Recent work

Tsukuba Central
21st April 2022

Jérémy Dubut

What I can talk about now

- Safety analysis for probabilistic non-linear systems ➔
  - S. Pruekprasert, T. Takisaka, C. Eberhart, A. Cetinkaya, J. D. Moment Propagation of Discrete-Time Stochastic Polynomial Systems using Truncated Carleman Linearization. IFAC'20
  - S. Pruekprasert, J. D., T. Takisaka, C. Eberhart, A. Cetinkaya. Moment Propagation Through Carleman Linearization with Application to Probabilistic Safety Analysis. Under review (Automatica)
- Self-triggered control synthesis ➔
  - S. Pruekprasert, C. Eberhart, J. D. Symbolic Self-triggered Control of Continuous-time Non-deterministic Systems without Stability Assumptions for 2-LTL Specifications. ICARCV'20 (best paper finalist)
  - S. Pruekprasert, C. Eberhart, J. D. Fast Synthesis for Symbolic Self-triggered Control under Right-recursive LTL Specifications. CDC'21
- Monotony and optimality properties with KeYmaera X ➔
  - J. Kolčák, J. D., I. Hasuo, S. Katsumata, D. Sprunger, A. Yamada. Relational Differential Dynamic Logic. TACAS'20
- Decision making using game theory ➔
  - S. Pruekprasert, X. Zhang, J. D., C. Huang, M. Kishida. Decision Making for Autonomous Vehicles at Unsignalized Intersection in Presence of Malicious Vehicles. ITSC'19
  - S. Pruekprasert, J. D., X. Zhang, C. Huang, M. Kishida. A Game Theoretic Approach to Decision Making for Multiple Vehicles at Roundabout. Preprint

What I can NOT talk about YET

- Spatio-temporal logic to monitor network connectivity (need more time for slides)
- Some work with Mazda (under IP review)
- Category theory and algebraic topology (would need more time)
- Isabelle (I guess Akihisa already talked about it)
- Commonroad (not sure what I can talk about)

Safety analysis for probabilistic non-linear systems

Safety analysis

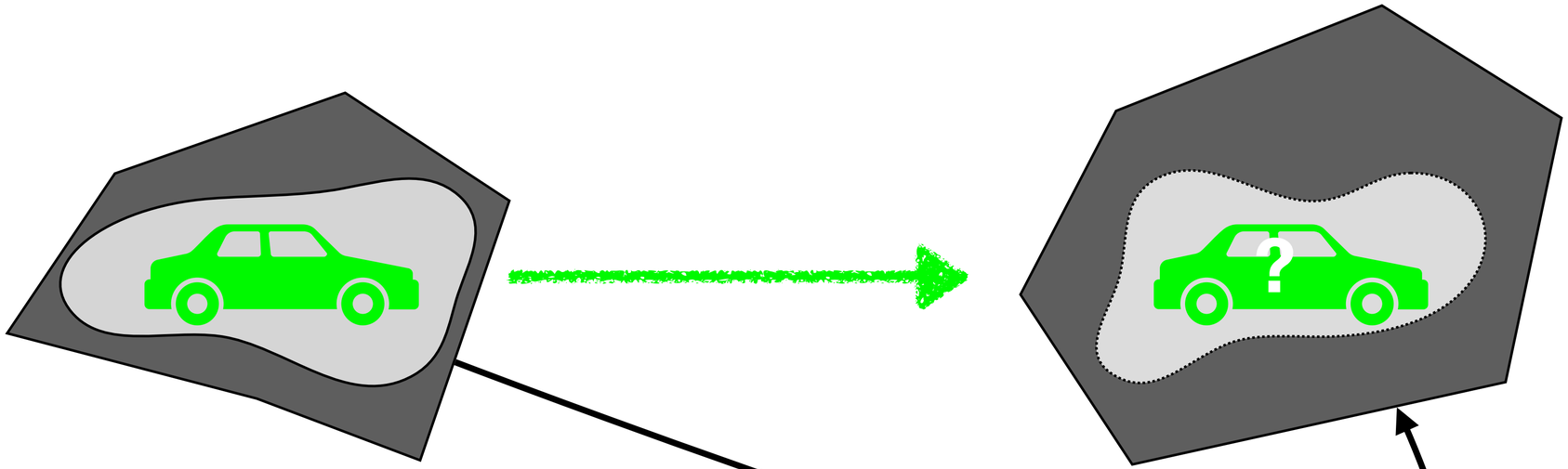
Goal: Compute a reachability zone for enforcing **safety**

Assume your car is in this zone
and its dynamics is given by
$\dot{X} = f(X)$

Compute a zone where the
car is guaranteed to be
in Δt time

Safety analysis

Goal: Compute a reachability zone for enforcing **safety**



Assume your car is in this zone and its dynamics is given by

$$\dot{X} = f(X)$$

Linearize the system

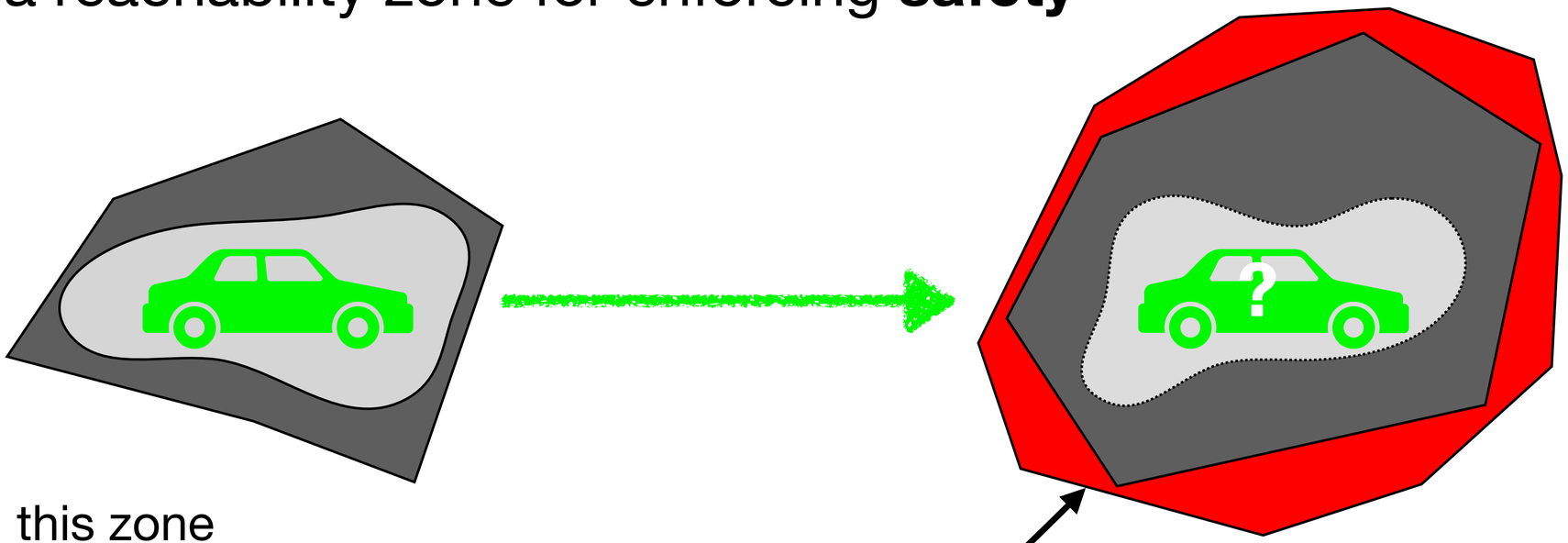
$$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t + \text{Errors}$$

Compute an over-approximation of the reachable zone

Safety analysis

Goal: Compute a reachability zone for enforcing **safety**

What about if the system has additive bounded noise?



Assume your car is in this zone and its dynamics is given by

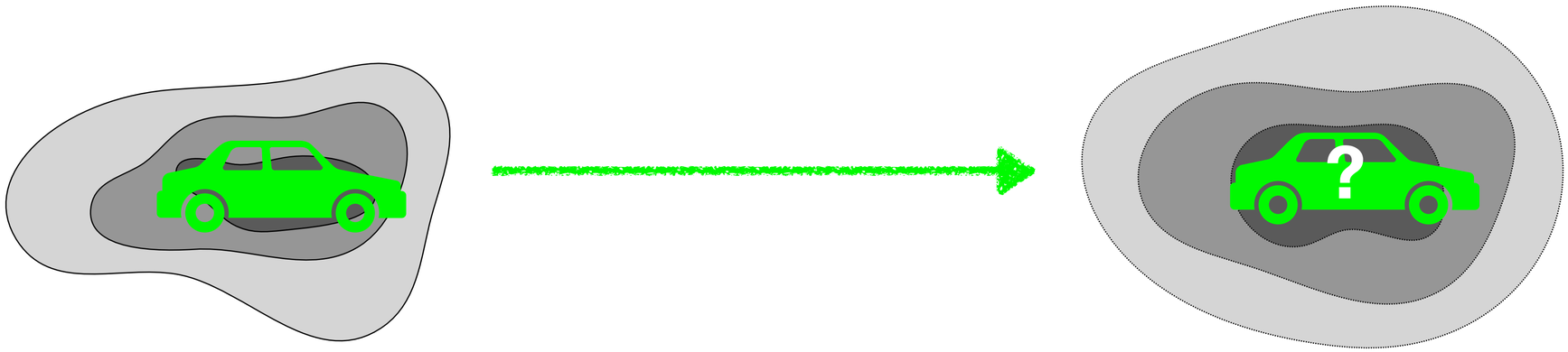
$$\dot{X} = f(X) + \text{Random}$$

Linearize the system

$$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t + \text{BiggerErrors}$$

Probabilistic safety analysis

Goal: Compute a reachability **probability** for enforcing safety



Assume the initial position follows this distribution (given by moments) and the dynamics is given by $\dot{X} = f(X)$

Estimate the distribution after Δt time (estimate the moments)

