A Symbolic Control Approach to the Programming of Cyber-Physical Systems

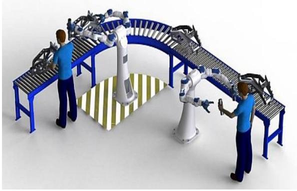
Antoine Girard

CNRS, Laboratoire des Signaux et Systèmes
Gif-sur-Yvette, France

ModeliScale project meeting
October, 15, 2020

Cyber-physical systems

Cyber-physical systems (CPS) consist of computational elements monitoring and controlling physical entities.



Design of CPS is challenging, time consuming and costly:

- Cyber/physical/human interactions
- Complex and multiple control objectives
- Critical safety requirements

Towards programmable CPS

Novel programming paradigm where the CPS (and not only its "cyber" component) is viewed as the execution platform:

- A *CPS program* describes the intended behavior of the CPS
  - Abstracts some characteristics of the cyber-physical execution platform
- A *CPS compiler* generates from a CPS program, control laws enforcing the specified behavior
  - Based on a model of the cyber-physical execution platform
  - Strong guarantees on the synthesized controller provided by the use of formal methods
- Rapid and dependable development/evolution of advanced functions of a CPS

Example - adaptive cruise control

Program

$$v_1^+ \geq v_1 + \alpha \quad \text{if } v_1 \leq v^* - \beta$$
$$v_1^+ \leq v_1 - \alpha \quad \text{if } v_1 \geq v^* + \beta$$

$d + l_2 < 0$

$d + l_2 \geq 0$

$d + l_2 < 0$

$$v_1^+ \leq v_1 - \alpha \quad \text{if } v_1 \geq v^* + \beta$$

$d + l_1 < 0$

Model

$$\begin{cases} d^+ &= d + \tau(v_1 - v_2) \\ v_1^+ &= f_1(v_1, u), u \in [u_{\min}, u_{\max}] \\ v_2^+ &= f_2(v_2, w), w \in [w_{\min}, w_{\max}] \end{cases}$$

Compiler
(controller synthesis)

Controller
(correct by design)
$u = \dots$

No Controller
(existing/found)

Outline of the talk

1. Formal controller synthesis from hybrid automata
   - A model matching problem
   - Symbolic control approach
   - Additional safety and reachability requirements
2. From hybrid automata to CPS programs
   - A proposal for a CPS programming language
   - Controller synthesis approaches to CPS compilation
3. Conclusions and perspectives

Transition systems

Definition

A *transition system* S is a tuple (X, U, Y, Δ, H), where

- X is a set of states
- U is a set of inputs
- Y is a set of outputs
- $\Delta : X \times U \rightrightarrows X$ is a set-valued transition map
- H: $X \longrightarrow Y$ is an output map

- The set of *enabled inputs* at state $x \in X$ is

$$\mathrm{enab}_\Delta(x) = \{u \in U | \ \Delta(x, u) \neq \emptyset\}$$

- The set of *non-blocking states* is

$$\mathrm{nbs}_\Delta = \{x \in X | \ \mathrm{enab}_\Delta(x) \neq \emptyset\}$$

Trajectories

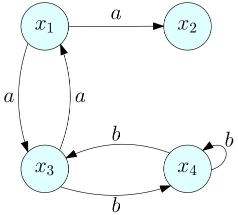
Definition

A *trajectory* of S is a sequence $(x_k, u_k)_{k=0}^{K}$, where $K \in \mathbb{N} \cup \{+\infty\}$ and

- $x_k \in X$, $u_k \in U$, for $0 \leq k \leq K$
- $x_{k+1} \in \Delta(x_k, u_k)$, for $0 \leq k < K$

A trajectory is called:

- *maximal*, if either $K = +\infty$ or $\Delta(x_K, u_K) = \emptyset$
- *complete*, if $K = +\infty$



$(x_4, b), (x_3, a), (x_1, b)$ is maximal

$(x_4, b), (x_3, a), (x_1, a), (x_3, b), (x_4, b), (x_4, b) \dots$ is complete

System and specification

System S_1:

$$x_{k+1} \in F(x_k, u_k)$$

$x_k \in X$, $u_k \in U$ where:

- $X \subseteq \mathbb{R}^{n_x}$ is the set of states
- $U \subseteq \mathbb{R}^{n_u}$ is the set of control inputs

Modeled by transition system

$$S_1 = (X, U, X, F, id_X)$$

Discrete time, continuous state.

