Formal Design Theories and Tools for Safety-Critical Cyber-Physical Systems

Naijun Zhan

School of Computer Science, Peking University

2024 Summer School on Trustworthy Software

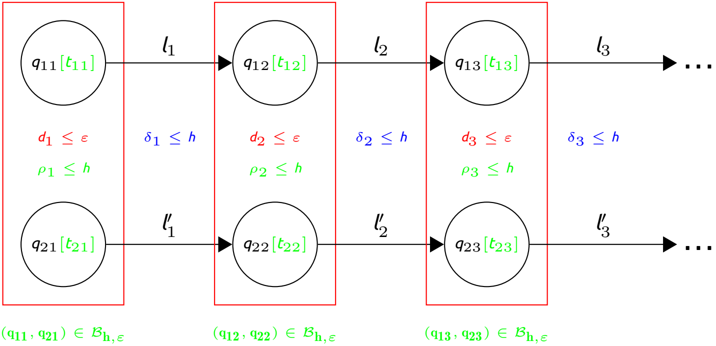
SEI of ECNU, Shanghai, July 9-12

Part 3 : Code Generation

Generating SystemC and ANSI-C from HCSP [FM'15, TOSEM'20, ICCPS'24]

Approximate Bisimulation [FM'15, TOSEM'20]

- Two HCSP processes $T_{P_i} = \langle Q_i, L_i, \twoheadrightarrow_i, Q_i^0, Y, H_i \rangle$ ($i = 1, 2$) with ouput set Y, are called **approximate bisimulation**, if there is a symmetric binary relation $\mathcal{B}_{h,\varepsilon} \subseteq Q_1 \times Q_2$ such that

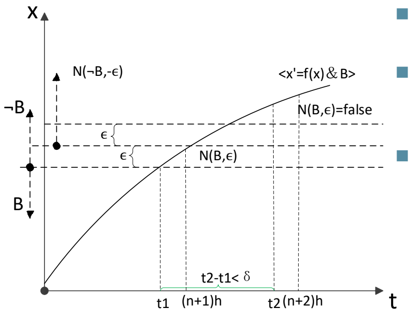


$d_i = \mathbf{d}(H_1(q_{1i}), H_2(q_{2i}))$, $\rho_i = \mathbf{d}(t_{1i}, t_{2i})$, and $\delta_i = \mathbf{d}(l_i, l'_i)$

Discretization of HCSP

Continuous Evolution : $\langle \dot{x} = f(x) \& B \rangle$

$$\frac{\langle \dot{\boldsymbol{x}} = f(\boldsymbol{x}) \& B \rangle}{\begin{array}{c}(N(B,\epsilon) \wedge N'(B,\epsilon) \rightarrow (\text{wait } h; \boldsymbol{x} := \boldsymbol{x} + h\Phi(\boldsymbol{x}, h)))^{\lfloor \frac{T}{h} \rfloor}; \\ (N(B,\epsilon) \wedge N'(B,\epsilon) \rightarrow (\text{wait } h'; \boldsymbol{x} := \boldsymbol{x} + h'\Phi(\boldsymbol{x}, h')); \\ N(B,\epsilon) \wedge N'(B,\epsilon) \rightarrow \text{stop}\end{array}}$$



- The ODE is robustly safe with respect to given precisions.
- when B turns false at t_1, the ODE continues to go away from B at least 2ϵ further within δ time.
- It is guaranteed to detect the change of B at next discretized step.

Discretization of HCSP

Continuous Interrupt $\langle \dot{\mathrm{x}} = \mathrm{e} \& B \rangle \trianglerighteq []_{i \in I}(ch_i * \to p_i)$

$$j_1 := 1; j_2 := -1;$$
$$(\neg(N(B,\varepsilon) \wedge N^n(B,\varepsilon)) \to j_1 := 0;$$
$$j_1 = 1 \wedge j_2 = -1 \to c := 0;$$
$$\langle \dot{c} = 1 \& c \leq h \rangle \trianglerighteq []_{i \in I}(ch_i * \to j_2 := i);$$
$$j_1 = 1 \wedge j_2 = -1 \to \mathrm{x} := \mathrm{x} + h\,(\mathrm{x}, h))^N;$$
$$j_2 \geq 0 \to \mathrm{x} := \mathrm{x} + c\,(\mathrm{x}, c); HtoD(p_{j_2});$$

Correctness Condtions

- HCSP process is (δ, μ)**-robustly safe**
- For any $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, f is **local Lipschitz**
- If unbounded time, it requires $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is **globally stable**

Code Generation (HCSP2SystemC, HCSP2C)

$$\mathrm{D}_{h,\epsilon}(\mathcal{P}) \quad \widehat{=} \quad \mathrm{D}_{h,\epsilon}(P_1)\|\mathrm{D}_{h,\epsilon}(P_2)\|\cdots\|\mathrm{D}_{h,\epsilon}(P_n)$$

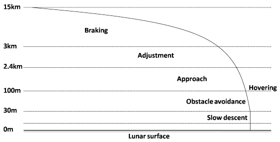
$$\begin{array}{rcl}
\mathrm{D}_{h,\epsilon}(\mathcal{P}) & \rightsquigarrow & \mathrm{SC_MODULE} \\
\mathrm{D}_{h,\epsilon}(P_i) & \rightsquigarrow & \mathrm{SC_THREAD}([\![\mathrm{D}_{h,\epsilon}(P_i)]\!]_{\mathrm{SC}}) \\
x := e & \rightsquigarrow & x = e \\
\mathrm{wait}\ d & \rightsquigarrow & \mathrm{wait}(d, \mathrm{SC_TU}) \\
\mathrm{D}_{h,\epsilon}(P); \mathrm{D}_{h,\epsilon}(Q) & \rightsquigarrow & [\![\mathrm{D}_{h,\epsilon}(P)]\!]_{\mathrm{SC}}; [\![\mathrm{D}_{h,\epsilon}(Q)]\!]_{\mathrm{SC}} \\
B \to \mathrm{D}_{h,\epsilon}(P) & \rightsquigarrow & \mathtt{if}(B)\{[\![\mathrm{D}_{h,\epsilon}(P)]\!]_{\mathrm{SC}}\} \\
\mathrm{D}_{h,\epsilon}(P) \sqcup \mathrm{D}_{h,\epsilon}(Q) & \rightsquigarrow & \mathtt{if}\ (\mathtt{rand()\%2})[\![\mathrm{D}_{h,\epsilon}(P)]\!]_{\mathrm{SC}}\ \mathtt{else}\ [\![\mathrm{D}_{h,\epsilon}(Q)]\!]_{\mathrm{SC}} \\
(\mathrm{D}_{h,\epsilon}(P))^* & \rightsquigarrow & \mathtt{for}\ (i = 1; i \leqslant \mathrm{num}(P^*); i++)[\![\mathrm{D}_{h,\epsilon}(P)]\!]_{\mathrm{SC}} \\
\cdots & &
\end{array}$$

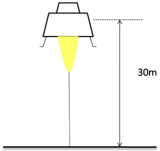
Correctness of HCSP2SystemC and HCSP2C

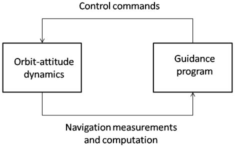
$$\mathcal{P} \quad \cong_{h,\epsilon} \quad \mathrm{D}_{h,\epsilon}(\mathcal{P}) \quad \cong \quad [\![\mathrm{D}_{h,\epsilon}(\mathcal{P})]\!]_{\mathrm{SC}}$$

GNC Control Program of Chang'e-3 [FM'14]

