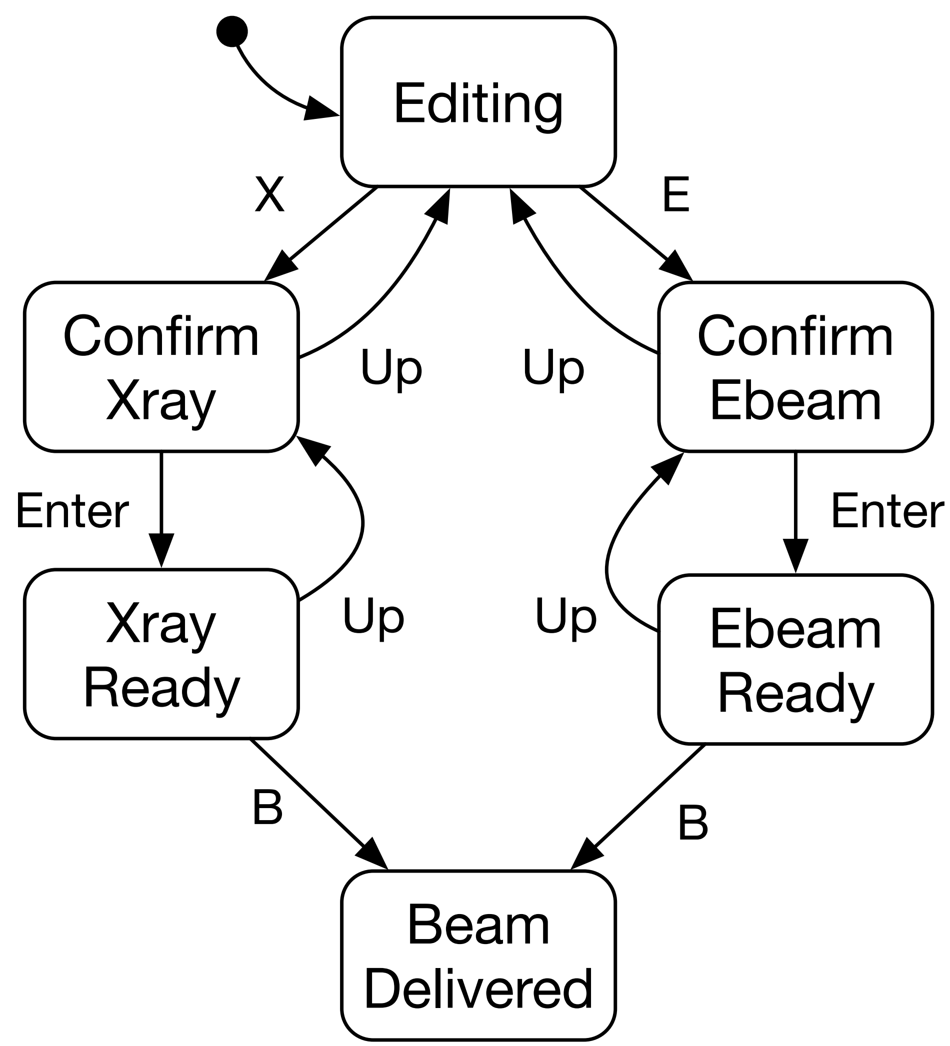
LTS

Safety property

$$M \parallel E \models P$$

LTS: Labelled transition system

Labelled Transition System



M

$$beh(M) \equiv \{ \langle\rangle, \langle \mathbf{X} \rangle, \langle \mathbf{E} \rangle, \langle \mathbf{X, Enter} \rangle, \langle \mathbf{X, Enter, B} \rangle, \langle \mathbf{E, Enter} \rangle \ldots \}$$

Simple but expressive formalism

Behaviors of an LTS ≡ Possible traces (event sequences)

Robustness: Behavioral View

LTS

LTS

Safety property

$$M \parallel E \models P$$

$$\delta$$

Traces (not in E)

$$E'$$

E with additional behaviors (δ)

Key Idea: Represent & compute δ as traces

Robustness: Behavioral View

LTS

LTS

Safety property

$$M \parallel E \models P$$

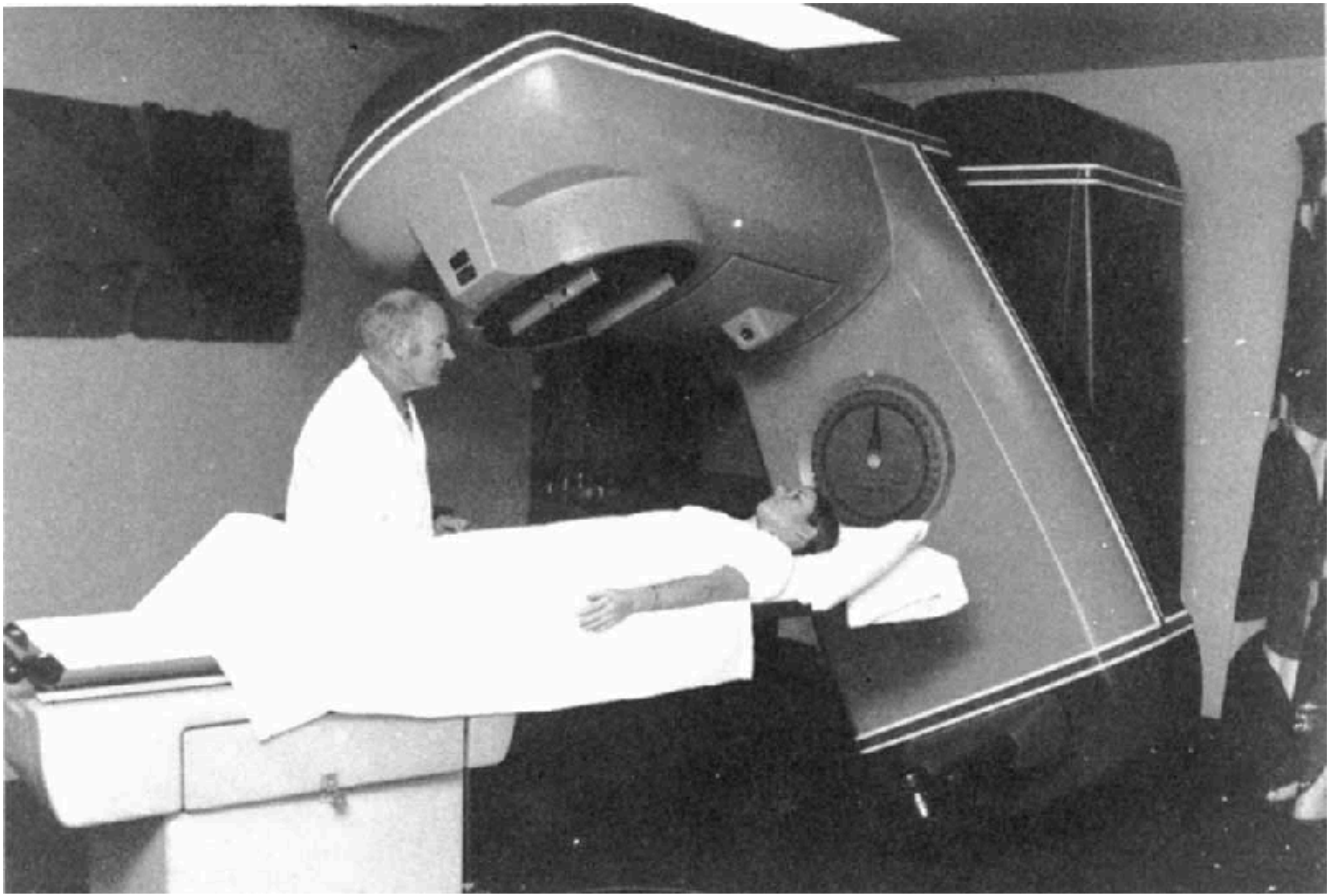
$$\Big\downarrow \Delta$$

$$M \parallel E' \models P$$

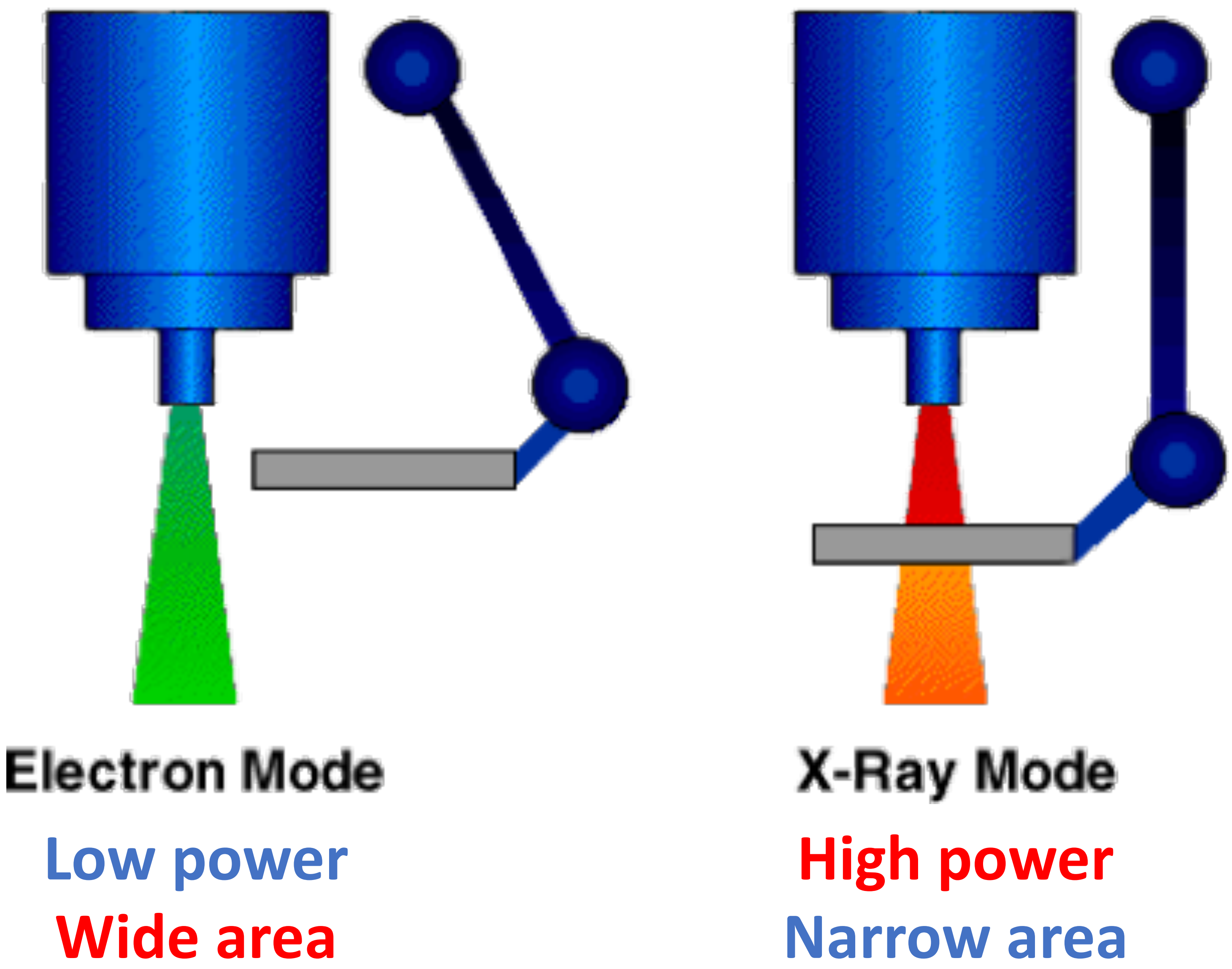
Robustness $\Delta(M, E, P)$

The largest set of possible deviations under which the system can ensure property P

Example: Therac-25



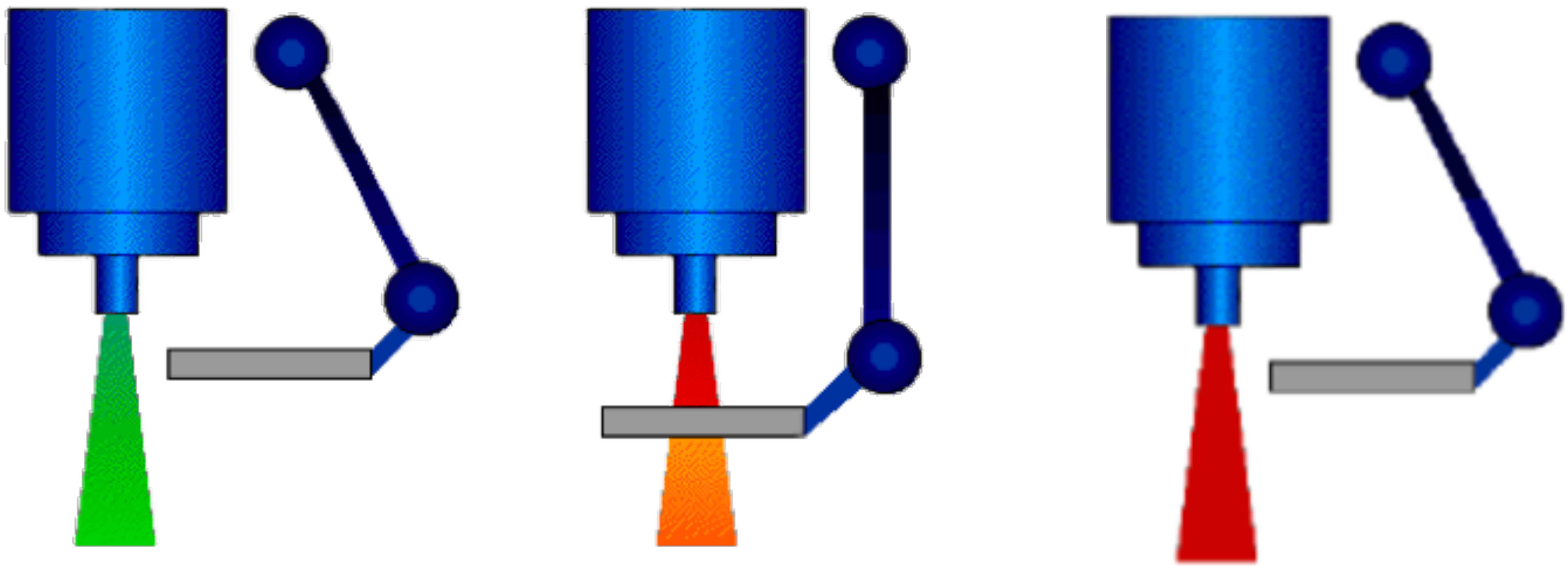
Radiation Modes in Therac-25



Electron ("Ebeam") and X-ray modes

Insert collimator during X-ray for safe radiation level

Safety Hazard



Electron Mode	X-Ray Mode	THE PROBLEM
Low power	High power	High power
Wide area	Narrow area	Wide area

X-ray mode & collimator out → Possible overdose!

Caused several fatal injuries in Therac-25

Establishing Safety Property

Software model
(interface + controller)

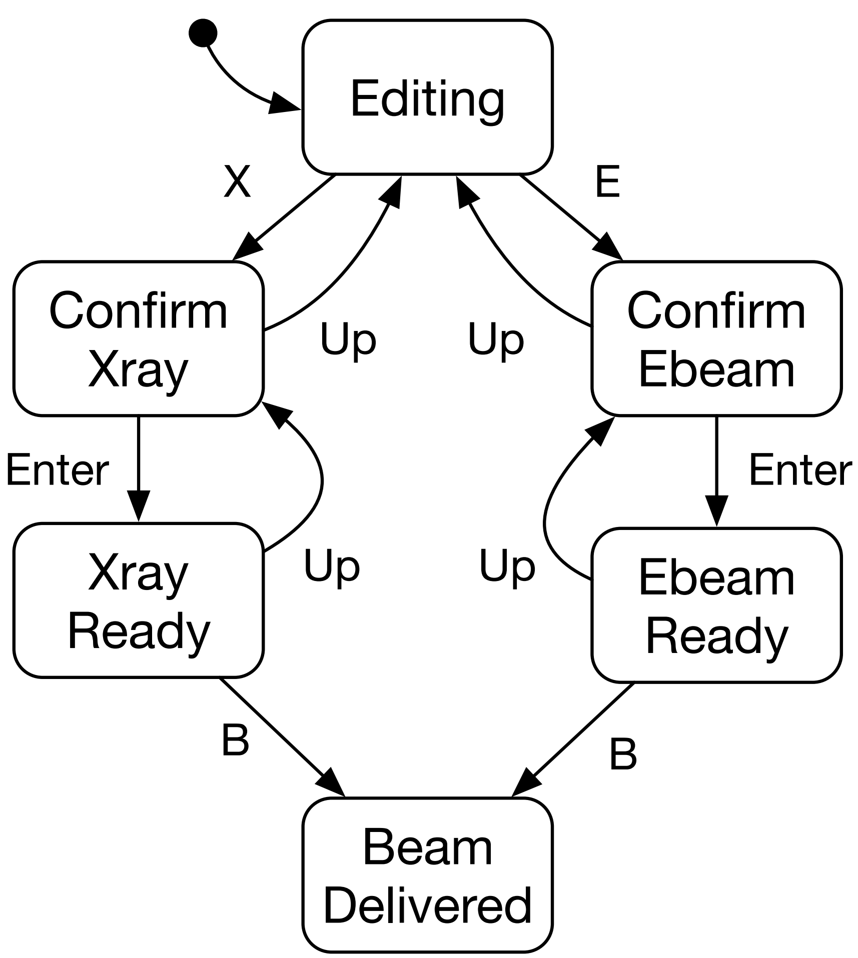
Safety property
“No radiation
overdose”

$$M \parallel E \models P$$

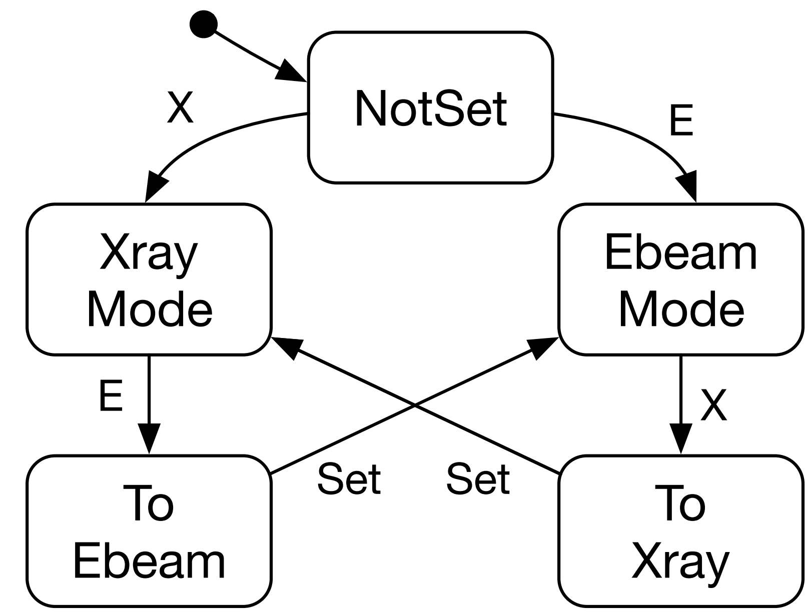
Operator task
description

M, E Labeled transition systems
P Safety property

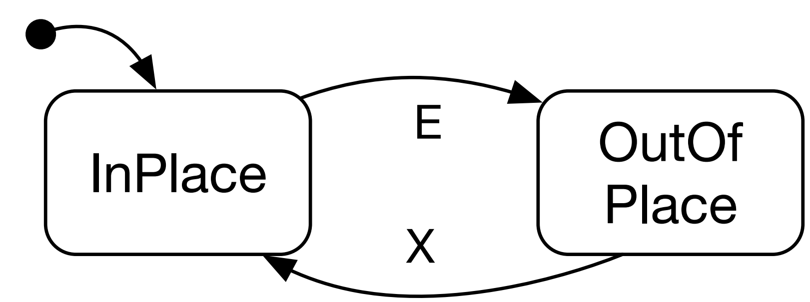
Therac-25 Design



Interface (M_I)



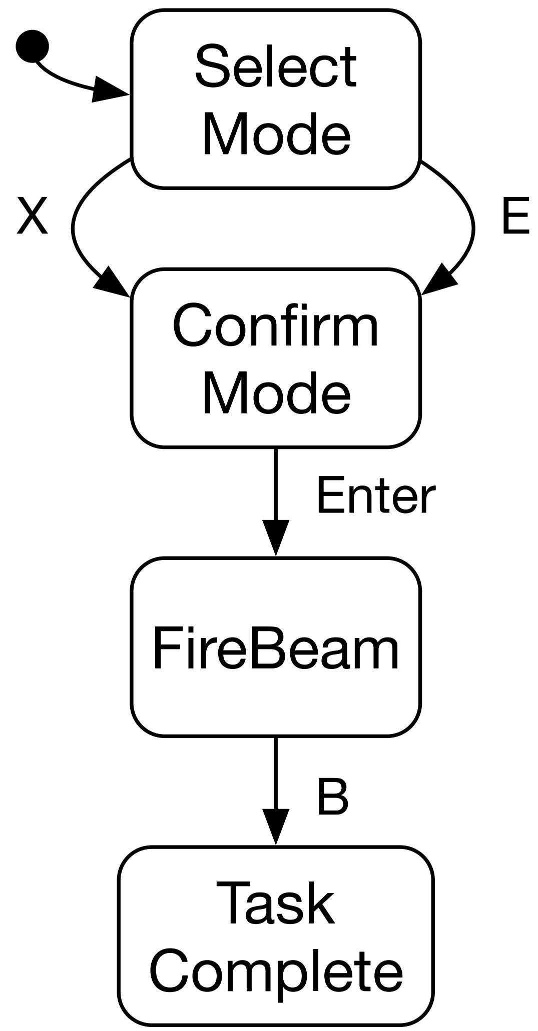
Mode setter (M_B)



Collimator (M_S)

$$M = M_I \parallel M_B \parallel M_S$$

Modeling Operator Behavior



Operator model (E)

Possible traces in E:

$\langle$X, Enter, B$\rangle$

$\langle$E, Enter, B$\rangle$

Captures expected sequences of users actions

Typically specified in a training manual or user instructions

Establishing Safety Property

Software Model

Safety property e.g., “No overdose”

$$M \parallel E \models P$$

Operator behavior

Under “expected” operator behavior, system satisfies the safety property!