System and specification

System S_1:

$$x_{k+1} \in F(x_k, u_k)$$

$x_k \in X$, $u_k \in U$ where:

- $X \subseteq \mathbb{R}^{n_x}$ is the set of states
- $U \subseteq \mathbb{R}^{n_u}$ is the set of control inputs

Modeled by transition system

$$S_1 = (X, U, X, F, id_X)$$

Discrete time, continuous state.

Specification S_2:

$$(x_{k+1}, p_{k+1}) \in G(x_k, p_k, v_k)$$

$x_k \in X$, $p_k \in P$, $v_k \in V$ where:

- P is a finite set of modes
- V is a finite set of external inputs

Modeled by transition system

$$S_2 = (X \times P, V, X, G, proj_X)$$

Discrete time, hybrid state.

Controller

Controller (θ, π) is a pair of set-valued maps:

$$\theta : X \times P \times V \rightrightarrows U \qquad \pi : X \times P \times X \times V \rightrightarrows P$$

Closed-loop system:

$$\begin{cases} x_{k+1} & \in & F(x_k, \theta(x_k, p_k, v_k)) \\ p_{k+1} & \in & \pi(x_k, p_k, x_{k+1}, v_k) \end{cases}$$

Compatibility condition: for all $x \in X$, $p \in P$, $v \in V$,

$$\theta(x, p, v) \subseteq \mathrm{enab}_F(x) \text{ and}$$
$$\forall x' \in F(x, \theta(x, p, v)),\ \pi(x, p, x', v) \neq \emptyset$$

Modeled by transition system

$$S_{cl} = (X \times P, V, X, \Delta_{cl}, proj_X)$$

Model matching problem

Problem (Model matching)

Synthesize:

- controller (θ, π) compatible with S_1
- controllable set $Z_c \subseteq X \times P$, $Z_c \neq \emptyset$

s.t. for every $(x_0, p_0) \in Z_c$, every maximal trajectory $(x_k, p_k, v_k)_{k=0}^{K}$ of S_{cl} is also a maximal trajectory of S_2.

Implication:

- every trajectory $(x_k, p_k, v_k)_{k=0}^{K}$ of S_{cl} is also a trajectory of S_2

Alternating simulation relation

Behavioral relationship between transition systems:

Definition (Tabuada 2008)

Let $S_a = (X_a, U_a, Y_a, \Delta_a, H_a)$, $S_b = (X_b, U_b, Y_b, \Delta_b, H_b)$ with $Y_a = Y_b$. $R \subseteq X_a \times X_b$ is an *alternating simulation relation* from S_a to S_b if:

1. for every $(x_a, x_b) \in R$, $H_a(x_a) = H_b(x_b)$
2. for every $(x_a, x_b) \in R$

$$\forall u_a \in \mathrm{enab}_{\Delta_a}(x_a),\ \exists u_b \in \mathrm{enab}_{\Delta_b}(x_b),$$
$$\forall x_b' \in \Delta_b(x_b, u_b),\ \exists x_a' \in \Delta_a(x_a, u_a),\ (x_a', x_b') \in R.$$

S_b *alternatingly simulates* S_a ($S_a \preceq_{AS} S_b$), if there exists an alternating simulation relation $R \neq \emptyset$ from S_a to S_b.

Solution characterization

Theorem

The model matching problem has a solution if and only if $S_2 \preceq_{AS} S_1$.