- Problem description







- System dynamics

$$\begin{cases} \dot{r} & = & v \\ \dot{v} & = & \frac{F_c}{m} - gM \\ \dot{m} & = & -\frac{F_c}{Isp_1} \\ \dot{F}_c & = & 0 \\ F_c & \in & [1500, 3000] \end{cases} \quad \text{and} \quad \begin{cases} \dot{r} & = & v \\ \dot{v} & = & \frac{F_c}{m} - gM \\ \dot{m} & = & -\frac{F_c}{Isp_2} \\ \dot{F}_c & = & 0 \\ F_c & \in & (3000, 5000] \end{cases}$$

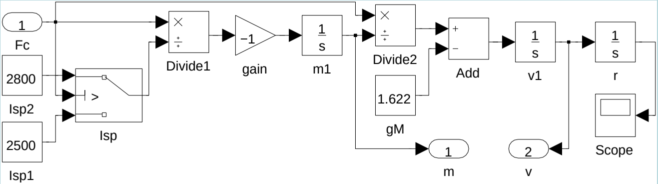
- Design objectives

(R1) $|v + 2| \leq 0.05$m/s during the slow descent phase and before touchdown;

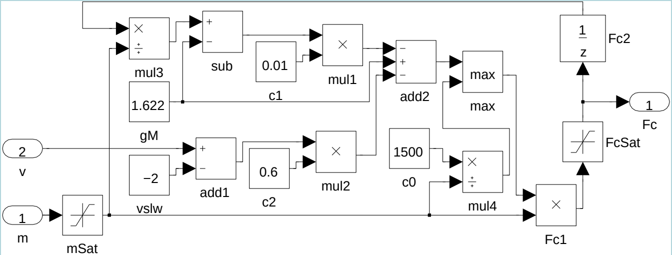
(R2) $|v| < 5$m/s at the time of touchdown;

Simulink Model of Powered Descent

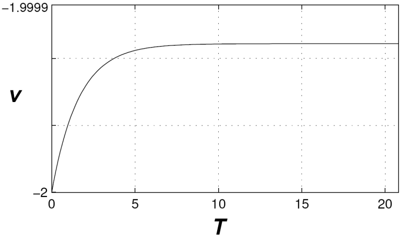
System dynamics

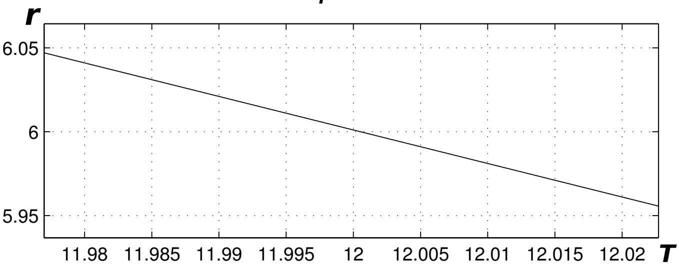


GNC conrol program



Simulation Results





From Simulink Model to HCSP Model

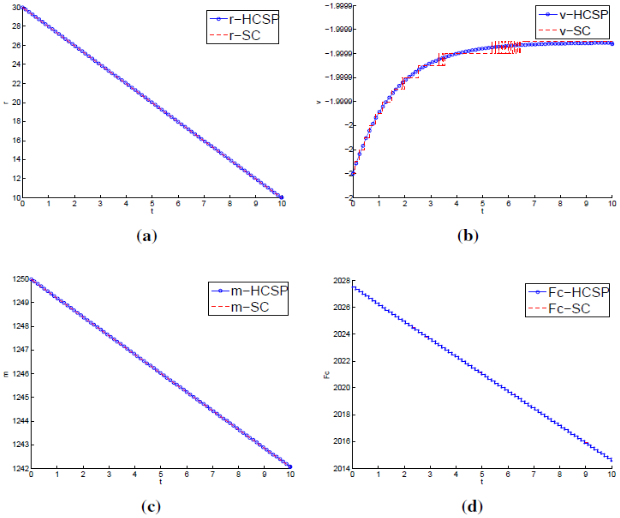
$$
\begin{array}{rcl}
P & \widehat{=} & PC \,\|\, PD \\
PC & \widehat{=} & v := -2;\ m := 1250;\ r := 30; \\
 & & (\ \langle Sys_1 \& f > 3000 \rangle \trianglerighteq CommI; \\
 & & \langle Sys_2 \& f \leq 3000 \rangle \trianglerighteq CommI\ )^* \\
PD & \widehat{=} & t := 0;\ g := 1.622;\ vslw := -2;\ f_1 = 2027.5; \\
 & & (\ ch_v?v_1;\ ch_m?m_1;\ f_1 := m_1 * aIC;\ ch_f!f_1; \\
 & & temp := t;\ \langle \dot{t} = 1 \& t < temp + 0.128 \rangle\ )^* \\
aIC & \widehat{=} & g - 0.01 * (f_1 / m_1 - g) - 0.6 * (v_1 - vslw) \\
Sys_1 & \widehat{=} & \dot{m} = -f/2548,\ \dot{v} = f/m - 1.622,\ \dot{r} = v \\
Sys_2 & \widehat{=} & \dot{m} = -f/2842,\ \dot{v} = f/m - 1.622,\ \dot{r} = v \\
CommI & \widehat{=} & ch_f?f \rightarrow \text{skip} \;[]\; ch_v!v \rightarrow \text{skip} \;[]\; ch_m!m \rightarrow \text{skip}
\end{array}
$$

Formal verification

Wtih **HHLProver** and **Invariant Generation** in **MARS**, we verified

$$\mathrm{P1} \qquad \{Pre\}\ P\ \{\mathrm{R}_2, \lceil \mathrm{R}_1 \rceil\}$$

Generated SystemC Code vs Legend C Code



Part 4 : Synthesis

Switching Logic Controller Synthesis [FM'12, FM'24]

Controller Synthesis

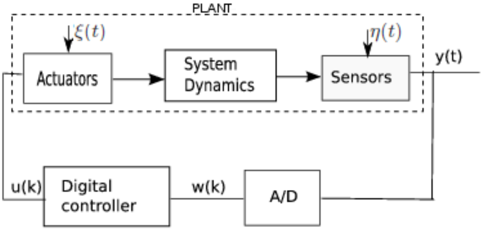
Controller Synthesis [from Wikipedia]

Given a model of the assumed behaviours of the environment and a system goal, controller synthesis means to construct an operational behaviour model for a component s.t. the system is guaranteed to satisfy the given goal when the environment is consistent with the given assumptions.

- An operation could be either inputs to dynamics, switching conditions, initial conditions, or reset functions.

Feedback controller

- Changing inputs impulsed on the dynamics (continuous or discrete)
- Steering the system to satisfy **stability**,**safety**, etc.



Controller Synthesis

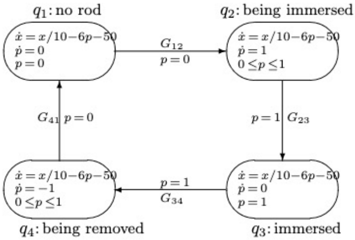
Controller Synthesis [from Wikipedia]

Given a model of the assumed behaviours of the environment and a system goal, controller synthesis means to construct an operational behaviour model for a component s.t. the system is guaranteed to satisfy the given goal when the environment is consistent with the given assumptions.

- An operation could be either inputs to dynamics, switching conditions, initial conditions, or reset functions.

Switching logic controller

- Refining the guard associated with each jump and the domain constraint in each mode
- Restricting the behavior so that the refined system satisfies the system objective



Controller Synthesis

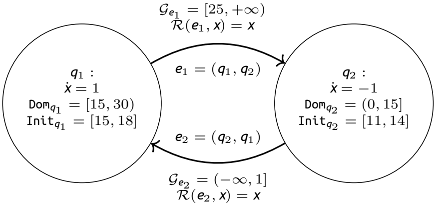
Controller Synthesis [from Wikipedia]

Given a model of the assumed behaviours of the environment and a system goal, controller synthesis means to construct an operational behaviour model for a component s.t. the system is guaranteed to satisfy the given goal when the environment is consistent with the given assumptions.