Example, a bicycle model

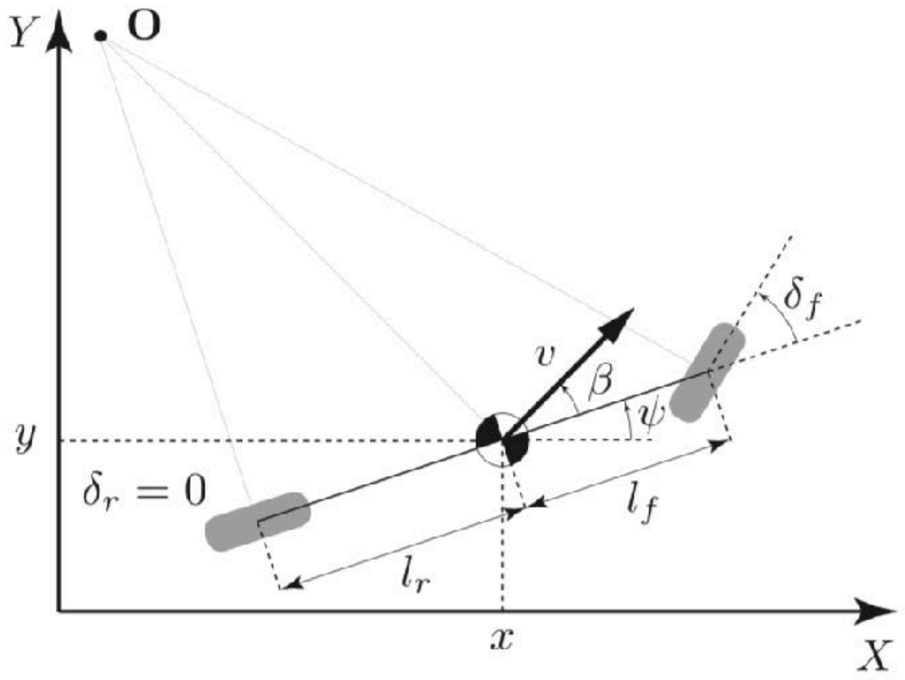


Fig. 1: Kinematic Bicycle Model

$$\dot{x}(t) = v(t) \cdot \cos(\psi(t) + \beta)$$

$$\dot{y}(t) = v(t) \cdot \sin(\psi(t) + \beta)$$

$$\dot{\psi}(t) = \frac{v(t)}{\ell} \cdot \sin\beta$$

$$\dot{v}(t) = a(t)$$

Kong et al., “Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design”, IEEE-IV 2015.

Example, a bicycle model

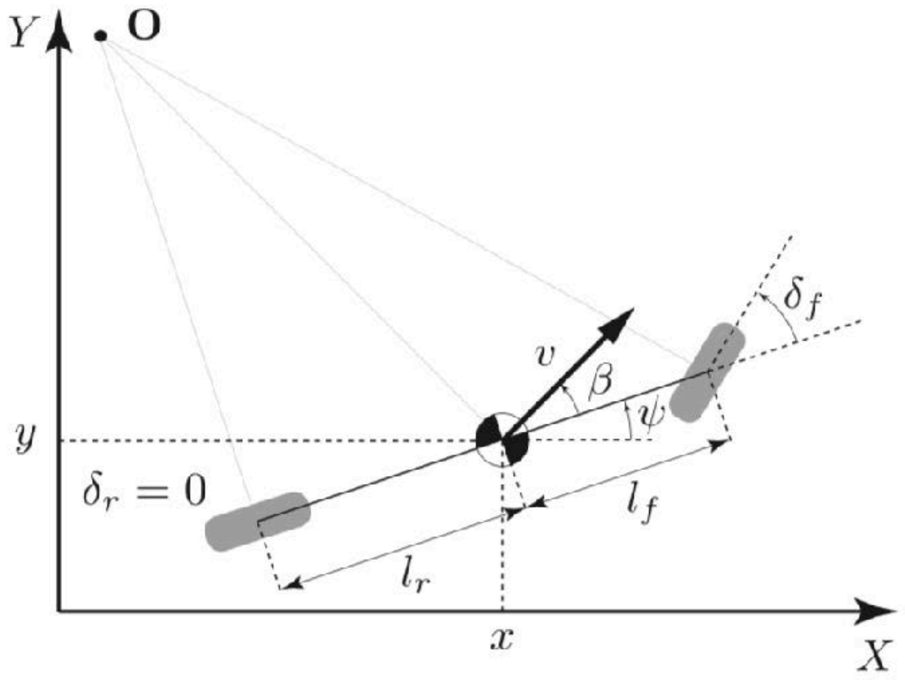


Fig. 1: Kinematic Bicycle Model

$$\dot{x}(t) = v(t) \cdot c(t)$$

$$\dot{y}(t) = v(t) \cdot s(t)$$

$$\dot{v}(t) = a(t)$$

$$\dot{c}(t) = -\frac{s(t) \cdot v(t) \cdot \sin\beta}{\ell}$$

$$\dot{s}(t) = \frac{c(t) \cdot v(t) \cdot \sin\beta}{\ell}$$

Example, a bicycle model

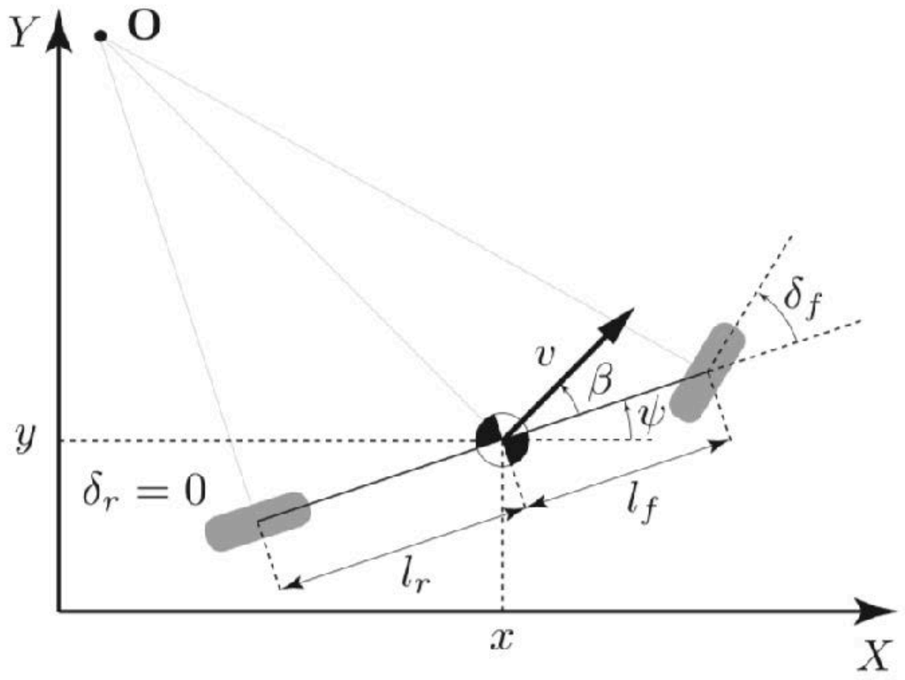


Fig. 1: Kinematic Bicycle Model

$$x(t+\Delta) = x(t) + \Delta c(t)v(t) + \frac{\Delta^2}{2}\left(a(t)c(t) - \frac{s(t)v^2(t)\sin\beta}{\ell}\right)$$

$$y(t+\Delta) = y(t) + \Delta s(t)v(t) + \frac{\Delta^2}{2}\left(a(t)s(t) + \frac{c(t)v^2(t)\sin\beta}{\ell}\right)$$

$$v(t+\Delta) = v(t) + \Delta a(t)$$

$$c(t+\Delta) = c(t) - \Delta\frac{s(t)v(t)\sin\beta}{\ell} - \frac{\Delta^2}{2}\left(\frac{c(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)s(t)\sin\beta}{\ell}\right)$$

$$s(t+\Delta) = s(t) + \Delta\frac{c(t)v(t)\sin\beta}{\ell} + \frac{\Delta^2}{2}\left(-\frac{s(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)c(t)\sin\beta}{\ell}\right)$$

Discrete time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Discrete time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

$$= \begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ c(t) \\ s(t) \end{bmatrix}$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Discrete time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

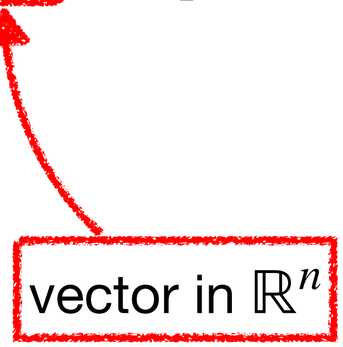
Discrete time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$



Discrete time polynomial system

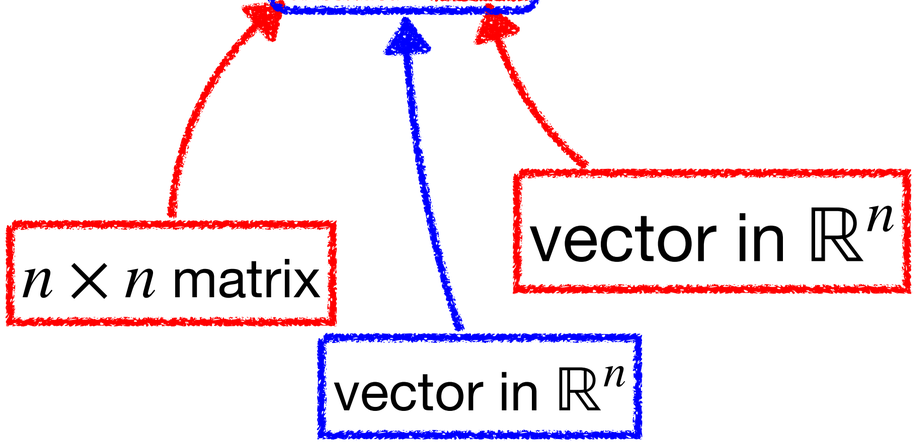
States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$

$$x(t+1) = F_0(t) + F_1(t)\, x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$



