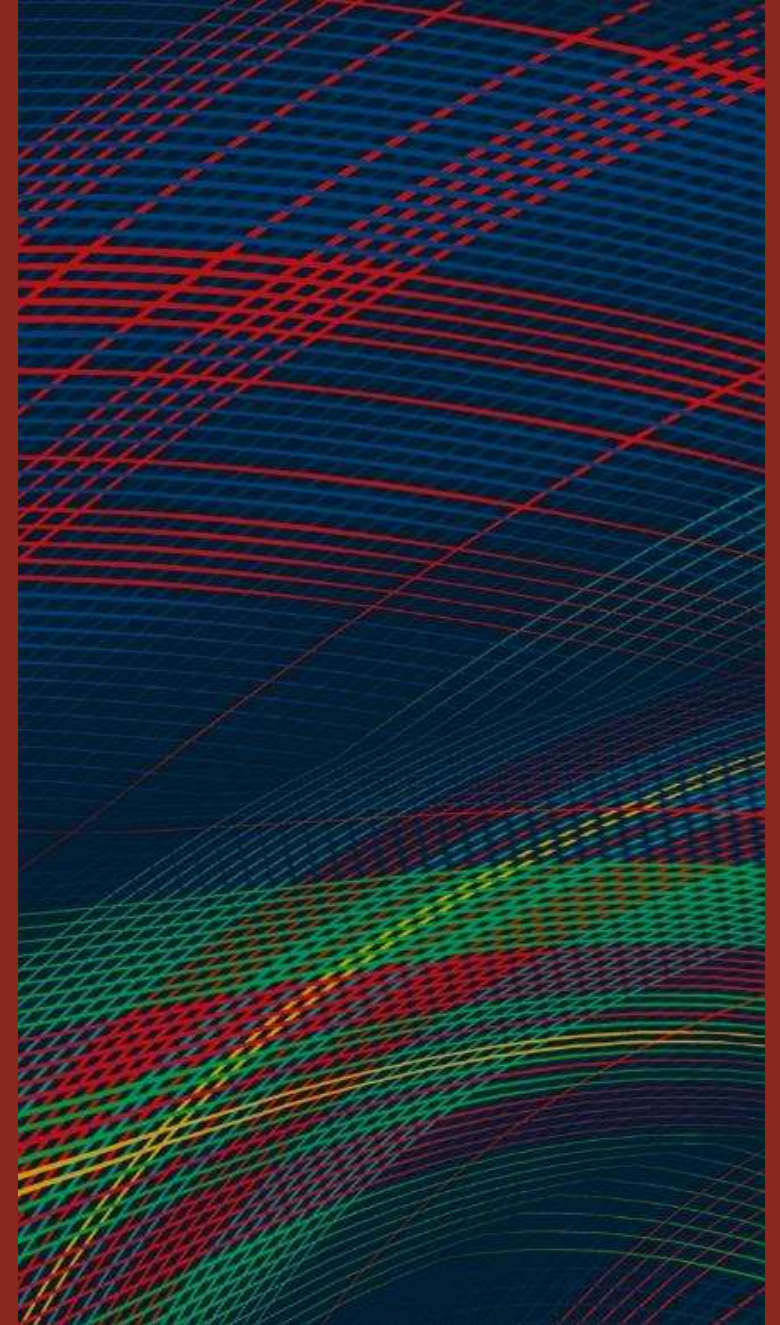


Delta Airlines

Verifying Drones with Enforcers

PI: Dr. Dionisio de Niz



Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM19-0579

Enforcement-Based Verification of Cyber-Physical Systems

Certification of Cyber-Physical Systems

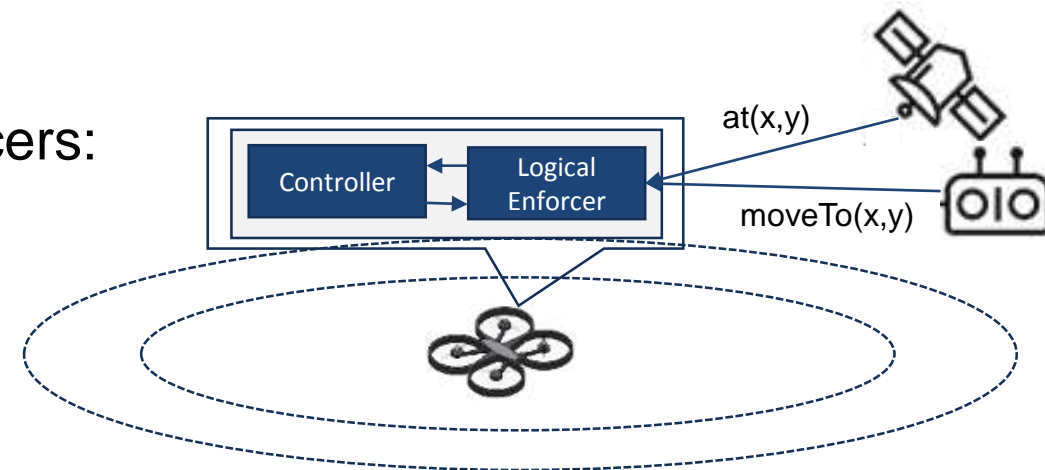
- Evidence of Safe Behavior
 - Logic: Correct actions (e.g., stop)
 - Timing: At the right time (e.g., before crash)
 - Control: according to physics (e.g., aerodynamics, wind, etc.)

Mathematical Verification to Provide Evidence for Certification:

- BUT: techniques do not scale to size of full systems

Our Solution:

- Add **simpler verified** runtime enforcers to make prevent unsafe actions