- An operation could be either inputs to dynamics, switching conditions, initial conditions, or reset functions.

Reset controller

- Redefining the reset map associated with each jump and refining the initial set of each mode
- Steering the modified system to achieve the system objective like **safety**, **stability**, **liveness**, etc.



Objective : $\mathcal{S}_1 = [15, 31)$, $\mathcal{S}_2 = (0, 14]$

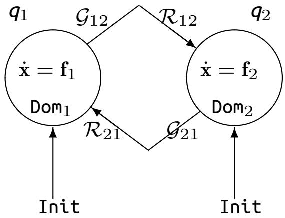
Part 4 : Synthesis

Switching Logic Controller Synthesis : w.r.t. Safety [FM'12]

Hybrid Automaton

$\mathcal{H} \,\widehat{=}\, (\mathcal{Q}, X, \mathbf{f}, \mathtt{Init}, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R})$ [Tomlin et al 00], where

- $\mathcal{Q} = \{q_1, \ldots, q_m\}$: discrete states, or modes
- $X = \{x_1, \ldots, x_n\}$: continuous state variables,
  $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$
- $\mathbf{f} : \mathcal{Q} \to (\mathbb{R}^n \to \mathbb{R}^n)$: continuous dynamics, $\mathbf{f}_q : \mathbb{R}^n \to \mathbb{R}^n$
- $\mathtt{Init} \subseteq \mathcal{Q} \times \mathbb{R}^n$: initial states
- $\mathtt{Dom} : \mathcal{Q} \to 2^{\mathbb{R}^n}$: domains $\mathtt{Dom}_q \subseteq \mathbb{R}^n$
- $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$: discrete transitions
- $\mathcal{G} : \mathcal{E} \to 2^{\mathbb{R}^n}$: switching guards
  $\mathcal{G}_e \subseteq \mathbb{R}^n$
- $\mathcal{R} : \mathcal{E} \to (\mathbb{R}^n \to \mathbb{R}^n)$: reset functions $\mathcal{R}(e, \cdot) : \mathbb{R}^n \to \mathbb{R}^n$



Problem Decription

- A safety requirement $\mathcal{S}$ assigns to each mode $q \in Q$ a safe region $S_q \subseteq \mathbb{R}^n$, i.e. $\mathcal{S} = \bigcup_{q \in Q}(\{q\} \times S_q)$.

Switching controller synthesis for safety [Asarin et al. 00]

Given a hybrid automaton $\mathcal{H}$ and a safety property S, find a hybrid automaton $\mathcal{H}' = (Q, X, f, D', E, G')$ such that

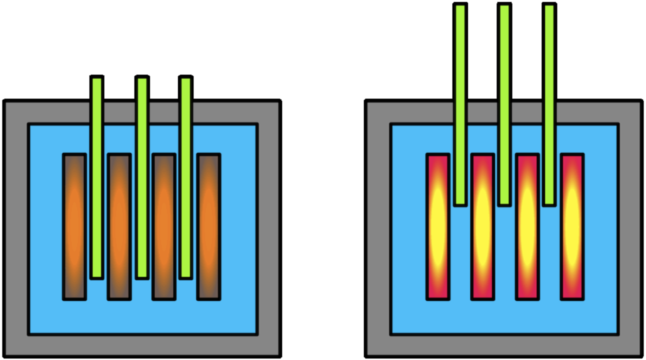
(r1) Refinement : for any $q \in Q$, $D'_q \subseteq D_q$, and for any $e \in E$, $G'_e \subseteq G_e$;

(r2) Safety : for any trajectory ω that $\mathcal{H}'$ accepts, if $(q, \mathbf{x})$ is on ω, then $\mathbf{x} \in S_q$;

(r3) Non-blocking : $\mathcal{H}'$ is non-blocking.

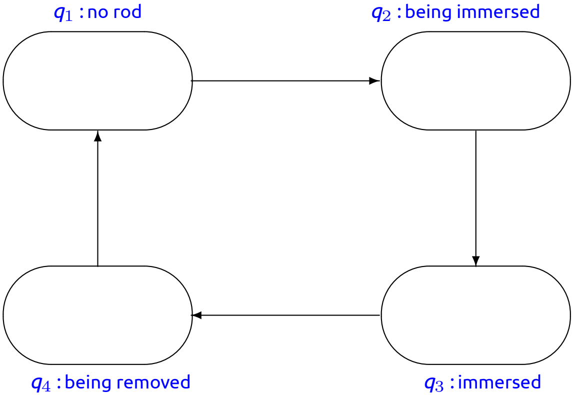
A Nuclear Reactor Example

The nuclear reactor system consists of a reactor core and a cooling rod which is immersed into and removed out of the core periodically to keep the temperature of the core in a certain range.



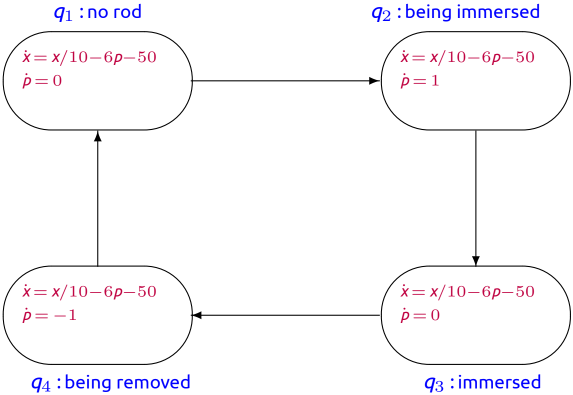
A Nuclear Reactor Example (Cont'd)

- x : temperature;
- p : proportion immersed



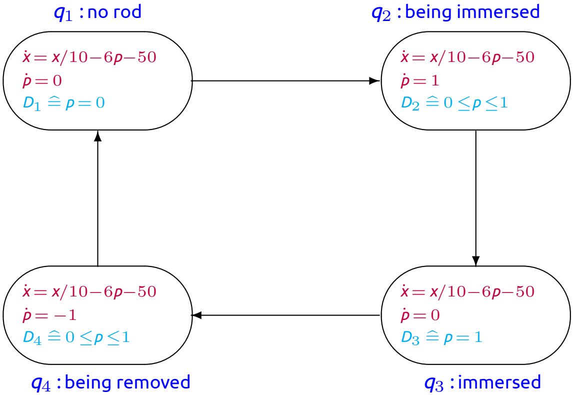
A Nuclear Reactor Example (Cont'd)

- x : temperature;
- p : proportion immersed



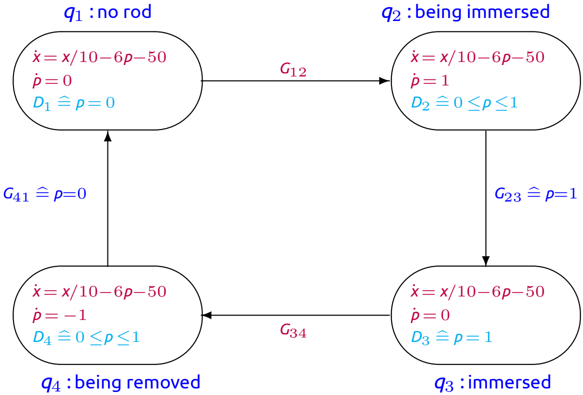
A Nuclear Reactor Example (Cont'd)

