Delta Airlines

Verifying Drones with Enforcers

PI: Dr. Dionisio de Niz

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.

DM19-0579

Enforcement-Based Verification of Cyber-Physical Systems

Certification of Cyber-Physical Systems

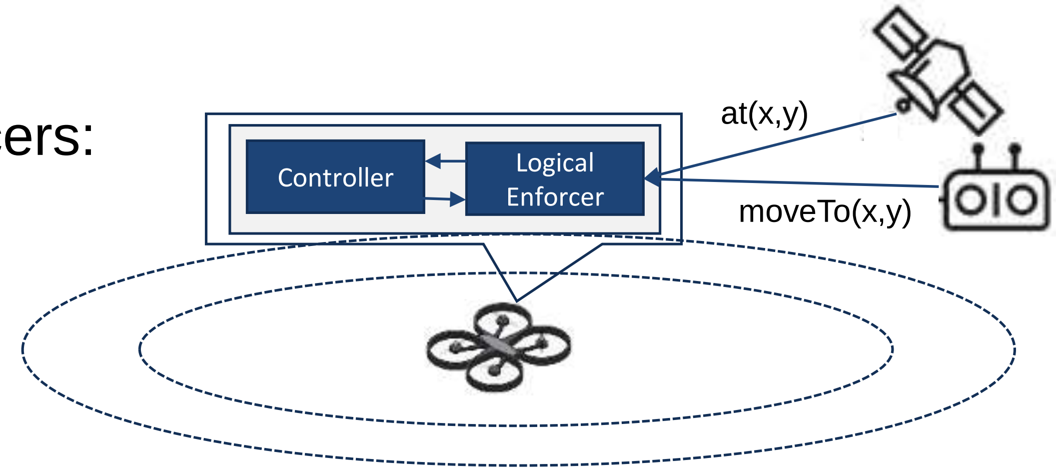
- Evidence of Safe Behavior
  - Logic: Correct actions (e.g., stop)
  - Timing: At the right time (e.g., before crash)
  - Control: according to physics (e.g., aerodynamics, wind, etc.)

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:

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

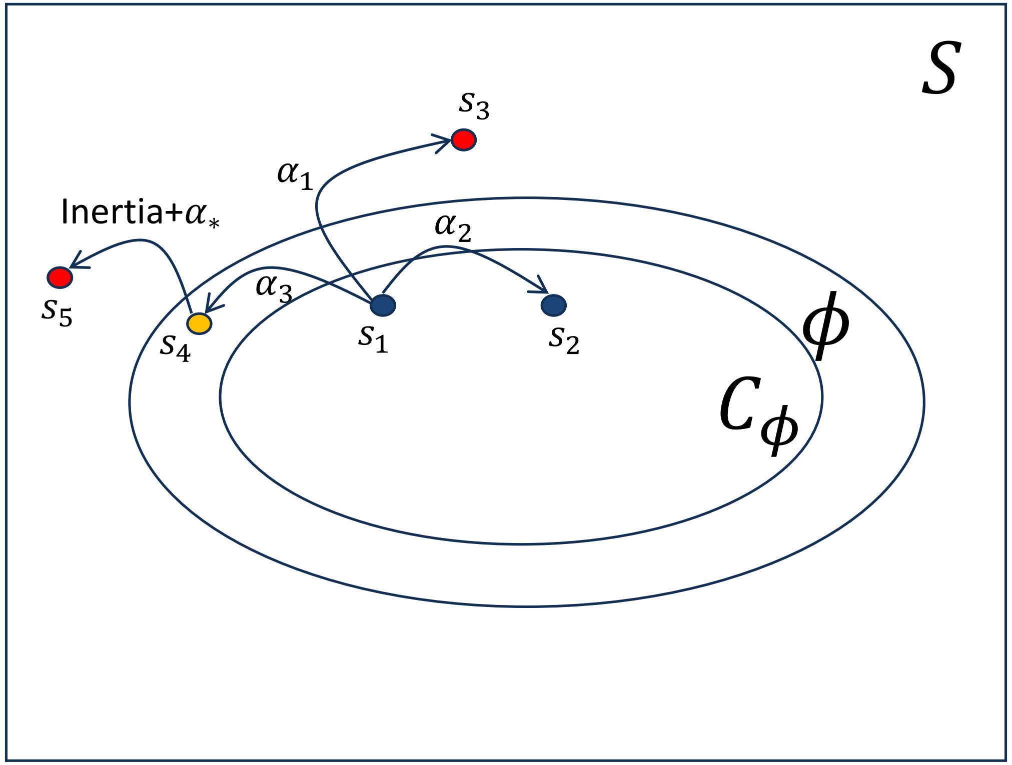
Logical Enforcer: $E = (P, C_\phi, \mu)$