- Formally: specify, verify, and compose multiple enforcers:
 - Logic: Enforcer **intercepts/replaces** unsafe action
 - Timing: at **right time**
- **Protect enforcers** against failures/ cyber-attacks



Mathematical Logical Model

Statespace

- $S = \{s\}$
- $\phi \subseteq S$

Periodic actions

- Transition: $R_P(\alpha) \subseteq S \times S$
- Destination state: $R_P(\alpha, s) = \{s' | (s, s') \in R(\alpha)\}$

Identify states too close to safety border

- Inertia lead to unsafe state even if enforced
- Enforceable states:

$$C_\phi = \{s | \exists \alpha: R_P(\alpha, s) \in C_\phi\}$$

Safe actions:

- $SafeAct(s) = \{\alpha | R_P(\alpha, s) \in C_\phi\}$

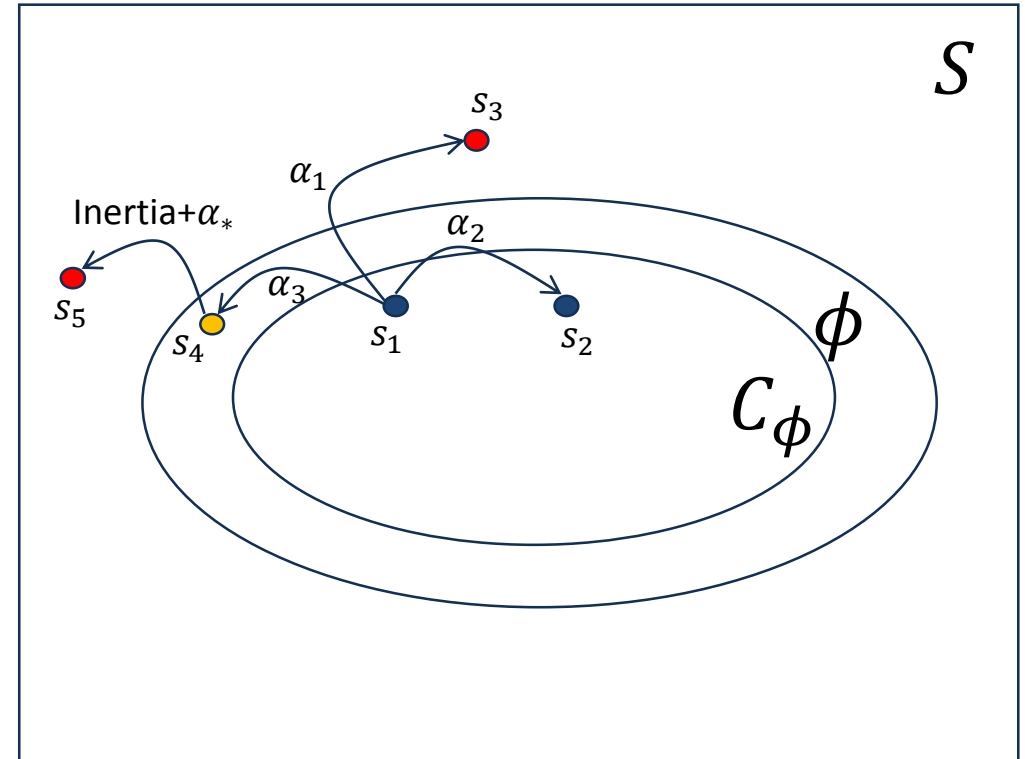
Logical Enforcer: $E = (P, C_\phi, \mu)$