- x : temperature ;
- p : proportion immersed



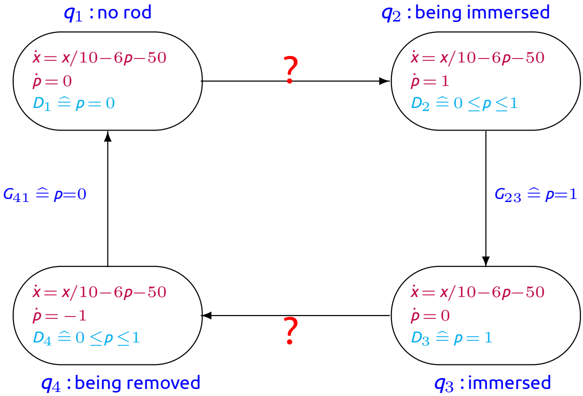
A Nuclear Reactor Example (Cont'd)

- x : temperature;
- p : proportion immersed



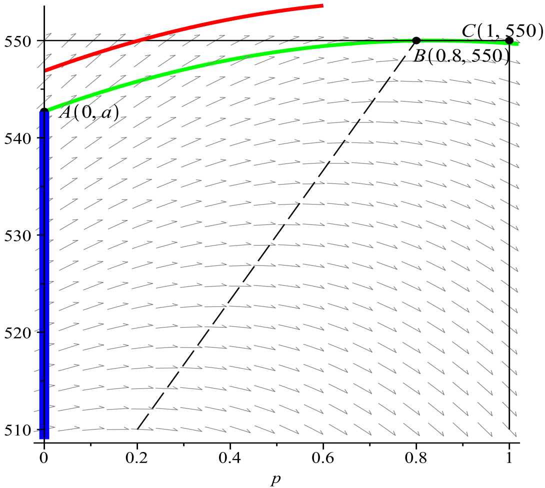
Switching Controller Synthesis for the Reactor

$S \widehat{=} 510 \leq x \leq 550$ for all modes



Bad Switching Violates Safety Property

Transition from mode q_1 to q_2



Solution to the Controller Synthesis Problem

Abstract Solution

Let $\mathcal{H}$ be a hybrid system and $\mathcal{S}$ be a safety property. If we can find a family of $D'_q \subseteq \mathbb{R}^n$ such that

(c1) for all $q \in Q$, $D'_q \subseteq D_q \cap S_q$;

(c2) for all $q \in Q$, D'_q is a continuous invariant of $(H_q, \mathbf{f}_q)$ with

$$H_q \mathrel{\widehat{=}} \Big(\bigcup_{e=(q,q')\in E} G'_e \Big)^c,$$

where $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$ for $e = (q, q')$, then the family of G'_e form a safe switching controller.

Template-Based Synthesis Framework

(s1) **Template assignment :** assign to each $q \in Q$ a template D'_q as the continuous invariant to be generated at mode q ;

(s2) **Guard refinement :** refine the transition guard G_e for each $e = (q, q') \in E$ by setting $G'_e \mathrel{\widehat{=}} G_e \cap D'_{q'}$;

(s3) **Deriving synthesis conditions :** encode (c1) and (c2) in the abstract solution into constraints on parameters appearing in the templates;

(s4) **Constraint solving :** solve the constraints derived from (s3) using quantifier elimination (QE);

(s5) **Parameters instantiation :** find an appropriate instantiation of D'_q and G'_e from the possible parameter values obtained at (s4)

Revisiting the Running Example (Cont'd)

The set of parameters : a, b, c, d

- $D_1' \mathrel{\widehat{=}} p = 0 \wedge 510 \leq x \leq a$
- $D_2' \mathrel{\widehat{=}} 0 \leq p \leq 1 \wedge x - b \geq p(d - b) \wedge$
  $x - 550 - \frac{36}{25}(a - 550)(p - \frac{5}{6})^2 \leq 0$
- $D_3' \mathrel{\widehat{=}} p = 1 \wedge d \leq x \leq 550$
- $D_4' \mathrel{\widehat{=}} 0 \leq p \leq 1 \wedge x - a \leq p(c - a) \wedge$
  $x - 510 - \frac{36}{25}(d - 510)(p - \frac{1}{6})^2 \geq 0$
- $G_{12}' \mathrel{\widehat{=}} p = 0 \wedge b \leq x \leq a$
- $G_{23}' \mathrel{\widehat{=}} p = 1 \wedge d \leq x \leq 550$
- $G_{34}' \mathrel{\widehat{=}} p = 1 \wedge d \leq x \leq c$
- $G_{41}' \mathrel{\widehat{=}} p = 0 \wedge 510 \leq x \leq a$
- $a = \frac{6575}{12} \wedge b = \frac{4135}{8} \wedge c = \frac{4345}{8} \wedge d = \frac{6145}{12}$.
- From this result we get that the cooling rod should be immersed before temperature rises to $\frac{6575}{12} = 547.92$, and removed before temperature drops to $\frac{6145}{12} = 512.08$.
- By solving differential equations explicitly, the corresponding exact bounds are 547.97 and 512.03

Part 4 : Synthesis

Switching Logic Controller Synthesis : against STL Specification [FM'24]

Signal Temporal Logic

Motivation

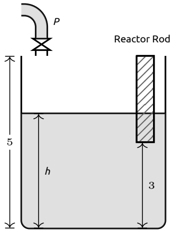
Many control objectives contain **timing constrains** in CPS.

A Chemistry Reactor System

- Pip P is ON : adding liquid, $\dot{h} = 1$
- Pip P is OFF : consuming liquid, $\dot{h} = -1$

Objective :

1. Keep liquid level in safe region (i.e., $0 \leq h \leq 4$)
2. Reaction between liquid and Reactor Rod happens at reaction phase $3 \leq t \leq 4$
3. Ensure $3 \leq h \leq 5$ when reaction happens



How to formulize this control objective ? —— STL
This can be written as STL formula :
$\varphi = (0 \leq h \leq 4)\mathcal{U}_{[3,4]}(3 \leq h \leq 5)$

Signal Temporal Logic

Syntax & Semantic

Syntax :

$$\varphi \widehat{=} \top \mid \mu(x, t) \geq 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi_1 \mathcal{U}_I \varphi_2$$

Semantic

$$\begin{array}{lll}
(\mathbf{x}, \tau) \models \mu(x, t) \geq 0 & \text{iff} & \mu(\mathbf{x}(\tau), \tau) \geq 0\,; \\
(\mathbf{x}, \tau) \models \neg\varphi & \text{iff} & (\mathbf{x}, \tau) \not\models \varphi\,; \\
(\mathbf{x}, \tau) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (\mathbf{x}, \tau) \models \varphi_1 \text{ and } (\mathbf{x}, \tau) \models \varphi_2\,; \\
(\mathbf{x}, \tau) \models \varphi_1 \mathcal{U}_I \varphi_2 & \text{iff} & \exists \tau' \geq \tau, \text{ such that } \tau' - \tau \in I,\ (\mathbf{x}, \tau') \models \varphi_2\,, \\
 & & \quad \text{and } \forall \tau'' \in [\tau, \tau'], (\mathbf{x}, \tau'') \models \varphi_1\,.
\end{array}$$

We considered a fragment of STL defined as above :

$$\phi \widehat{=} \mu(x, t) \geq 0 \mid \neg\phi \mid \phi \wedge \phi$$

$$\varphi \widehat{=} \phi_1 \mathcal{U}_I \phi_2$$

Problem Formulation

We considered the following switched system : $\Phi = (Q, F, \Theta, \pi)$, where

- $Q \widehat{=} \{ q_1, q_2, \ldots, q_m \}$ is a finite set of discrete modes.
- $F \widehat{=} \{ f_q \mid q \in Q \}$ is a set of vector fields, and each mode $q \in Q$ endows with a unique vector field f_q which specifies how system evolves in mode q.
- $\Theta \subseteq \mathbb{R}^n$ is a set of initial states.
- $\pi : \Theta \to (\mathbb{R}_{\geq 0} \to Q)$ is a switching controller. The controller maps each initial state $x_0 \in \Theta$ to a piecewise constant function $\pi(x_0)$, which in turn maps a time t to the corresponding control mode $\pi(x_0)(t)$.