- Set of safe actions:

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

- Monitor and enforce safe action:

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



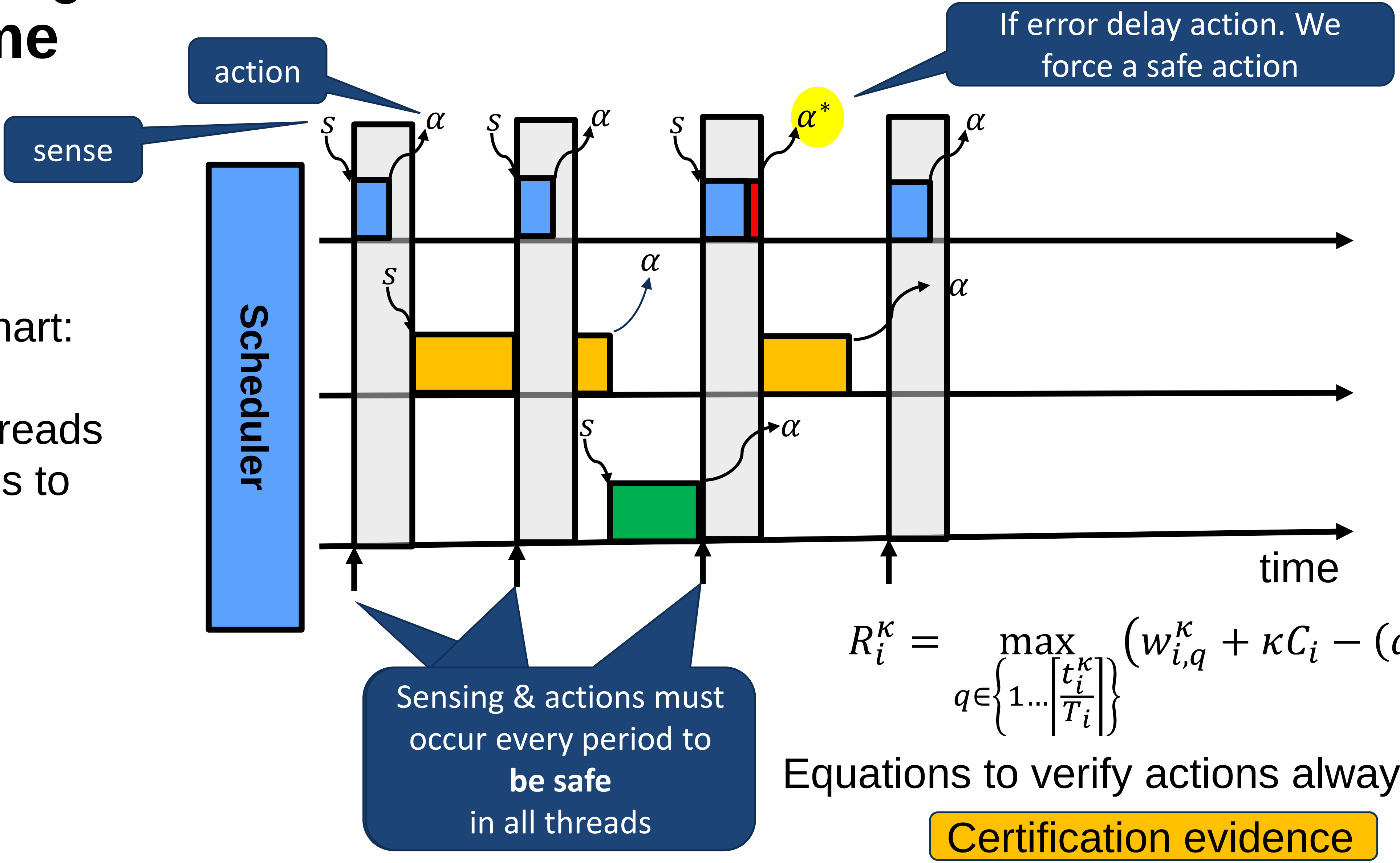
We use tools that directly verify C source code