- Set of safe actions:

$$\mu(s) \subseteq SafeAct(s)$$

- Monitor and enforce safe action:

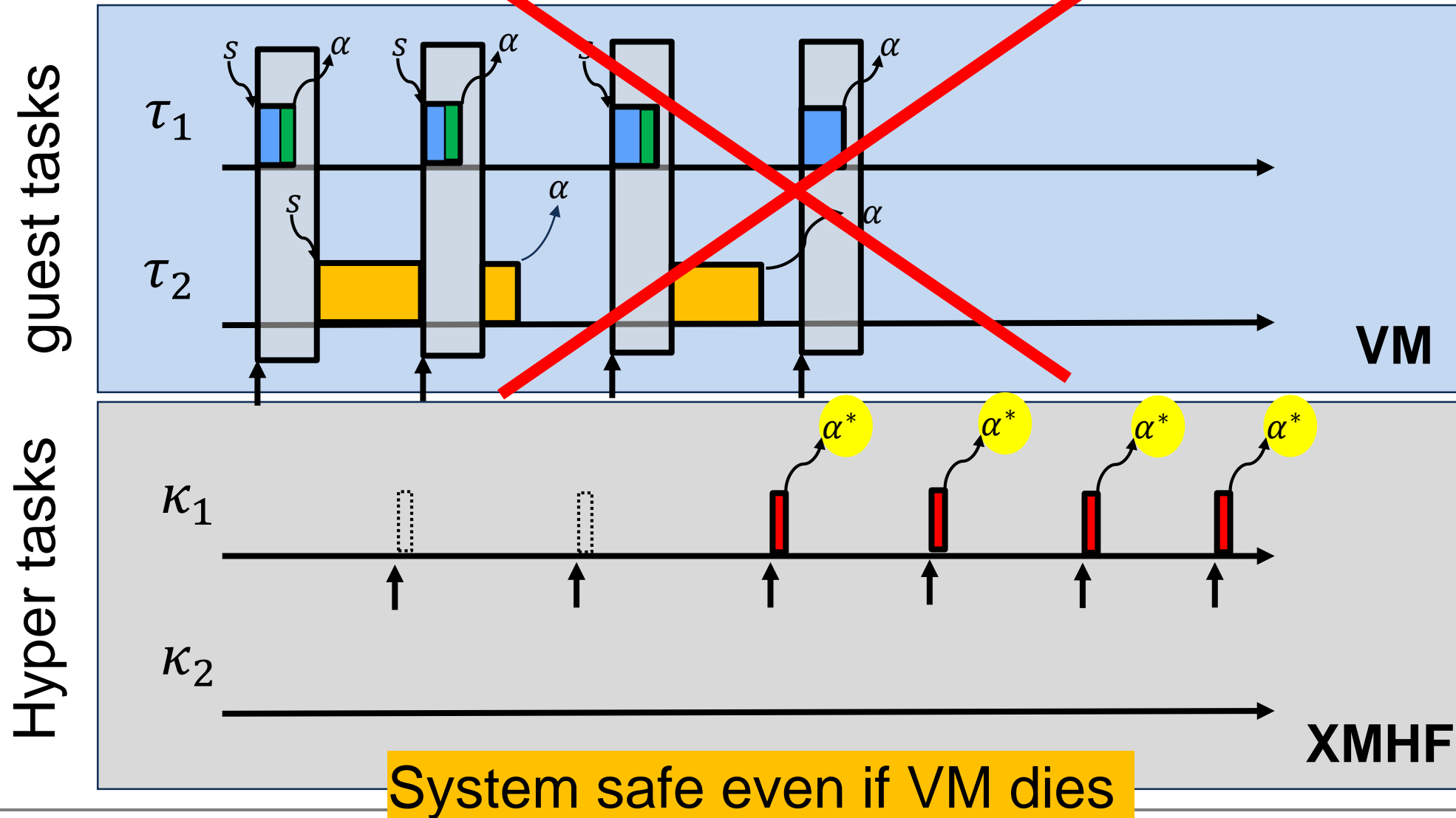
$$\tilde{\alpha} = \begin{cases} \alpha, & \alpha \in \mu(s) \\ pick(\mu(s)), & otherwise \end{cases}$$