Synthesis of Switching controller

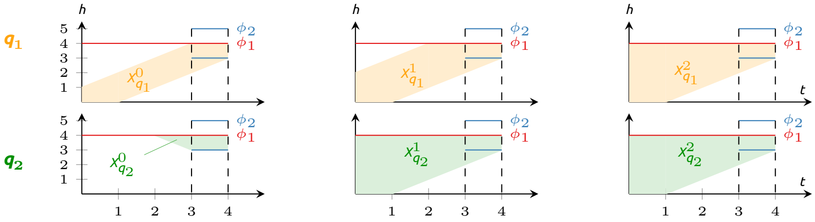
Given a finite set of discrete modes $Q = \{ q_1, q_2, \ldots, q_m \}$, a set of vector fields $F = \{ f_{q_1}, f_{q_2}, \ldots, f_{q_m} \}$, and an STL formula $\varphi = \phi_1 \mathcal{U}_I \phi_2$, the switched system synthesis problem aims to synthesize a switched system $\Phi = (Q, F, \Theta, \pi)$, such that $\Phi \models \varphi$.

We aim to identify a controller capable of specifying the **timing of discrete transitions**, ensuring the satisfaction of the provided STL formula from a given initial state.

State-Time Set

For a given STL formula φ, X_q^i denotes the set of state-time pairs (x, τ) where a switched system starting from state x at time τ in mode q can satisfy φ within i switches.

State-Time Set $X_{q_1}^i$, $X_{q_2}^i$ of Chemistry Reactor System



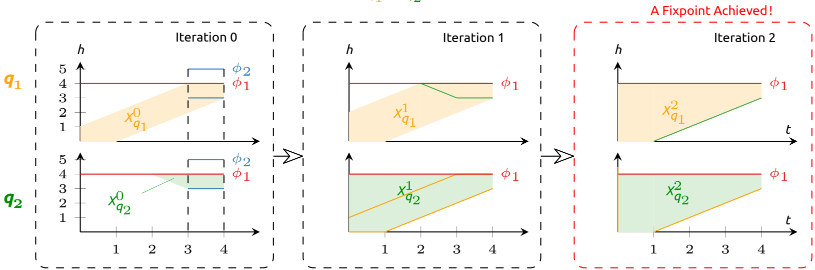
Start in q_1/q_2, initiate from $X_{q_1}^0$/$X_{q_2}^0$, φ can be satisfied **without switch**.

Start in mode q_1/q_2, initiate from $X_{q_1}^1$/$X_{q_2}^1$, φ can be satisfied **within one switch**.

Start in mode q_1/q_2, initiate from $X_{q_1}^2$/$X_{q_2}^2$, φ can be satisfied **within two switches**.

Computing State-Time Sets

State-Time Set $X^i_{q_1}$, $X^i_{q_2}$ Inductively Calculation



(x, τ) in $X^0_{q_1}$:
Satisfy $\phi_1 \mathcal{U}_I \phi_2$ in q_1.
(x, τ) in $X^0_{q_2}$:
Satisy $\phi_1 \mathcal{U}_I \phi_2$ in q_2.

(x, τ) in $X^1_{q_1}$:
Satisfy $\phi_1 \mathcal{U} X^0_{q_2}$ in q_1.
(x, τ) in $X^1_{q_2}$:
Satisy $\phi_1 \mathcal{U} X^0_{q_1}$ in q_2.

(x, τ) in $X^2_{q_1}$:
Satisfy $\phi_1 \mathcal{U} X^1_{q_2}$ in q_1.
(x, τ) in $X^2_{q_2}$:
Satisy $\phi_1 \mathcal{U} X^1_{q_1}$ in q_2.

Computing State-Time Sets

Theorem

For any $q \in Q$, suppose the solution of ODE $\dot{\mathrm{x}}(t) = f_q(\mathrm{x}(t))$ with initial x at time τ is denoted by $\Psi(\,\cdot\,; x, \tau, q)$, then the state-time sets can be inductively represented by

$$X_q^0 = \mathrm{QE}\Big(\exists\delta \geq 0,\ \Big(\phi_2[x, t{=}\Psi(t+\delta; x, t, q), t+\delta] \wedge (t+\delta \in I)\Big) \wedge \Big(\forall 0 \leq h \leq \delta,\ \phi_1[x, t{=}\Psi(t+h; x, t, q), t+h]\Big)\Big) \quad (1)$$

$$X_q^i = \bigvee_{q' \neq q} \mathrm{QE}\Big(\exists\delta \geq 0,\ \Big(X_{q'}^{i-1}[x, t{=}\Psi(t+\delta; x, t, q), t+\delta]\Big) \wedge \Big(\forall 0 \leq h \leq \delta,\ \phi_1[x, t{=}\Psi(t+h; x, t, q), t+h]\Big)\Big) \quad (2)$$

for any $q \in Q$ and any $i \in \mathbb{N}$.

- **Dynamics are Constant :** state-time set can be explicitly calculated using quantifier elimination in polynomial complexity.
- **Dynamics are Non-constant :** state-time set can be approximated using reach-avoid analysis [Xue et al, 2023]

Synthesizing Switched System

Given initial state,

- First, identify the minimal transitions required to satisfy the STL formula using the initial set $\Theta(q)^i := (X_q^i \setminus X_q^{i-1})[t=0]$ —— Algorithm 1
- Second, identify the time interval a discrete transition can happen using **reachability analysis** —— Algorithm 2

Algorithm 1 Synthesis of Switched system

Require: Q, F, $\varphi = \phi_1 \mathcal{U}_I \phi_2$, and k ▷ k is the upper bound of switching time
Ensure: A switched system $\Phi = (Q, F, \mathtt{Init}, \pi)$, such that $\Phi \models \varphi$
for all $q \in Q$ **do**
  $X_q^0 \leftarrow$ inner-approximate/explicitly calculate X_q^0
  $\mathtt{Init}(q)^0 \leftarrow X_q^0[t=0]$
end for
for $i = 1, 2, \cdots, k$ **do**
  for all $q \in Q$ **do**
    $X_q^i \leftarrow$ inner-approximate/explicitly calculate X_q^i
    $\mathtt{Init}(q)^i \leftarrow (X_q^i \setminus X_q^{i-1})[t=0]$ ▷ $\mathtt{Init}(q)^i$ is recorded for controller synthesis
  end for
end for
$\mathtt{Init} \leftarrow \cup_{q \in Q} \cup_{i=0}^{k} \mathtt{Init}(q)^i$ ▷ Initial set
Call Alg. 2 to obtain controller π ▷ Given any $x_0 \in \mathtt{Init}$, Alg. 2 computes the controller that drives x_0 to satisfy φ

Algorithm 2 Switching controller synthesis

Require: x_0, $\{X_q^i\}_{i=0}^k$, and $\{\texttt{Init}(q)^i\}_{i=0}^k$ ▷ x_0 is the initial state

Ensure: $\pi(x_0)$ ▷ The switching controller

Find the initial set $\texttt{Init}(q_0)^l$ that includes x_0 and has the smallest index l

Select q_0 as initial mode, $t_0 \leftarrow 0$

for $j = 1, \cdots, l$ **do**

    for $q \in Q$ **do**

        if $\texttt{Reach}(\widetilde{t}; x_{j-1}, t_{j-1}, q_{i-1}) \subseteq X_{\widetilde{q}}^{l-j}[t = \widetilde{t}]$ for some $\widetilde{t} > t_{j-1}, \widetilde{q} \in Q$ **then**

