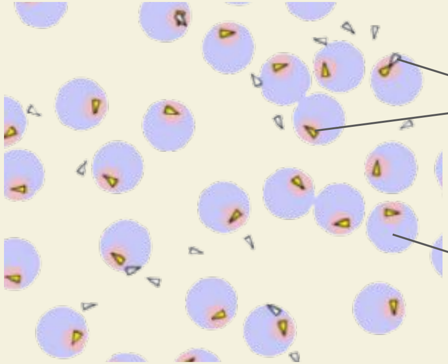
Accounting for communication delays

Guruprerana Shabadi
Undergraduate student at École Polytechnique | Intern with Sergio Mover @ Cosynus Team, LIX

Motivation

- Consider the collision avoidance problem with drones navigating in space
- Drones need to communicate their positions to other drones to avoid collision
- Delays arising with communication protocols are inevitable

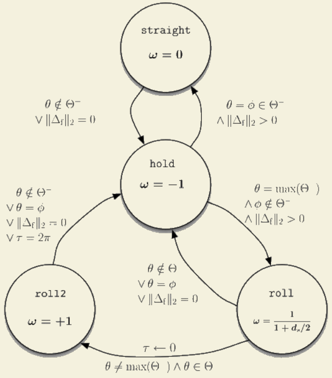


Drones trying to navigate without collision

Reserved regions which cannot overlap

This automata can help each drone navigate safely…

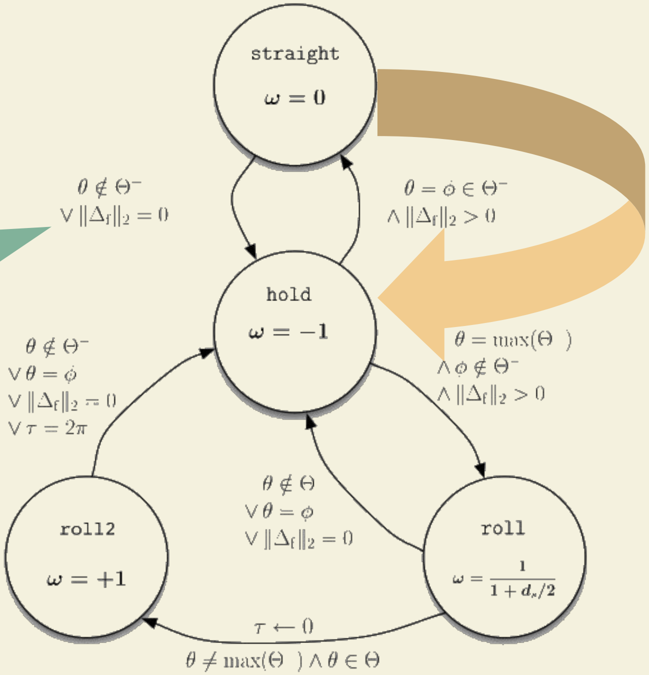
if there were no communication delays



Figures from Pallottino, L., Scordio, V. G., Bicchi, A., and Frazzoli, E. 2007. Decentralized cooperative policy for conflict resolution in multivehicle systems. IEEE Transactions on Robotics, 23(6):1170–1183.

Hybrid automata cannot capture delays

The positions of other drones evolve in other systems and appear as ‘inputs’ in our system



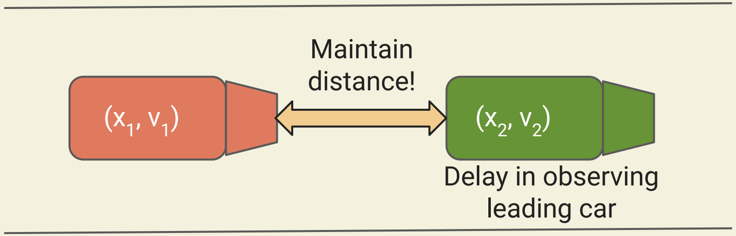
When two drones come close, they need to switch from *straight* mode to *hold* mode

But what if there is a delay in knowing the position of the other drone?

Collision avoidance policy from Pallottino, L., Scordio, V. G., Bicchi, A., and Frazzoli, E. 2007. Decentralized cooperative policy for conflict resolution in multivehicle systems. IEEE Transactions on Robotics, 23(6):1170–1183.

