

## Challenge

Assure Safety of Distributed Cyber-Physical Systems

- Unpredictable Algorithms (Machine Learning)
- Coordinating multiple vehicles (distributed) to achieve mission

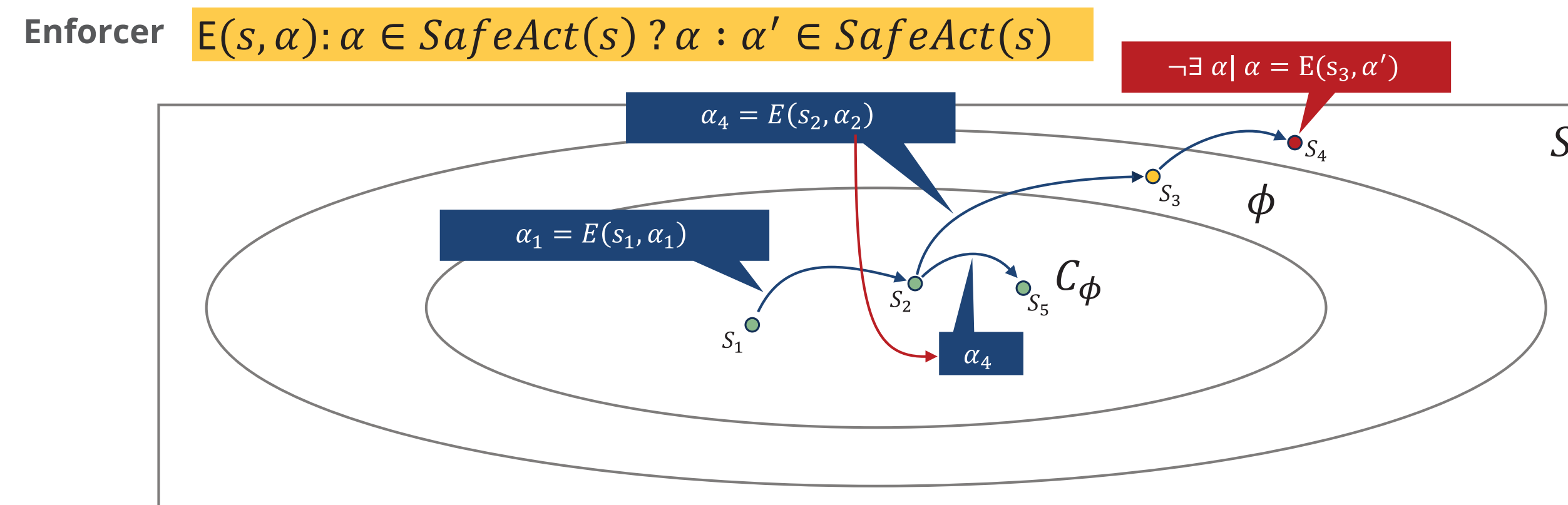
## Solution

- Add simpler (verifiable) runtime enforcer to make algorithms predictable
- Formally specify, verify, and compose multiple enforcers
- Enforcer intercepts/replaces unsafe action at right time

