Certifiable Distributed Runtime Assurance

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: \( V_A \)
e.g., move-to\((x,y)\) action
- System state: \( s: V_S \rightarrow D \in S \)
- Actions: \( \alpha: V_A \rightarrow D \)
- Behavior: periodic state transition \( R_P(\alpha) \subseteq S \times S \)
\( R_P(\alpha,s) = \{s' | (s,s') \in R_P(\alpha)\} \)
- Safe state: \( \phi \subseteq S \)
- Enforceable state:
  \( \phi \supseteq C_\phi \)
- Safe Actuation
  \( SafeAct(s) = \{ \alpha | R_P(\alpha, s) \in C_\phi \} \)

Timing enforcement
- Unverified software may never finish!
- \( \Rightarrow \) 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
- Enforcer intercepts/replaces unsafe action at right time

Timing guarantees
- Never allow task to exceed budget
- Always execute actuation

Enforcing Drone Virtual Fence
Controller
Logical enforcer

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

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

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
\( \mu_{1,2}: \mu_1 \cap \mu_2 \neq \emptyset \) \( \arg \max_{\alpha \in \mu_1} \sum U_1(s,\alpha) : \arg \max_{\alpha \in \mu_2} \sum U_1(s,\alpha') \)

Enforcers allows verification of complex CPS: Autonomous Vehicles
- Limit misbehavior
- With Verifiable Enforcers
- Result: Verified whole system