Nature of the delays

- Do not affect dynamics within a discrete mode
- Only affect *which* discrete mode we are in, i.e., they affect transitions



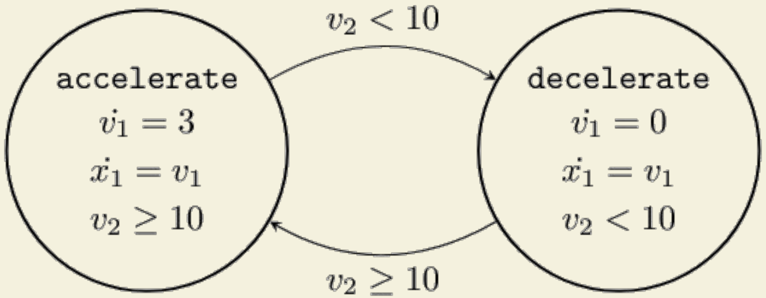


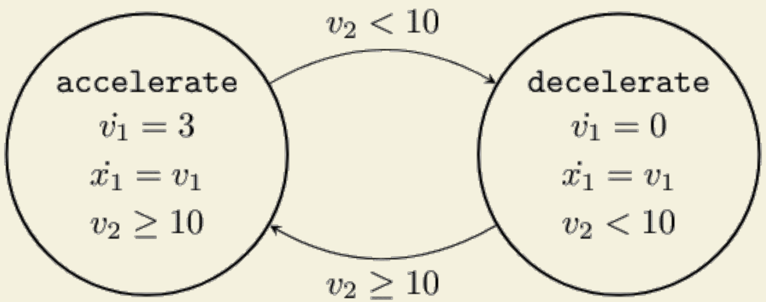
Fig: An oversimplified cruise control system

How do we model these delays?

Preliminary: Hybrid automata

A hybrid automata is composed of

- Vector of system variables $(x_1, \ldots, x_n)$
- Set of discrete modes $Q = \{q_1 \ldots, q_k\}$ and an associated invariant I_{q_i}
- Transitions between them $E \subseteq Q \times Q$
- A vector field associated to each discrete mode F_{q_i}
- A guard set associated to each transition $G_{(q_i, q_j)}$
- A reset map associated to each transition $R_{(q_i, q_j)}$



How can we extend the semantic of hybrid automata

A lazy hybrid automata LHA (in the continuous semantic) is composed of

- Vector of system variables $(x_1, \ldots, x_m)$
- A vector of 'input' variables $(y_1, \ldots, y_n)$
- Set of discrete modes $Q = \{q_1 \ldots, q_k\}$ and an associated invariant I_{q_i}
- Transitions between them $E \subseteq Q \times Q$
- A vector field associated to each discrete mode F_{q_i}
- A guard set associated to each transition $G_{(q_i,q_j)}$
- A reset map associated to each transition $R_{(q_i,q_j)}$
- Set of time bounds associated to each 'input' variable

$$B = ((l_1, u_1), \ldots, (l_n, u_n))$$

Inspired from the discrete semantic presented in *Agrawal, M., Thiagarajan, P.: Lazy rectangular hybrid automat. In: 7th HSCC, LNCS 2993, Springer (2003) 1–15.*

A different semantic for guard evaluation at transitions

Hybrid automata

Can take transition if

$$(X(t), y_1(t), \ldots, y_n(t)) \in G_{(q_i,q_j)}$$

Lazy hybrid automata

Can take transition if

$$\exists t_1 \in [t - l_1, t - u_1], \ldots, \exists t_n \in [t - l_n, t - u_n]$$

$$(X(t), y_1(t_1), \ldots, y_n(t_n)) \in G_{(q_i,q_j)}$$

Intuitively...