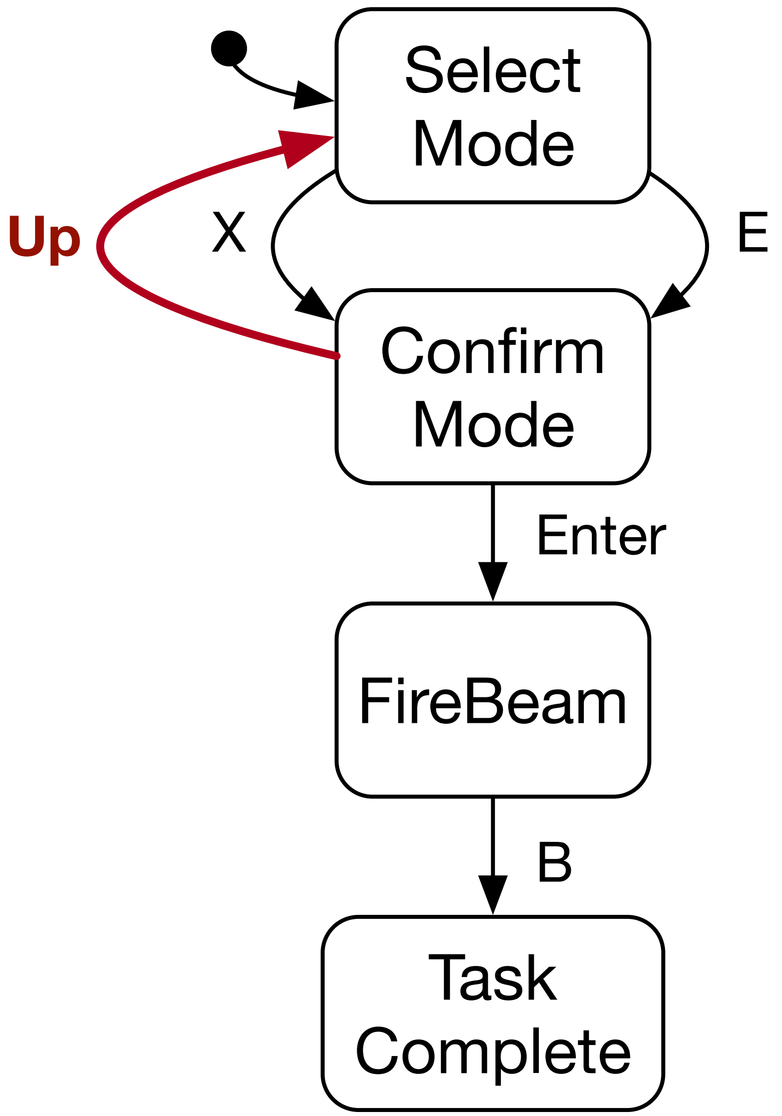
Deviations

Operator Error

“…[Therapist] noticed that for mode she had typed "x" (for X ray) when she had intended "e" (for electron)…the mistake was easy to fix; she merely used the cursor up key to edit the mode entry.”

An Investigation of the Therac-25 Accidents
Leveson & Turner, IEEE Computer, 1993

Modeling Operator Error

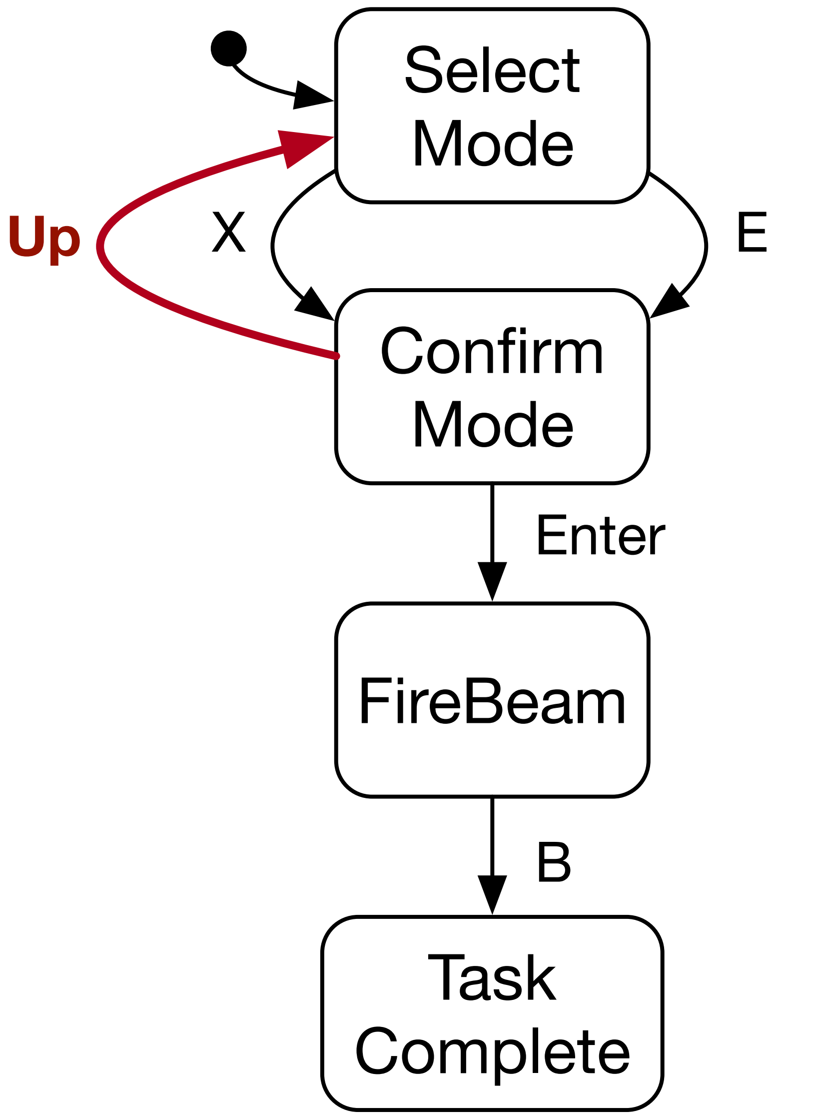


Erroneous operator (E')

What if the operator commits an error?

1. Selects X-ray mode by mistake
2. Realizes error, presses UP to go back
3. Selects Ebeam mode and proceeds

Deviation as Additional Behavior



Erroneous operator (E')

$\langle$X, Enter, B$\rangle$

$\langle$X, Up, E, Enter, B$\rangle$

A new trace (deviation) absent in E

System under Deviated Environment

Perfect operator

$$M \parallel E \models P$$

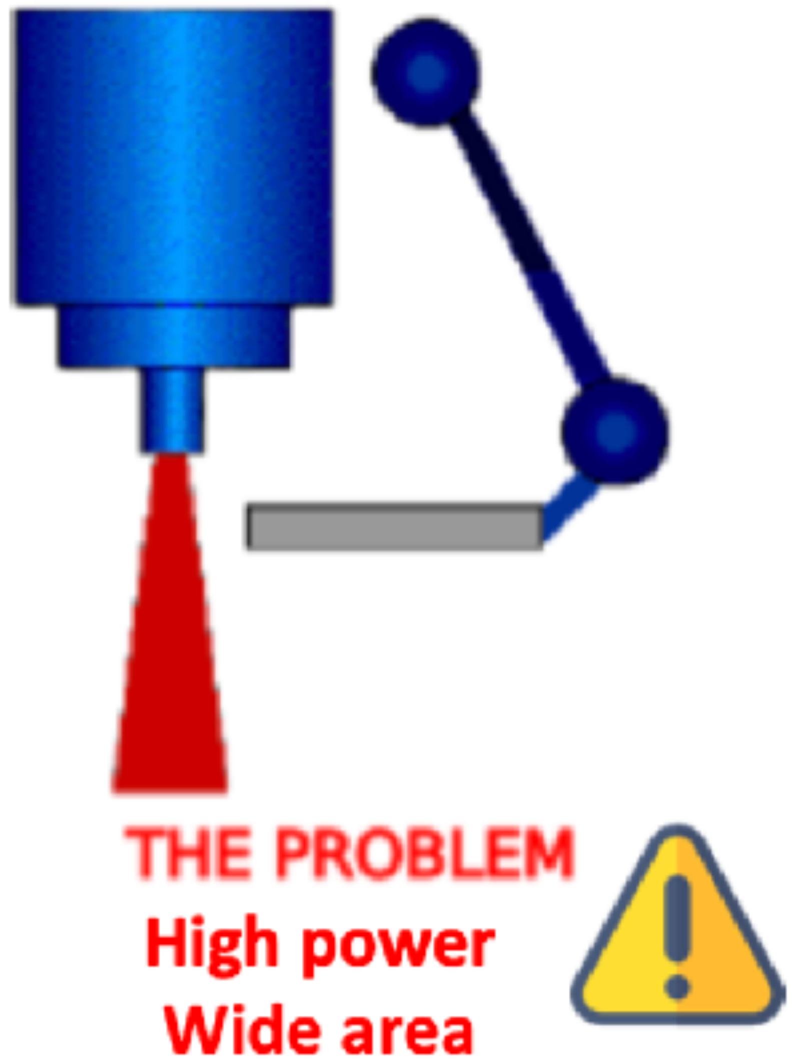
$$\Big\downarrow \delta$$

Operator mistake

$$M \parallel E' \models P\,?$$

Erroneous operator

Safety Violation



What if the operator commits an error?

1. Selects X-ray mode by mistake
2. Realizes error, presses UP to go back
3. Selects Ebeam mode and proceeds
4. Collimator is removed for Ebeam
5. When user fires, radiation setting is still transitioning from X-ray to Ebeam
6. **Safety violation!**

System under Deviated Environment

Perfect operator

$$M \parallel E \models P$$

Erroneous operator

$$\delta$$

Operator mistake

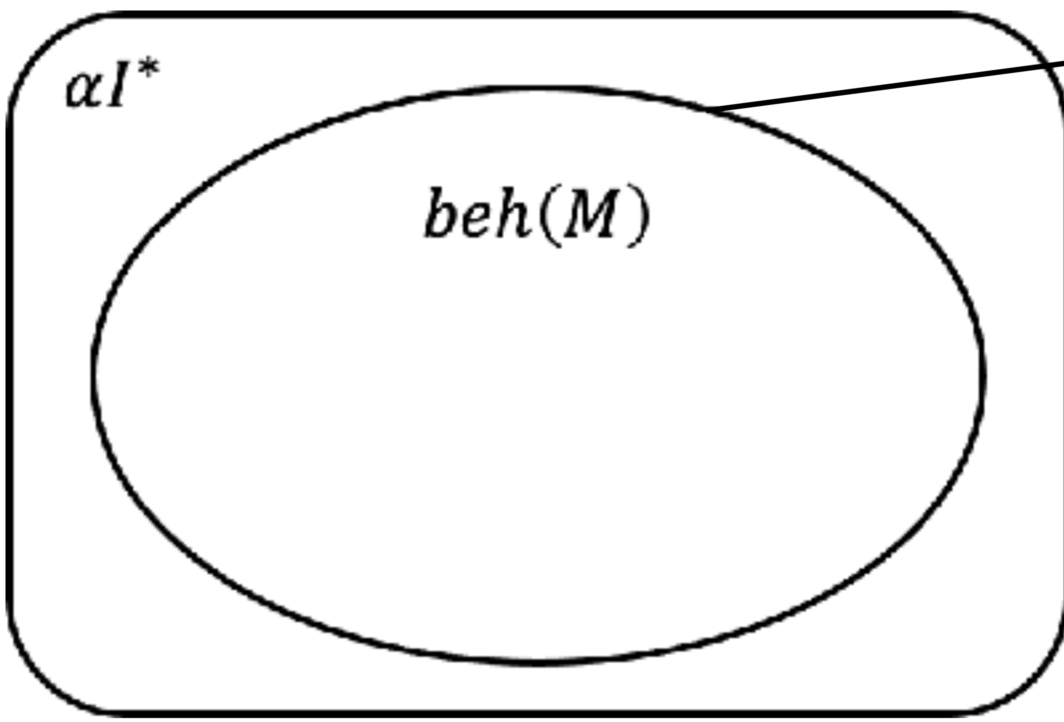
$$M \parallel E' \not\models P$$

System is not robust; i.e., fails to be safe under the deviated environment!

Robustness: Another View

M : Radiation therapy system
E : Operator behavior
P : “No overdose”
αI : Interface events
αI^*: All possible traces over αI

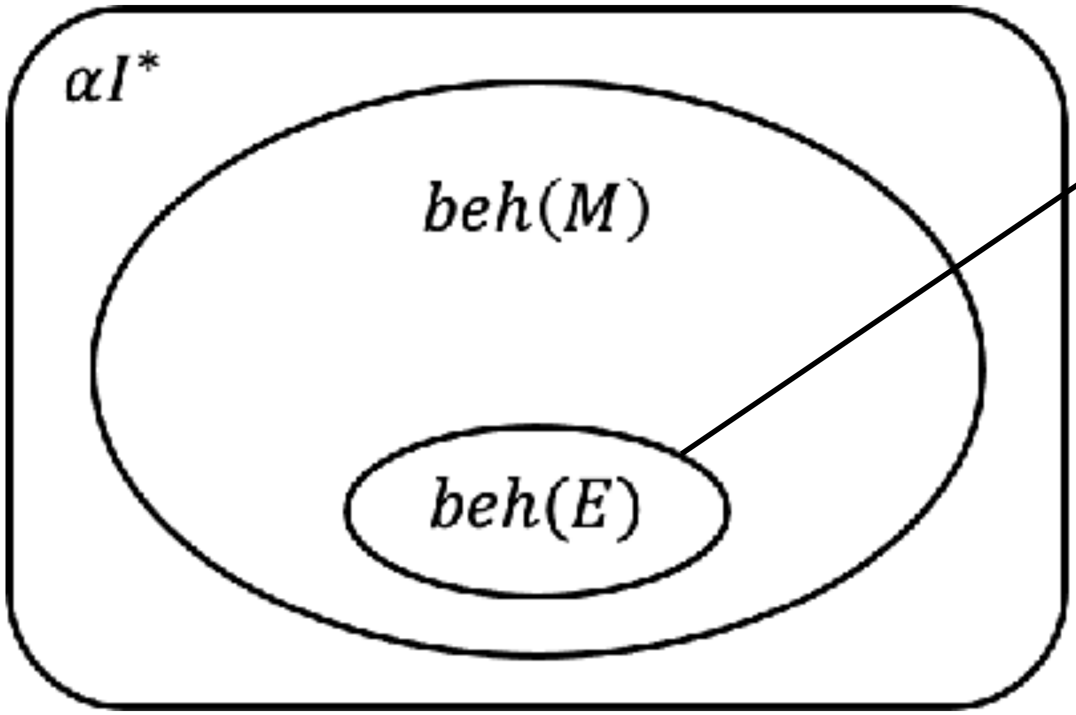
Robustness: Another View



All environmental behaviors accepted by system

⟨X, Enter, B⟩
⟨E, Enter, B⟩
⟨X, Up, E, Enter, B⟩
⟨E, Up, X, Enter, B⟩...

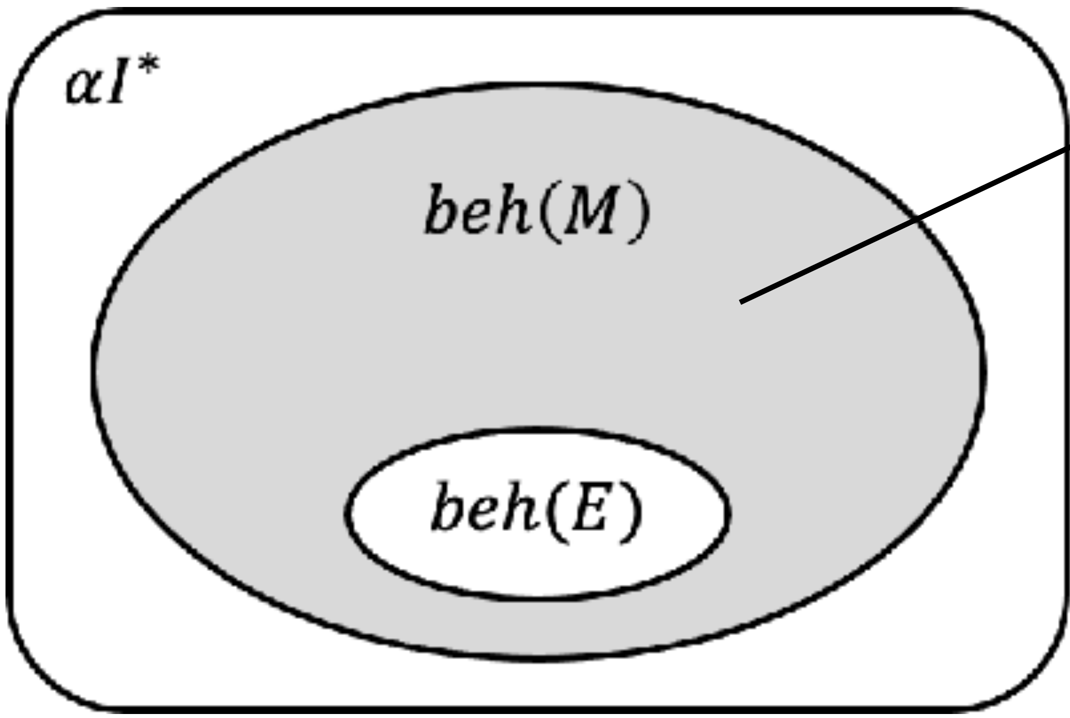
Robustness: Another View



Normative (i.e., expected) environment behavior
⟨**X, Enter, B**⟩
⟨**E, Enter, B**⟩

System satisfies its property ("no overdose") under these env. behaviors

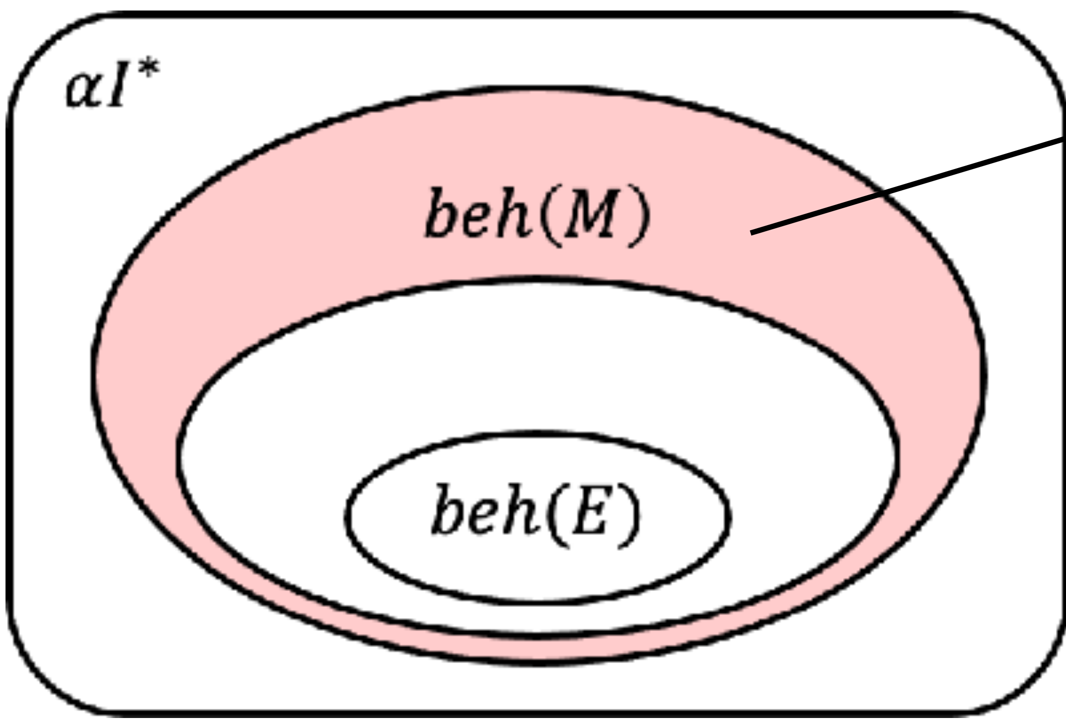
Robustness: Another View



System may or may not satisfy its property under these deviations

⟨X, Up, E, Enter, B⟩
⟨E, Up, X, Enter, B⟩...

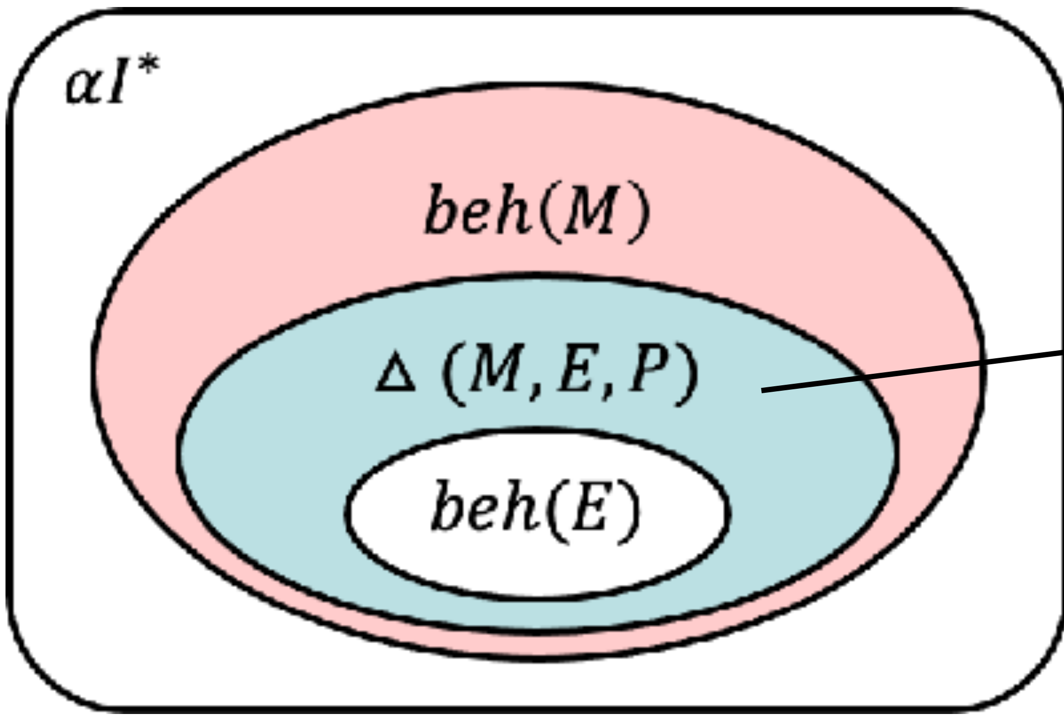
Robustness: Another View



System may violate property under these behaviors (called intolerable deviations)

⟨X, Up, E, Enter, B⟩

Robustness: Defintion



Robustness $\Delta(M, E, P)$
System satisfies its property under these deviations

⟨E, Up, X, Enter, B⟩...