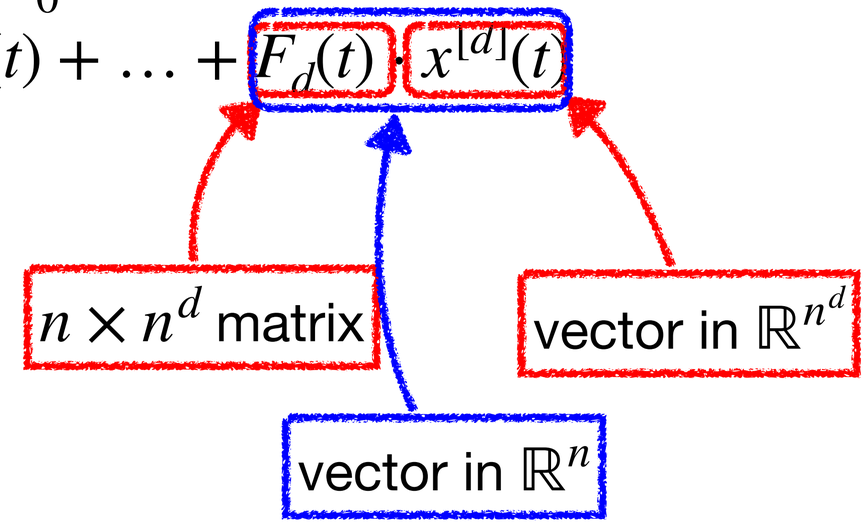
Discrete time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{\lfloor d \rfloor}(t)$$



d-th Kronecker power of $x(t)$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{[2]} = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

Discrete time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Usual assumption: the $F_i(t)$s do not depend on t nor on x_0

Example, a bicycle model

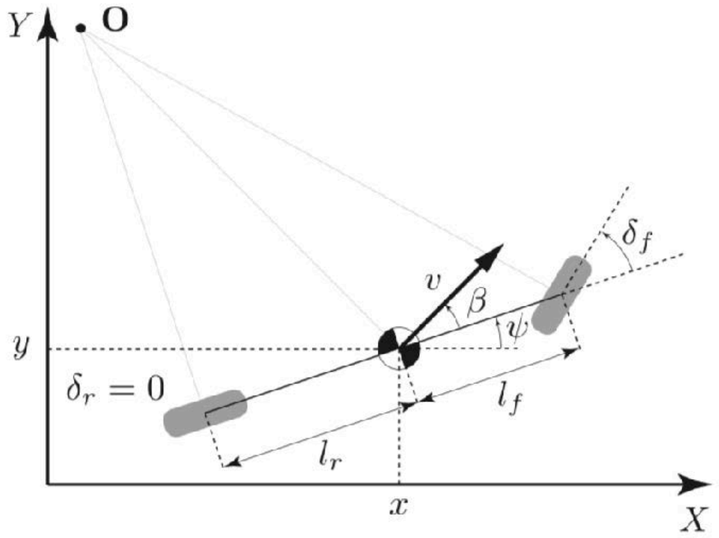


Fig. 1: Kinematic Bicycle Model

$$
\begin{aligned}
x(t+\Delta) &= x(t) + \Delta c(t) v(t) + \frac{\Delta^2}{2}\left(a(t)c(t) - \frac{s(t)v^2(t)\sin\beta}{\ell}\right) \\
y(t+\Delta) &= y(t) + \Delta s(t) v(t) + \frac{\Delta^2}{2}\left(a(t)s(t) + \frac{c(t)v^2(t)\sin\beta}{\ell}\right) \\
v(t+\Delta) &= v(t) + \Delta a(t) \\
c(t+\Delta) &= c(t) - \Delta\frac{s(t)v(t)\sin\beta}{\ell} - \frac{\Delta^2}{2}\left(\frac{c(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)s(t)\sin\beta}{\ell}\right) \\
s(t+\Delta) &= s(t) + \Delta\frac{c(t)v(t)\sin\beta}{\ell} + \frac{\Delta^2}{2}\left(-\frac{s(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)c(t)\sin\beta}{\ell}\right)
\end{aligned}
$$

$$
F_0(t) = \begin{bmatrix} 0 \\ 0 \\ \Delta a(t) \\ 0 \\ 0 \end{bmatrix}
$$

$$
F_1(t) = \begin{bmatrix}
1 & 0 & 0 & 0 & \frac{\Delta^2 a(t)}{2} & 0 \\
0 & 1 & 0 & 0 & 0 & \frac{\Delta^2 a(t)}{2} \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -\frac{\Delta^2 a(t)\sin\beta}{2\ell} \\
0 & 0 & 0 & 0 & -\frac{\Delta^2 a(t)\sin\beta}{2\ell} & 1
\end{bmatrix}
$$

Discrete time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Discrete time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Random vector

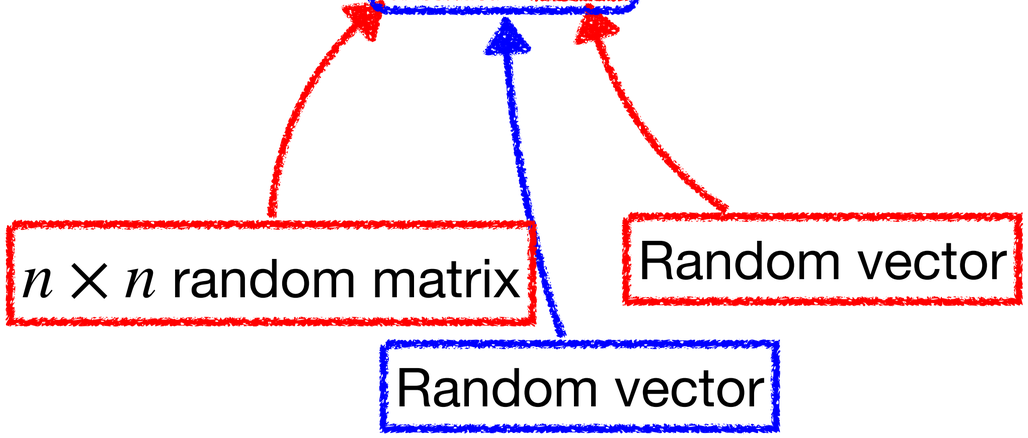
Discrete time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t)\, x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$



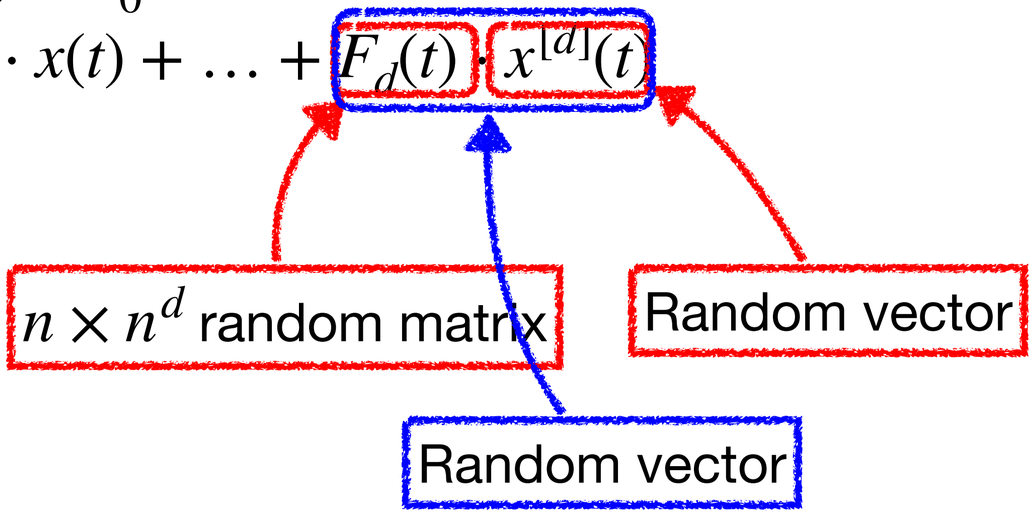
Discrete time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{\lfloor d \rfloor}(t)$$



Discrete time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Usual assumption: the $F_i(t)$s do not depend on t

($F_i(t)$ and $F_j(s)$ are independent for $t \neq s$)

($F_i(t)$ and $F_i(s)$ are identically distributed)

nor on x_0

(x_0 and $F_i(t)$ are independent)

Carleman linearisation

Main idea: transform a finite-dimensional polynomial system into a infinite-dimensional linear system

$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Computing the Kronecker products:

$$x^{[j]}(t+1) = \sum_{k=0}^{jd} A_{j,k}(t) \cdot x^{[k]}(t)$$

where $A_{j,k}(t)$ is computed from the $F_i(t)$. Using:

$$y(t) = [1 \quad x(t) \quad x^{[2]}(t) \quad \ldots \quad]$$

we obtain the following infinite-dimensional linear system:

$$y(t+1) = A(t) \cdot y(t)$$

where $A(t)$ is computed from the $F_i(t)$.

Moment equations

The j-th moments of $x(t)$ are given by $\mathbb{E}(x^{[j]}(t))$.

Equations between the moments:

$$\mathbb{E}(x^{[j]}(t+1)) = \sum_{k=0}^{jd} \mathbb{E}(A_{j,k}(t) \cdot x^{[k]}(t))$$

using the assumptions:

$$\mathbb{E}(x^{[j]}(t+1)) = \sum_{k=0}^{jd} \mathbb{E}(A_{j,k}(t)) \cdot \mathbb{E}(x^{[k]}(t))$$

and $\mathbb{E}(A_{j,k}(t))$ is independent of t. We then obtain:

$$\mathbb{E}(y(t+1)) = E \cdot \mathbb{E}(y(t))$$

where E is computed from the moments of the $F_i(t)$.

Truncated system

M. Forest, A. Pouly, “Explicit error bounds for Carleman linearization”, arXiv:1711.02552, 2017.

Fix N, and define E_N the restriction of E to the $\sum_{k=0}^{N} n^k$ raws and columns.

Our estimation of the N-th first moments of $x(t)$ is given by the following system:

$$\widetilde{y}(0) = [1 \quad \mathbb{E}(x_0) \quad \dots \quad \mathbb{E}(x_0^{[N]})]$$
$$\widetilde{y}(t+1) = E_N \cdot \widetilde{y}(t)$$

That is, $\widetilde{y}(t)$ is an approximation of $[1 \quad \mathbb{E}(x(t)) \quad \dots \quad \mathbb{E}(x^{[N]}(t))]$.

Furthermore, we have efficient ways of computing bounds of the errors.