We use tools that directly verify C source code

Certification evidence

We protect the enforcers to prevent virus from modifying them
(with **verified** hypervisor **evidence for certification**)



Verifying Interaction with Environment (control)

- Math prove that enforcer we can always recover safety

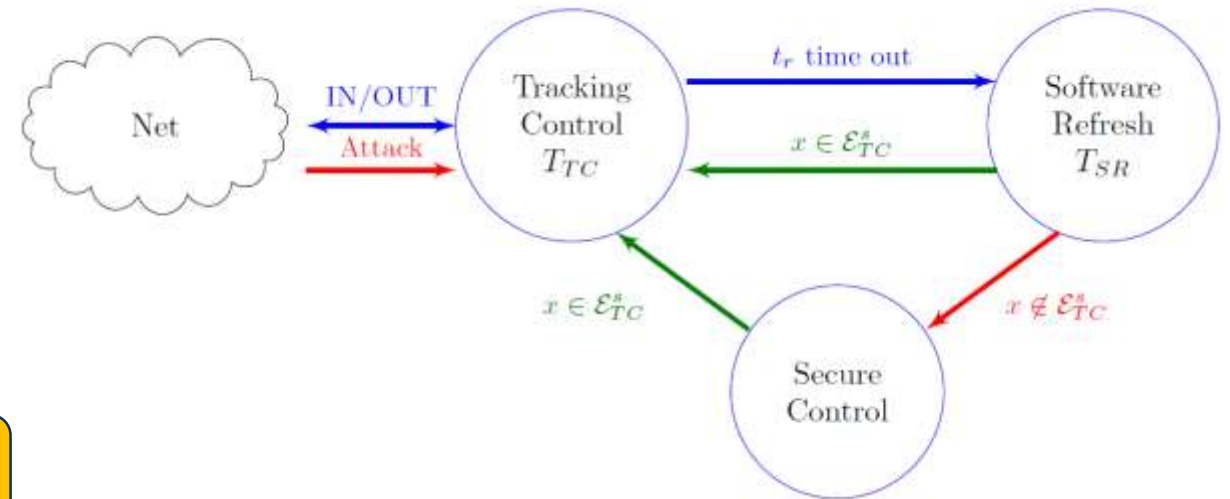
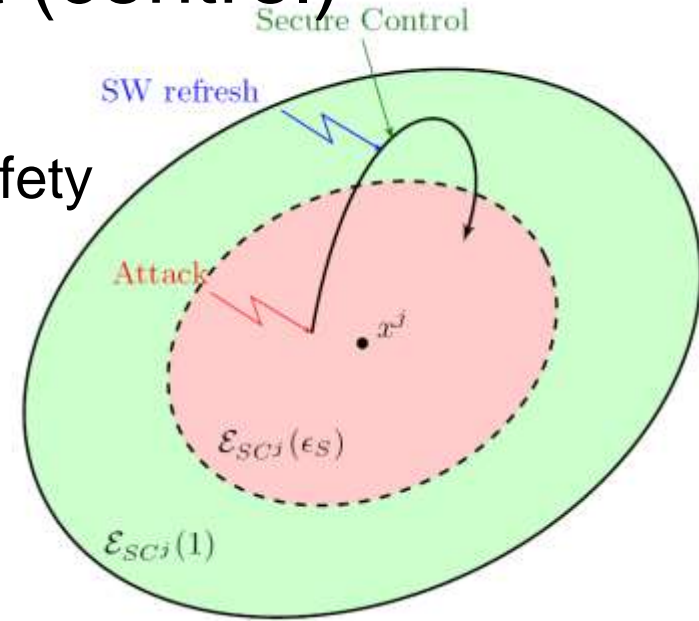
$\mathcal{E}_{SCj}(1)$ Lyapunov Theory and Positively Invariant Sets

- Safety region $\mathcal{E}_{SCj}(\epsilon_s) \triangleq \epsilon_s \mathcal{E}_{SCj}(1)$

$\epsilon_s \quad T_{UC}$

$$\mathcal{R}(T_{UC}; \mathcal{E}_{SCj}(\epsilon_s), U) \subseteq \mathcal{E}_{SCj}(1)$$

Evidence for Certification



Enforcers detect and correct unsafe behavior

With mathematical evidence