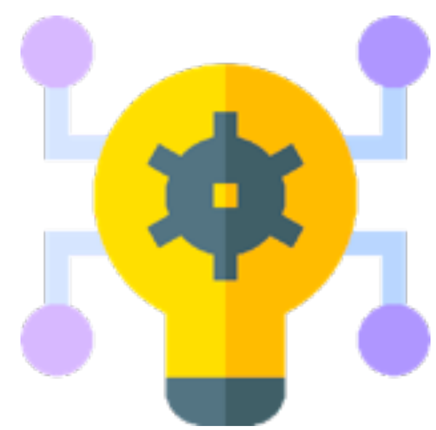
Robustness (Δ) is a computable, first-class property of a system!

Our definition enables new types of design analysis tasks

Robust Software Design: Roadmap

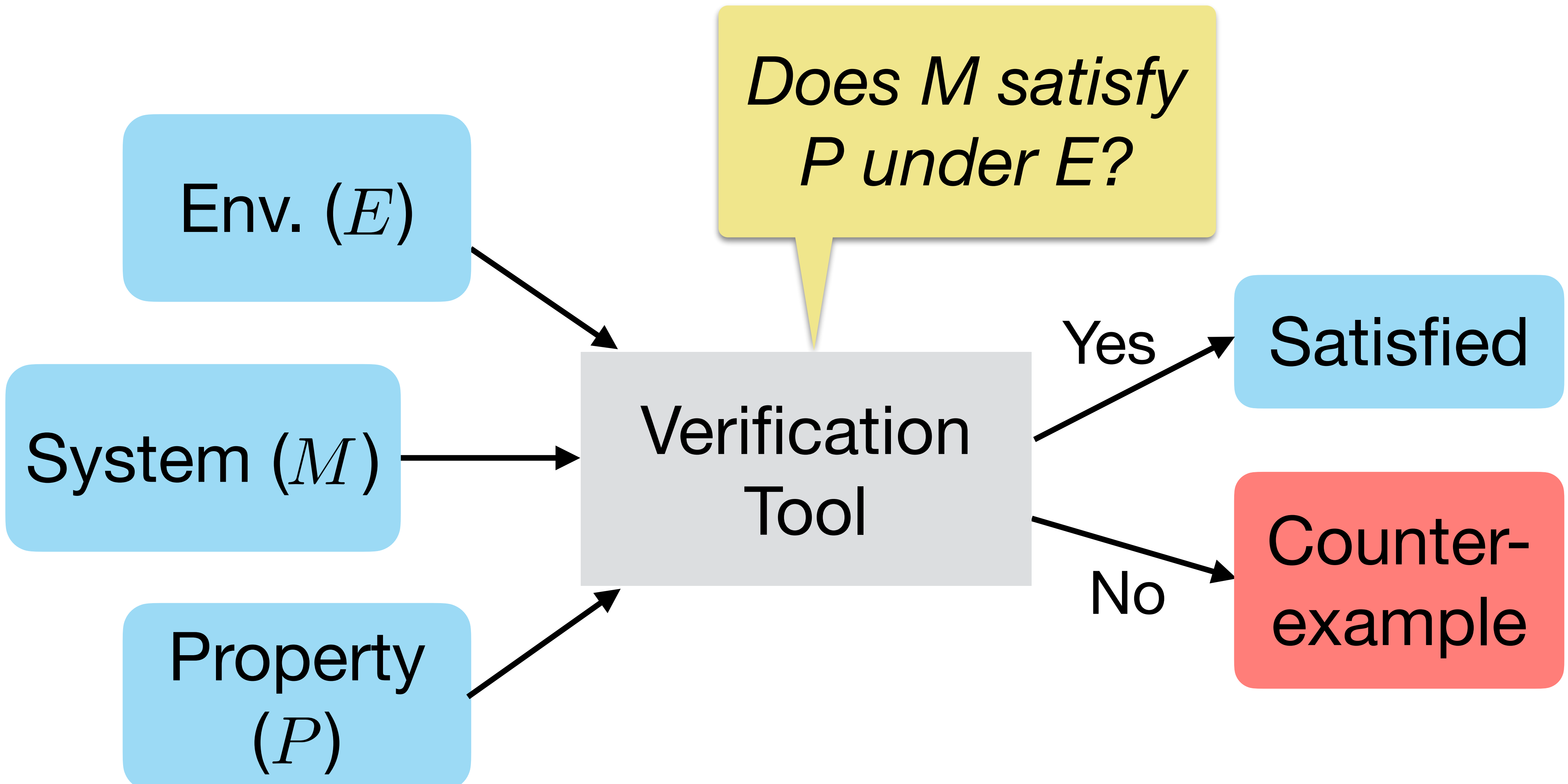
Specification
What does it mean for
our system to be robust?

Analysis
How robust is our
system?

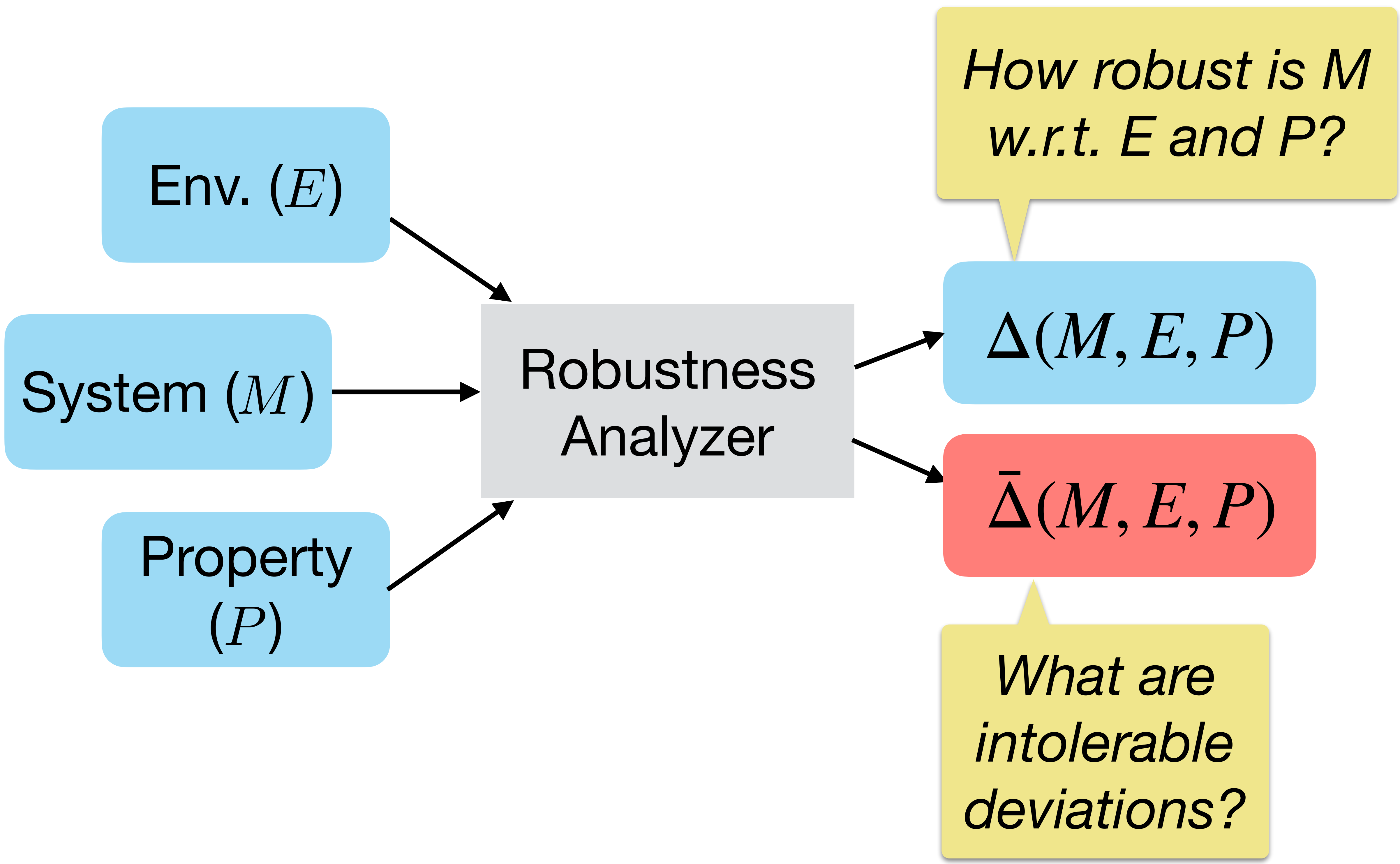


Robustification
How do we improve
its robustness?