Computation tasks

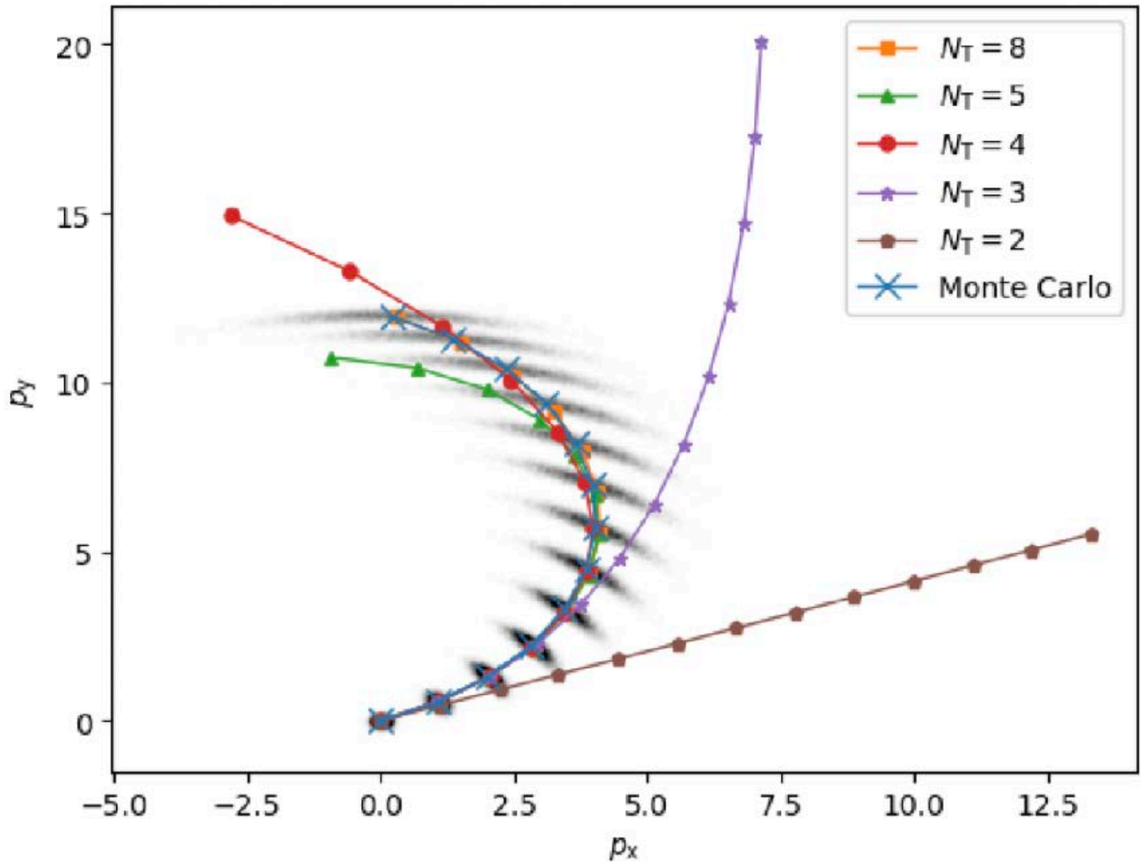
- Compute E_N for a fixed N
  - Kronecker product algebra + computing moments
  - Very expensive
  - **OFFLINE**

- Compute the approximations $\tilde{y}(1), \ldots, \tilde{y}(h)$
  - Matrix multiplications
  - Very cheap (even cheaper if done on a GPU?)
  - **ONLINE**

- Computing a bound on the error
  - Basic computations
  - Quite cheap
  - **ONLINE**

Our method vs sampling

Method	Monte Carlo		Moment propagation			
Parameters	num. samples		N_T			
	10	10^4	4	16	64	256
Time (μs)	2.9e10^3	3.4e10^6	11	14	30	93



Tail probability analysis

Proposition:

If $||\,.\,||$ is the Euclidian norm:

$$P(||x(t)-\tilde{x}_1(t)|| \geq \alpha) \leq \frac{\sum_{i=1}^{n} (\tilde{x}_2(t))_{i,i} + \epsilon_{i,i} - \max\{0,(|(\tilde{x}_1(t))_i| - \epsilon_i)^2\}}{(\alpha - \epsilon)^2}$$

Proof:

Chebyshev's inequality

Meaning: we can compute α depending on our approximations and error bounds such that the Euclidian ball centred at $\tilde{x}_1(t)$ with radius α contains $x(t)$ with a given probability

Tail probability analysis

Proposition:

If $||\,.\,||$ is the Euclidian norm:

$$P(||x(t)-\tilde{x}_1(t)|| \geq \alpha) \leq \frac{\sum_{i=1}^{n} (\tilde{x}_2(t))_{i,i} + \epsilon_{i,i} - \max\{0,(|(\tilde{x}_1(t))_i| - \epsilon_i)^2\}}{(\alpha - \epsilon)^2}$$

Proof:
Chebyshev's inequality

Meaning: we can compute α depending on our approximations and error bounds such that the Euclidian ball centred at $\tilde{x}_1(t)$ with radius α contains $x(t)$ with a given probability

Tail probability analysis

Proposition:

If $||\,.\,||$ is the Euclidian norm:

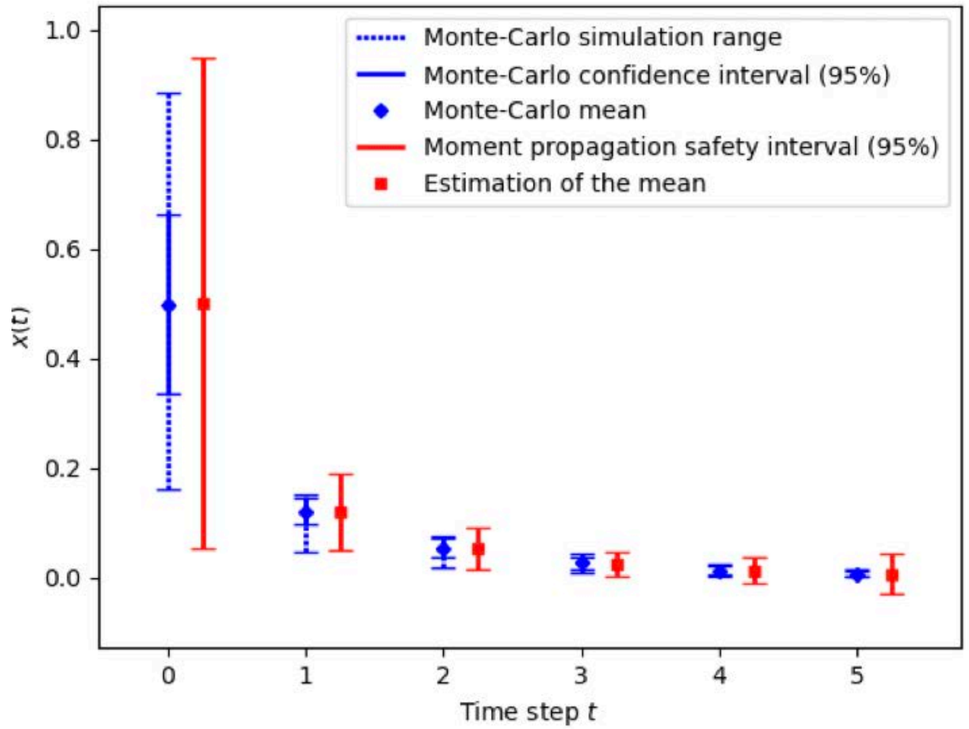
$$P(||x(t)-\tilde{x}_1(t)|| \geq \alpha) \leq \frac{\sum_{i=1}^{n} (\tilde{x}_2(t))_{i,i} + \epsilon_{i,i} - \max\{0,(|(\tilde{x}_1(t))_i| - \epsilon_i)^2\}}{(\alpha - \epsilon)^2}$$

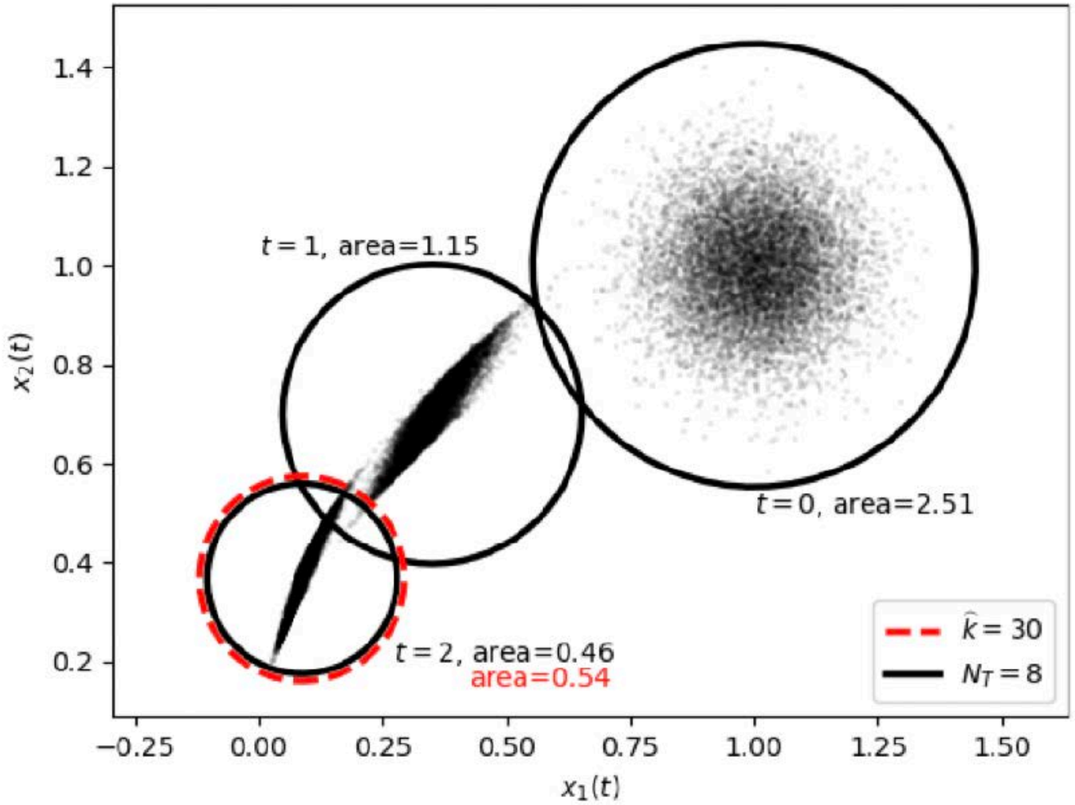
Proof:

Chebyshev's inequality

Meaning: we can compute α depending on our approximations and error bounds such that the Euclidian ball centred at $\tilde{x}_1(t)$ with radius α contains $x(t)$ with a given probability

Experimental results





Better tail probability analysis

Proposition:

If $||x||_Q$ is the norm $QxQ^\top$ with Q symmetric definite positive:

$$P(||x(t)-\tilde{x}_1(t)||_Q \geq \alpha) \leq \frac{\sum_{i,j=1}^{n} Q_{i,j}((\tilde{x}_2(t))_{i,j} - (\tilde{x}_1(t))_i(\tilde{x}_1(t))_j)}{(\alpha - \epsilon_Q)^2}$$

when the truncation is big enough.