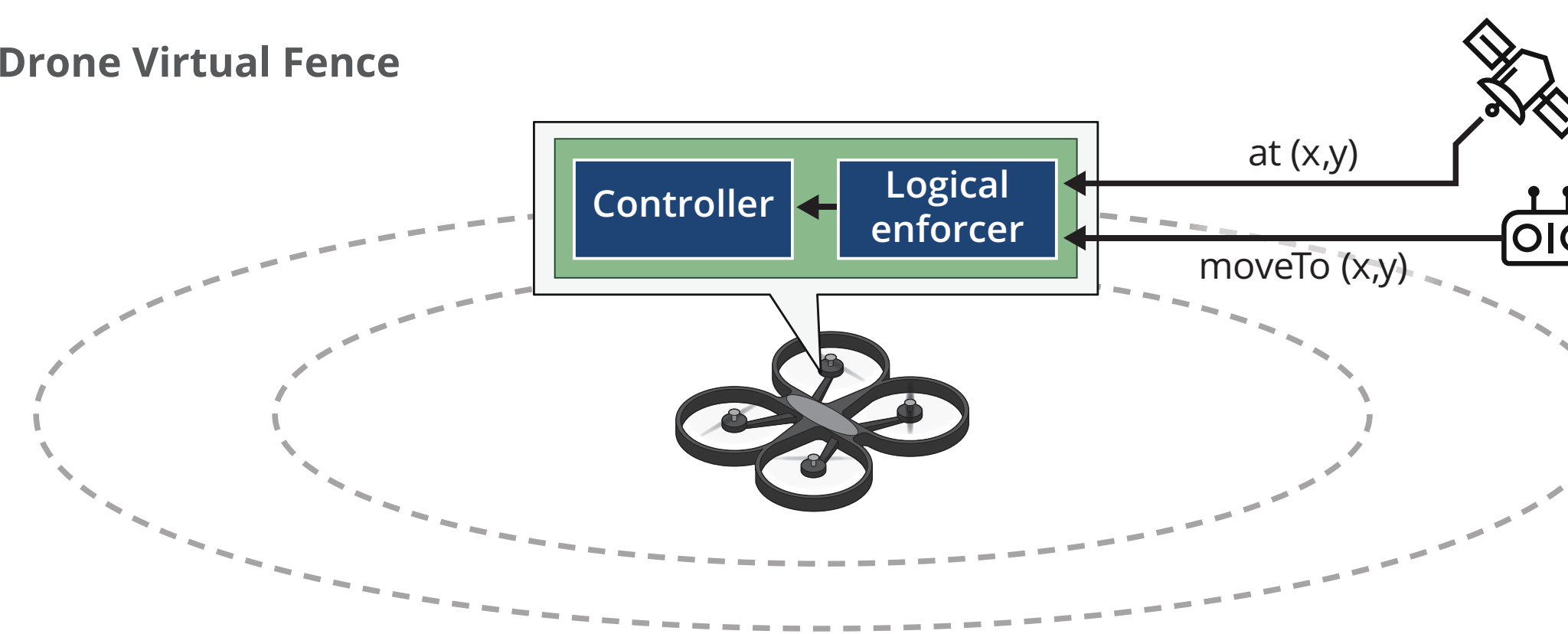
## Formalization (Time-Aware Logic)

State of system: variable values

- State variables:  $V_S$  e.g.,  $(x,y)$  position
- Action variables:  $V_\Sigma$  e.g.,  $\text{move-to}(x,y)$  action
- System state:  $s:V_S \mapsto D \in S$
- Actions:  $\alpha:V_\Sigma \mapsto D$
- Behavior: periodic state transition  $R_p(\alpha) \subseteq S \times S$   
 $R_p(\alpha, s) = \{s' \mid (s, s') \in R_p(\alpha)\}$
- Safe state:  $\phi \subseteq S$
- Enforceable state:  $\phi \supseteq C_\phi = \{s \mid \exists \alpha \in \Sigma: R_p(\alpha, s) \in C_\phi\}$
- Safe Actuation  $\text{SafeAct}(s) = \{\alpha \mid R_p(\alpha, s) \in C_\phi\}$



## Enforcing Drone Virtual Fence



## Timing enforcement

- Unverified software may never finish!
- => No action produced to be enforced!