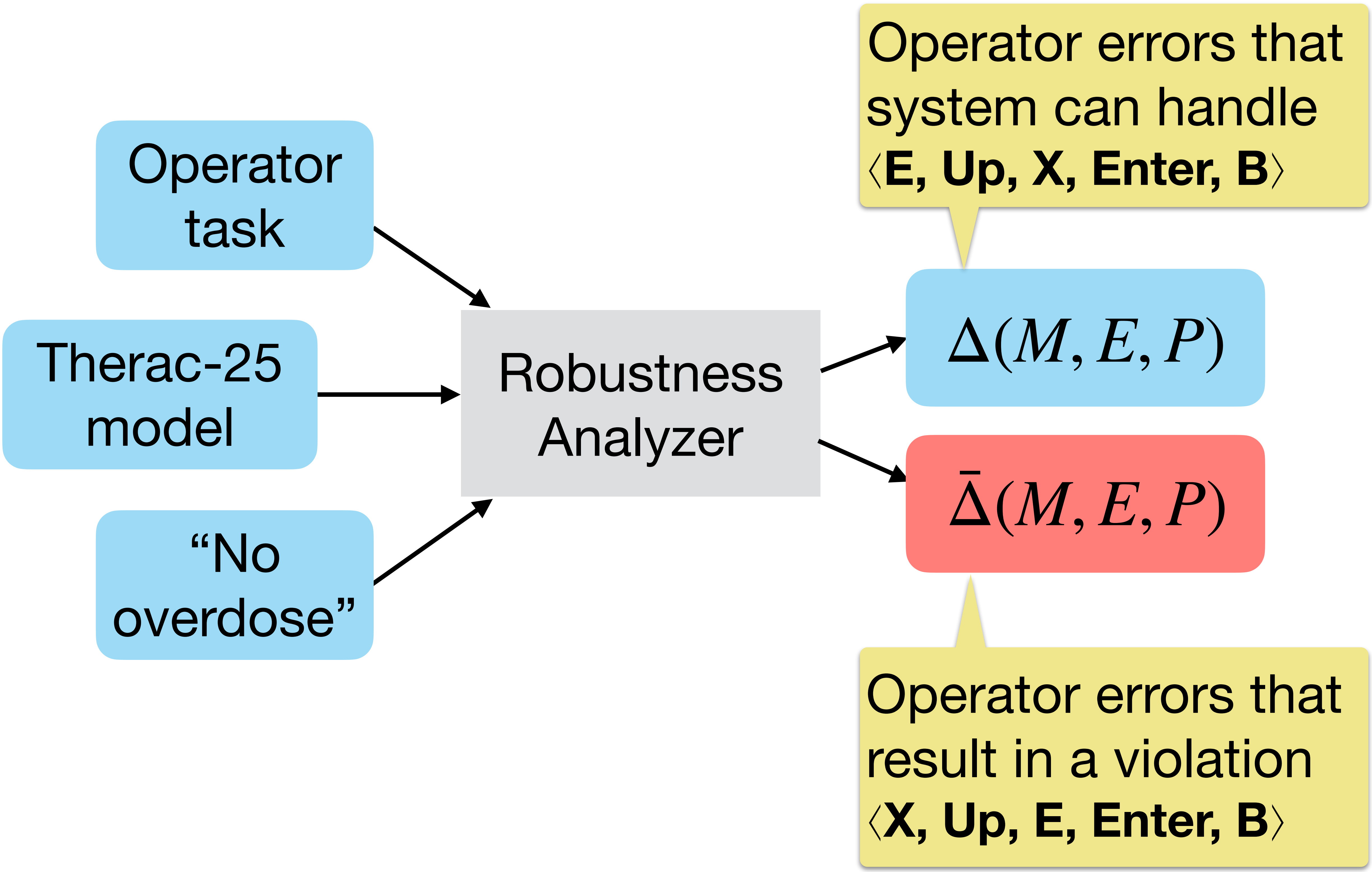
Verification Problem



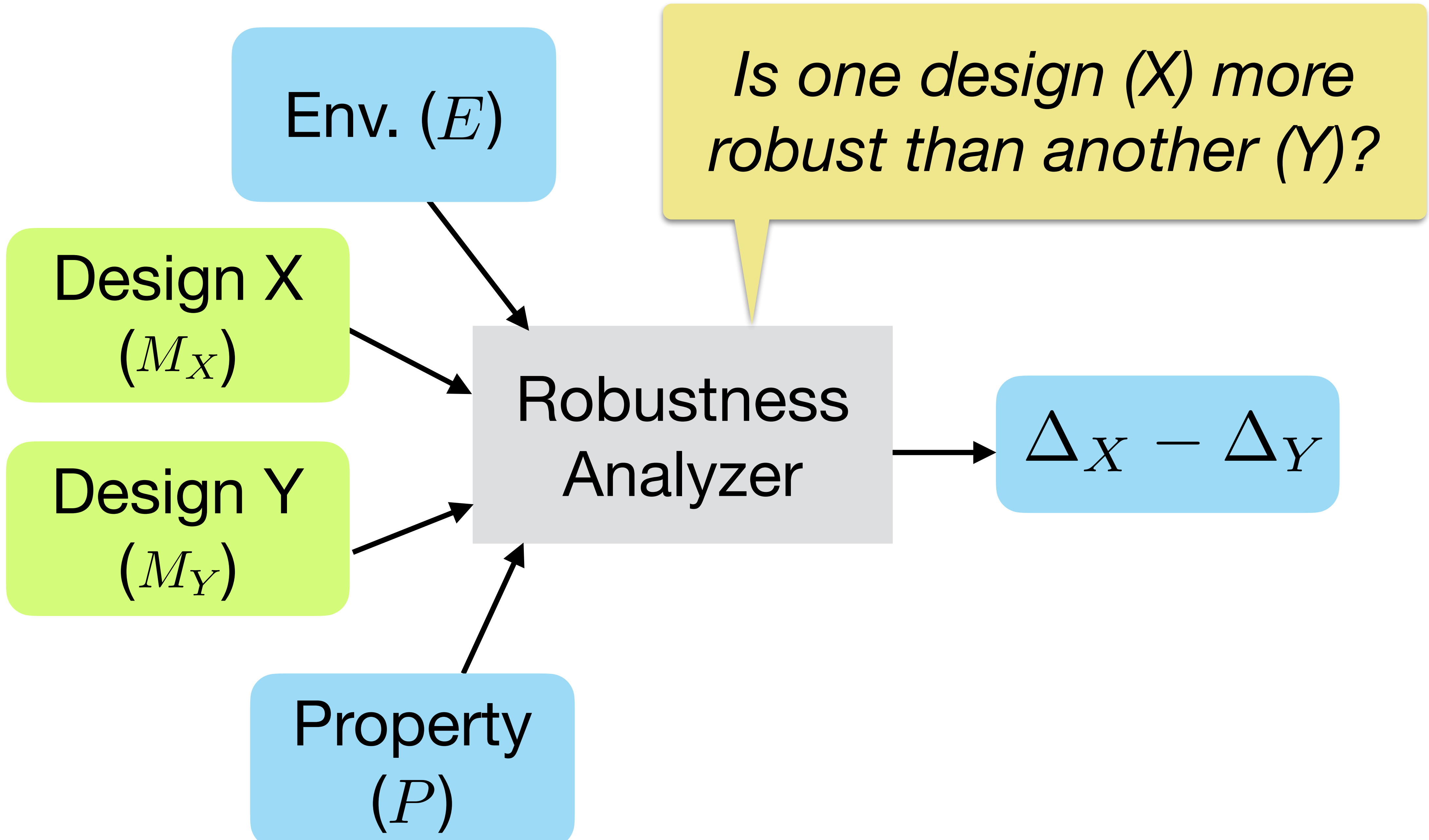
Robustness Analysis



Robustness Analysis

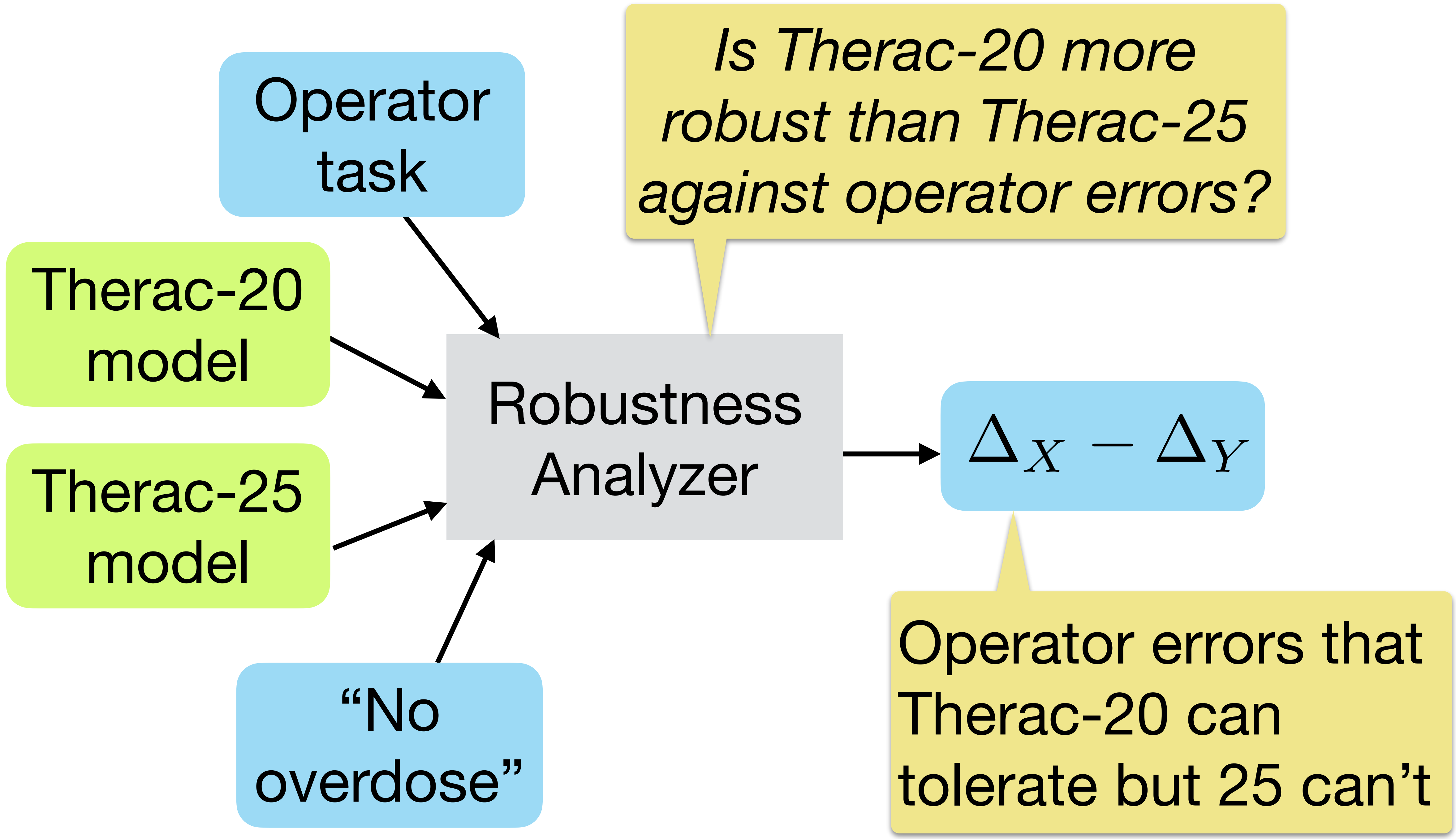


Robustness Comparison



$$\Delta_X = \Delta(M_X, E, P)$$
$$\Delta_Y = \Delta(M_Y, E, P)$$

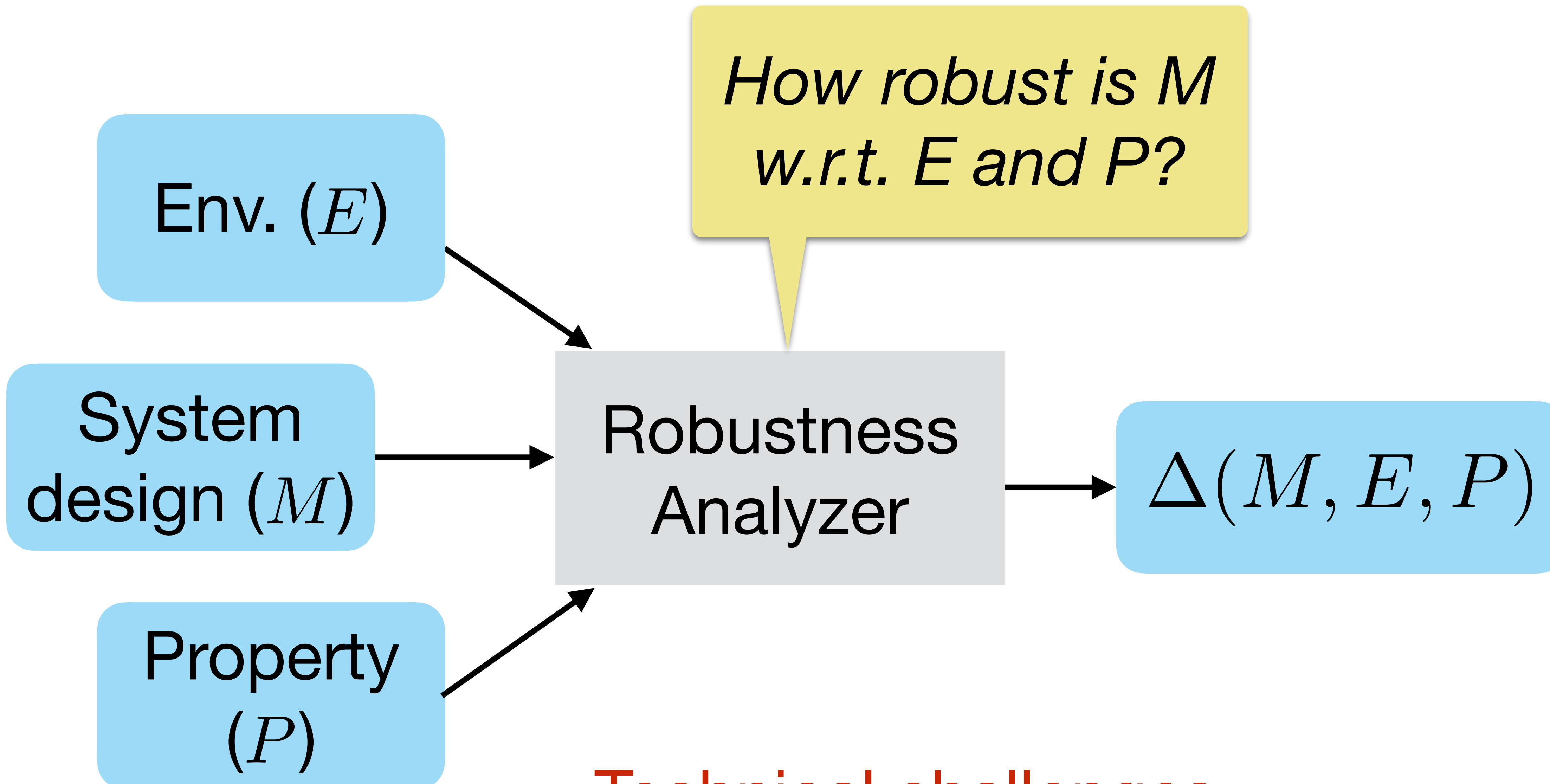
Robustness Comparison



$$\Delta_X = \Delta(M_X, E, P)$$
$$\Delta_Y = \Delta(M_Y, E, P)$$

Analyzing Robustness

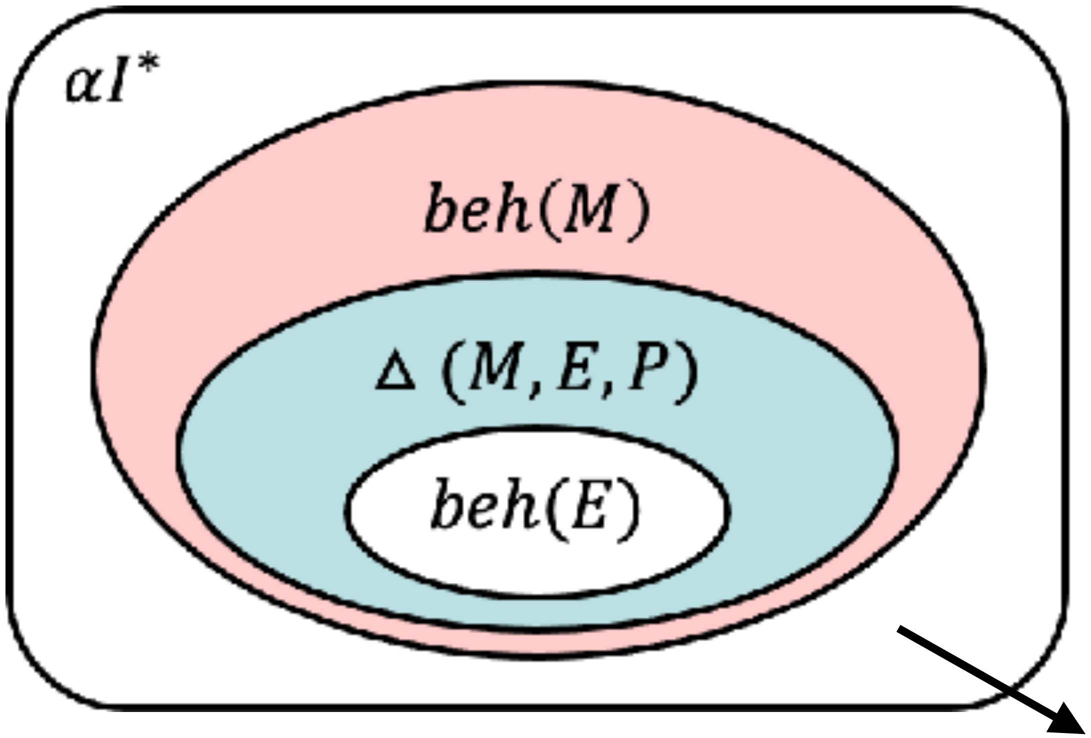
Robustness Analysis



Technical challenges

1. Computing $\Delta(M, E, P)$
2. Choosing its representation

Computing Robustness

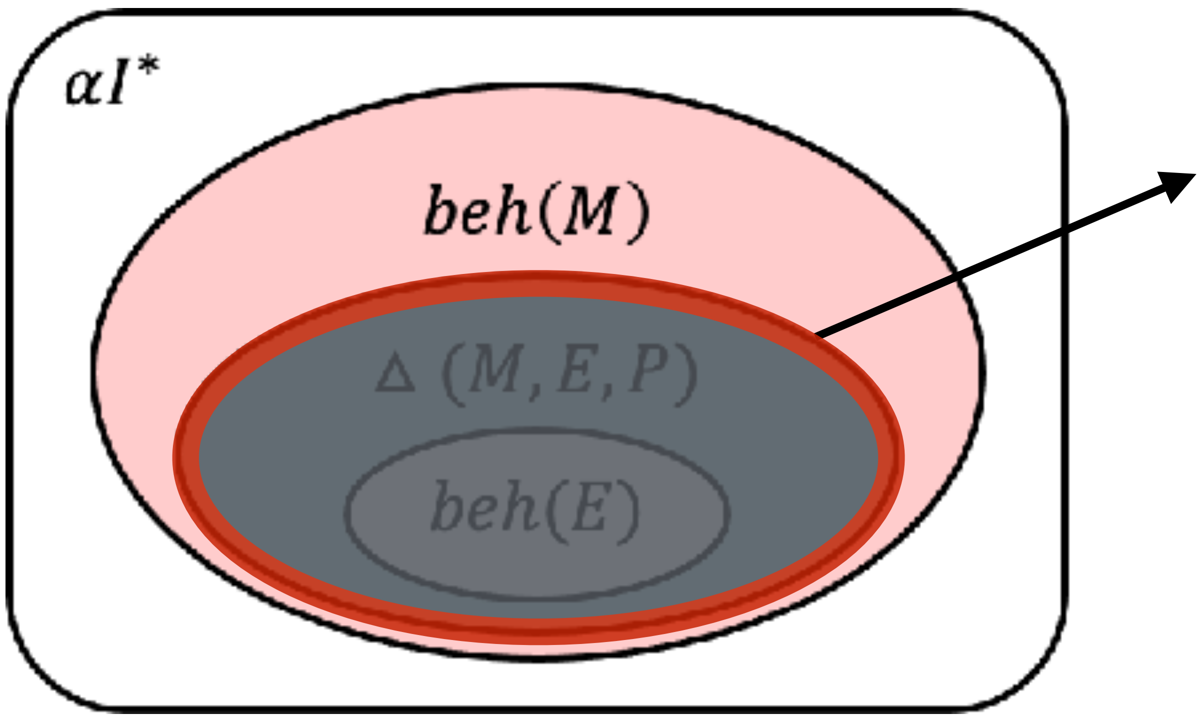


Challenge #1

Infinite number of possible deviations

How do we find a maximum set that the system is robust against?

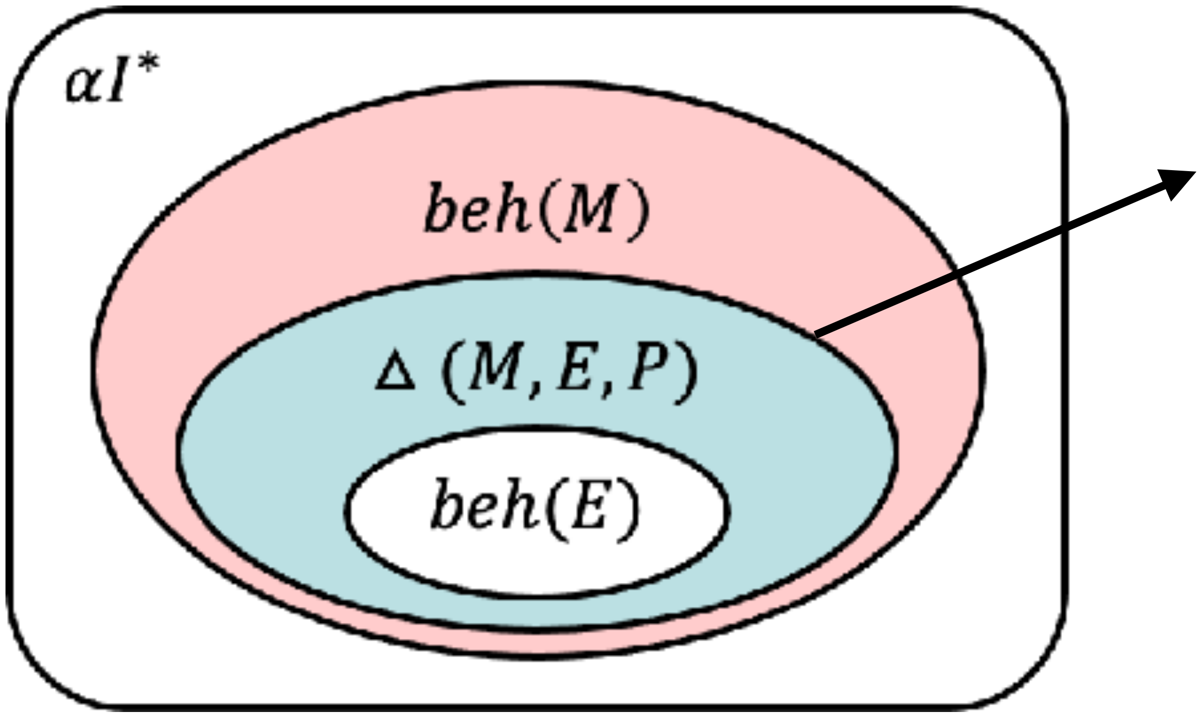
Computing Robustness



Weakest assumption (WA)
Set of all environmental behaviors under which M satisfies P

Assumption generation for software component verification. Giannakopoulou, Pasareanu, and Barringer. ASE 2003.

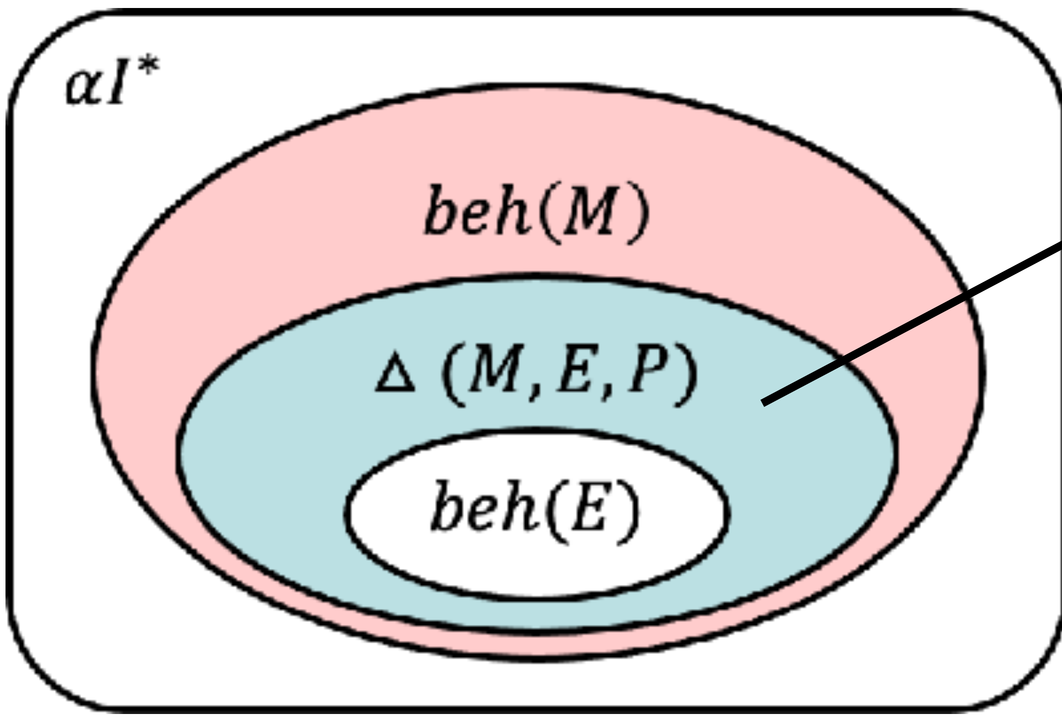
Computing Robustness



Weakest assumption (WA)
Set of all environmental behaviors under which M satisfies P

1. Compute WA using assumption generation method
2. Compute the difference over E (i.e., $WA - E$)

Representing Robustness

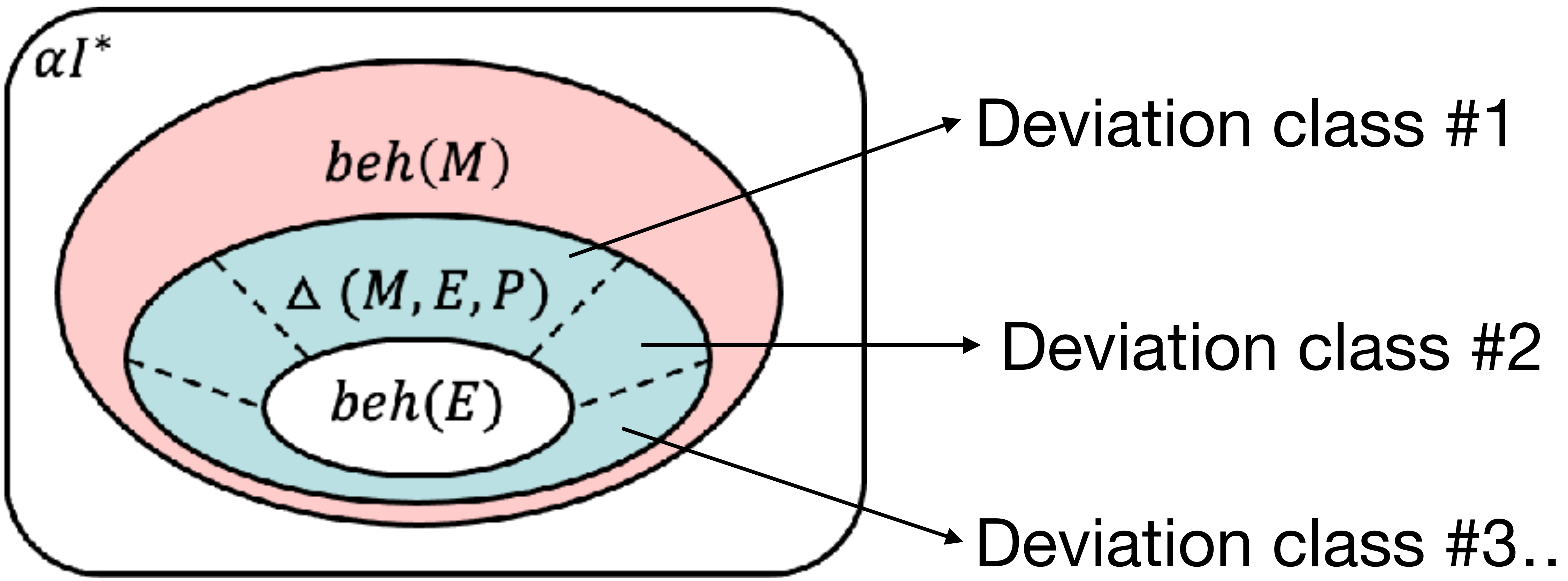


Challenge #2

Δ is an infinite set of traces

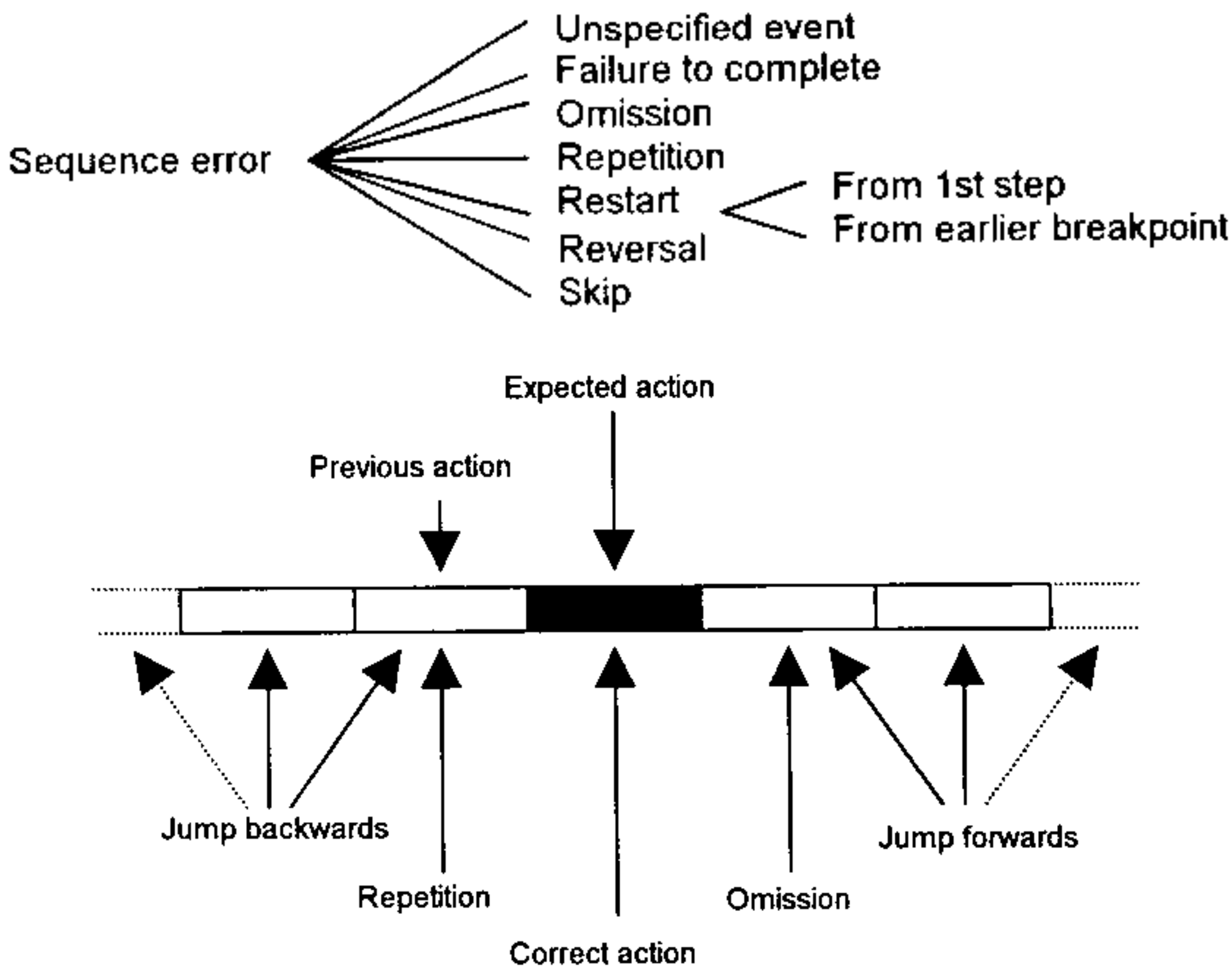
How do we represent this information to the designer?

Trace Partitioning



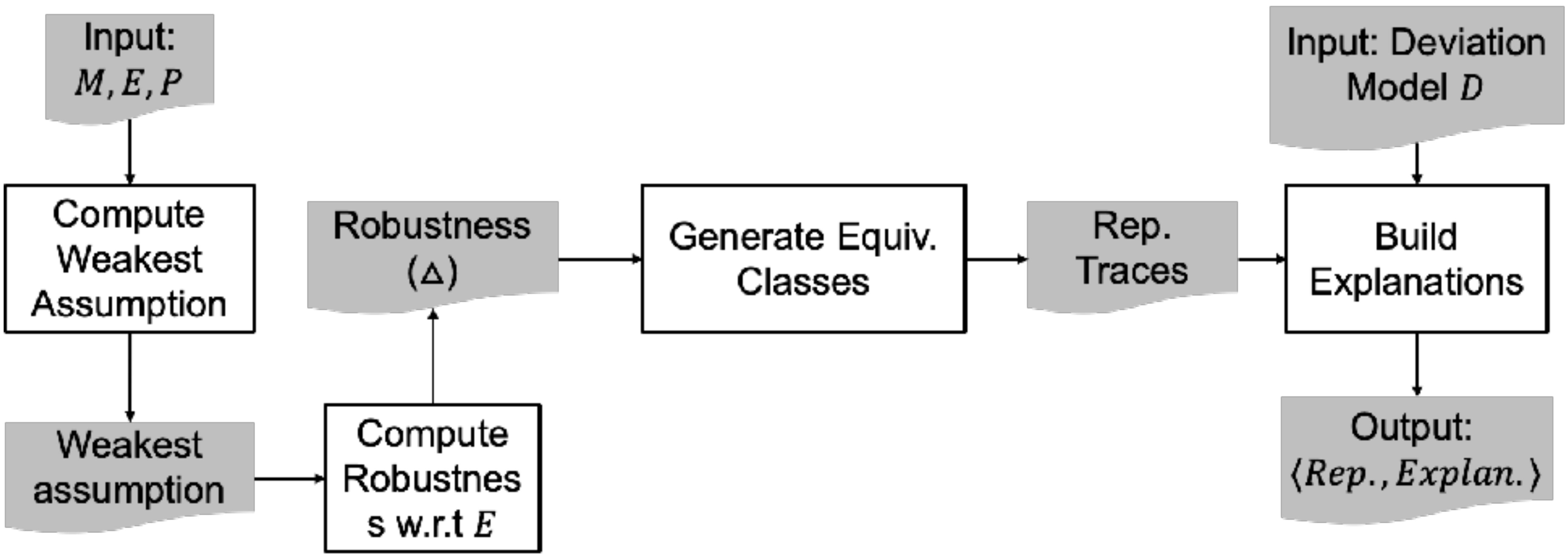
Group Δ into a finite number of equivalence classes
Each class represents a particular type of deviation (e.g., omission, repetition, intrusion error...)