Proof:

Chebyshev's inequality

ϵ_Q can be bounded using our error bounds and the Eigen values of Q

Meaning: we can compute Q depending on our approximations and error bounds such that the an ellipse centred at $\tilde{x}_1(t)$ with axises depending on α and Q contains $x(t)$ with a given probability

Better tail probability analysis

Proposition:

If $||x||_Q$ is the norm $QxQ^\top$ with Q symmetric definite positive:

$$P(||x(t)-\tilde{x}_1(t)||_Q \geq \alpha) \leq \frac{\sum_{i,j=1}^{n} Q_{i,j}((\tilde{x}_2(t))_{i,j} - (\tilde{x}_1(t))_i(\tilde{x}_1(t))_j)}{(\alpha - \epsilon_Q)^2}$$

when the truncation is big enough.

Proof:
Chebyshev's inequality

ϵ_Q can be bounded using our error bounds and the Eigen values of Q

Meaning: we can compute Q depending on our approximations and error bounds such that the an ellipse centred at $\tilde{x}_1(t)$ with axises depending on α and Q contains $x(t)$ with a given probability

Better tail probability analysis

Proposition:

If $||x||_Q$ is the norm $QxQ^\top$ with Q symmetric definite positive:

$$P(||x(t)-\tilde{x}_1(t)||_Q \geq \alpha) \leq \frac{\sum_{i,j=1}^{n} Q_{i,j}((\tilde{x}_2(t))_{i,j} - (\tilde{x}_1(t))_i(\tilde{x}_1(t))_j)}{(\alpha - \epsilon_Q)^2}$$

when the truncation is big enough.

Proof:

Chebyshev's inequality

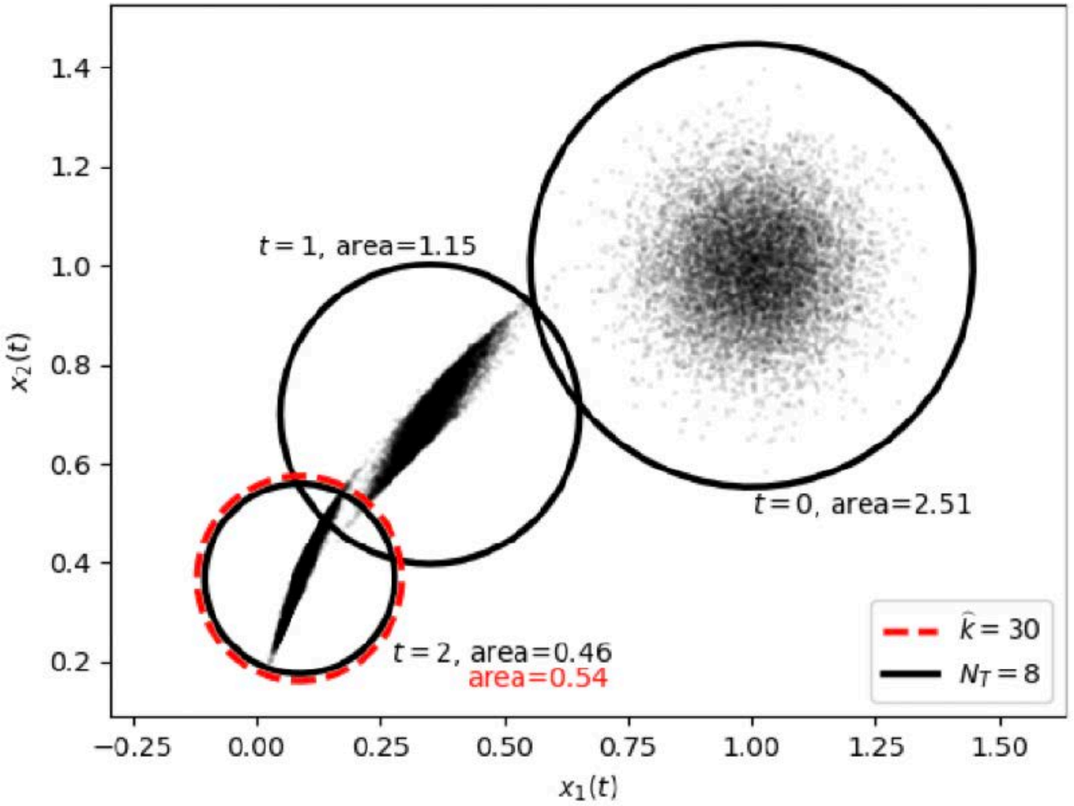
ϵ_Q can be bounded using our error bounds and the Eigen values of Q

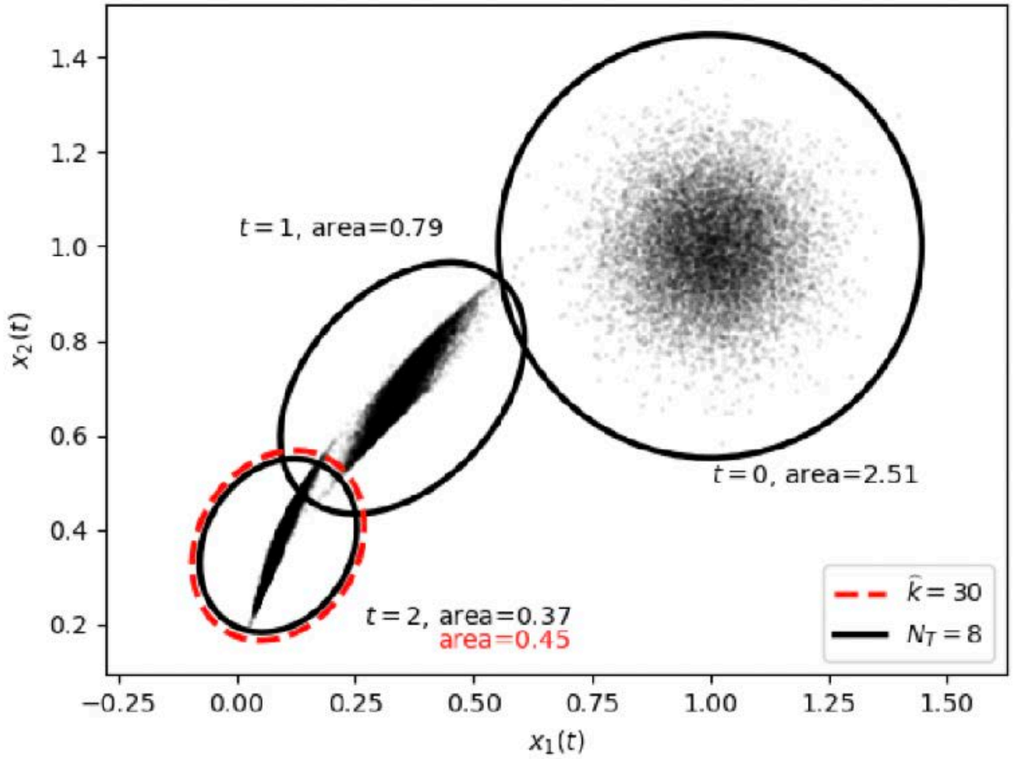
Meaning: we can compute Q depending on our approximations and error bounds such that the an ellipse centred at $\tilde{x}_1(t)$ with axises depending on α and Q contains $x(t)$ with a given probability

How to compute Q?

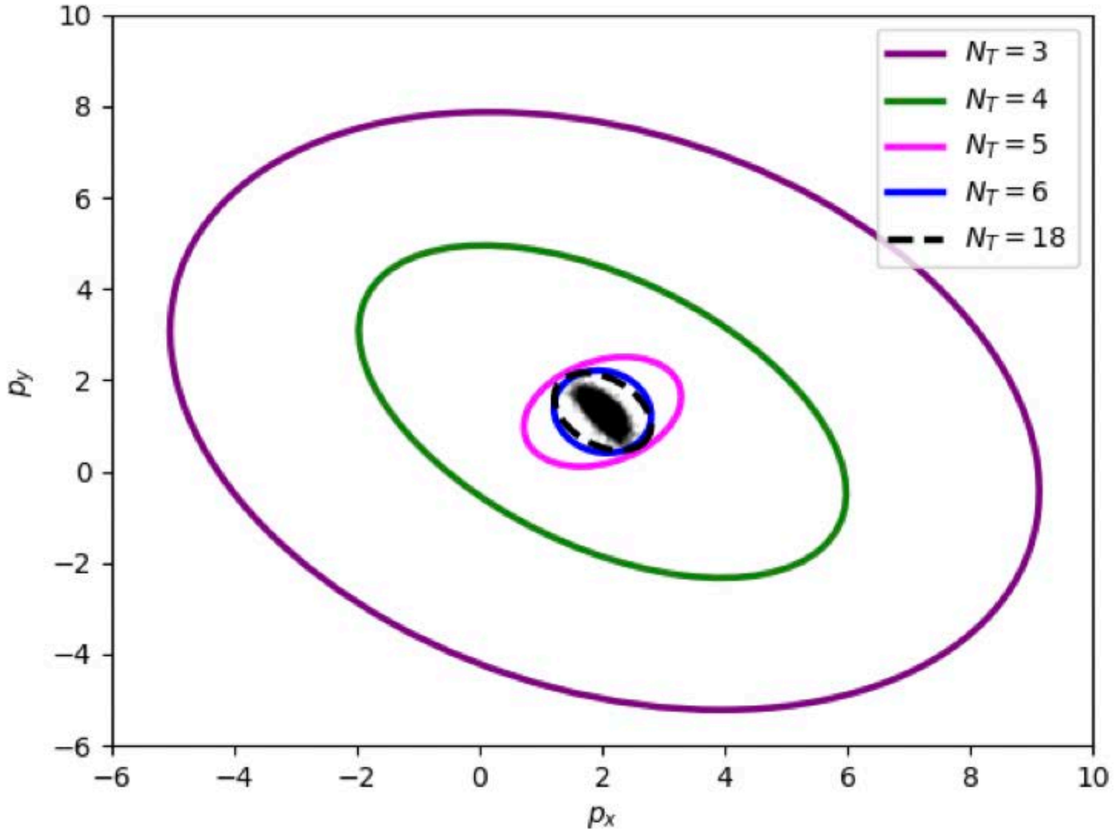
- Solve the following convex optimisation problem:
  minimise $\det(Q^{-1})$ under
  Q is definite-positive
  $$\frac{\sum_{i,j=1}^{n} Q_{i,j}((\tilde{x}_2(t))_{i,j} - (\tilde{x}_1(t))_i(\tilde{x}_1(t))_j)}{\alpha^2} \leq p$$
- Compute ϵ_Q
- Then $x(t)$ is guaranteed to be in the ellipsoid
  $$\{x \mid ||x - \tilde{x}_1(t)||_Q \leq \alpha + \epsilon_Q\}$$
  With probability at least $1 - p$.