Controllers given alternating simulation relation $R \subseteq (X \times P) \times X$:

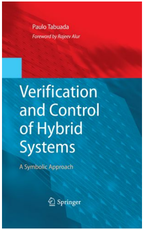
$$
\begin{aligned}
Z_c &= proj_{(X \times P)}(R) \\
\theta(x,p,v) &= \left\{ u \in \mathrm{enab}_F(x) \;\middle|\; \begin{array}{l} \forall x' \in F(x,u),\ \exists p' \in P : \\ (x',p') \in G(x,p,v) \cap Z_c \end{array} \right\} \\
\pi(x,p,x',v) &= \left\{ p' \in P \;\middle|\; (x',p') \in G(x,p,v) \cap Z_c \right\}
\end{aligned}
$$

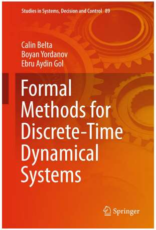
The model matching problem reduces to computing an alternating simulation relation from S_2 to S_1.

An approach based on symbolic control

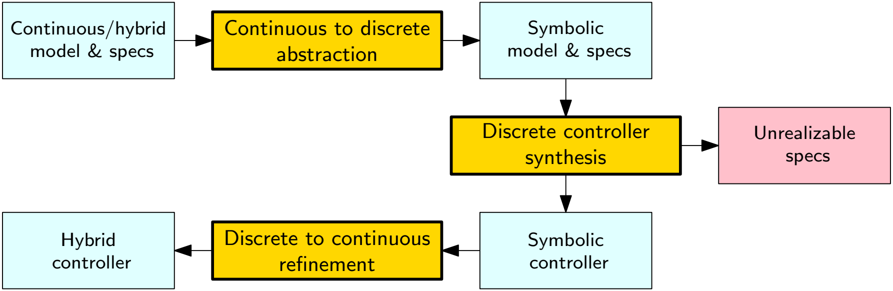
Symbolic control is a computational approach to controller synthesis:

- based on symbolic (i.e. finite state) abstractions of systems and specifications
- mathematical correctness of synthesized controllers
- applies to nonlinear systems with input/state constraints and bounded uncertainties
- heavy offline/light online computations





Symbolic control workflow



Approach to model matching problem

Main steps:

1. compute a symbolic abstraction $\hat{S}_1$ of system S_1: $\hat{S}_1 \preceq_{AS} S_1$
2. compute a symbolic abstraction $\hat{S}_2$ of specification S_2: $S_2 \preceq_{AS} \hat{S}_2$
3. compute alternating simulation relation from $\hat{S}_2$ to $\hat{S}_1$

If $\hat{S}_2 \preceq_{AS} \hat{S}_1$, transitivity of alternating simulation gives $S_2 \preceq_{AS} S_1$

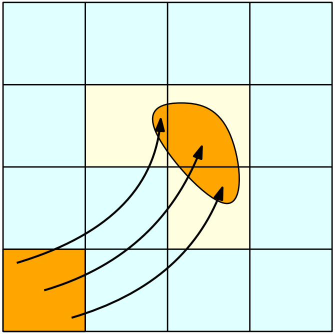
For abstraction, we use:

- a finite partition of X: $(X_q)_{q \in Q}$ where Q is a finite set of symbols
- a finite subset of control inputs $\hat{U} \subseteq U$

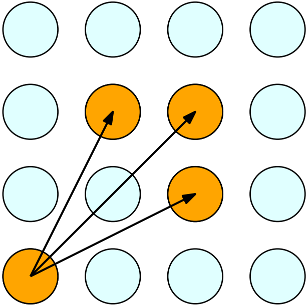
Abstraction of the system

Transition system $\hat{S}_1 = (X, \hat{U}, X, \hat{F}, id_X)$ with:

$$x' \in \hat{F}(x, \hat{u}) \iff x \in X_q,\ x' \in X_{q'},\ q' \in \Delta_1(q, \hat{u})$$