Deviation Patterns in Human Errors



The phenotypes of erroneous actions. E. Hollnagel. Int. J. Man-Machine Studies (1993)

Analysis Process



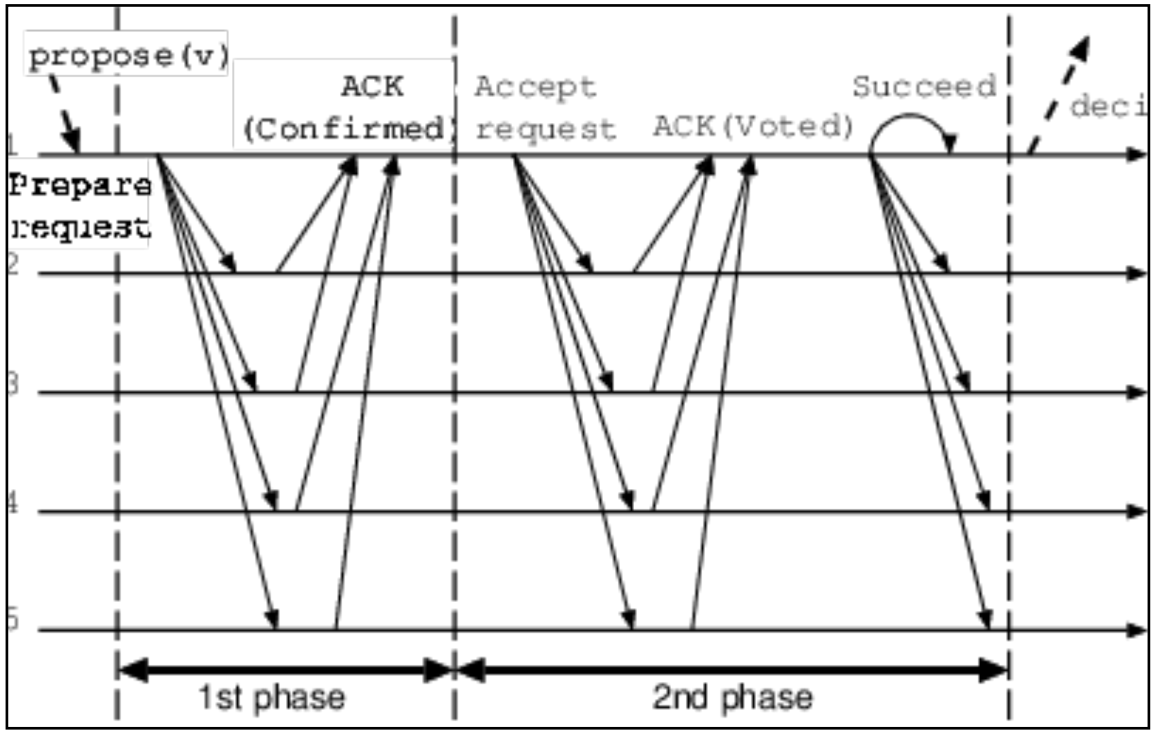
More details in our paper!

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

Analysis Case Studies

Safety-critical interfaces

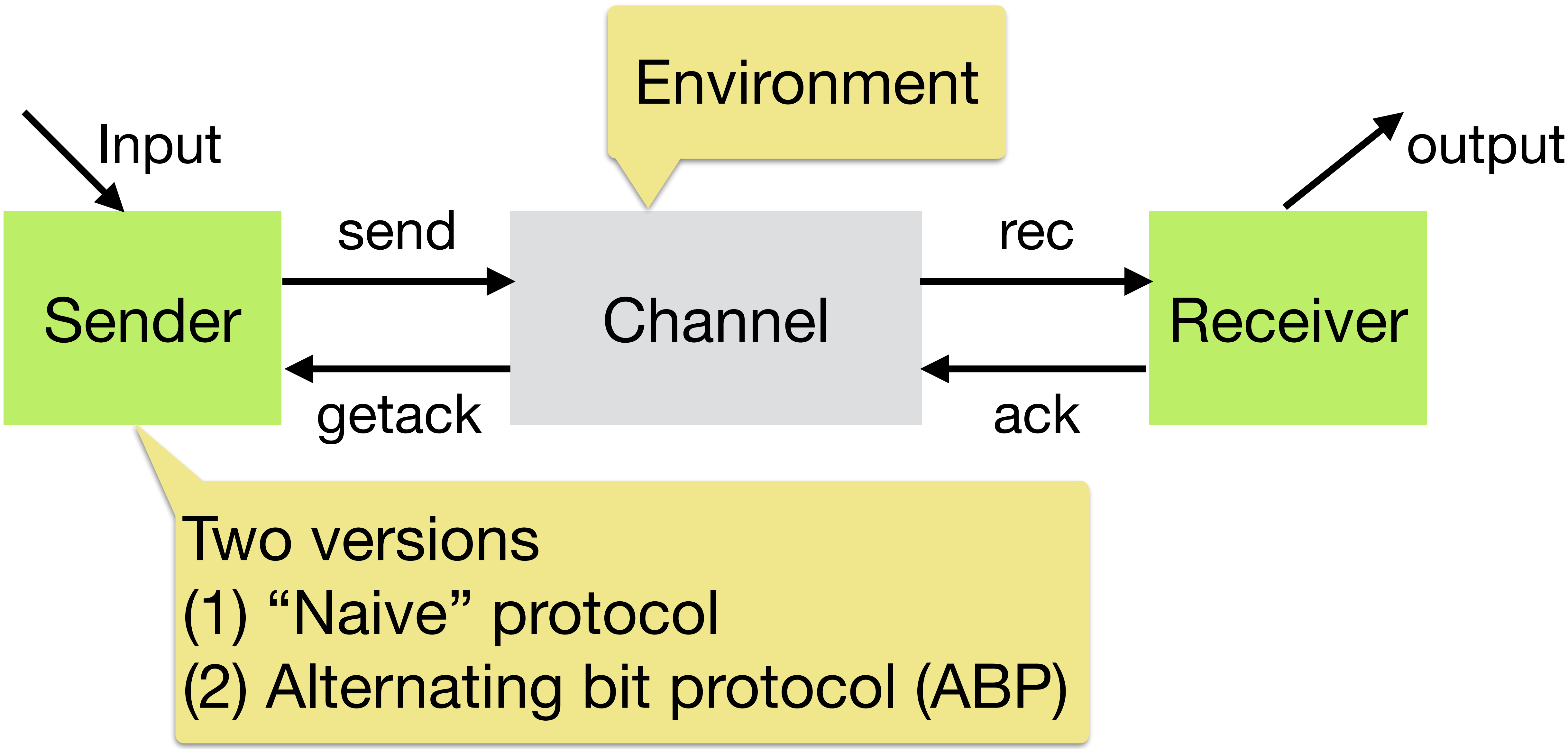
Robustness against human errors



Network protocols

Robustness against unreliable network faults

Network Communication Protocols



Property: “Message delivered in correct order”

Network Faults as Trace Deviations

What could happen in an unreliable channel?

Packet duplication

$\langle$**send[0], rec[0], ack[0], getack[0]**$\rangle$

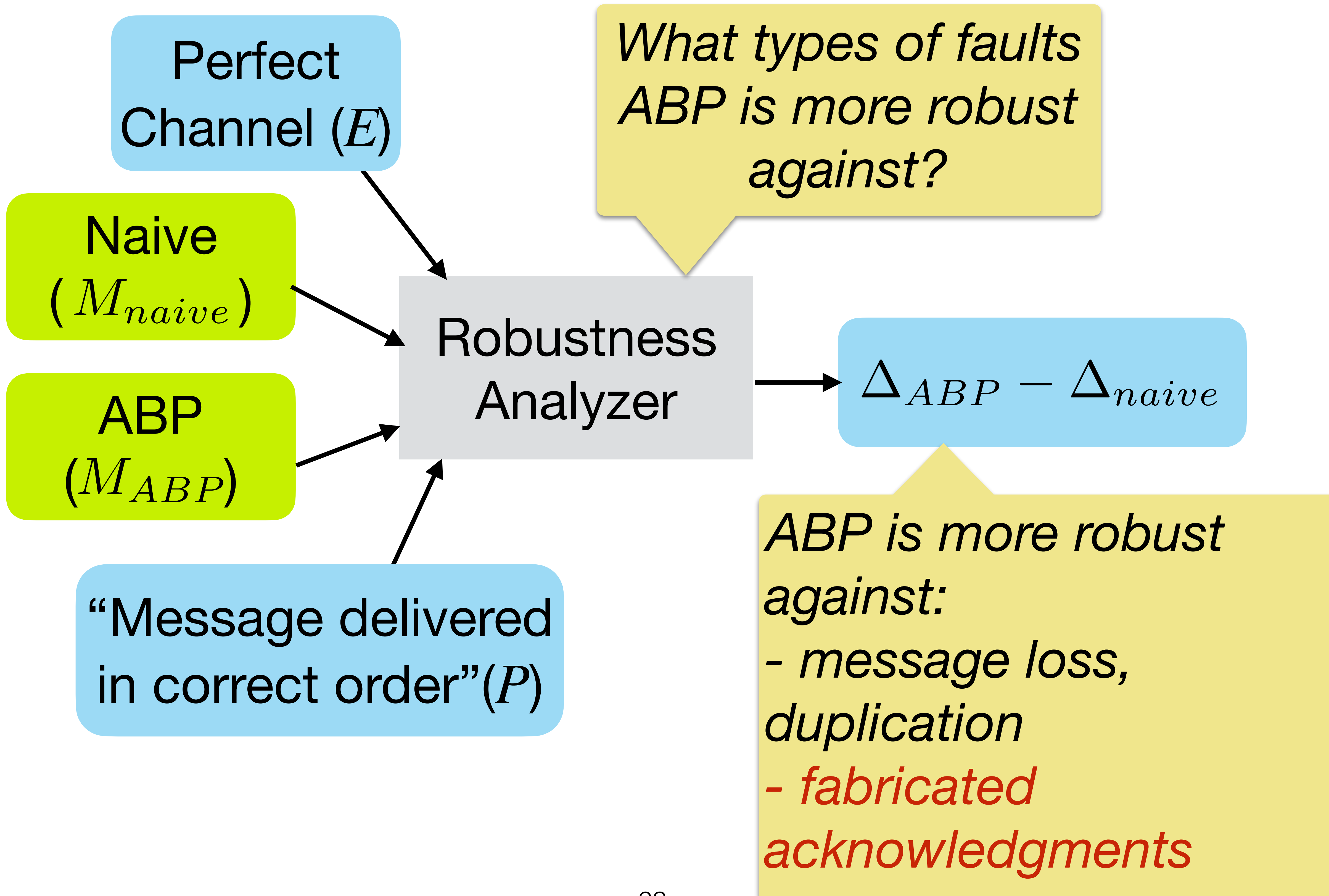
$\langle$**send[0], rec[0], rec[0], ack[0], getack[0]**$\rangle$

Packet corruption

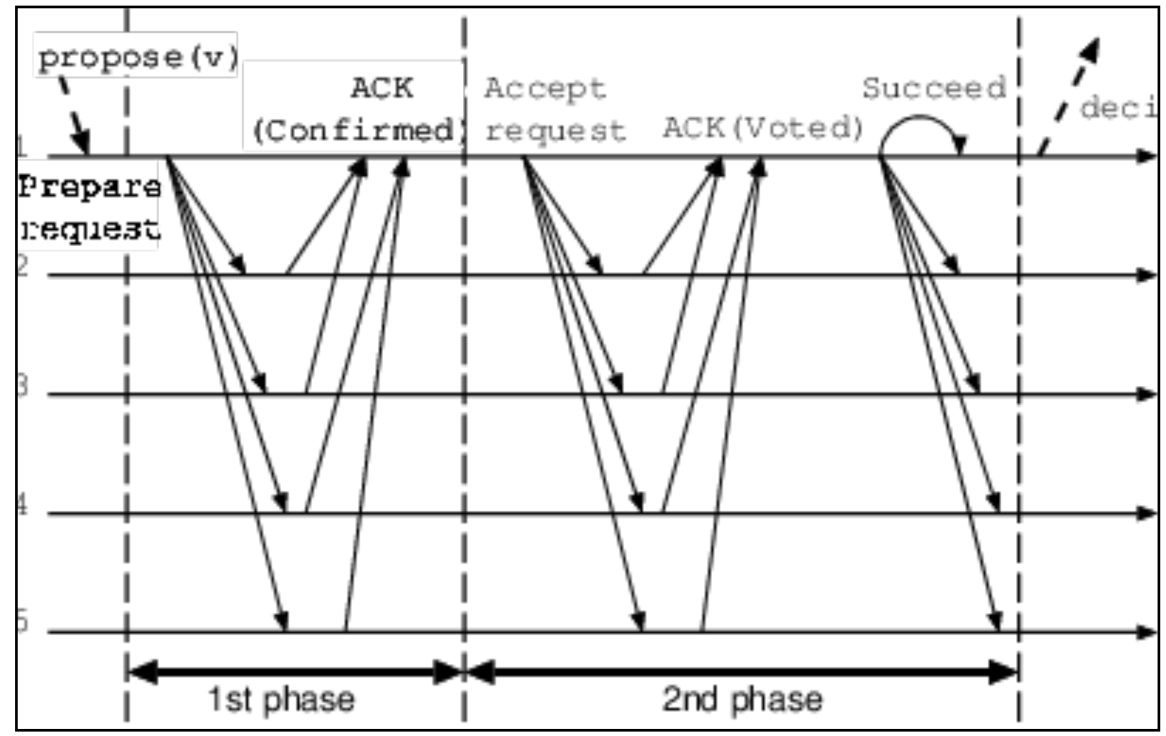
$\langle$**send[0], rec[0], ack[0], getack[0]**$\rangle$

$\langle$**send[0], rec[1], ack[0], getack[0]**$\rangle$

Comparing Network Protocols



Analysis Case Studies



Safety-critical interfaces

Robustness against human errors

Network protocols

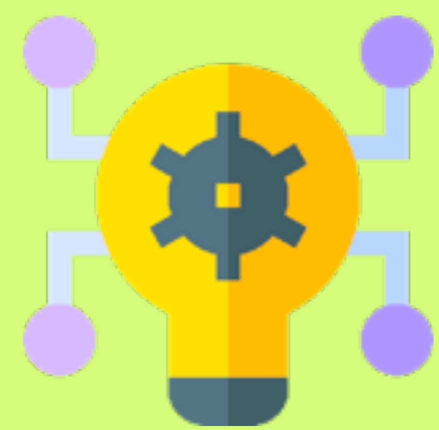
Robustness against unreliable network faults

Our definition captures deviations in multiple domains
Robustness can be computed under several seconds

Robust Software Design: Roadmap

Specification
What does it mean for
our system to be robust?

Analysis
How robust is our
system?



Robustification
How do we improve
its robustness?

Robustifying Systems

$$M \parallel E \models P$$

$$\Big\downarrow \delta$$

Intolerable deviations

$$M \parallel E' \not\models P$$

Robustification

$$M \parallel E \models P$$

Enhance

$$\Big\downarrow \delta$$

$$M' \parallel E' \not\models P$$

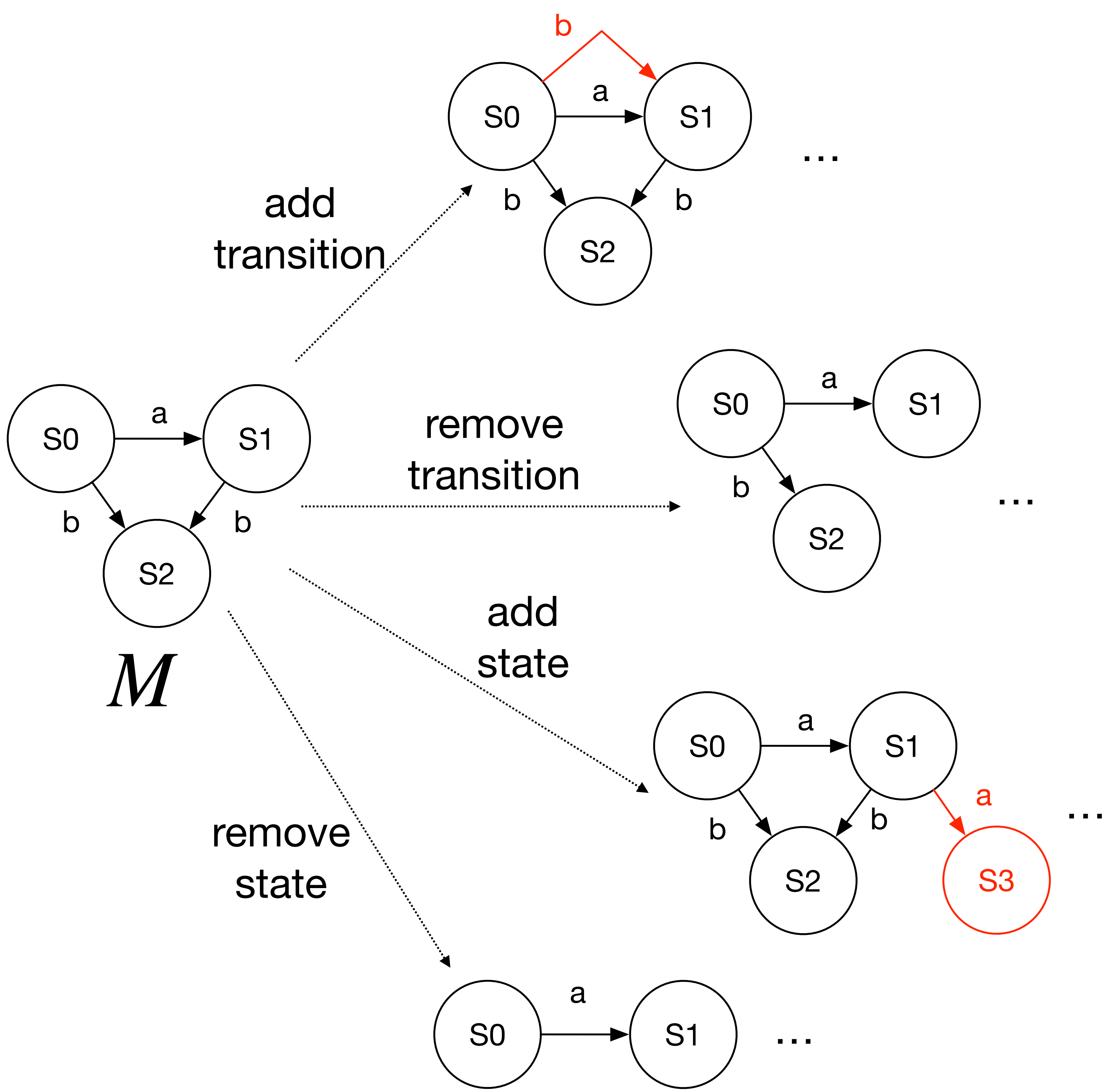
Can we generate suggestions for enhancing the original design to handle additional deviations?

Robustification

$$\begin{tikzcd}
M \parallel E \models P \arrow[d, dashed, "\delta"] \arrow[d, bend right=60, swap, "\text{Transform}"] \\
M' \parallel E' \models P
\end{tikzcd}$$

Can we generate suggestions for enhancing the original design to handle additional deviations?
Treat it as a model transformation problem!

LTS Transformation

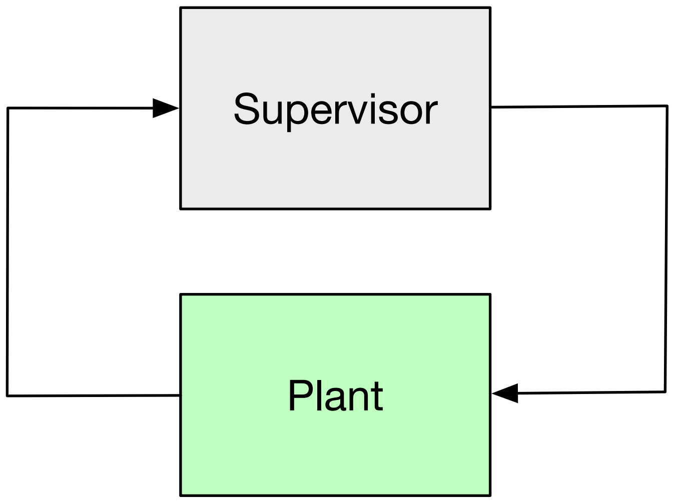


Challenge
Infinite number of possible modifications!

Robustification as Supervisory Control

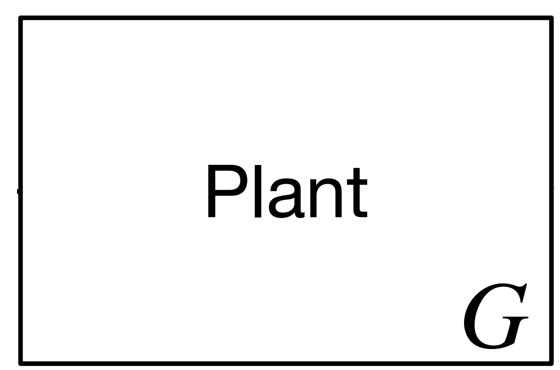
Robustification problem

reduce



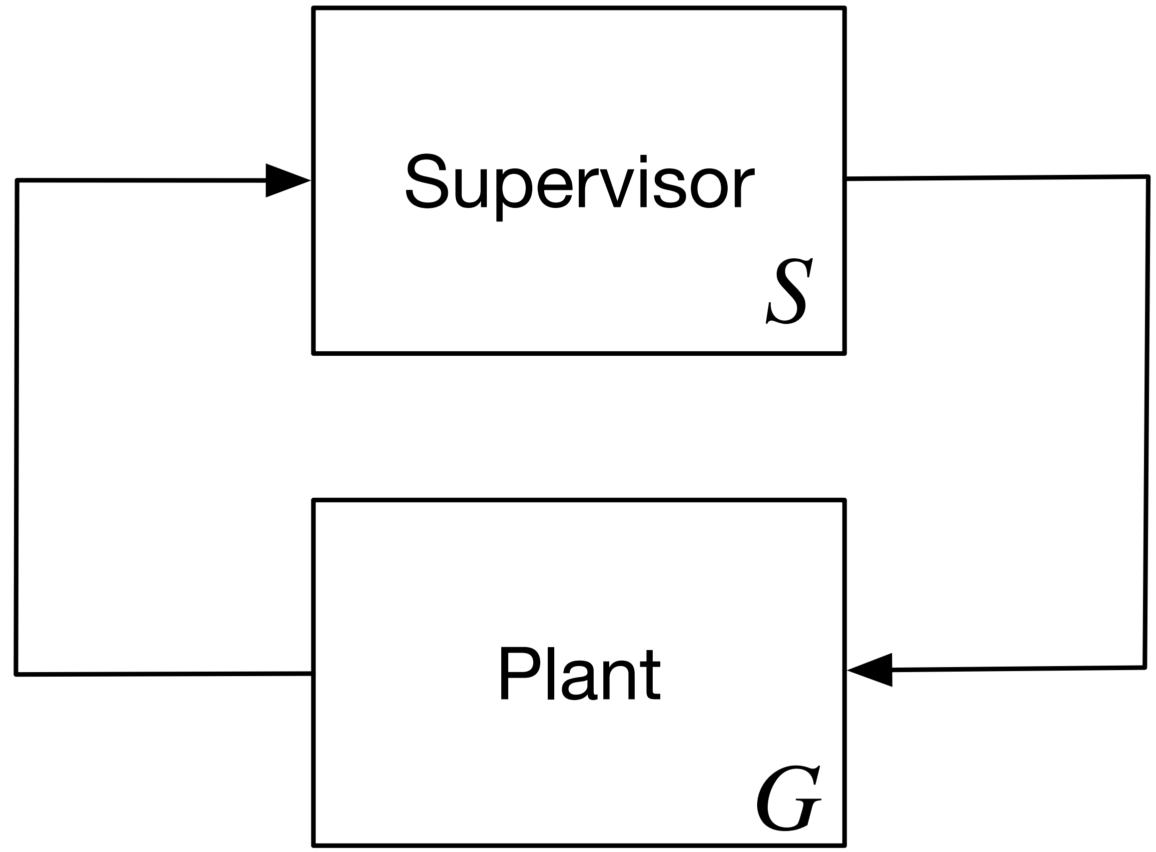
Supervisory control problem

Supervisory Control



Supervisory control of a class of discrete event processes.
Ramadge & Wonham, SIAM Journal on Control and Optimization (1987).