            Select $t_j \leftarrow \widetilde{t},\ \ q_j \leftarrow \widetilde{q}$

            $x_j \leftarrow \texttt{Reach}(t_j; x_{j-1}, t_{j-1}, q_{j-1})$

            Break

        end if

    end for

end for

$\pi(x_0) = (q_0, t_0)(q_1, t_1) \cdots (q_l, t_l)$ ▷ Representing a piecewise constant function such that $\pi(x_0)(t) = q_i$ if $t_i \leq t < t_{i+1}$

Example

In the Chemistry Reactor System, we have obtained the state-time sets $\{X^i_{q_1}, X^i_{q_2}\}$ for $i \leq 2$, thus, according to Alg. 1 (with $k = 2$), we have

$$\begin{array}{lll} \Theta(q_1)^0 = [0,1], & \Theta(q_1)^1 = (1,2], & \Theta(q_1)^2 = (2,4] \\ \Theta(q_2)^0 = \emptyset, & \Theta(q_2)^1 = [0,4], & \Theta(q_2)^2 = \emptyset. \end{array}$$

Based on this, we can synthesize a switched system Φ with $\Theta = \{h \mid 0 \leq h \leq 4\}$. The corresponding switching controller π is defined by

$$\pi(x_0) = \begin{cases} (q_1, 0), & \text{if } 0 \leq x_0 \leq 1 \\ (q_2, 0)(q_1, \frac{x_0 - 1}{2}), & \text{if } 1 < x_0 \leq 4. \end{cases}$$

Part 4 : Synthesis

Reset Controller Synthesis [RDCPS,Automatica]

Problem Formulation

Given an HA $\mathcal{H}$, we are interested in the following two types of reset controller synthesis problems :

Problem I : only with safety

Given a safe set $\mathcal{S} \subseteq \mathcal{Q} \times \mathcal{X}$, whether one can redefine $\mathtt{Init}$ and $\mathcal{R}$, and obtain a redesigned HA $\mathcal{H}' = (\mathcal{Q}, X, \mathbf{f}, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$, which is safe w.r.t. $\mathcal{S}$, and $\mathtt{Init}^r \subseteq \mathtt{Init}$;

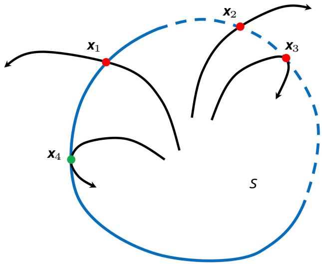
Problem II : safety+liveness

Given a safe set $\mathcal{S} \subseteq \mathcal{Q} \times \mathcal{X}$ and a target set $\mathcal{T} \subseteq \mathcal{Q} \times \mathcal{X}$, whether one can redefine $\mathtt{Init}$ and $\mathcal{R}$, and obtain a redesigned HA $\mathcal{H}' = (\mathcal{Q}, X, \mathbf{f}, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$, s.t. for any $(q, \boldsymbol{x}) \in \mathtt{Init}^r$, any trajectory starting from $(q, \boldsymbol{x})$ must reach $\mathcal{T}$, $\mathcal{H}'$ is safe w.r.t. $\mathcal{S}$ before reaching into $\mathcal{T}$, and $\mathtt{Init}^r \subseteq \mathtt{Init}$.

Transverse Set

Given a vector field $\mathbf{f}$ and a set $S \subseteq \mathbb{R}^n$, ***the transverse set*** of S w.r.t. $\mathbf{f}$, denoted by $\mathtt{trans}_{\mathbf{f}\uparrow S}$ of $\mathbf{f}$ over S, is defined by

- $\boldsymbol{x}_1 \in \mathtt{trans}_{\mathbf{f}\uparrow S}$
- $\boldsymbol{x}_2 \in \mathtt{trans}_{\mathbf{f}\uparrow S}$
- $\boldsymbol{x}_3 \in \mathtt{trans}_{\mathbf{f}\uparrow S}$
- $\boldsymbol{x}_4 \notin \mathtt{trans}_{\mathbf{f}\uparrow S}$



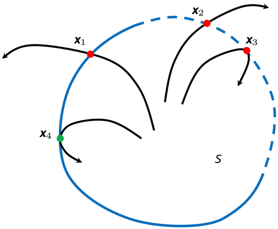
Differential Invariant (DI)

A set C is a ***differential invariant*** of vector field $\mathbf{f}$ w.r.t. a set S if for all $\boldsymbol{x} \in C$ and $T \geq 0$

$$\begin{pmatrix} \forall t \in [0, T]. \\ \phi(\boldsymbol{x}, t) \in S \end{pmatrix} \implies \begin{pmatrix} \forall t \in [0, T]. \\ \phi(\boldsymbol{x}, t) \in C \end{pmatrix}$$

In other words, $\mathtt{trans}_{\mathbf{f}\uparrow S \cap C} = \emptyset$.

- $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ do not belong to any DI w.r.t. S
- $\boldsymbol{x}_4$ belong to a DI w.r.t. S



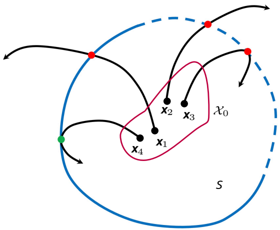
Reach-Avoid Set

Generalized Reach-Avoid Set

Given a vector field f, an initial set $\mathcal{X}_0$, a safe set S and a target set $\mathcal{T}$, the generalized (maximal) reach-avoid set $\mathtt{GRA}(\mathcal{X}_0 \xrightarrow[\mathrm{f}]{S} \mathcal{T})$ is defined

$$\mathtt{GRA}(\mathcal{X}_0 \xrightarrow[\mathrm{f}]{S} \mathcal{T}) \mathrel{\widehat{=}} \left\{ \boldsymbol{x} \in \mathcal{X}_0 \cap S \,\middle|\, \begin{array}{l} \exists T \geq 0. \forall t \in [0, T). \phi(\boldsymbol{x}, t) \in S \wedge \\ \forall \epsilon > 0. \exists t \in [T, T+\epsilon). \phi(\boldsymbol{x}, t) \in \mathcal{T} \end{array} \right\}.$$

- $\boldsymbol{x}_1 \in \mathtt{GRA}(\mathcal{X}_0 \xrightarrow[f]{S} \mathtt{trans}_{\mathrm{f}\uparrow S})$
- $\boldsymbol{x}_2 \in \mathtt{GRA}(\mathcal{X}_0 \xrightarrow[f]{S} \mathtt{trans}_{\mathrm{f}\uparrow S})$
- $\boldsymbol{x}_3 \in \mathtt{GRA}(\mathcal{X}_0 \xrightarrow[f]{S} \mathtt{trans}_{\mathrm{f}\uparrow S})$
- $\boldsymbol{x}_4 \notin \mathtt{GRA}(\mathcal{X}_0 \xrightarrow[f]{S} \mathtt{trans}_{\mathrm{f}\uparrow S})$



Computing TS, DI and GRA by SDP

Theorem

For a semialgebraic set S, let $C \widehat{=} S \setminus \mathtt{GRA}(S \xrightarrow[\mathrm{f}]{S} \mathtt{trans}_{\mathrm{f}\uparrow S})$, then C is a semialgebraic DI of f w.r.t. S, if f is polynomial.

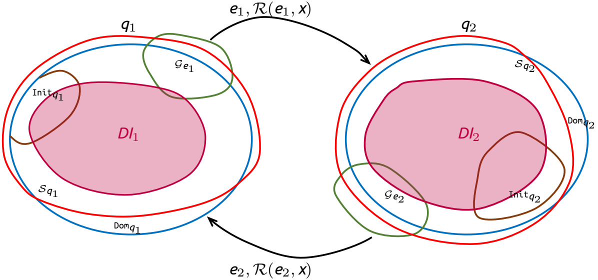
Theorem

Let S and D be a semialgebraic set, and f be polynomial, then $\mathtt{trans}_{f\uparrow S}$, $\mathtt{GRA}(S \xrightarrow[f]{S} D)$, and DI defined by $S \setminus \mathtt{GRA}(S \xrightarrow[f]{S} D)$ can be computed efficiently by SDP.