$F(X_q, \hat{u}),\ \hat{F}(X_q, \hat{u})$



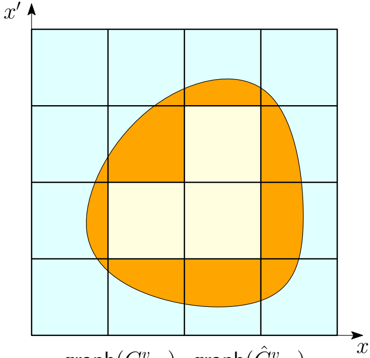
$\Delta_1(q, \hat{u})$

Abstraction of the specification

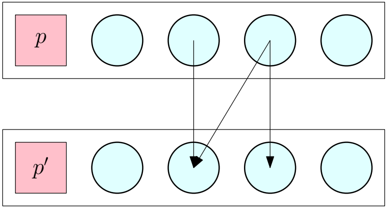
Transition system $\hat{S}_2 = (X \times P, V, X, \hat{G}, proj_X)$ with:

$$(x', p') \in \hat{G}(x, p, v) \quad \Leftrightarrow x \in X_q,\ x' \in X_{q'},\ (q', p') \in \Delta_2(q, p, v)$$

Rewrite $G(x, p, v) = \bigcup_{p' \in P} G^v_{p,p'}(x) \times \{p'\}$



$\text{graph}(G^v_{p,p'}),\ \text{graph}(\hat{G}^v_{p,p'})$



$(q', p') \in \Delta_2(q, p, v)$

Abstraction results

Proposition

For the proposed constructions:

- $\hat{S}_1 \preceq_{AS} S_1$
- *if, on their domain,* $G^v_{p,p'}$ *are Lipschitz and have images with non-empty interior, then we can build a partition* $(X_q)_{q \in Q}$ *such that* $S_2 \preceq_{AS} \hat{S}_2$

Alternating simulation relation

Given $\hat{Z} \subseteq Q \times P$, let the *predecessor* set be given by

$$\mathrm{Pre}(\hat{Z}) = \left\{ (q,p) \in Q \times P \;\middle|\; \begin{array}{c} \forall v \in \mathrm{enab}_{\Delta_2}(q,p), \exists u \in \mathrm{enab}_{\Delta_1}(q): \\ \forall q' \in \Delta_1(q,u), \exists p' \in P: \\ (q',p') \in \Delta_2(q,p,v) \cap \hat{Z} \end{array} \right\}$$

Fixed point algorithm:

$$\hat{Z}^0 = Q \times P, \; \hat{Z}^{k+1} = \mathrm{Pre}(\hat{Z}^k)$$

Theorem

The sequence $(\hat{Z}^k)_{k \in \mathbb{N}}$ reaches its fixed point $\hat{Z}^\infty = \bigcap_{k \in \mathbb{N}} \hat{Z}_k$ in finite number of iterations. The relation R given by

$$R = \{((x,p),x') \in (X \times P) \times X \mid x = x' \in X_q, \; (q,p) \in \hat{Z}^\infty \; \}$$

is an alternating simulation relation from $\hat{S}_2$ to $\hat{S}_1$ and also from S_2 to S_1.

Additional requirements

Limitations of the model matching problem formulation:

- No mechanism to avoid blocking states of the specification
  $\implies$ blocking states are winning states
- No possibility to specify termination conditions
  $\implies$ tasks run forever unless a blocking state is reached

We introduce a set of *terminal states* $Z_f \subseteq X \times P$:

- The task terminates when Z_f is reached
- Two termination semantics:
  1. blocking states that are outside Z_f should be avoided
     $\implies$ *Safety requirement*
  2. states in Z_f should be reached
     $\implies$ *Reachability requirement*

Safety requirement

Problem (Model matching problem with safety requirement)

Synthesize:

- controller (θ, π) compatible with S_1
- controllable set $Z_c \subseteq X \times P$, $Z_c \neq \emptyset$

s.t. for any $(x_0, p_0) \in Z_c$, any maximal trajectory $(x_k, p_k, v_k)_{k=0}^{K}$ of S_{cl}:

1. $(x_k, p_k, v_k)_{k=0}^{K}$ is a trajectory of S_2, $K \in \mathbb{N}$ and $(x_K, p_K) \in Z_f$; or
2. $(x_k, p_k, v_k)_{k=0}^{K}$ is a maximal trajectory of S_2, and either is complete, or $\mathrm{enab}_G(x_K, p_K) \neq \emptyset$

Implication:

- every maximal trajectory $(x_k, p_k, v_k)_{k=0}^{K}$ of S_{cl}, where for all $0 \leq k \leq K$, such that $(x_k, p_k) \notin Z_f$, $v_k \in \mathrm{enab}_G(x_k, p_k)$ is a trajectory of S_2 and either is complete or $(x_K, p_K) \in Z_f$.