## Temporal enforcer

- Protect other tasks from bogus never-ending (or large) executions
- Produce default safe actuation if task takes too long

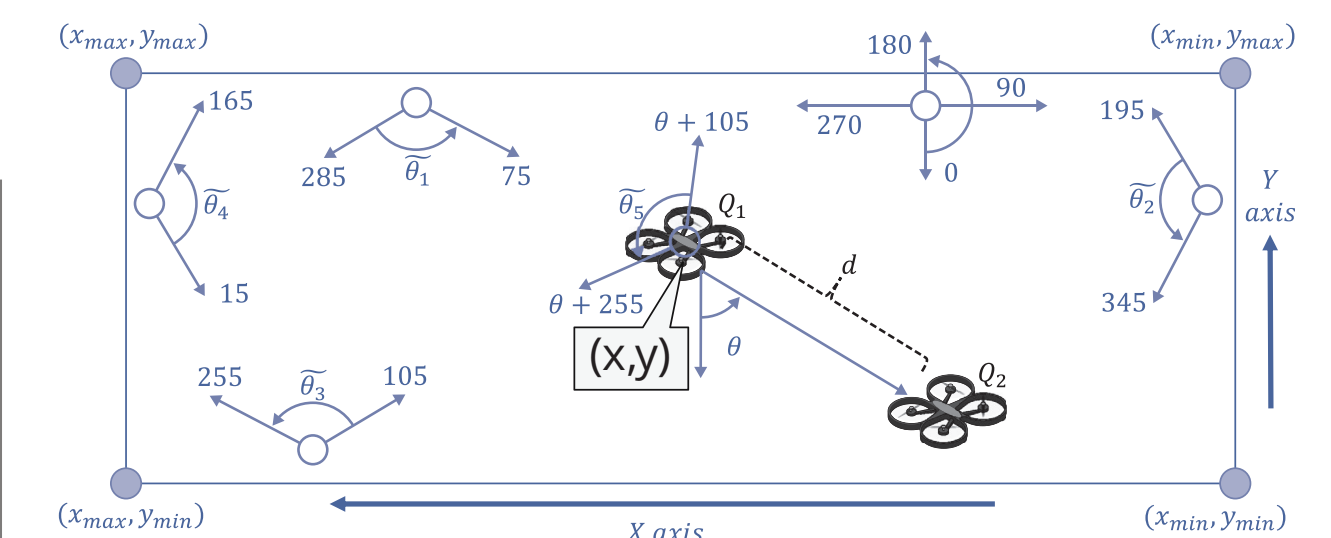
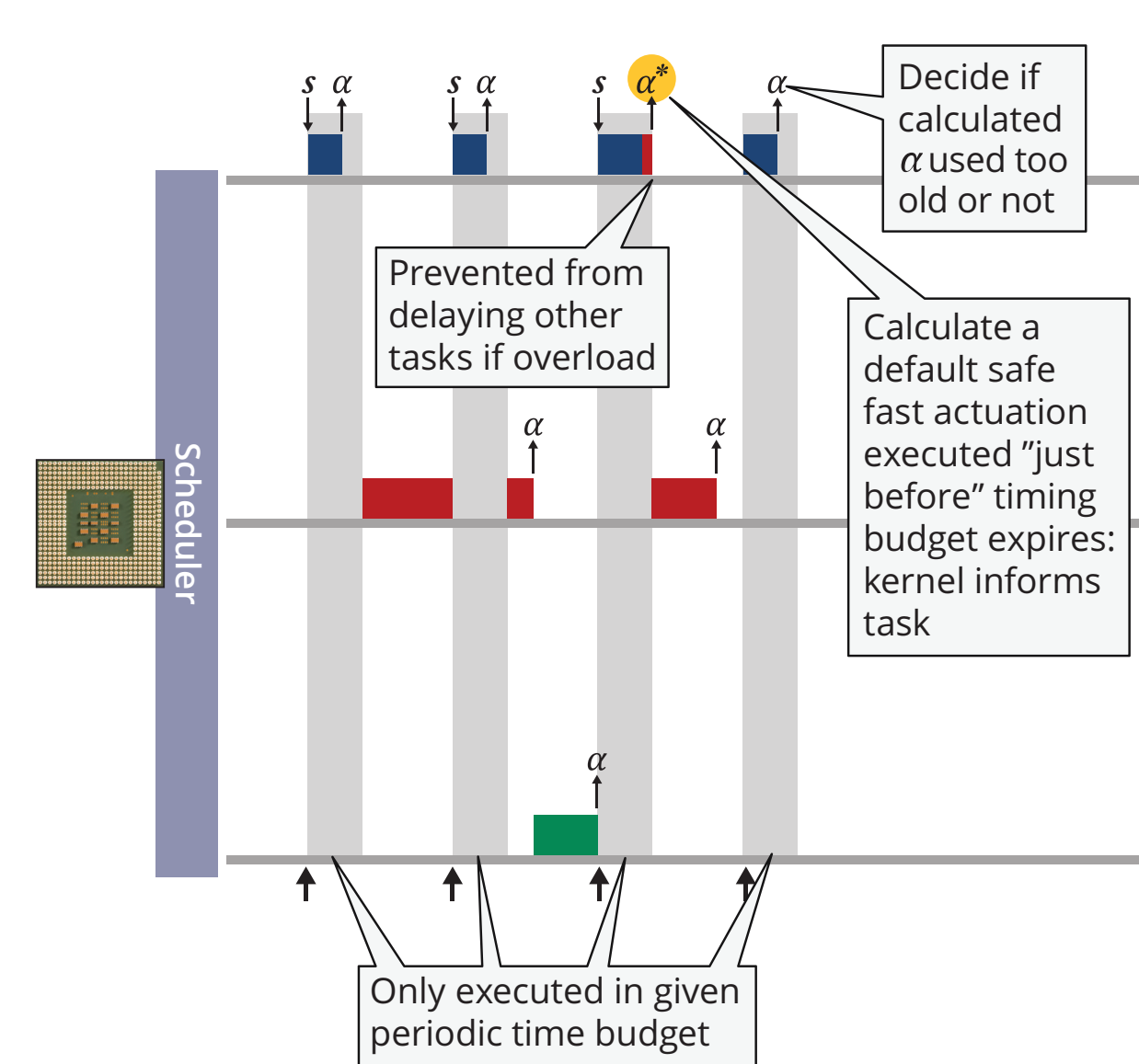
## How