Supervisory Control

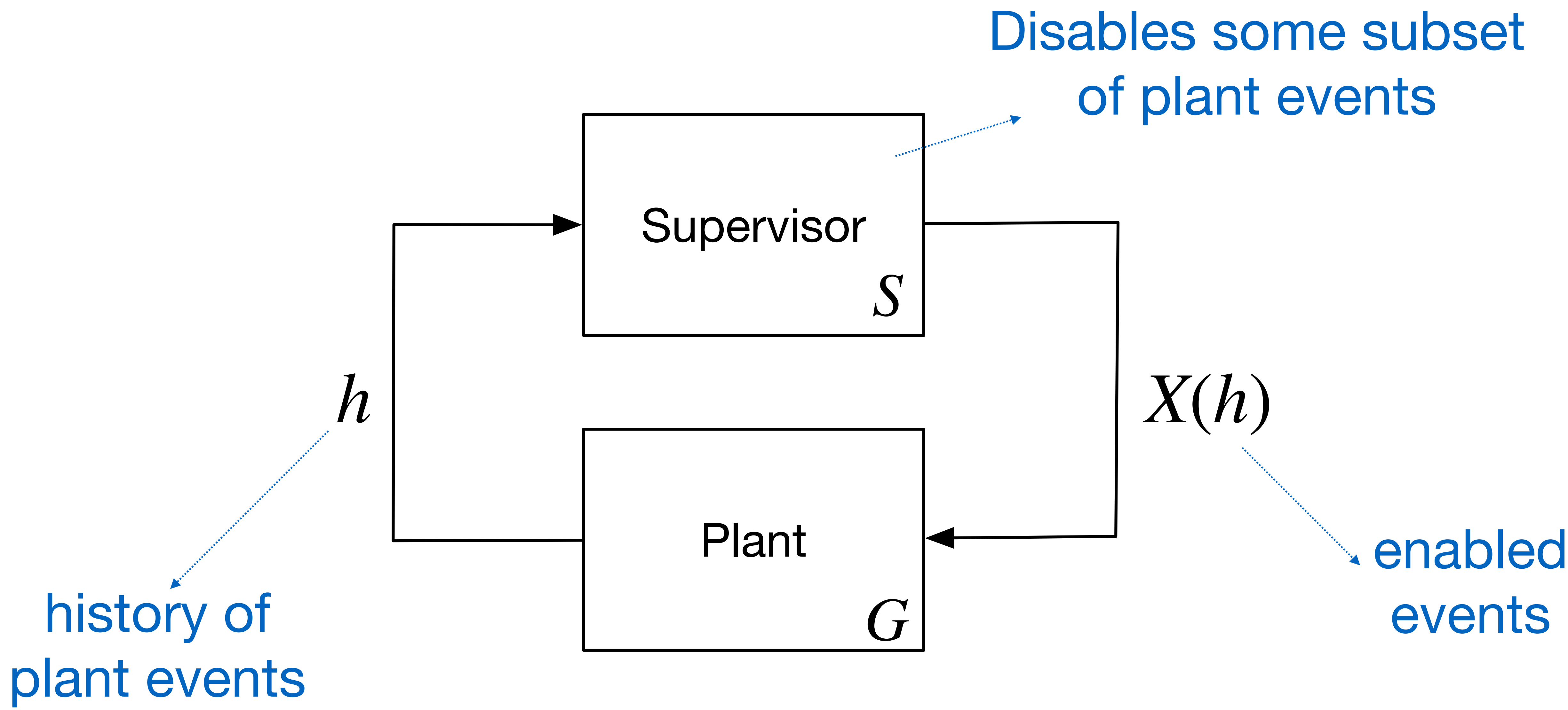


$S \| G \vDash P$

Given plant G, find supervisor S such that G under the control of S satisfies property P

Supervisory control of a class of discrete event processes.
Ramadge & Wonham, SIAM Journal on Control and Optimization (1987).

Supervisory Control

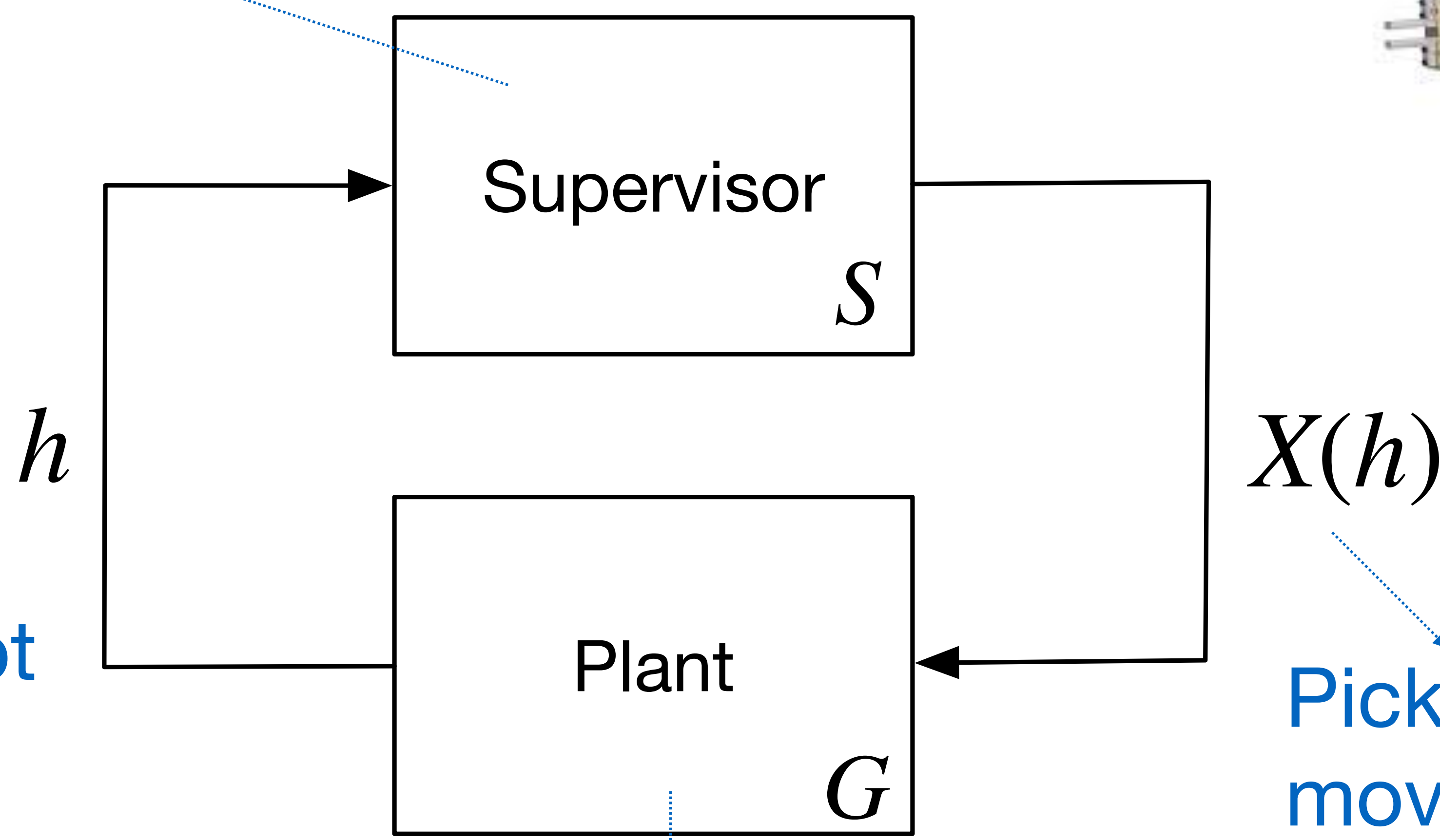


Based on plant's past behavior, supervisor restricts the set of events that it is allowed to perform

Supervisory Control: Applications

What's safe to do in the current conveyor belt state?

P: "No incorrect assembly"