Suppose we can have *upto* 3s delay in observing the velocity of the leading car

`if`

$$\exists t' \in [t-3, t], v_2 < 10$$

`then`

`switch to 'slow'`

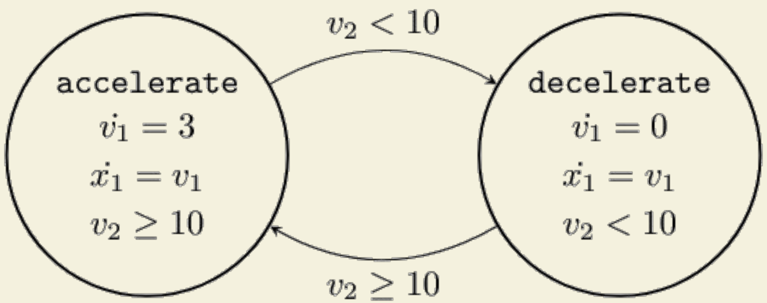


Fig: The running cruise control example

We can take a transition if the guard was satisfied at some point in the past!

Same with invariants

Hybrid automata

`Invariant holds if` $\forall t \in [0, T]$

$$(x_1(t), \ldots, x_n(t)) \in I_{q_i}$$

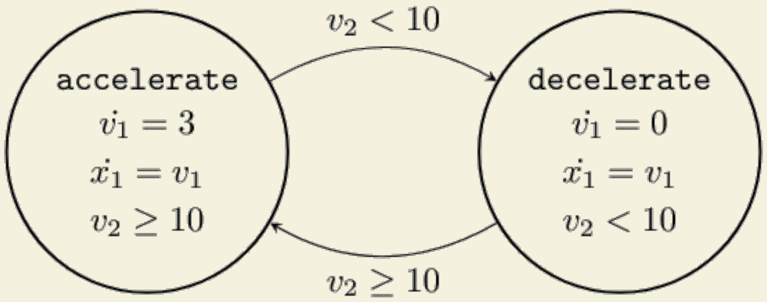
Lazy hybrid automata

`Invariant holds if` $\forall t \in [0, T]$

$$\exists t_1 \in [t - l_1, t - u_1], \ldots, \exists t_n \in [t - l_n, t - u_n]$$

$$(x_1(t_1), \ldots, x_n(t_n)) \in I_{q_i}$$

An example trace in a lazy hybrid automaton



- In this case, we fix the dynamics of v_2 to evolve at a constant rate of -1 starting from an initial value of 15, i.e., $v_2(t) = 15 - t$
- We have a delay of upto *2s* in the observation of v_2

***t* = 0**

mode = **`accelerate`**
$v_1 = 0$
$x_1 = 0$
$v_2 = 15$

Invariant satisfied for all *t* in [0, 6]

***t* = 6**

mode = **`accelerate`**
$v_1 = 18$
$x_1 = 54$
$v_2 = 9$

$v_2(t - 0.5) = 9.5 < 10$

***t* = 6**

mode = **`decelerate`**
$v_1 = 18$
$x_1 = 54$
$v_2 = 9$

Challenges

- Understanding the expressive power of the model
- Can we translate these into regular hybrid automata?
- Identifying reasonable restrictions/assumptions to allow translations, i.e., under what conditions are they equivalent to regular hybrid automata?

Translations!

Translation of *lazy* hybrid automata

What works

✔ New guard evaluation semantic

What does not work

❌ *Almost non-zeno* condition required: upper bound on number of transitions within a given time period