Certification evidence

Ensuring drone senses environment and corrects actions **on time**

GANTT Chart:

Multiple threads taking turns to execute

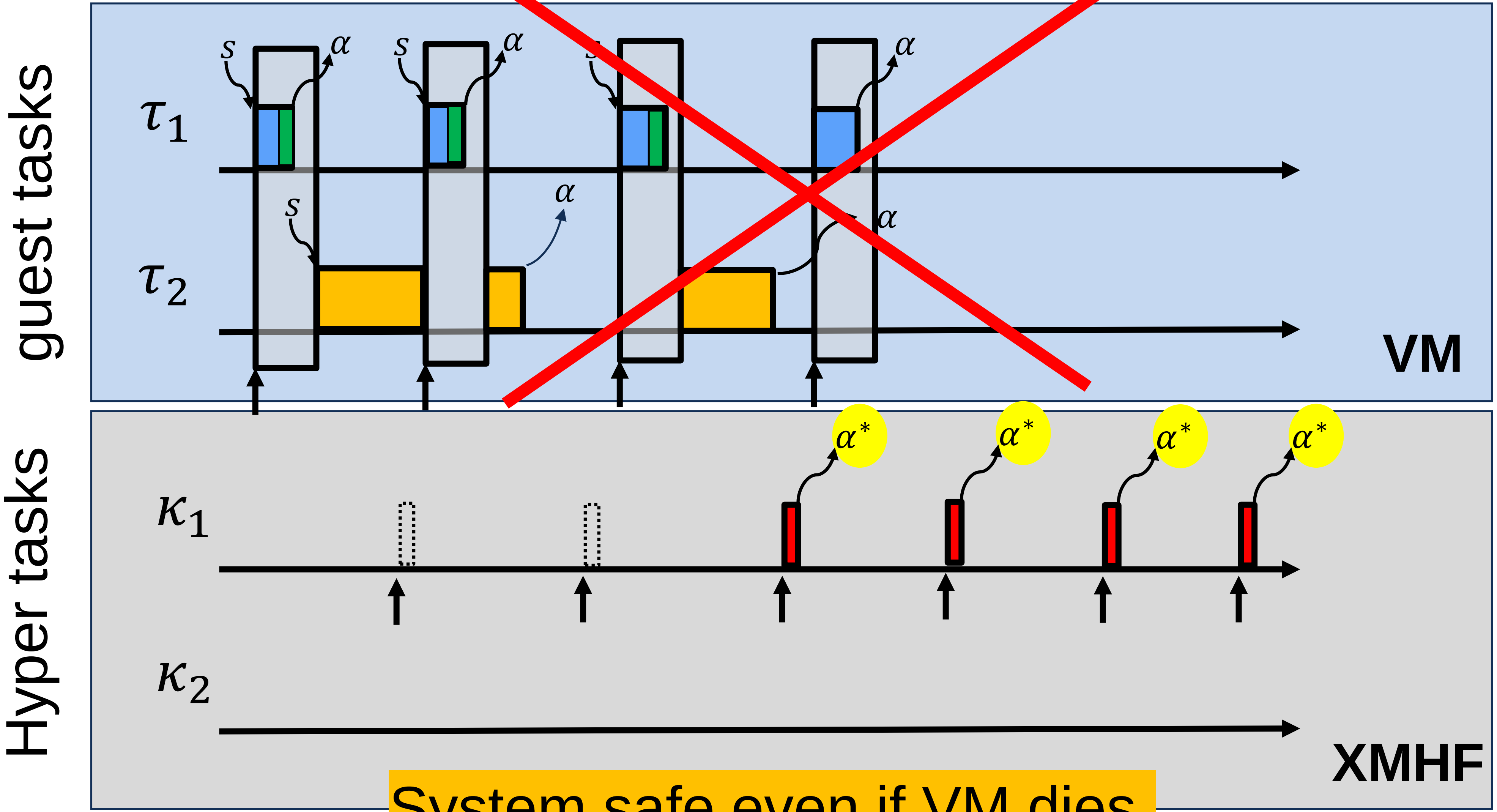


$$R_i^{\kappa} = \max_{q \in \left\{1 \ldots \left\lceil \frac{t_i^{\kappa}}{T_i} \right\rceil \right\}} \left(w_{i,q}^{\kappa} + \kappa C_i - (q-1) T_i \right)$$