Reset synthesis only with safety

Basic idea

- **Stay within each mode forever**



$$\mathtt{Init}^r_{q_1} = \mathtt{Init}_{q_1} \cap (\mathtt{SD}_{q_1} \setminus \mathtt{GRA}(\mathtt{SD}_{q_1} \xrightarrow[\mathtt{f}_{q_1}]{\mathtt{SD}_{q_1}} \mathtt{trans}_{\mathtt{f}_{q_1}} \uparrow \mathtt{SD}_{q_1}))$$

$$\mathtt{Init}^r_{q_2} = \mathtt{Init}_{q_2} \cap (\mathtt{SD}_{q_2} \setminus \mathtt{GRA}(\mathtt{SD}_{q_2} \xrightarrow[\mathtt{f}_{q_2}]{\mathtt{SD}_{q_2}} \mathtt{trans}_{\mathtt{f}_{q_2}} \uparrow \mathtt{SD}_{q_2}))$$

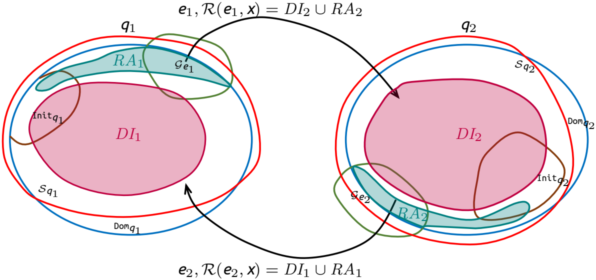
$$DI_1 = \mathtt{SD}_{q_1} \setminus \mathtt{GRA}(\mathtt{SD}_{q_1} \xrightarrow[\mathtt{f}_{q_1}]{\mathtt{SD}_{q_1}} \mathtt{trans}_{\mathtt{f}_{q_1}} \uparrow \mathtt{SD}_{q_1}),\ \mathtt{SD}_{q_1} = \mathcal{S}_{q_1} \cap \mathtt{Dom}_{q_1}$$

$$DI_2 = \mathtt{SD}_{q_2} \setminus \mathtt{GRA}(\mathtt{SD}_{q_2} \xrightarrow[\mathtt{f}_{q_2}]{\mathtt{SD}_{q_2}} \mathtt{trans}_{\mathtt{f}_{q_2}} \uparrow \mathtt{SD}_{q_2}),\ \mathtt{SD}_{q_2} = \mathcal{S}_{q_2} \cap \mathtt{Dom}_{q_2}$$

Reset synthesis only with safety

Basic idea

- **Reset to the switching part of the post-mode, but still inside a global invariant of the whole system.**



$$\mathtt{Init}^r_{q_1} = \mathtt{Init}_{q_1} \cap ((\mathtt{SD}_{q_1} \setminus \mathtt{GRA}(\mathtt{SD}_{q_1} \xrightarrow[\mathtt{f}_{q_1}]{\mathtt{SD}_{q_1}} \mathtt{trans}_{\mathtt{f}_{q_1}} \uparrow \mathtt{SD}_{q_1})) \cup RA_1)$$

$$\mathtt{Init}^r_{q_2} = \mathtt{Init}_{q_2} \cap ((\mathtt{SD}_{q_2} \setminus \mathtt{GRA}(\mathtt{SD}_{q_2} \xrightarrow[\mathtt{f}_{q_2}]{\mathtt{SD}_{q_2}} \mathtt{trans}_{\mathtt{f}_{q_2}} \uparrow \mathtt{SD}_{q_2})) \cup RA_2)$$

$$DI_1 = \mathtt{SD}_{q_1} \setminus \mathtt{GRA}(\mathtt{SD}_{q_1} \xrightarrow[\mathtt{f}_{q_1}]{\mathtt{SD}_{q_1}} \mathtt{trans}_{\mathtt{f}_{q_1}} \uparrow \mathtt{SD}_{q_1}), \ \mathtt{SD}_{q_1} = \mathcal{S}_{q_1} \cap \mathtt{Dom}_{q_1}$$

$$DI_2 = \mathtt{SD}_{q_2} \setminus \mathtt{GRA}(\mathtt{SD}_{q_2} \xrightarrow[\mathtt{f}_{q_2}]{\mathtt{SD}_{q_2}} \mathtt{trans}_{\mathtt{f}_{q_2}} \uparrow \mathtt{SD}_{q_2}), \ \mathtt{SD}_{q_2} = \mathcal{S}_{q_2} \cap \mathtt{Dom}_{q_2}$$

Reset synthesis only with safety

Algorithm

Algorithm Reset Control Synthesis Only with Safety

Require : $\mathcal{H} = (\mathcal{Q}, \mathcal{X}, \mathrm{f}, \mathtt{Init}, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R})$ and safe set $\mathcal{S}$

Ensure : $\mathcal{H}^r = (\mathcal{Q}, \mathcal{X}, \mathrm{f}, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$ satisfying $\mathcal{S}$

for each $q \in \mathcal{Q}$ **do**
- $\mathtt{SD}_q \leftarrow \mathcal{S}_q \cap \mathtt{Dom}_q$;
- $\mathtt{Dom}^r_q \leftarrow \mathtt{SD}_q \setminus \mathtt{GRA}(\mathtt{SD}_q \xrightarrow[\mathrm{f}_q]{\mathtt{SD}_q} \mathtt{trans}_{\mathrm{f}_q \uparrow \mathtt{SD}_q})$;
- **for** each $p \in \mathtt{Post}(q)$ **do**
  - $\mathtt{Dom}^r_q \leftarrow \mathtt{Dom}^r_q \cup \mathtt{GRA}(\mathtt{SD}_q \xrightarrow[\mathrm{f}_q]{\mathtt{SD}_q} \mathtt{Dom}^c_q \cap \mathcal{G}_e))$;
- **end for**
- **for** each $p \in \mathtt{Pre}(q)$ **do**
  - set $\mathcal{R}^r(e = (p, q), x) \subset \mathtt{Dom}^r_q$, for $x \in \mathcal{G}_e$;
- **end for**
- $\mathtt{Init}^r_q \leftarrow \mathtt{Init}_q \cap \mathtt{Dom}^r_q$;

end for

return $\mathcal{H}^r = (\mathcal{Q}, \mathcal{X}, \mathrm{f}, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$;

Reset Synthesis only with Safety

Correctness

Correctness

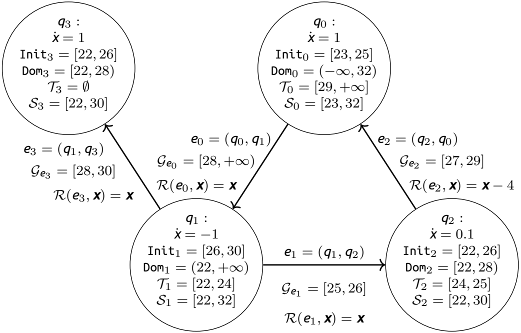
Problem I is solvable if and only if $\mathtt{Init}^r$ obtain from the above algorithm is not empty.

- **Soundness :** If $\mathtt{Init}^r$ obtained from the above algorithm is not empty, the resulting $\mathcal{H}^r$ solves **Problem I**.
- **Completeness :** If **Problem I** can be solved by some reset controller, $\mathtt{Init}^r$ obtained from the above algorithm is not empty.