Reachability requirement

Problem (Model matching problem with reachability requirement)

Synthesize:

- controller (θ, π) compatible with S_1
- controllable set $Z_c \subseteq X \times P$, $Z_c \neq \emptyset$

s.t. for any $(x_0, p_0) \in Z_c$, any maximal trajectory $(x_k, p_k, v_k)_{k=0}^{K}$ of S_{cl}:

1. $(x_k, p_k, v_k)_{k=0}^{K}$ is a trajectory of S_2, $K \in \mathbb{N}$ and $(x_K, p_K) \in Z_f$; or
2. $(x_k, p_k, v_k)_{k=0}^{K}$ is a maximal trajectory of S_2, $K \in \mathbb{N}$ and $\text{enab}_G(x_K, p_K) \neq \emptyset$

Implication:

- every maximal trajectory $(x_k, p_k, v_k)_{k=0}^{K}$ of S_{cl}, where for all $0 \leq k \leq K$, such that $(x_k, p_k) \notin Z_f$, $v_k \in \text{enab}_G(x_k, p_k)$ is a trajectory of S_2 and satisfies $K \in \mathbb{N}$ and $(x_K, p_K) \in Z_f$.

Controller synthesis

We use similar approaches based on symbolic abstractions $\hat{S}_1$ and $\hat{S}_2$. Consider the states of symbolic terminal states

$$\hat{Z}_f = \{(q,p) \in Q \times P |\ X_q \times \{p\} \subseteq Z_f\}.$$

Fixed point algorithms:

$$\hat{Z}_s^0 = \hat{Z}_f \cup \mathrm{nbs}_{\Delta_2}, \qquad \hat{Z}_s^{k+1} = \hat{Z}_f \cup \left(\mathrm{nbs}_{\Delta_2} \cap \mathrm{Pre}(\hat{Z}_s^k)\right)$$
$$\hat{Z}_r^0 = \hat{Z}_f, \qquad \hat{Z}_r^{k+1} = \hat{Z}_f \cup \left(\mathrm{nbs}_{\Delta_2} \cap \mathrm{Pre}(\hat{Z}_r^k)\right)$$

Theorem

The sequences $(\hat{Z}_s^k)_{k\in\mathbb{N}}$, $(\hat{Z}_r^k)_{k\in\mathbb{N}}$ reach their fixed points $\hat{Z}_s^\infty = \bigcap_{k\in\mathbb{N}} \hat{Z}_k$, $\hat{Z}_r^\infty = \bigcup_{k\in\mathbb{N}} \hat{Z}_k$ in finite number of iterations. Controllers solving the model matching problem with safety or reachability requirement can be constructed from these sets.

Example 1 - adaptive cruise control

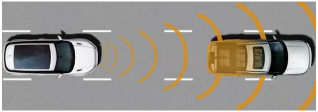
System

$$\begin{cases} d^+ &= d + \tau(v_1 - v_2) \\ v_1^+ &= f_1(v_1, u), u \in [u_{\min}, u_{\max}] \\ v_2^+ &= f_2(v_2, w), w \in [w_{\min}, w_{\max}] \end{cases}$$