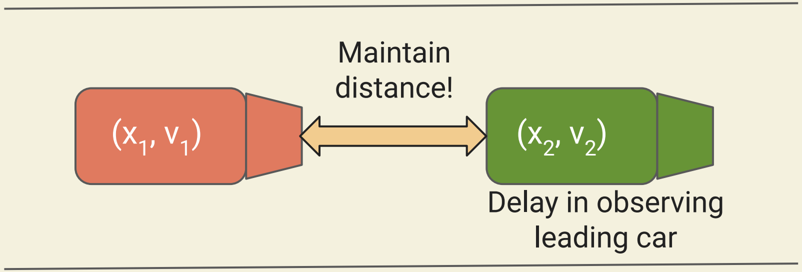
❌ Invariants

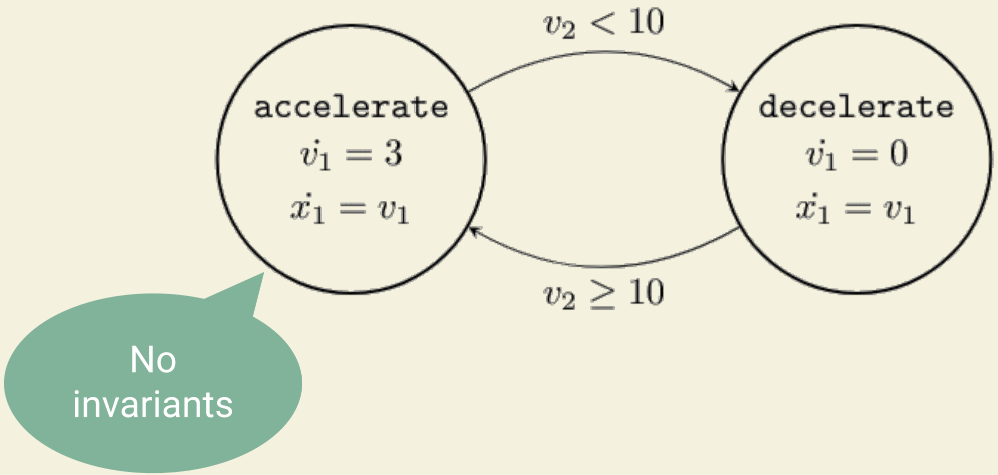
To translate

Consider our example of cruise control with four state variables x_1, v_1, x_2, v_2 such that the time bounds for v_2 are $[t-2, t]$, i.e., there can be upto a *2s* delay in the communication of v_2

Almost non-zeno condition:

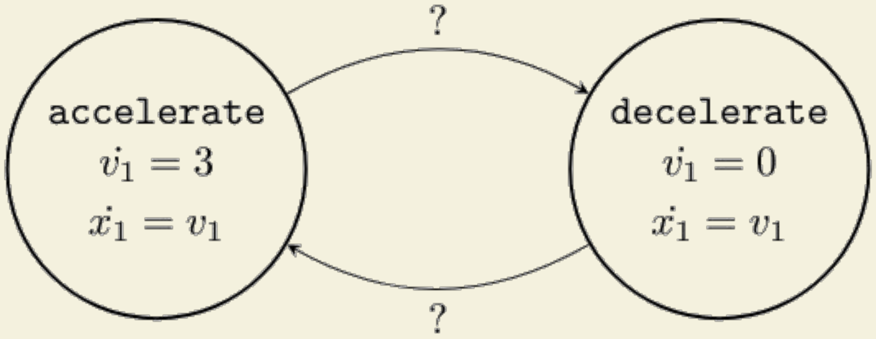
Suppose there can be *at most 3* transitions in a *2s* window





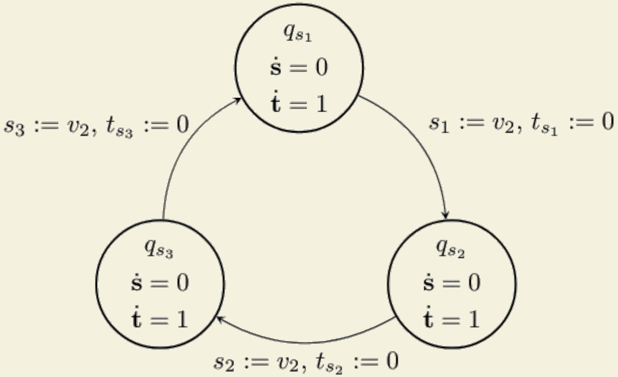
Translation: the new hybrid automaton

Introduce 3 new variables s_1, s_2, and s_3 to save the state of v_2



||

Parallel composition



We *save* the state of v_2 when needed

Translation: new guard conditions

$$v_2 < 10$$

Was $v_2 < 10$ sometime in the past $2s$?
Well, we can check the saved states of v_2!

$$t_{s_1} < 2 \wedge s_1 < 10$$
$$\bigvee t_{s_2} < 2 \wedge s_2 < 10$$
$$\bigvee t_{s_3} < 2 \wedge s_3 < 10$$

Invariants are hard to translate in the new semantic

Cannot hope to store every value taken on by a variable within the time bounds, i.e., we would require infinite storage/computation

Same problem with removing the almost non-zeno condition!