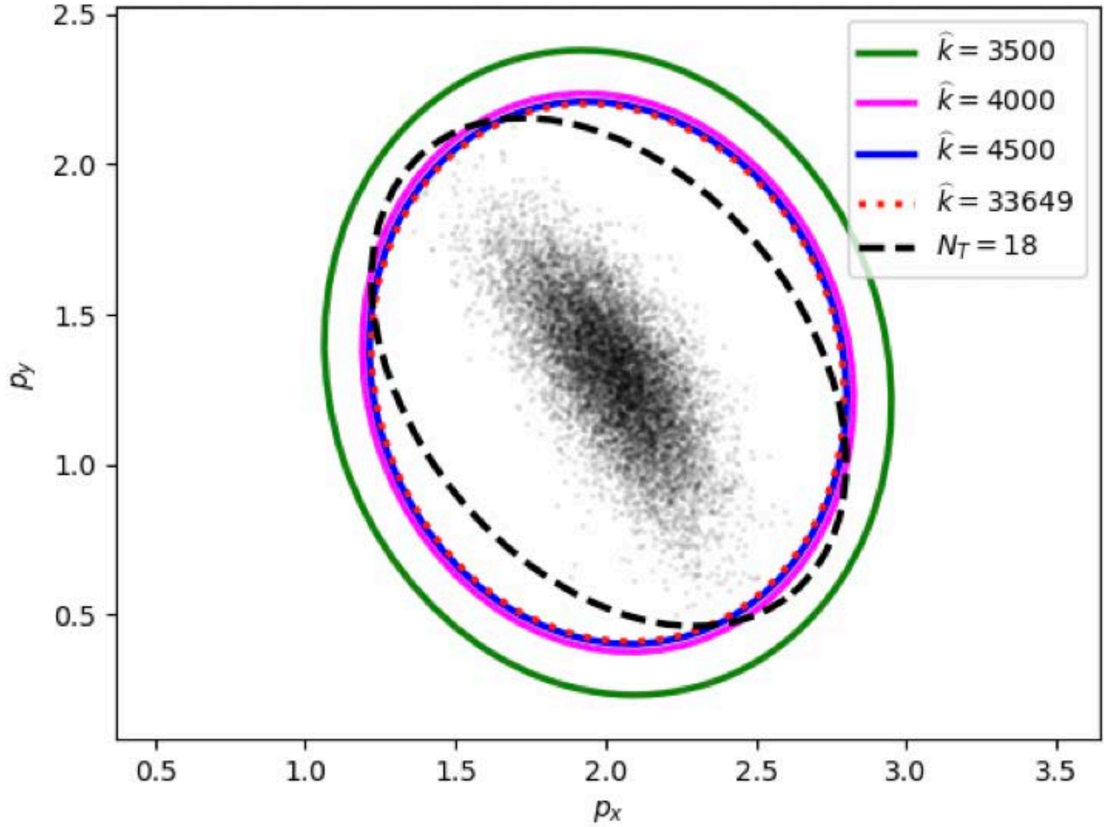
Experiment results





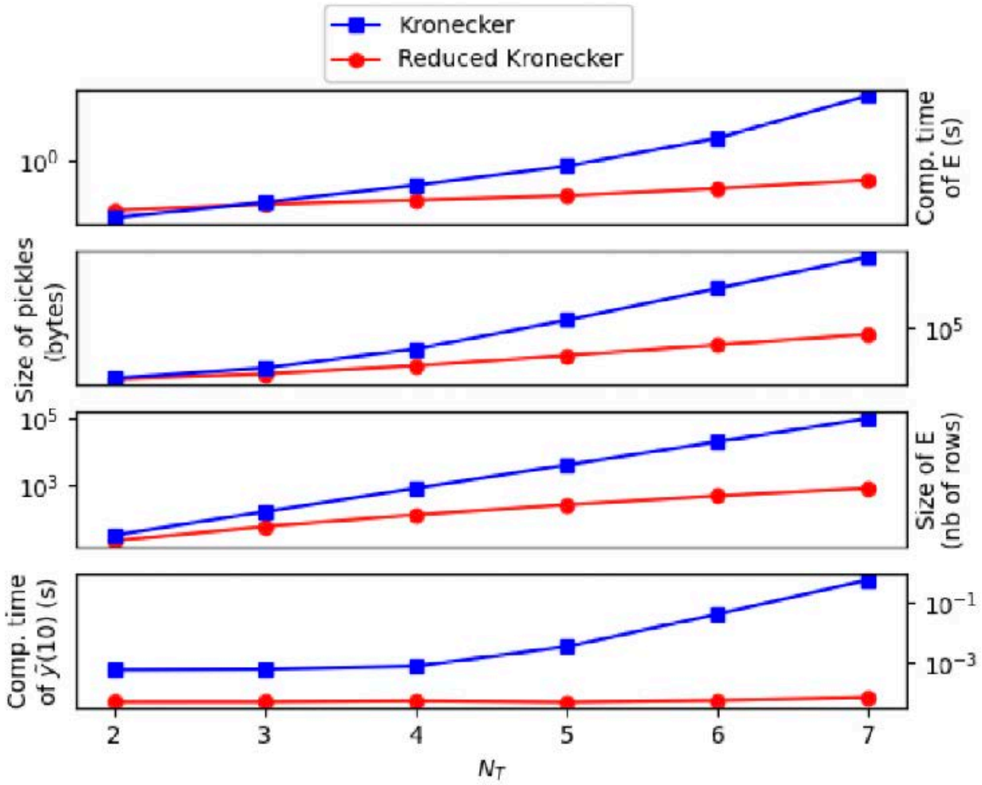
Experiment results





Optimisations

- Bottleneck: offline computation is slow and too much memory consuming
  - Kronecker algebra is bad (a lot of duplications, slow libraries, …)
- Optimisation 1: replace Kronecker algebra with something more optimised
  - Basically based on multivariate polynomials
- Optimisation 2: a lot of work can be parallelised



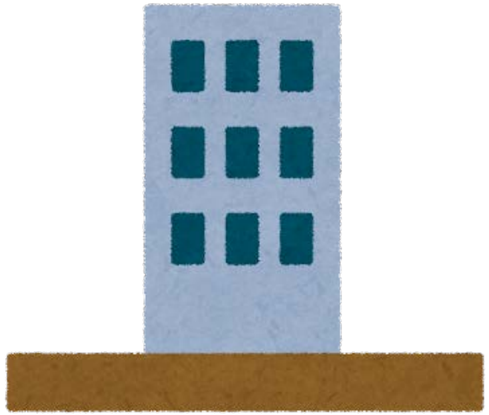
Self-triggered control

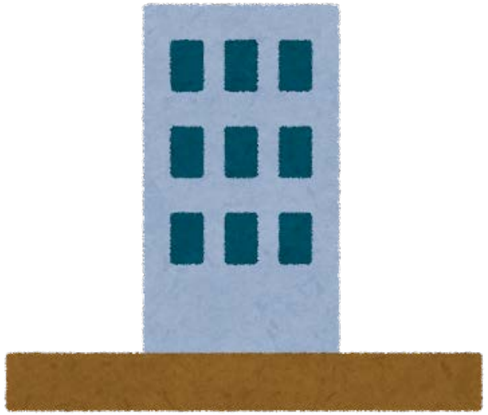
References

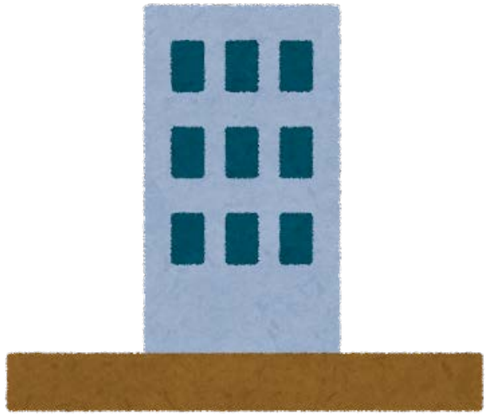
- S. Pruekprasert, C. Eberhart, J. D.
  Symbolic Self-triggered Control of Continuous-time Non-deterministic Systems without Stability Assumptions for 2-LTL Specifications.
  ICARCV'20 (best paper finalist)

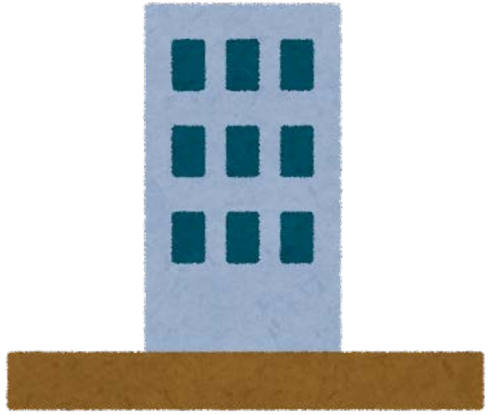
- S. Pruekprasert, C. Eberhart, J. D.
  Fast Synthesis for Symbolic Self-triggered Control under Right-recursive LTL Specifications.
  CDC'21