Nonlinear dynamics

Input/state constraints

Bounded uncertainties



Specification

$d + l_2 < 0$

$v_1^+ \geq v_1 + \alpha$ if $v_1 \leq v^* - \beta$

$v_1^+ \leq v_1 - \alpha$ if $v_1 \geq v^* + \beta$

$d + l_2 \geq 0$ $\qquad$ $d + l_2 < 0$

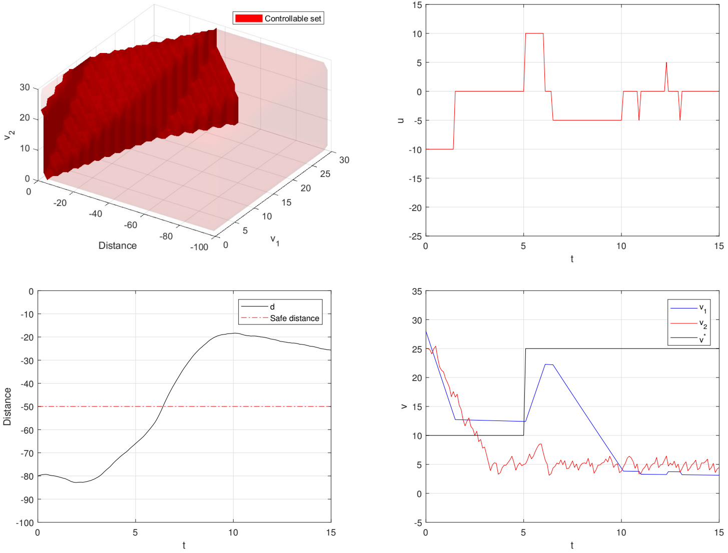
$d + l_1 < 0$

$v_1^+ \leq v_1 - \alpha$ if $v_1 \geq v^* + \beta$

Terminal set : $Z_f = \emptyset$

Termination semantics : safety

Example 1 - adaptive cruise control



Example 2 - take-over maneuver

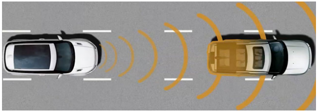
System

$$\begin{cases} d^+ & = & d + \tau\nu(v_1, v_2, |y - u_y|) \\ v_1^+ & = & f_1(v_1, u_v), u_v \in [u_{\min}, u_{\max}] \\ v_2^+ & = & f_2(v_2, w), w \in [w_{\min}, w_{\max}] \\ y^+ & = & u_y, u_y \in \{1, 2\} \end{cases}$$

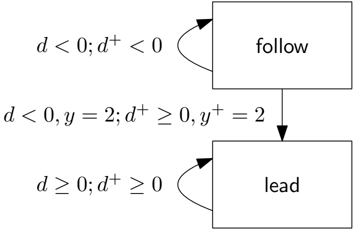
Nonlinear dynamics

Input/state constraints

Bounded uncertainties



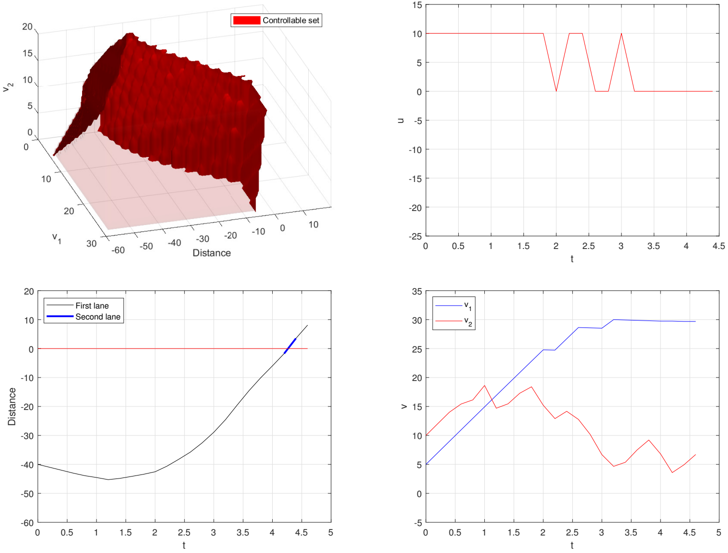
Specification



Terminal set : $Z_f = \{\text{lead}\} \times \{d \geq 0, y = 1\}$

Termination semantics : reachability

Example 2 - take-over maneuver



Outline of the talk

1. Formal controller synthesis from hybrid automata
   - A model matching problem
   - Symbolic control approach
   - Additional safety and reachability requirements
2. From hybrid automata to CPS programs
   - A proposal for a CPS programming language
   - Controller synthesis approaches to CPS compilation
3. Conclusions and perspectives

CPS programs

Consider a CPS modeled by transition system $S = (X, U, X, F, id_X)$.