- Each task gets a CPU budget stop task if budget exceeded
- If task about to exceed budget execute safe action

## Timing guarantees

- Never allow task to exceed budget
- Always execute actuation

## Safe actuation on timing enforcement



## Composing enforcers

- System with multiple enforcers: virtual fence+ collision avoidance
- Safe actions from different enforcers may conflict Drone at fence limit + other drone approaching
- Enforcer composition to resolve conflicts (1) priority based (2) utility maximization

## Formalization

Enforcer:  $E:(P, C_\phi, \mu, U)$   
 $\mu(s) \subseteq \text{SafeAct}(s); U$ : utility

No conflict:

$E_1(P_1, C_{\phi_1}, \mu_1, U_1), E_2(P_2, C_{\phi_2}, \mu_2, U_2)$   
 $\mu_{1,2}: \mu_1 \cap \mu_2$

Conflict: Priority resolution

$\mu_{1,2}: \mu_1 \cap \mu_2 \neq \emptyset ? \mu_1 \cap \mu_2 : \mu_1$

Conflict: Utility maximization

$\mu_{1,2}: \mu_1 \cap \mu_2 \neq \emptyset ? \text{argmax}_{\alpha \in \mu_1 \cap \mu_2} \sum U_i(s, \alpha') : \text{argmax}_{\alpha \in \mu_1} \sum U_i(s, \alpha')$

Enforcers allows

verification of complex CPS:  
Autonomous Vehicles

- Limit misbehavior
- With Verifiable Enforcers
- Result: Verified whole system

Copyright 2017 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.

Internal use:\* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:\* 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 external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

\* These restrictions do not apply to U.S. government entities.

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

DM17-0730

Certifiable Distributed Runtime Assurance