Usual control loop

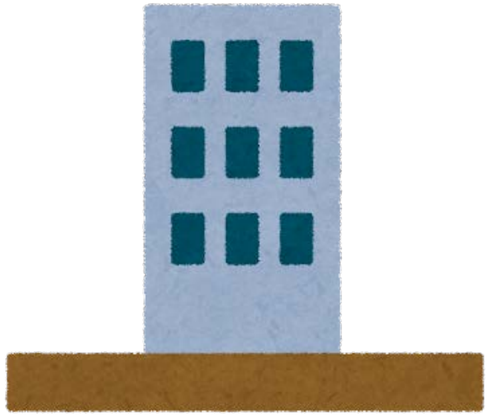


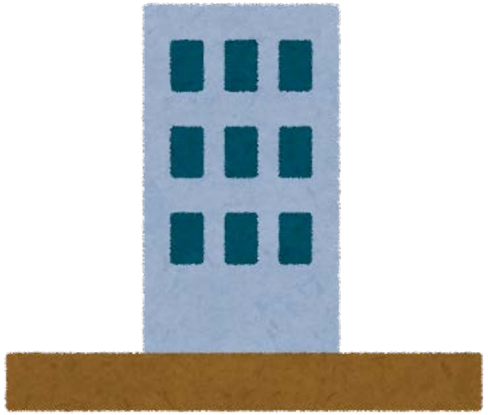


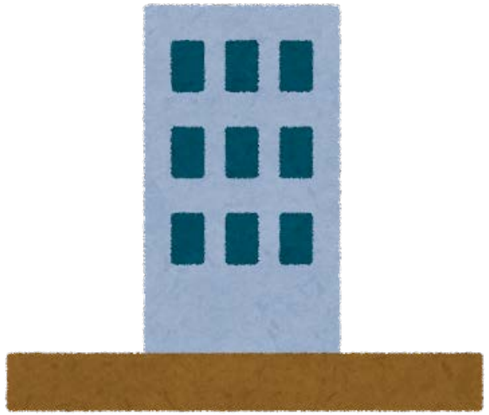


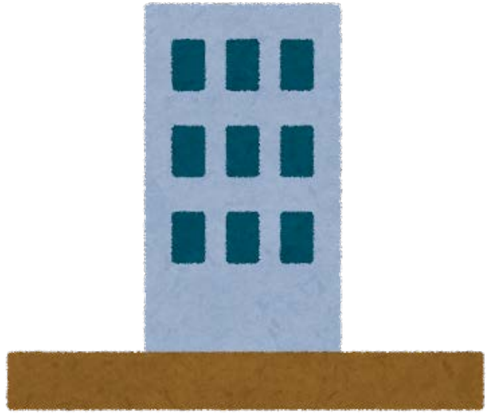


Usual control loop



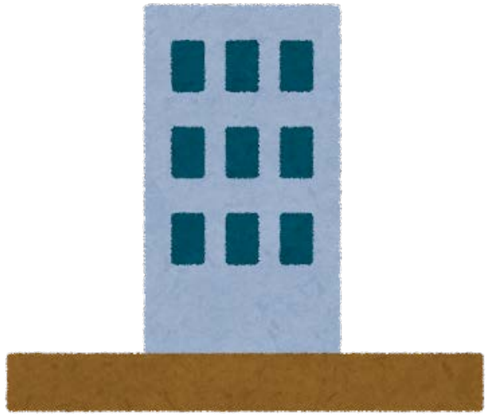


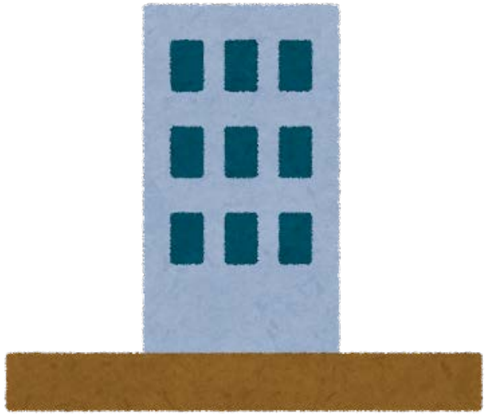


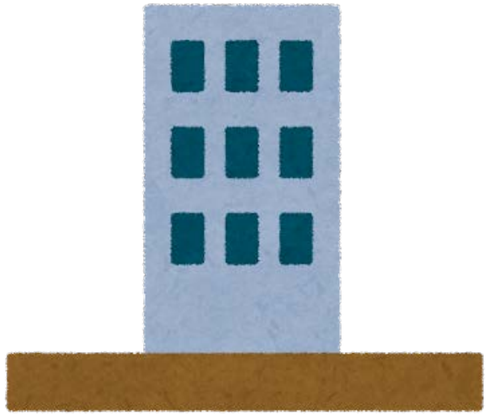