A *program* for S consists of

- A collection of *tasks* $\mathcal{T}_1, \dots, \mathcal{T}_N$ each defined by
  - a transition system $S_i = (X \times P_i, V, X, G_i, proj_X)$
  - a set of terminal states $Z_{f,i} \subseteq X \times P_i$
  - a termination semantics "safety" or "reachability"
- A *scheduler* $\Sigma : Z_f \rightrightarrows P$ where

$$Z_f = Z_{f,1} \cup \cdots \cup Z_{f,N} \text{ and } P = P_1 \cup \cdots \cup P_N$$

- A set of *terminal states* $Z_{f,0} \subseteq Z_f$

Program executions

Executions of a CPS program:

- Execution of the current task specified by model matching problem with appropriate termination semantics
- Upon termination of the task:
    - If a terminal state of the program $(x, p) \in Z_{f,0}$ is reached then the program terminates
    - Otherwise, use the scheduler to select $p' \in \Sigma(x, p)$ and execute new task starting in state (x, p')

Programs and executions can also be defined inductively.

Controllers from CPS programs

- Synthesize a controller for each task, let $Z_{c,i}$ the associated controllable set
- The controllers are *schedulable* if

$$\forall (x,p) \in Z_f \setminus Z_{f,0},\ \exists p' \in \Sigma(x,p):\ (x,p') \in Z_c$$

  where $Z_c = Z_{c,1} \cup \dots \cup Z_{c,N}$.
- If controllers are not schedulable, it is possible to use fixed point algorithms to refine terminal conditions of tasks and re-synthesize controllers, until schedulability (maximal or anytime synthesis).

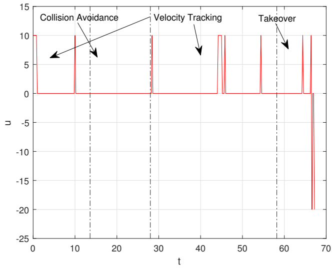
Example

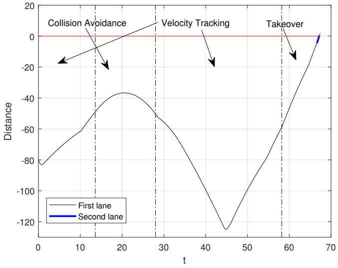
Task 1: adaptive cruise control

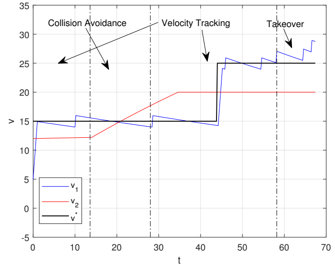
- $Z_{f,1} = \{d + 60 \geq 0, v_1 - v_2 \geq 5\}$
- Safety semantics

Task 2: take-over maneuver

- $Z_{f,2} = \{d \geq 0, y = 1\}$
- Reachability semantics







Outline of the talk

1. Formal controller synthesis from hybrid automata
   - A model matching problem
   - Symbolic control approach
   - Additional safety and reachability requirements
2. From hybrid automata to CPS programs
   - A proposal for a CPS programming language
   - Controller synthesis approaches to CPS compilation
3. Conclusions and perspectives

Conclusion and perspectives

- A proposal for a language to program CPS
  - Intuitive description of elementary tasks using hybrid automata[1]
  - Specification of complex behaviors by scheduling elementary tasks[2]
- Feasibility of CPS program compilation
  - Automatic model based controller synthesis using formal methods
  - Proof of concept using symbolic control techniques[3]
- Future work
  - Infeasible specifications: synthesis of least violating controllers (with distance certificate)
  - Performance optimization by combining symbolic approaches with model predictive control or deep neural networks

[1] Sinyakov & Girard, Formal controller synthesis from specifications given by discrete-time hybrid automata, hal-02361404v1

[2] Sinyakov & Girard, Formal synthesis from control programs, CDC 2020

[3] Co4Pro toolbox: `https://github.com/girardan/Co4Pro`