past robot actions

Pick up item, move arm, …

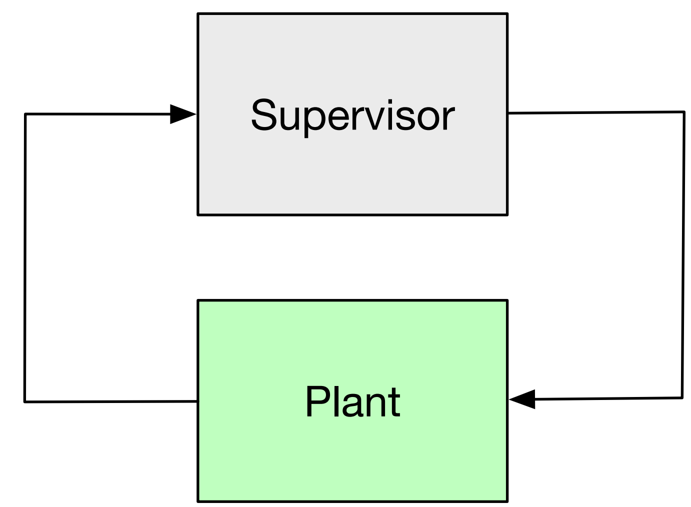
Manufacturing robot

Others: Network security, concurrency control, protocol synthesis…

Robustification as Supervisory Control

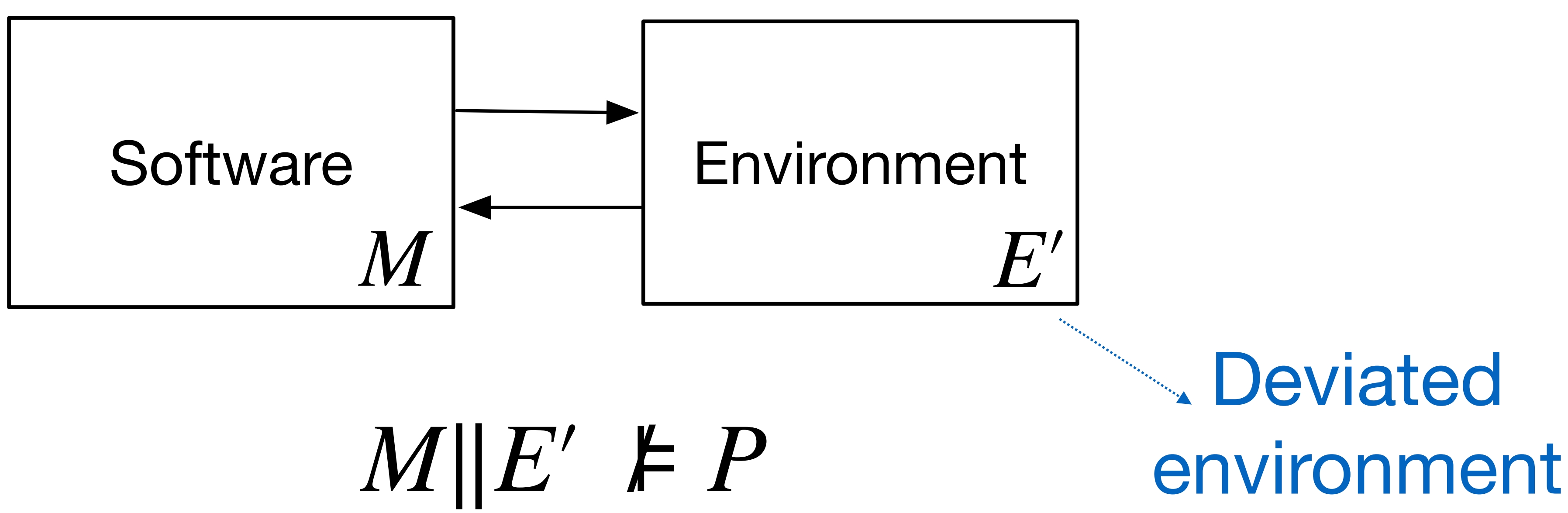
reduce

Robustification problem



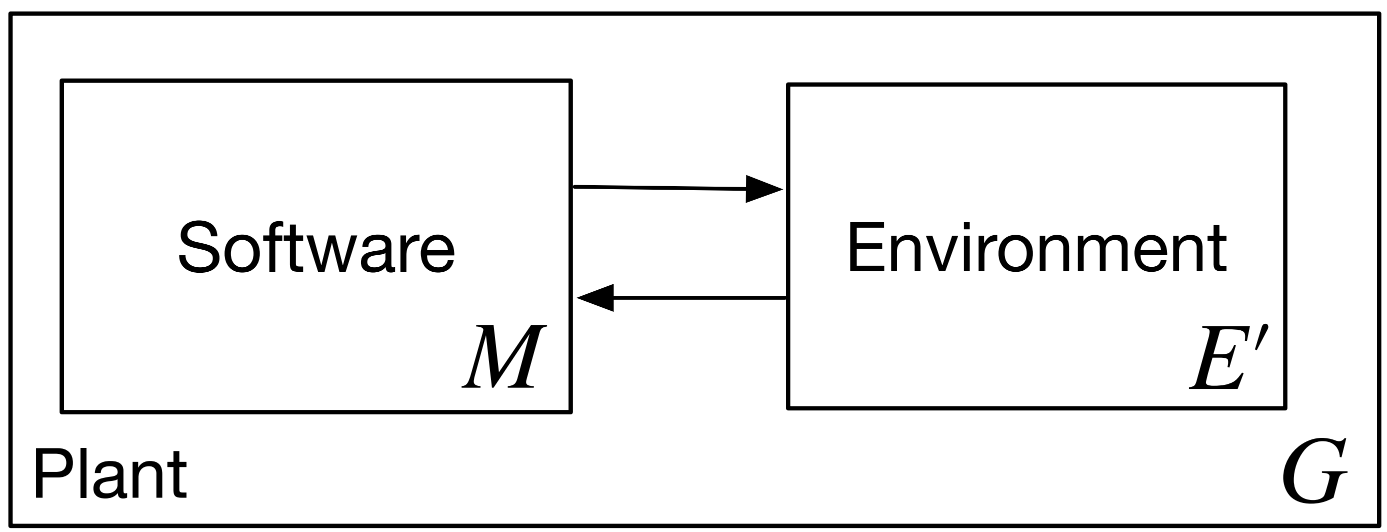
Supervisory control problem

Robustification as Supervisory Control

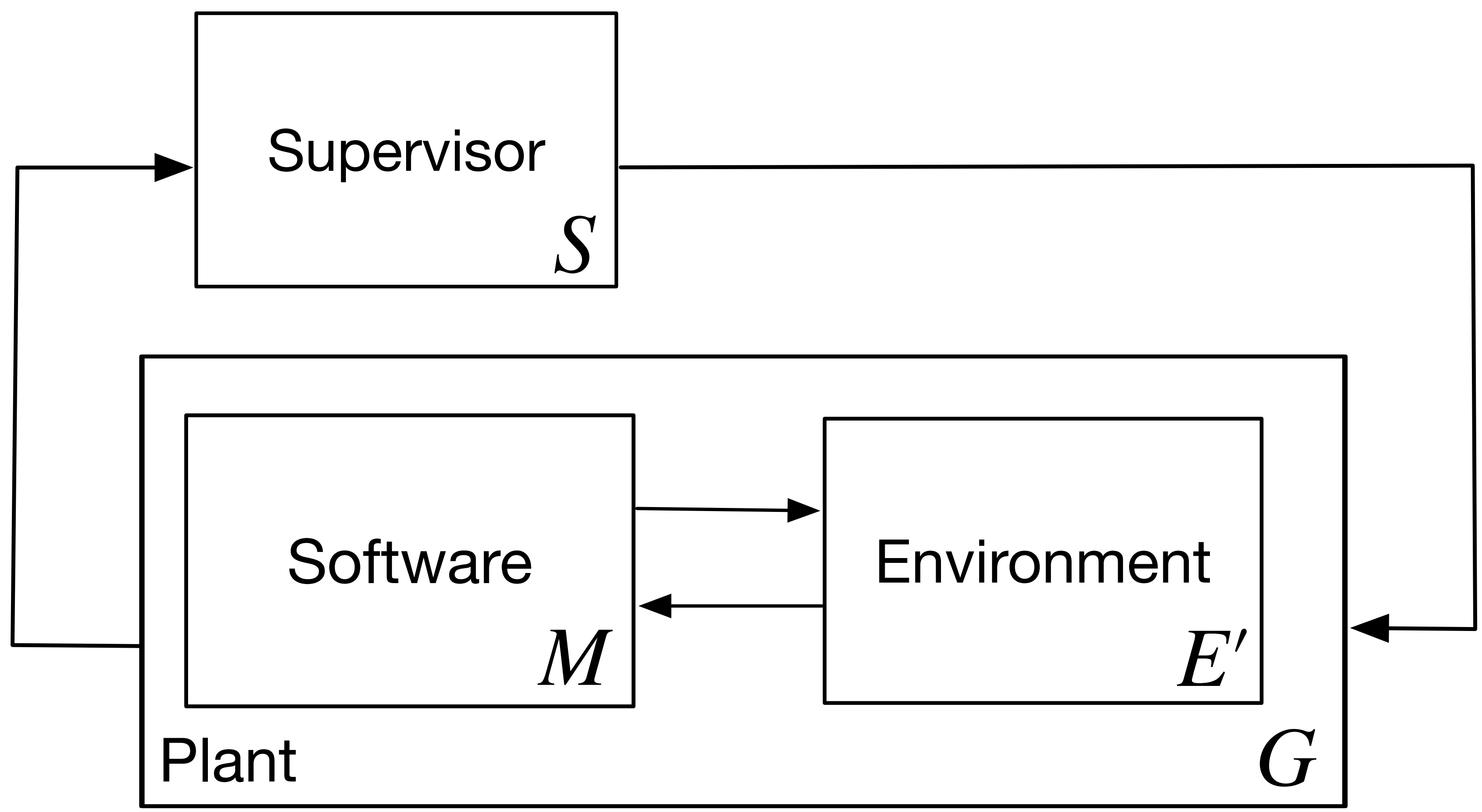


$$M \| E' \not\models P$$

Robustification as Supervisory Control



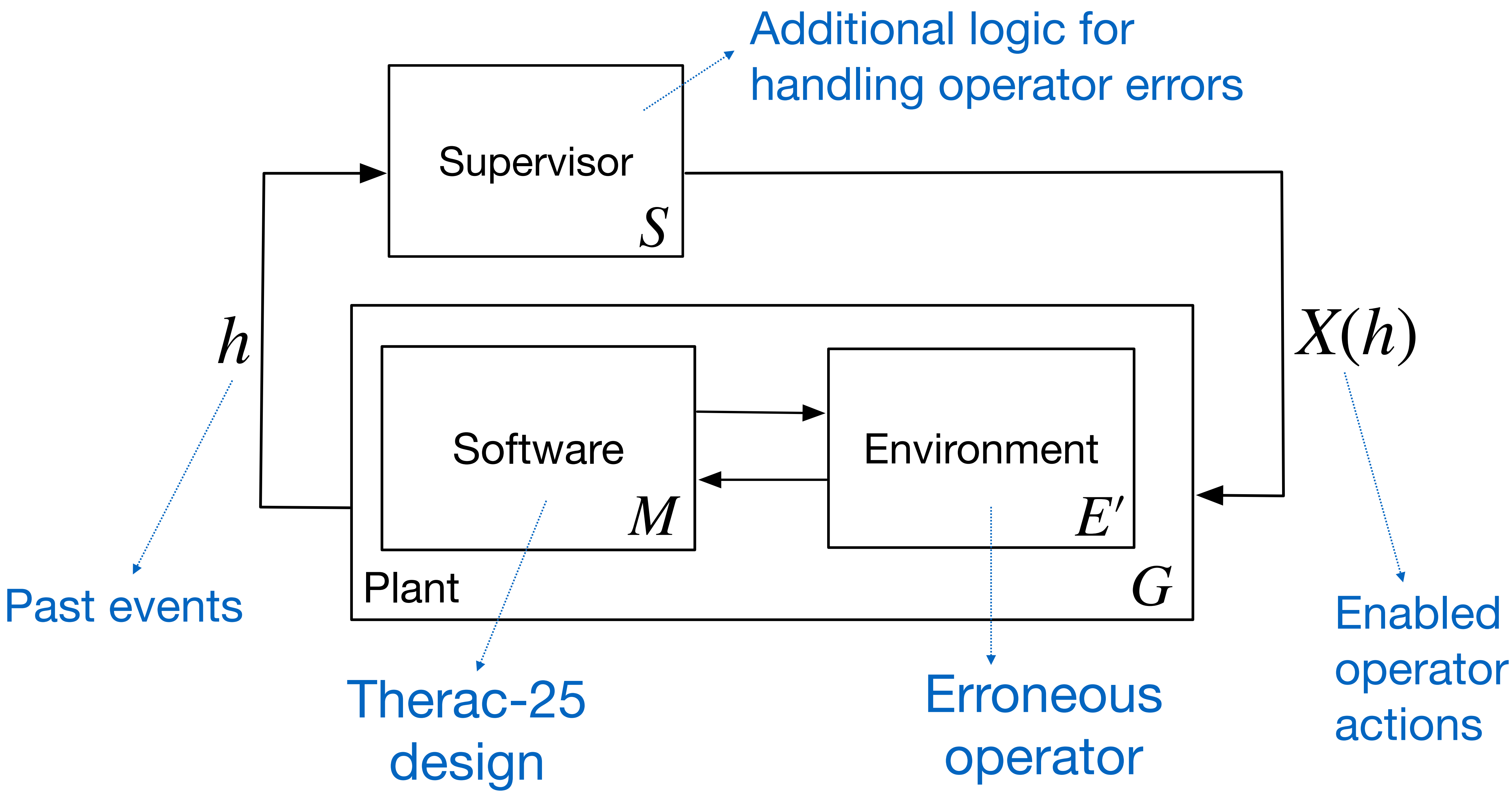
Robustification as Supervisory Control



$$S \| G \models P$$

$$\iff S \| (M \| E') \models P$$

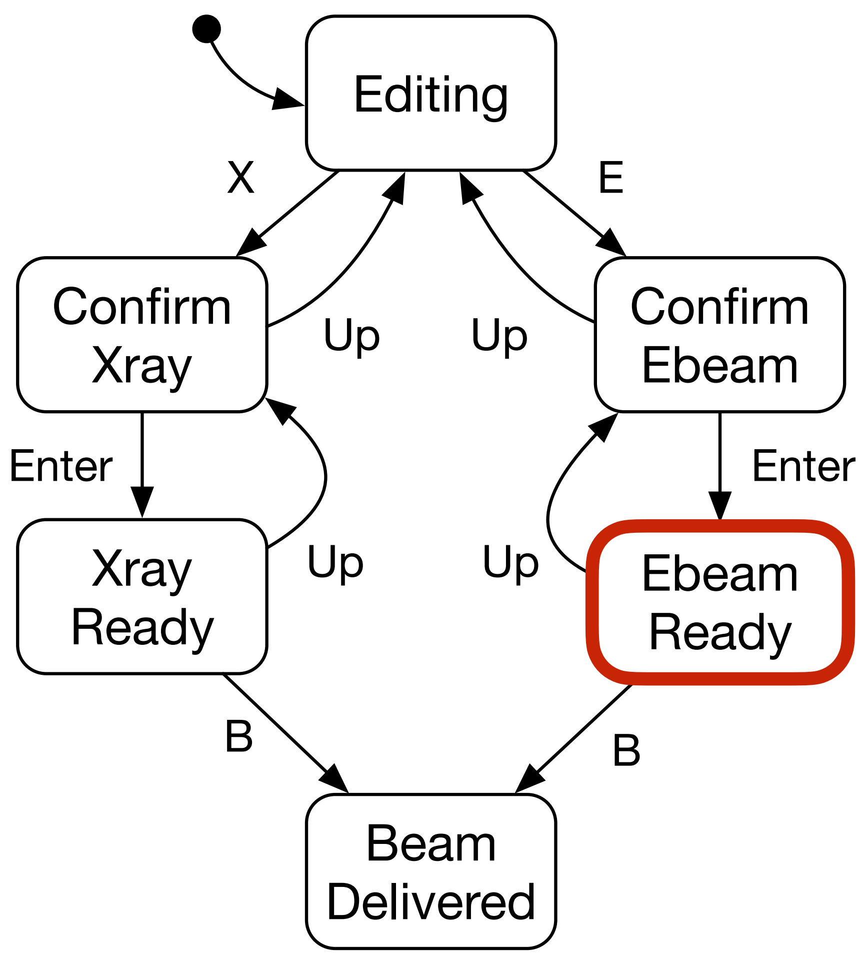
Therac-25 Example



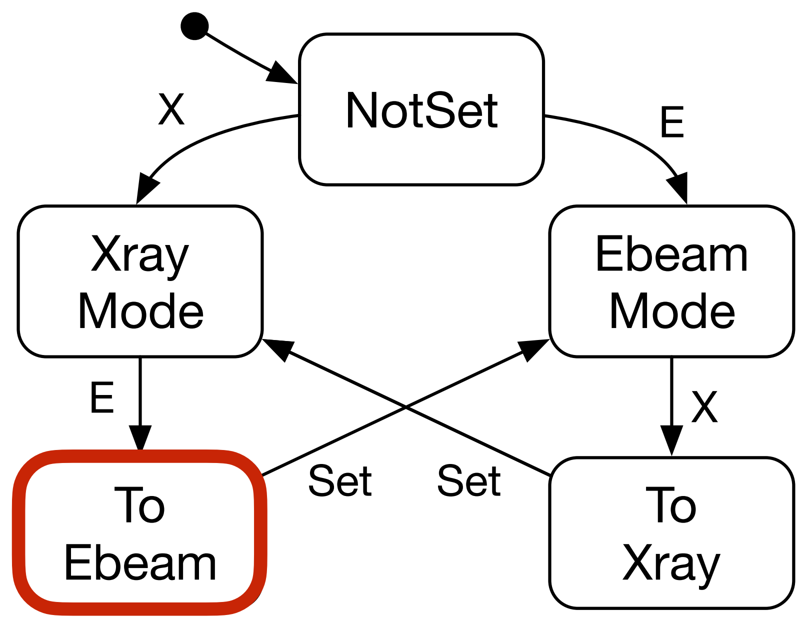
P: “No overdose”

Therac-25 Example

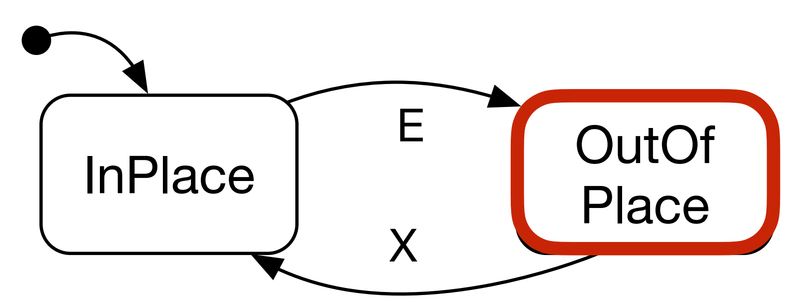
Recall: Safety violation under erroneous operator



Interface (M_I)

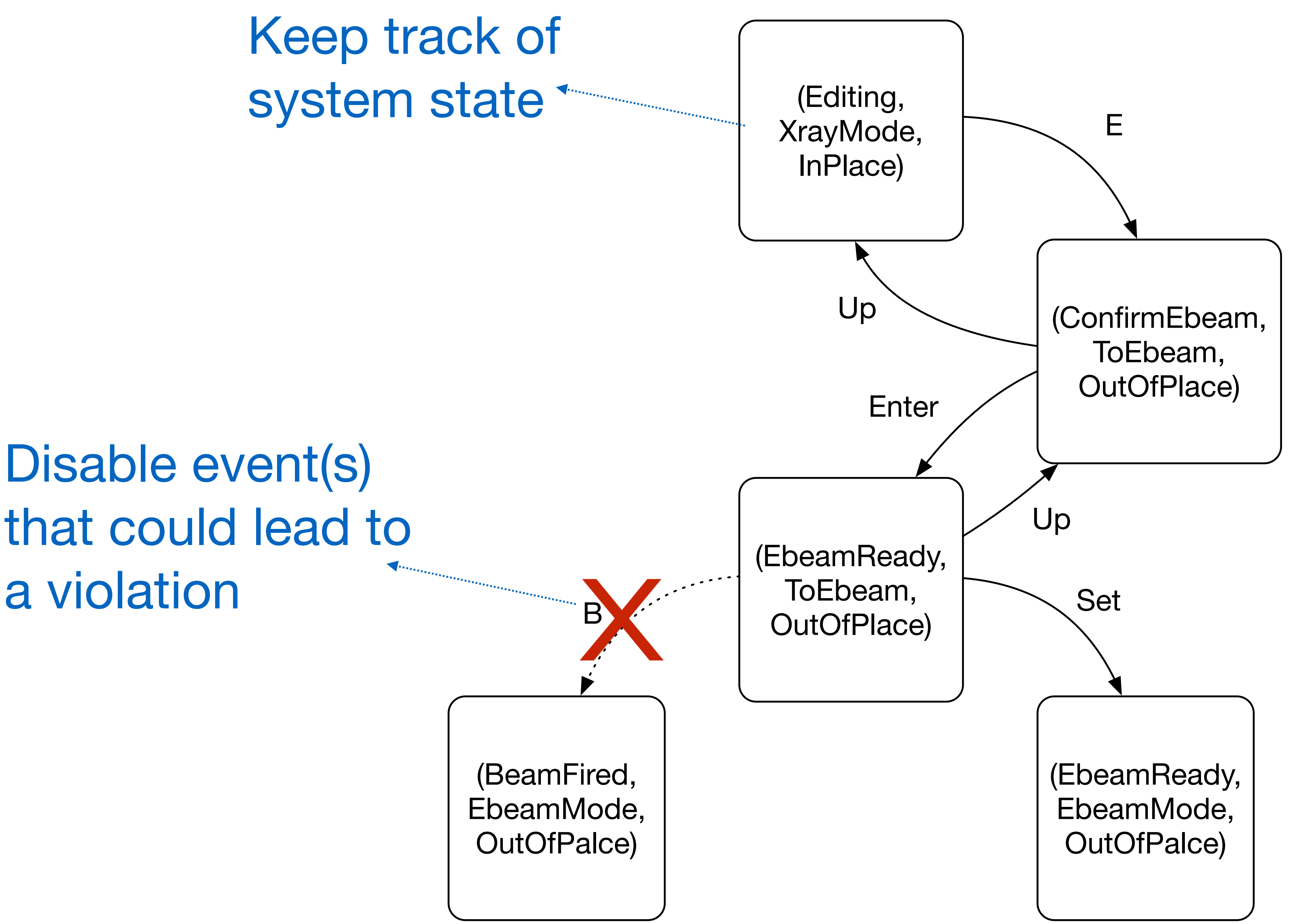


Mode setter (M_B)



Collimator (M_S)

Supervisor Behavior

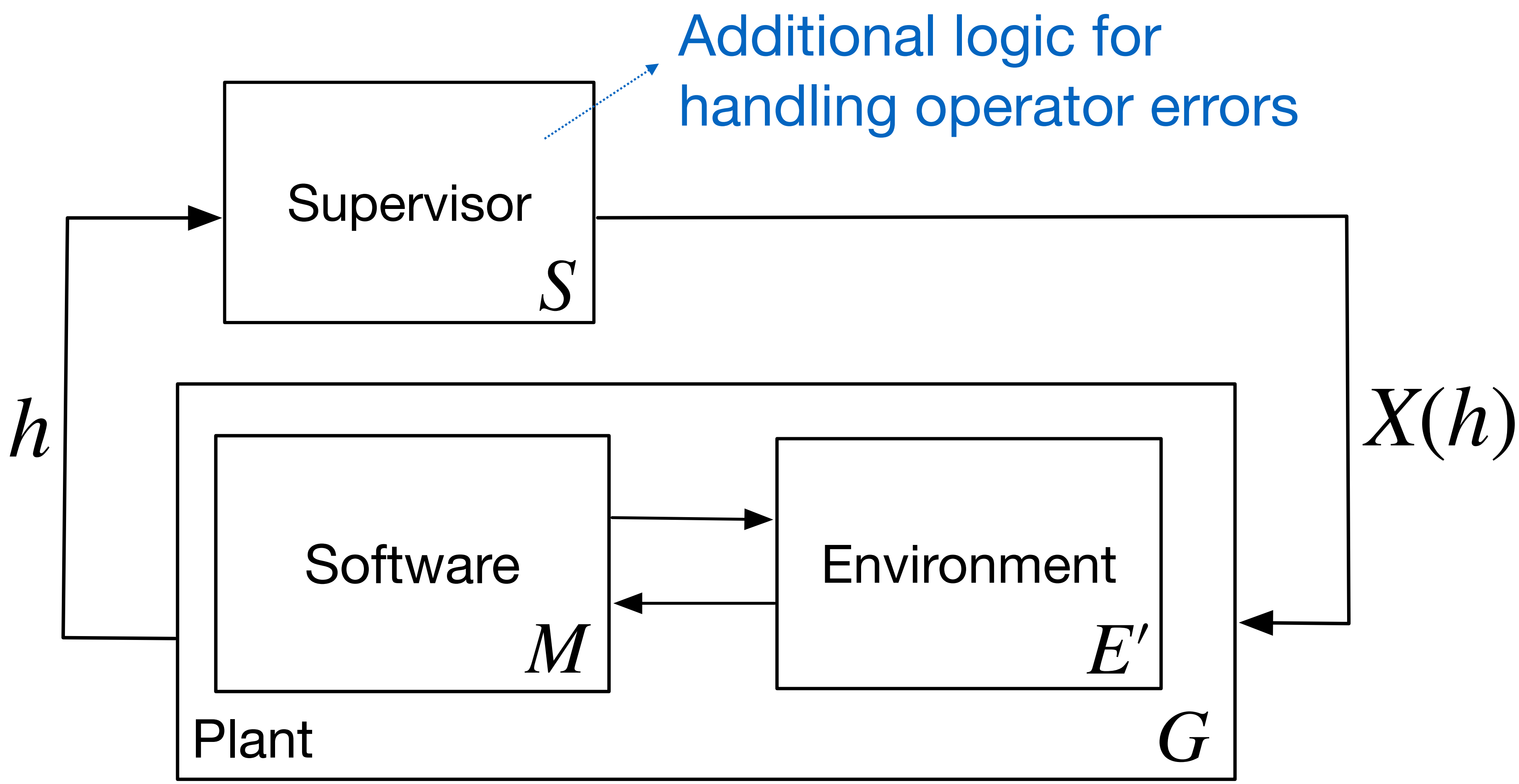


Complications

Not all events are observable/controllable by supervisor

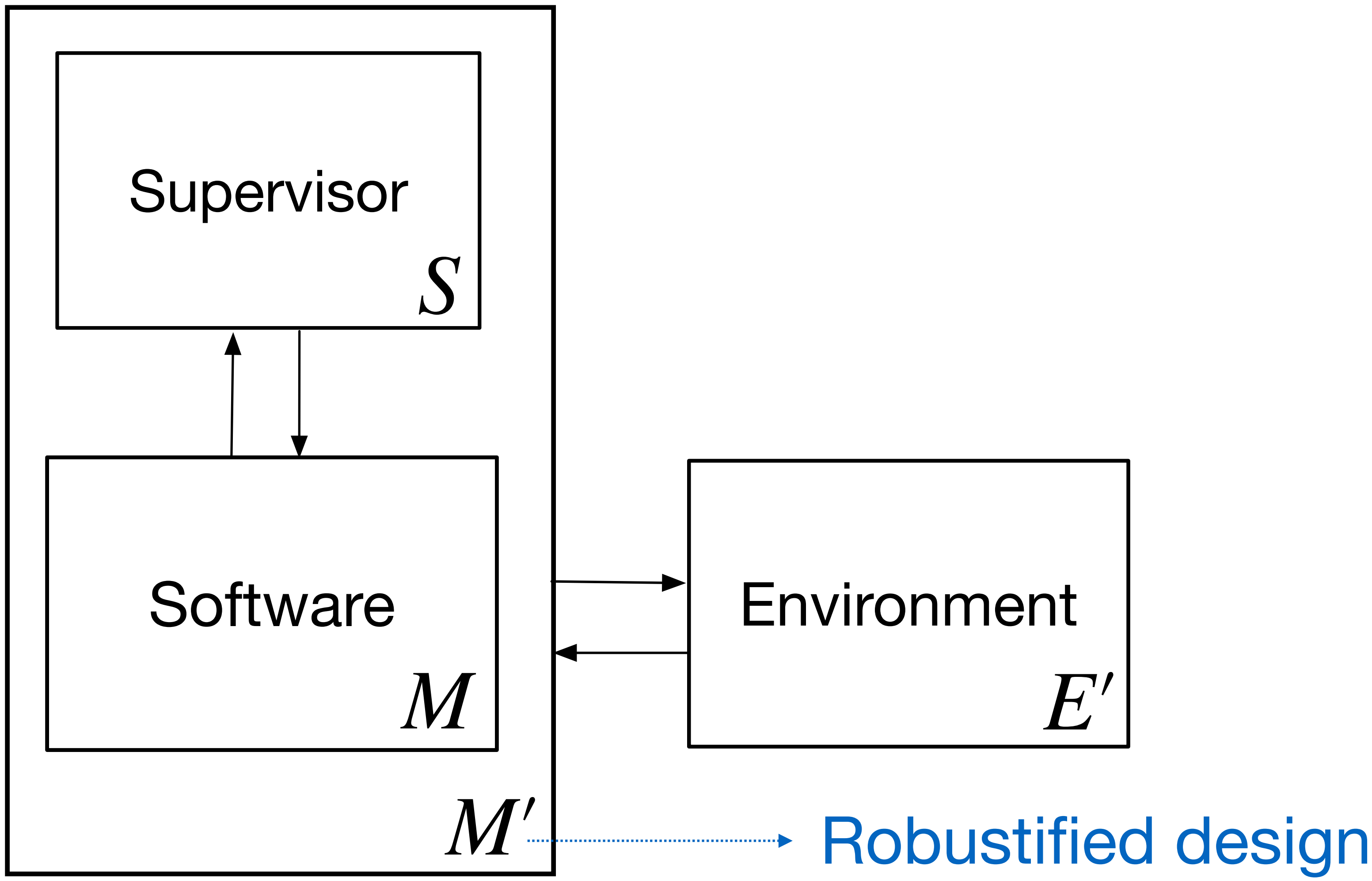
Non-determinism: Multiple possible states to keep track of

Therac-25 Example



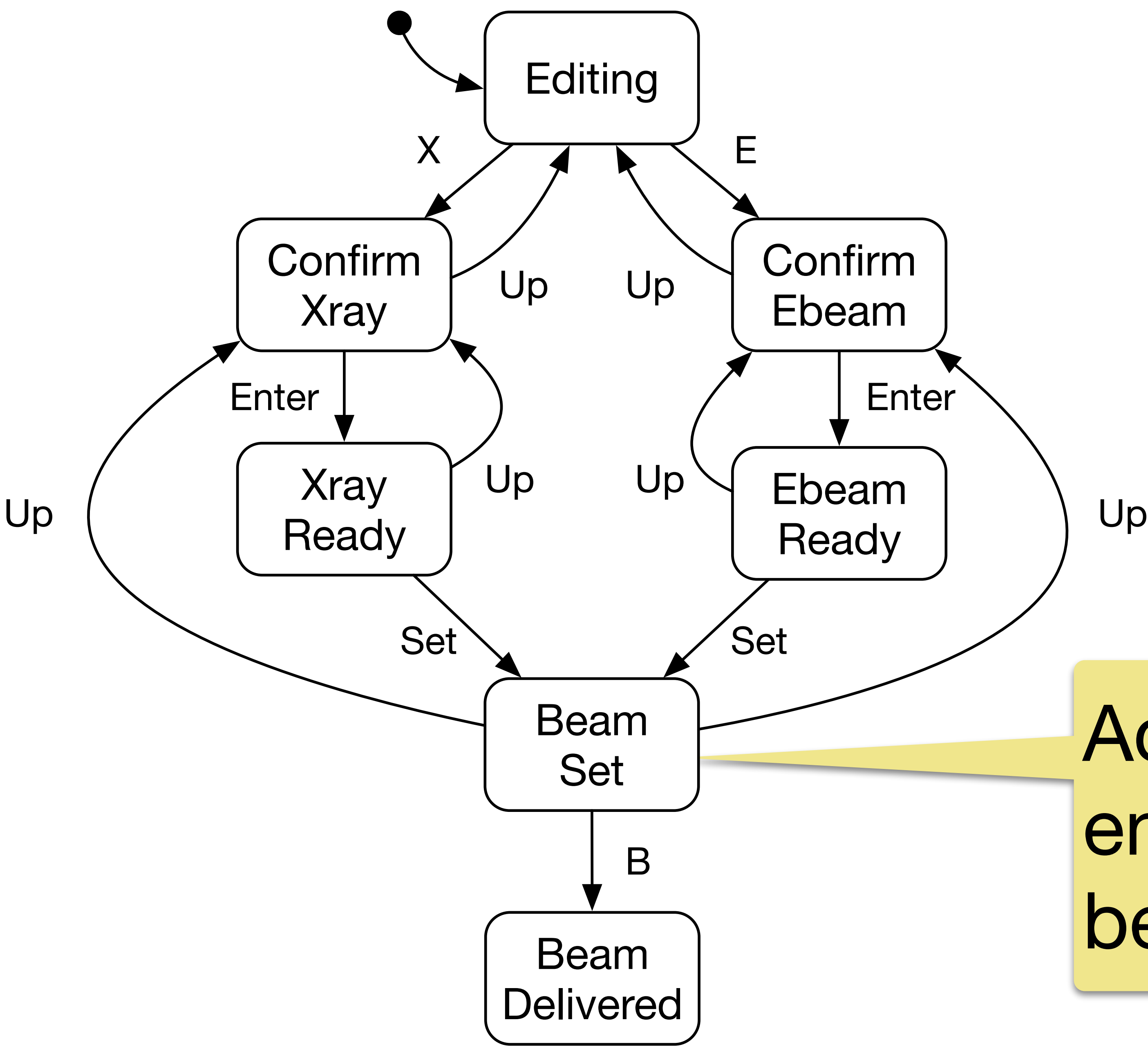
$$S \| G \models P$$
$$\iff S \| (M \| E') \models P$$

Robustification as Supervisory Control



$$S \| (M \| E') \models P$$
$$\iff (S \| M) \| E' \models P$$
$$\iff M' \| E' \models P$$

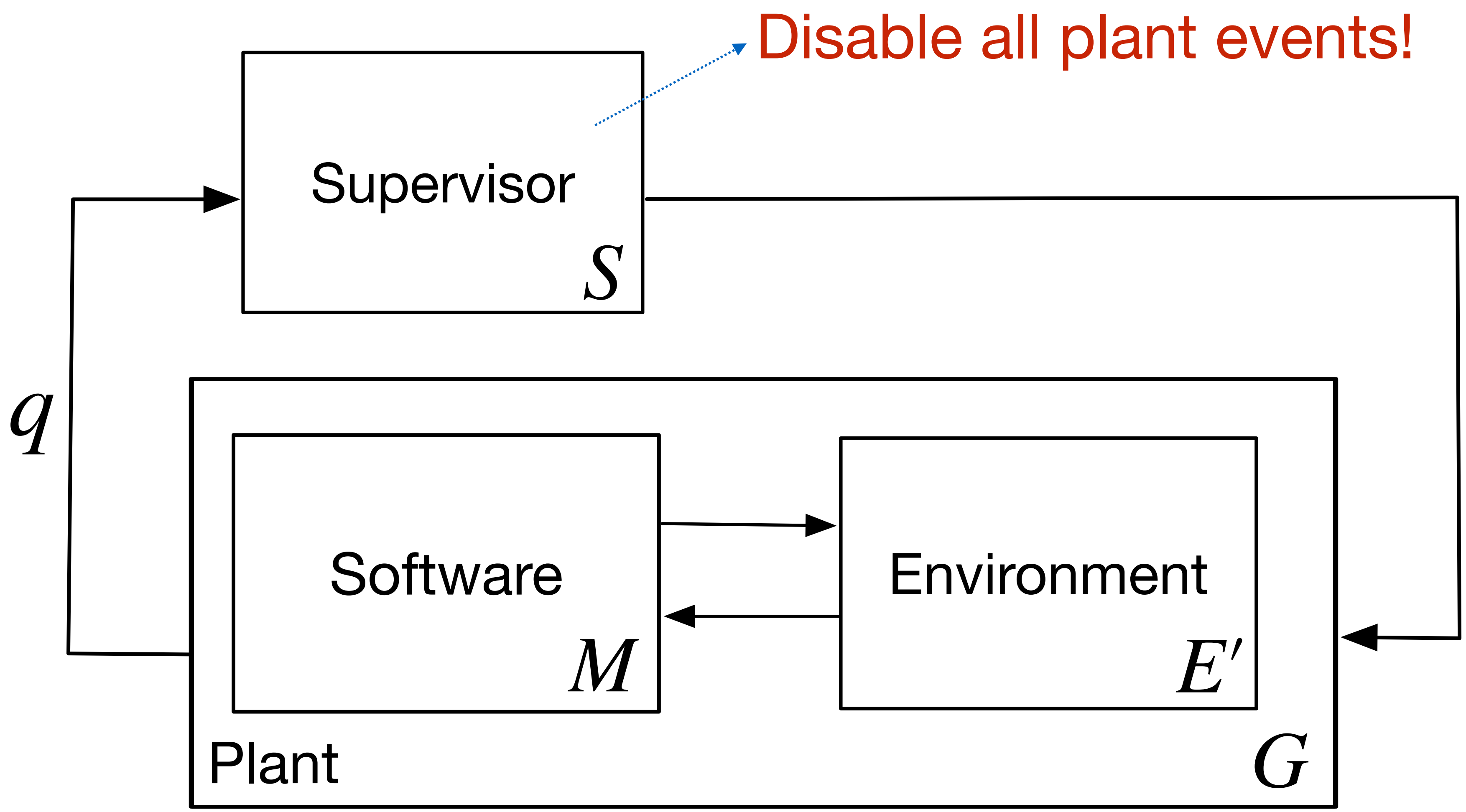
Synthesized Solution for Therac-25



Additional check to ensure mode transition before beam firing

Redesigned Interface (M_I)