Translation of *lazy* timed automata

What works

✔ Invariants along with their new semantic

✔ New guard evaluation semantic

What does not work

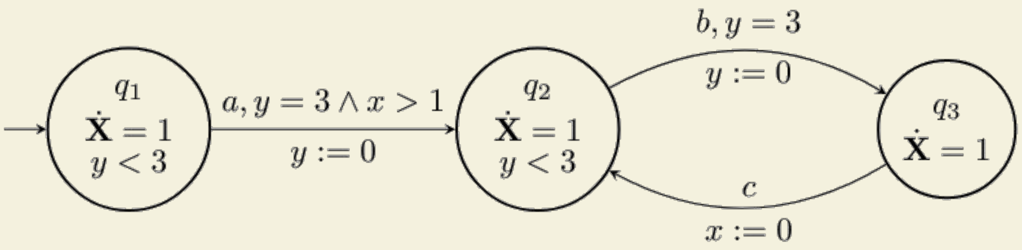
❌ *Almost non-zeno* condition required: upper bound on number of transitions within a given time period

To translate

Consider a timed automata with two clocks *x, y* such that the time bounds for *y* are [*t*-10, *t*], i.e., there can be upto a *10s* delay in the observation of *y*

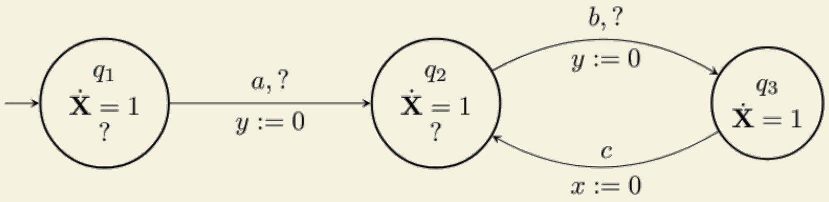
Almost non-zeno condition:

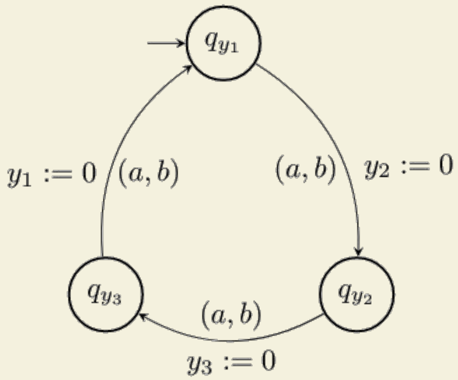
Suppose there can be *at most 2* transitions in a *10s* window



Translation: the new timed automaton

Introduce three new clocks y_1, y_2, and y_3 to track resets





Parallel composition
Transitions with same labels are synchronized

We track every reset of *y* in the last 10s

Translation: new predicates (both invariants and guards!)

$$y = 3$$

Was $y = 3$ sometime in the past 10s?
Well, we can compute every value of y in the last 10s!

$$\begin{aligned}
&(y_1 > y_2) \wedge (3 \in [\max(y_1 - 10, 0), y_1 - y_2]) \\
\bigvee &(y_2 > y_3) \wedge (3 \in [\max(y_2 - 10, 0), y_2 - y_3]) \\
\bigvee &(y_3 > y_1) \wedge (3 \in [\max(y_3 - 10, 0), y_3 - y_1])
\end{aligned}$$

Translations

Subclass of lazy hybrid automata	Restrictions	Translation
Lazy timed automata	‘Almost non-xeno’ condition	Regular timed automata
Lazy hybrid automata	‘Almost non-xeno’ condition and no invariants	Regular hybrid automata

To summarize

LHA > HA?

- Lazy hybrid automata allow us to **capture delays** in observation of variables which determine which discrete mode that we are present in
- Allowed to take transitions based on a state held in the past
- Tough to capture the new invariants in translations (except timed automata)

Next objectives

- Understanding the relationship between the continuous semantic and the discrete semantic of lazy HA presented in *Agrawal, M., Thiagarajan, P.: Lazy rectangular hybrid automat. In: 7th HSCC, LNCS 2993, Springer (2003) 1–15*
- Reachability analysis on lazy HA
- Delays within the dynamics - DDEs
- Proving that Lazy HA > HA