Safety together with Liveness

Basic idea

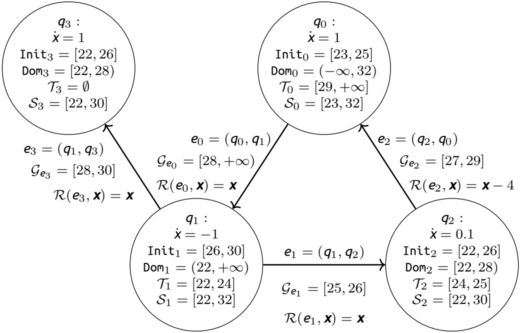
- Blocking executions like $\langle q_3 \rangle, \langle q_1, q_3 \rangle$: with $\mathcal{T}_3 = \emptyset$. This implies the liveness cannot be satisfied along these trajectories.
- Blocking all trajectories from Θ_3 ;
- Blocking all trajectories that can reach to q_3 via e_3.



Safety together with Liveness

Basic idea

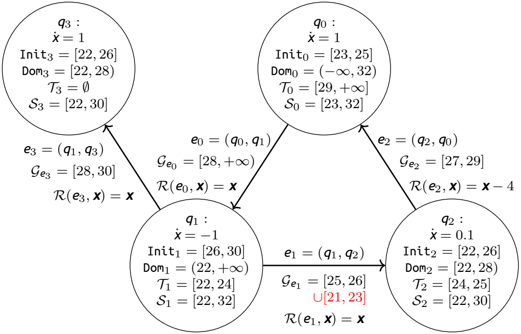
- Blocking executions like $\langle q_0, q_1, q_2, q_0 \rangle$: as it is possible that the trajectory keeps evolving along the loop safely forever.
- Blocking a selected discrete transition on the simple loop by redefining the reset maps associated with all incoming edges to and the initial set of the pre-mode of the transition.



Safety together with Liveness

Basic idea

- There may not exist a reset controller.



Safety together with Liveness

Algorithm

Algorithm Reset Control Synthesis With Safety and Liveness

Require : $\mathcal{H} = (\mathcal{Q}, \mathcal{X}, \mathrm{f}, \mathtt{Init}, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R})$, safe set $\mathcal{S}$ and target set $\mathcal{T}$

Ensure : $\mathcal{H}^r = (\mathcal{Q}, \mathcal{X}, \mathrm{f}, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$ that can guarantee that all trajectories can reach to $\mathcal{T}$ and satisfy $\mathcal{S}$ before reaching $\mathcal{T}$, or "No Such Reset Controllers Exist"

for each $q \in \mathcal{Q}$ **do**

    $\mathtt{SD}_q \leftarrow \mathcal{S}_q \cap \mathtt{Dom}_q$;

    $\mathtt{Dom}^r_q \leftarrow \mathtt{GRA}(\mathtt{SD}_q \xrightarrow[\mathrm{f}_q]{\mathtt{SD}_q} \mathcal{T}_q)$;

    for each $p \in \mathtt{Post}(q)$ **do**

    $\mathtt{Dom}^r_q \leftarrow \mathtt{Dom}^r_q \cup \mathtt{GRA}(\mathtt{SD}_q \xrightarrow[\mathrm{f}_q]{\mathtt{SD}_q} \mathtt{Dom}^c_q \cap \mathcal{G}_{e=(q,p)}))$ **end for**;

    $\mathtt{Init}^r_q \leftarrow \mathtt{Init}_q \cap \mathtt{Dom}^r_q$; $\mathtt{ST}_q \leftarrow \mathtt{ST}_q$ computed by (1);

end for

for each q with $\mathtt{Init}^r_q \neq \emptyset$ **do** *Refining_Dom*(q) **end for**;

for each $q \in \mathcal{Q}$ **do** $\mathtt{Init}^r_q \leftarrow \mathtt{Init}^r_q \cap \mathtt{Dom}^r_q$ **end for**;

for each $e = (p, q) \in \mathcal{E}$ **do**

    $\mathcal{R}^r(e, x) \subseteq \mathtt{Dom}^r_q$ **if** $\mathcal{R}^r(e, x)$ is not redefined in Algorithm 3;

end for

if $\mathtt{Init}^r = \cup_{q \in Q} \mathtt{Init}^r_q \neq \emptyset$ **then**

Safety together with Liveness

Correctness

Theorem [Correctness]

Problem II is solvable if and only if $\mathtt{Init}^r$ obtain from the above algorithm is not empty.

- **Soundness :** Our approach is sound, that is, any reset controller synthesized by the above approach does solve **Problem II** ;
- **Completeness :** Our approach is also complete, that is, if **Problem II** can be solved by some reset controller, the above approach does synthesize such one.

Reset Synthesis with Time Delay

- **Reset controller synthesis** with respect to **time delay** [Automatica]

Summing UP

On-going and Future Work

Modeling and specification

- To develop an IDE for AADL$\oplus$Simulink/Stateflow
- To extend HCSP and HHL for security and privacy
- To extend π-calculus to hybrid system for mobile IoT, including modeling, verification, security and privacy

Verification

- Complicated properties related to delay, probability, stochasticity
- More efficient invariant generation
- More efficient reachable set computation
- Simplification of HHL and improvement of the automation of HHLProver

Code generation

- Integrated controller synthesis, including reset controller, switching logic controller, and feedback controller
- From HCSP to RUST
- Improvement of MARS 2.0 and more case studies

More Readings

- N. Zhan, S. Wang and H. Zhao (2017) : Formal Verification of Simulink/Stateflow Diagrams, Springer-Verlag.
- J. Liu, J. Lv, Z. Quan, N. Zhan, H. Zhao, C. Zhou and L. Zou (2010) : A calculus for HCSP. Proc. of APLAS 2010, LNCS 6461.
- J. Liu, N. Zhan and H. Zhao (2011) : Computing semi-algebraic invariants for polynomial dynamical systems. Proc. of EMSOFT'11.
- H. Zhao, N. Zhan, D. Kapur, and K.G. Larsen (2012) : A "hybrid" approach for synthesizing optimal controllers of hybrid systems : A case study of the oil pump industrial example. Proc. of FM 2012, LNCS 7436.
- H. Zhao, M. Yang, N. Zhan, B. Gu, L. Zou and Y. Chen (2014) : Formal verification of a descent guidance control program of a lunar lander. Proc. of FM 2014, LNCS 8442.
- L. Zou, M. Fraenzle, N. Zhan and P. Mosaad (2015) : Automatic stability and safety verification for delay differential equations. Proc. of CAV 2015, LNCS.
- Q. Wang, M. Chen, B. Xue, N. Zhan, J.-P. Katoen (2022) : Encoding inductive invariants as barrier certificates synthesis via difference-of-convex programming. Information and Computation 289 :104965
- X. Xu, B. Zhan, S. Wang, J.-P. Talpin and N. Zhan (2022) : Semantics foundation for cyber-physical systems using higher-order UTP. ACM Transactions on Software Engineering and Methodology, 32(1), Article No. 9 :1-48
- G. Yan, L. Jiao, S. Wang, L. Wang and N. Zhan (2020) : Automatically generating SystemC code from HCSP formal mdoels. ACM Transactions on Software Engineering and Methodology, 29(1), Article 4 :1-39
- S. Wang, Z. Ji, B. Zhan, X. Xu, Q. Gao and N. Zhan (2024) : Formally Verified C Code Generation from Hybrid Communicating Sequential Processes. ICCPS 2024
- W. Wu, S. Feng, T. Gan, J. Wang, B. Xia and N. Zhan (2024) : On Completeness of SDP-Based Barrier Certificate Synthesis over Unbounded Domains. FM 2024
- H. Su, S. Feng, S, Zhan and N. Zhan (2024) : Switching Controller Synthesis for Hybrids Systems Against STL Formulas. FM 2024
- H. Su, J. Zhu, S. Feng, Y. Bai, B. Gu, J. Liu, M. Yang and N. Zhan (2024) : Reset Controller Synthesis by Reach-avoid Analysis for Delay Hybrid Systems. CoRR abs/2309.05908, conditional by AUTOMATICA

Thanks & Questions?