Quality of Redesign

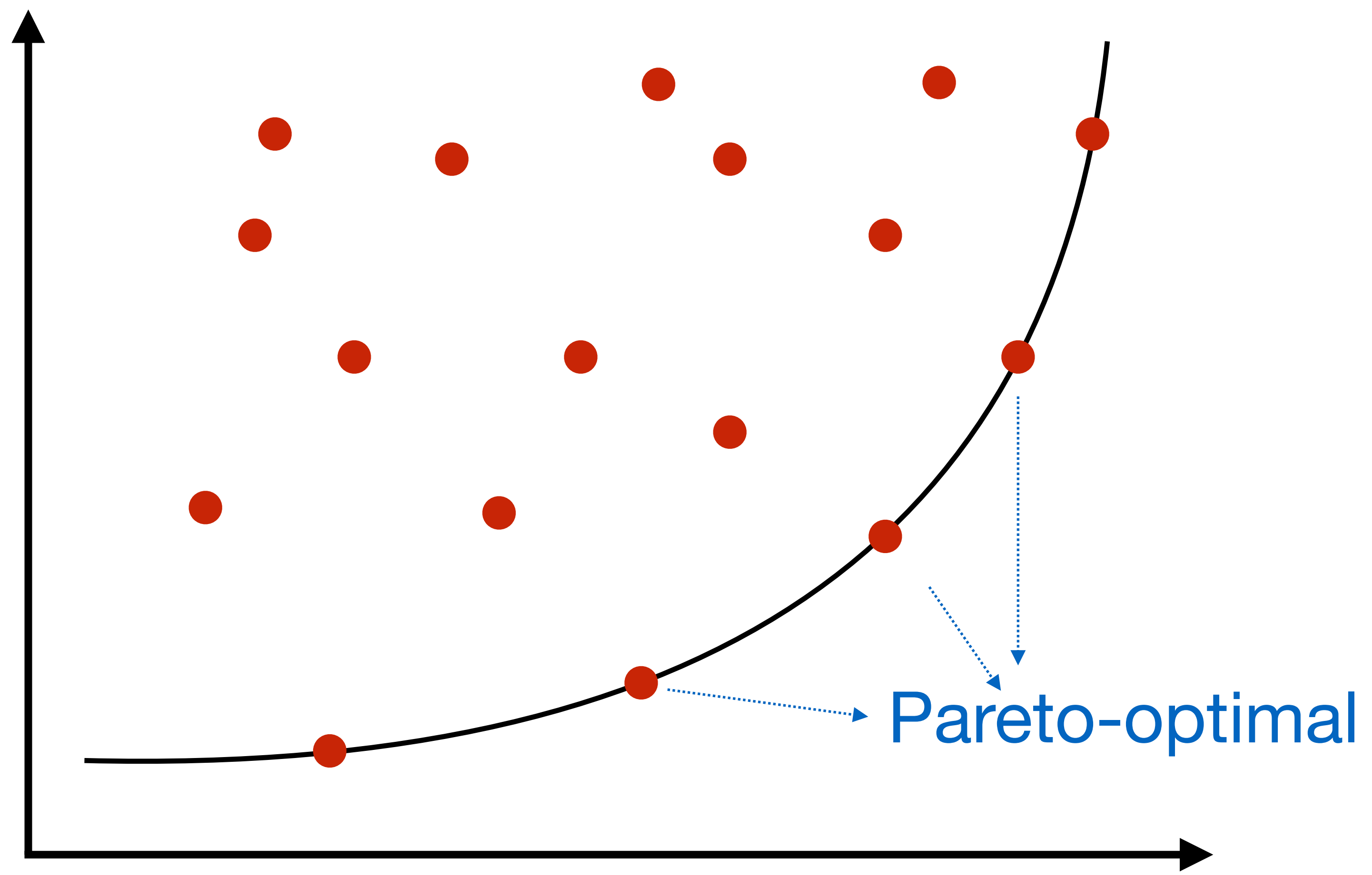


Multiple possible solutions, not all desirable!

Some supervisors may disable more events than needed

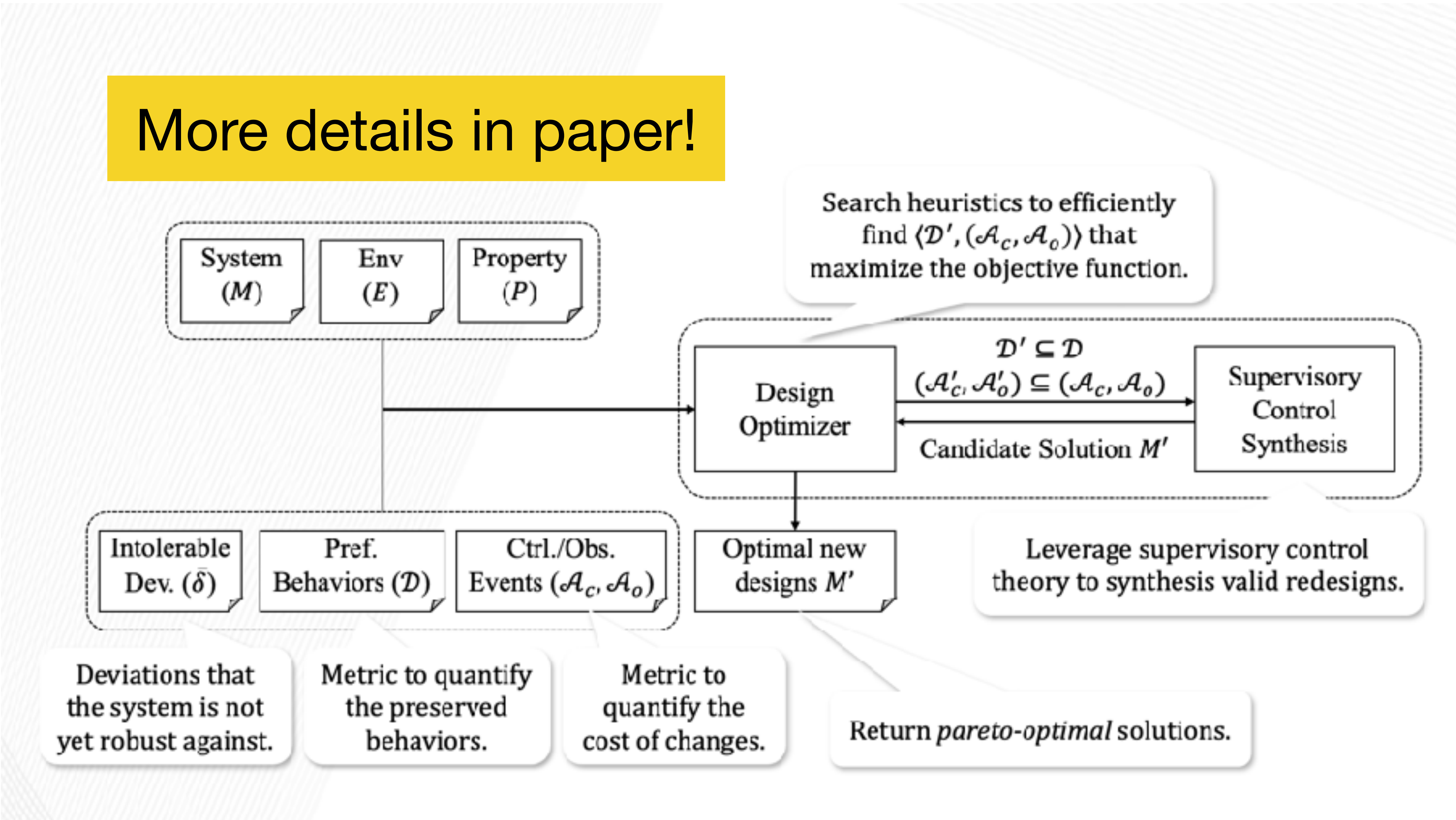
Ideally, find a solution that
(1) Preserves as many existing behaviors in M as possible
(2) Picks the simplest supervisor possible

Optimal Robustification



Trade-offs between these dimensions!
Multiple, possible Pareto-optimal solutions

Robust Design Framework



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

Robustification Case Studies

Electronic voting

Voter errors

Malicious officials

Infusion pump

Therapist errors

Power and alarm failure

Electronic Voting System

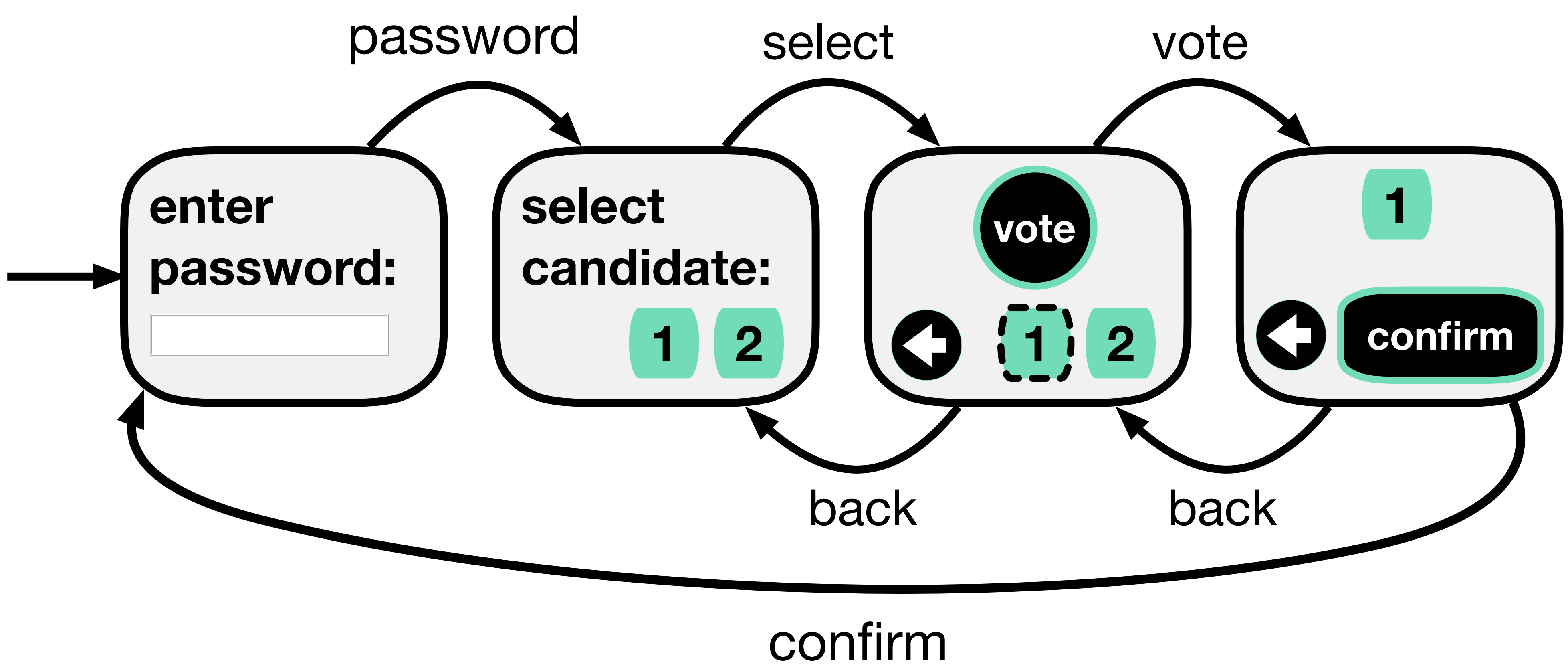
FRANKFORT — A former Clay County precinct worker testified Friday that top election officers in the county taught her how to change people's choices on voting machines to steal votes in the May 2006 primary.

ES&S iVotronic, Kentucky

Voters exits the voting booth before pressing “confirm”

Malicious official enters booth, press “back” & modify the vote

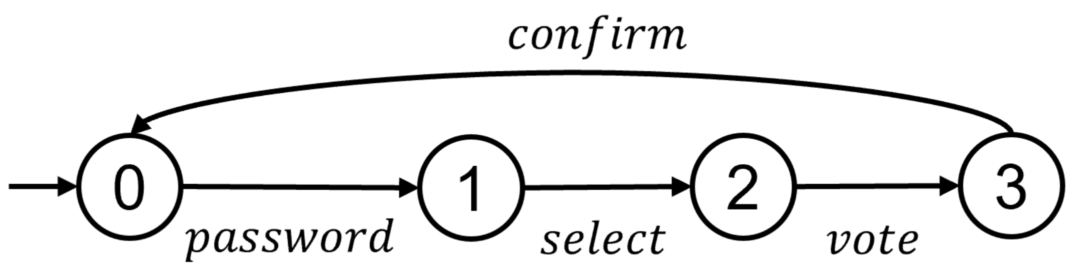
Electronic Voting Interface

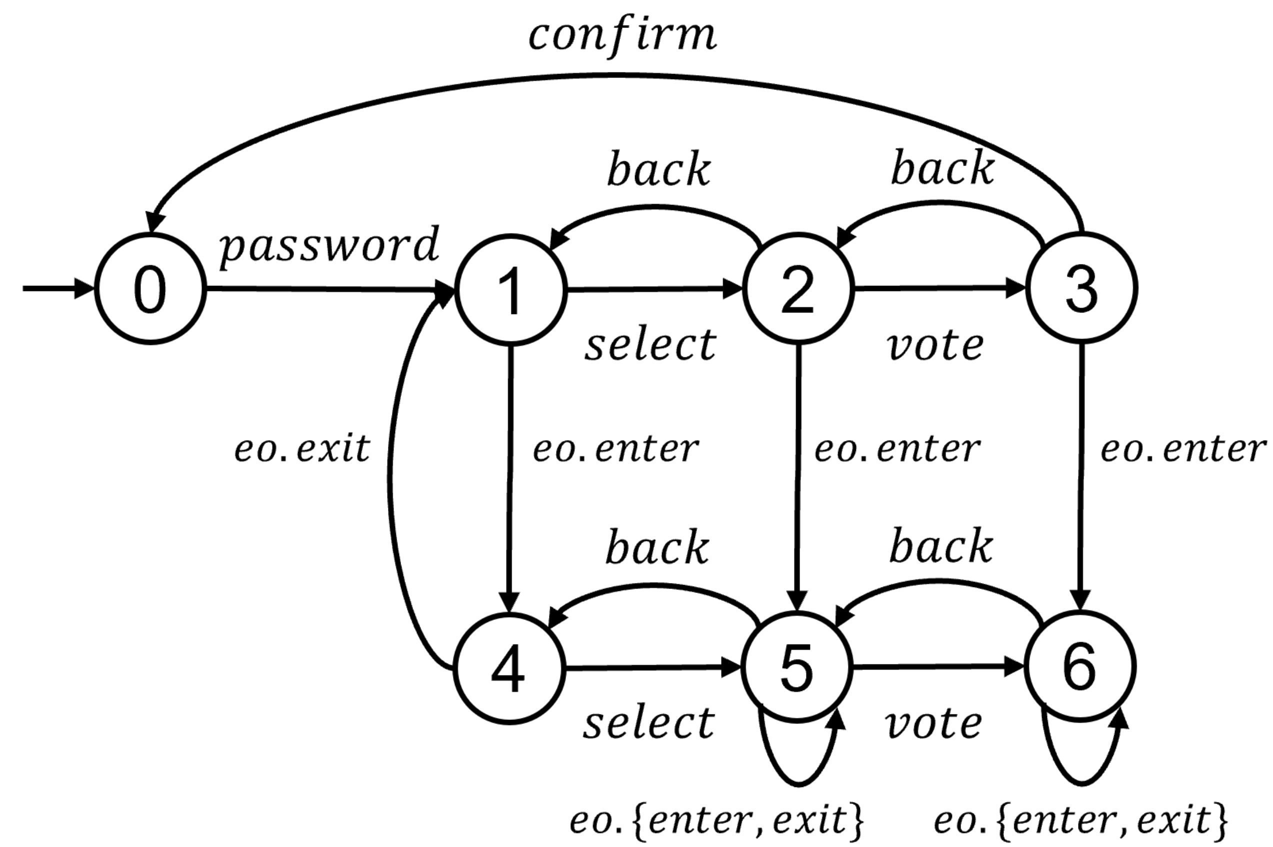


Property: The machine must record the vote as selected by the voter

Introduce a deviation to capture voters omitting “confirm”
Use robustification to generate suggested enhancements

Synthesized Solutions





Redesign #1

Disables “back” action

Simple, but not permissive

Does not allow the voter to modify their selection

Redesign #2

Disables confirm while the official is in the booth

More permissive: Allows vote change

But also more complex: Requires keeping track of booth occupant

Robustification Case Studies

Electronic voting

Infusion pump

Our method can automatically synthesize optimal robustification solutions
For complex models (~760 states), < 20 secs

Other On-going Works

Robustification

$$\begin{tikzcd}
M \arrow[d, bend right=60, "\text{strengthen}"'] & \parallel & E \arrow[d, dashed, "\delta"] & \models & P \\
M' & \parallel & E' & \models & P
\end{tikzcd}$$

Can we enhance the original design to tolerate additional deviations in the environment?

Robustness through Req. Weakening

$$M \parallel E \models P \xrightarrow{\text{weaken}} M \parallel E' \models P'$$

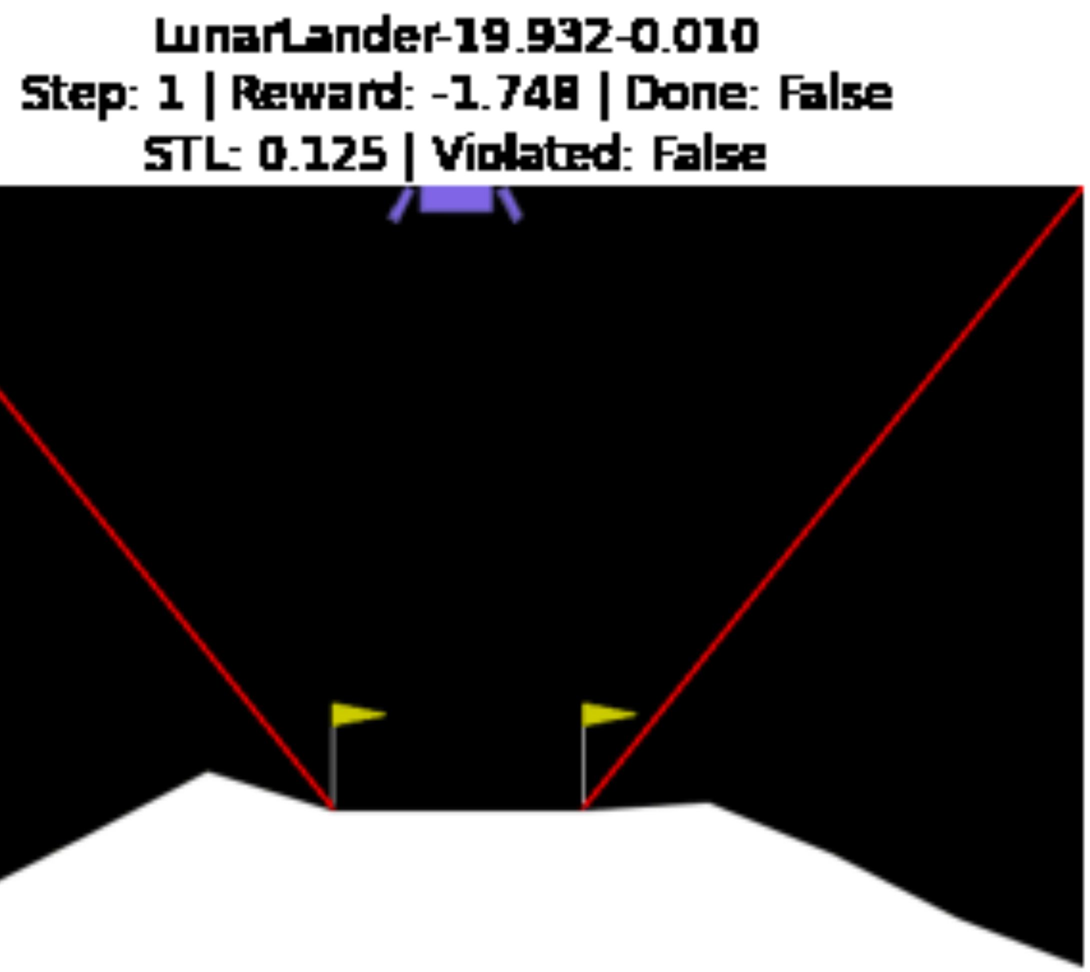
$$E \overset{\delta}{\dashrightarrow} E'$$

Self-adaptive framework

Temporarily weaken P to a weaker variant (P') that is (1) acceptable to the user & (2) satisfiable in the deviated environment

SEAMS 2023

Robustness of AI-based Systems



System as a composition of AI & “traditional” SW

How robust is the overall system against:
(1) Deviations in the environment?
(2) Mistakes in the learning-based components?

How do we validate & measure robustness in such systems?

Takeaway

Software-intensive systems depend on various assumptions about the environment

The environment may deviate from its expectations due to misbehavior or changes

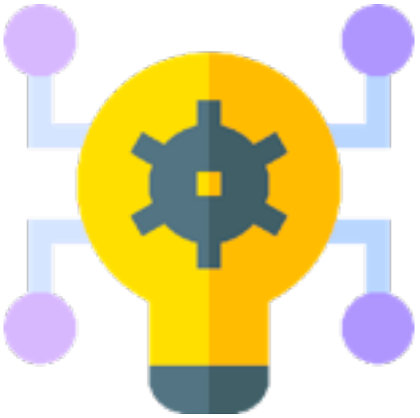
To ensure critical properties, systems should be designed to be robust against possible deviations

Robust Software Design: Roadmap

Specification
What does it mean for our system to be robust?

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

Analysis
How robust is our system?



Robustification
How do we improve its robustness?