Equations to verify actions always on time

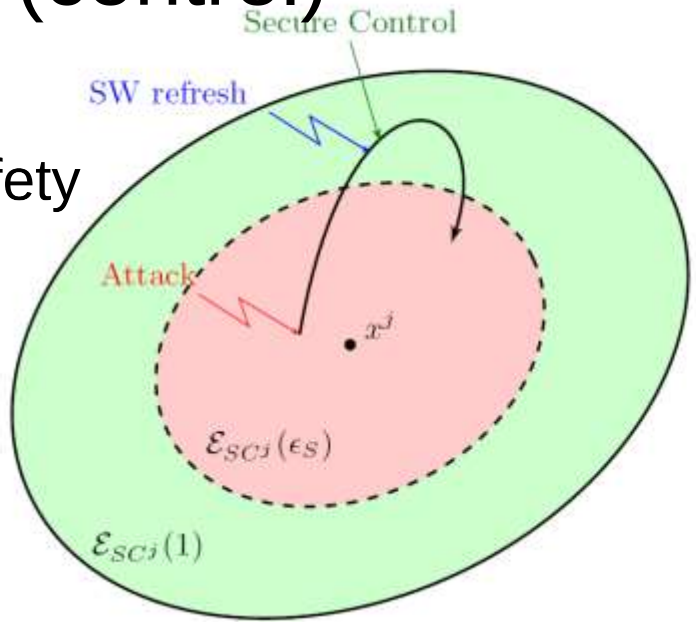
Certification evidence

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



System safe even if VM dies

Verifying Interaction with Environment (control)



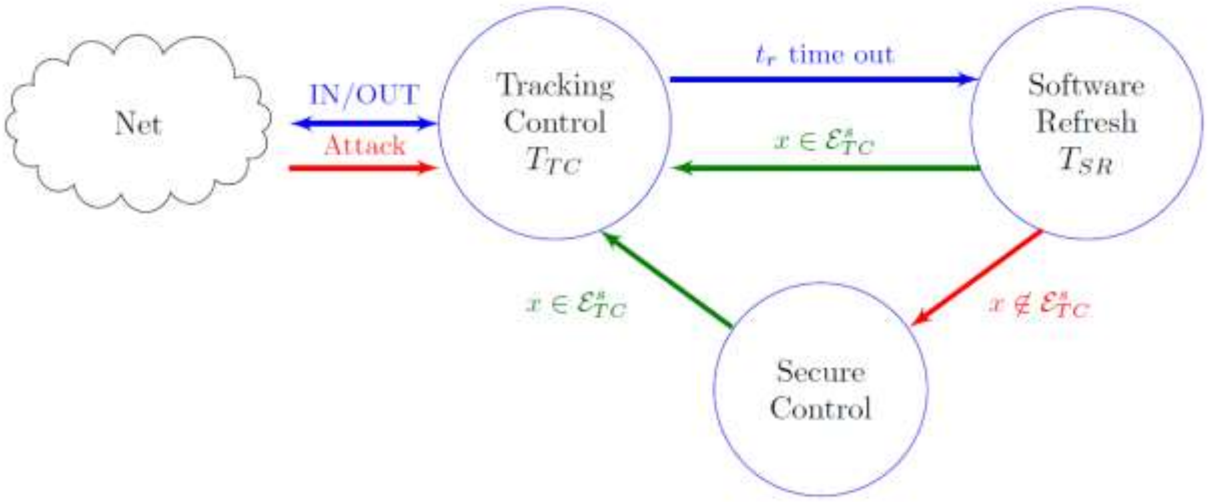
- Math prove that enforcer we can always recover safety

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

- Safety region $\mathcal{E}_{SC^j}(\epsilon_s) \triangleq \epsilon_s \mathcal{E}_{SC^j}(1)$

ϵ_S T_{UC}

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



Evidence for Certification

Enforcers detect and correct unsafe behavior

With mathematical evidence