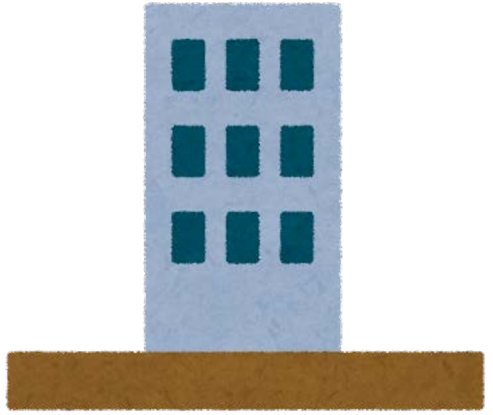
Usual control loop

Control input:
- **Acceleration**
- **Angles**
- **…**

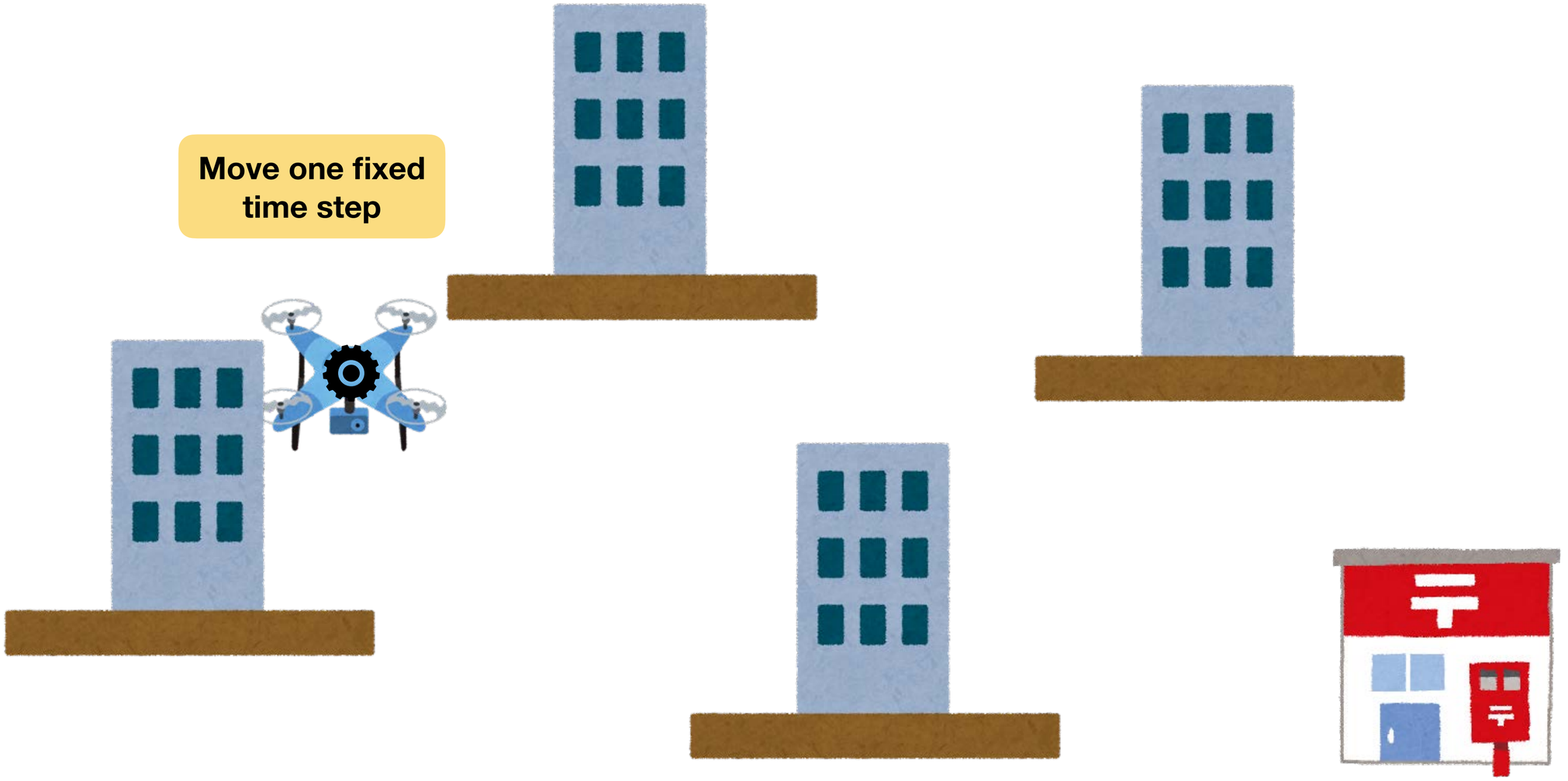




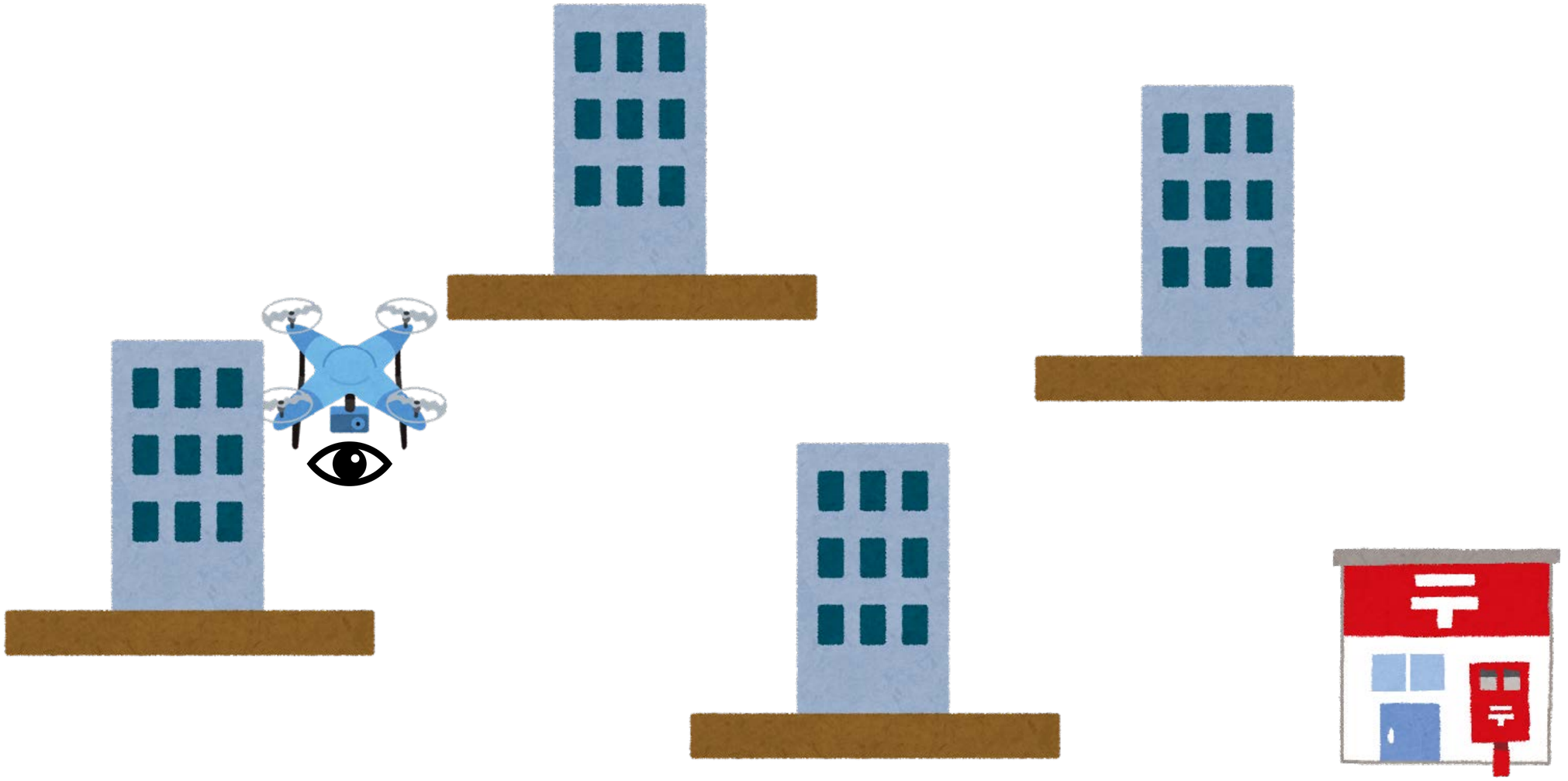




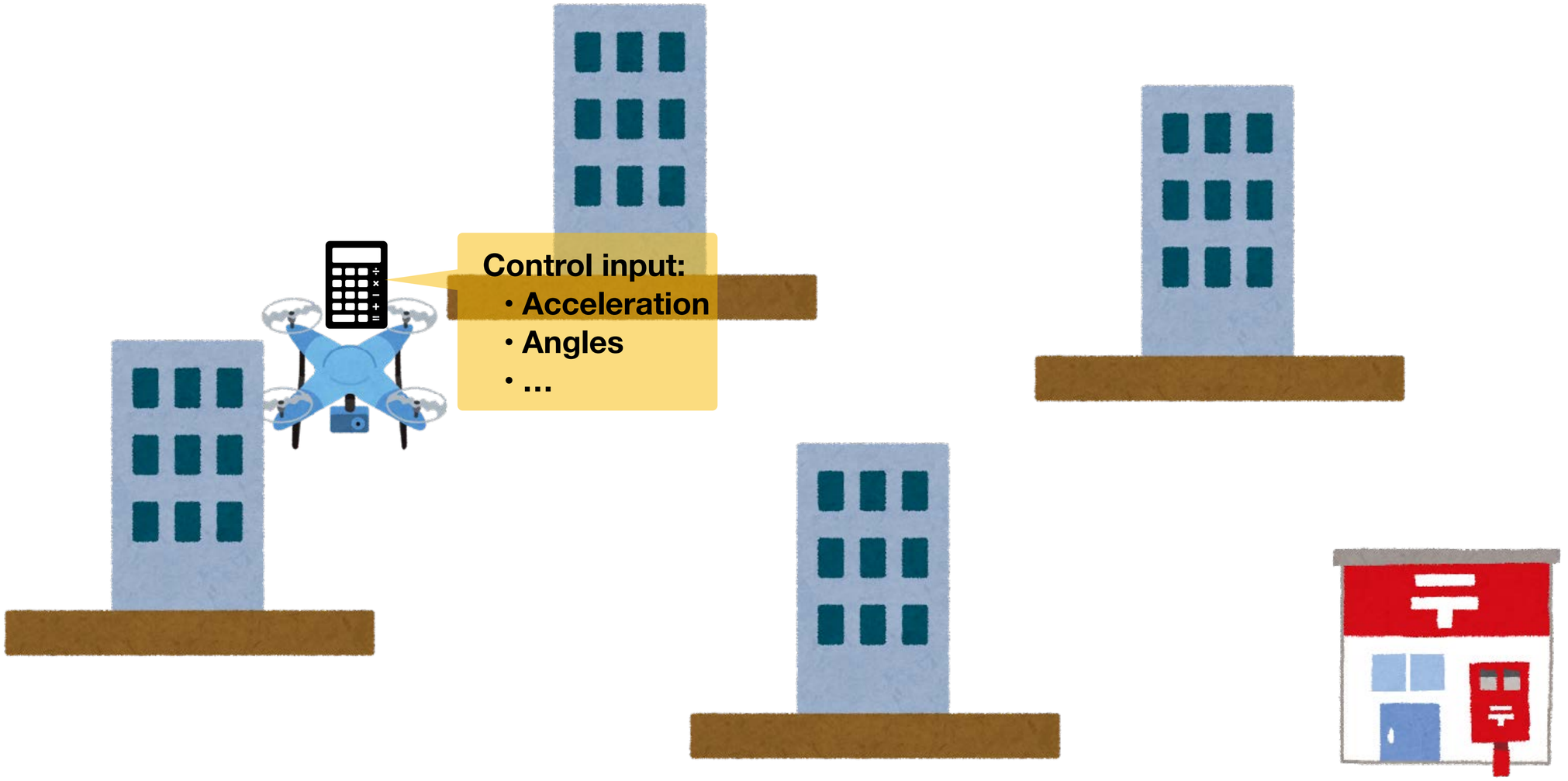
Usual control loop



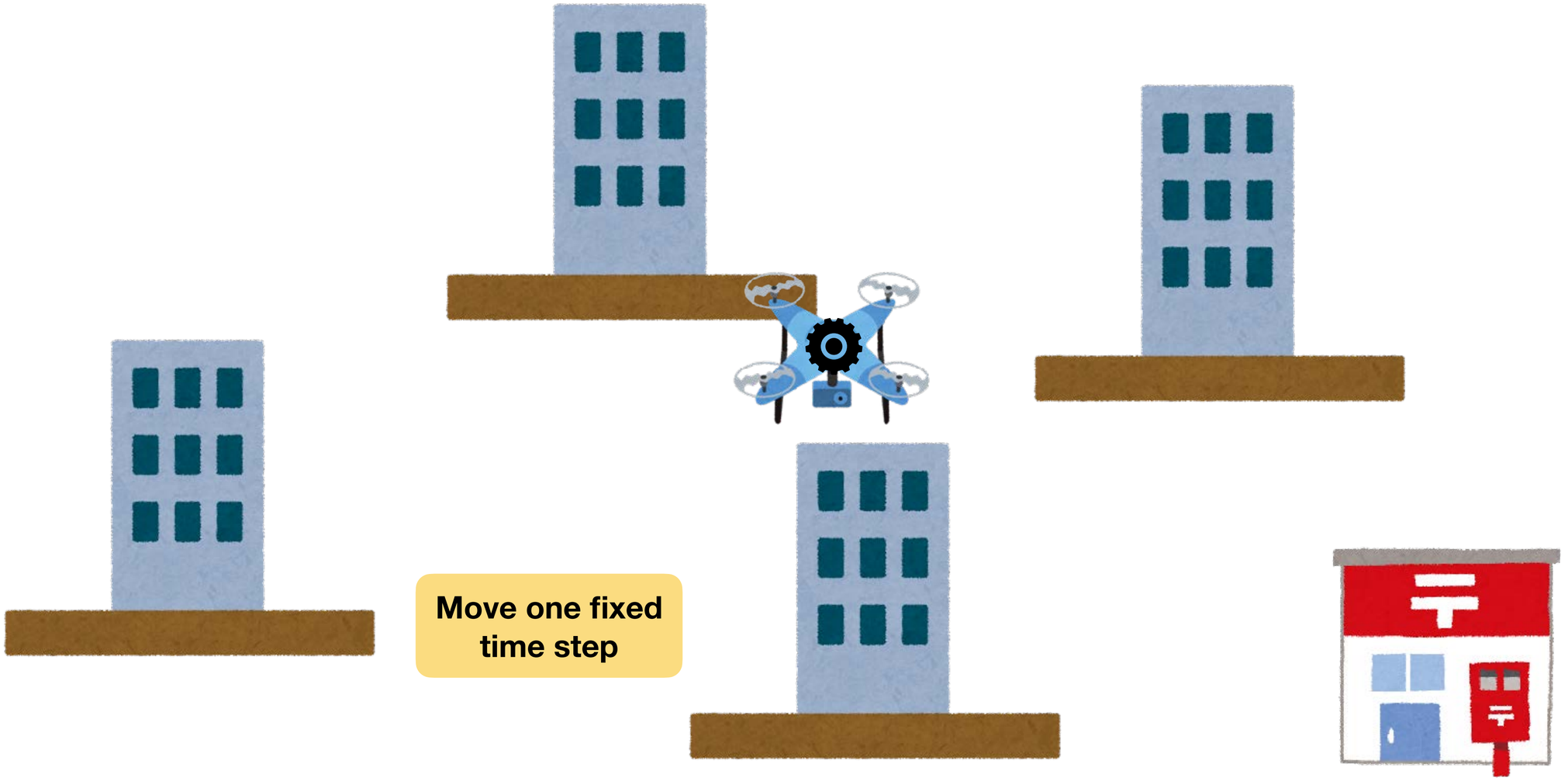
Usual control loop



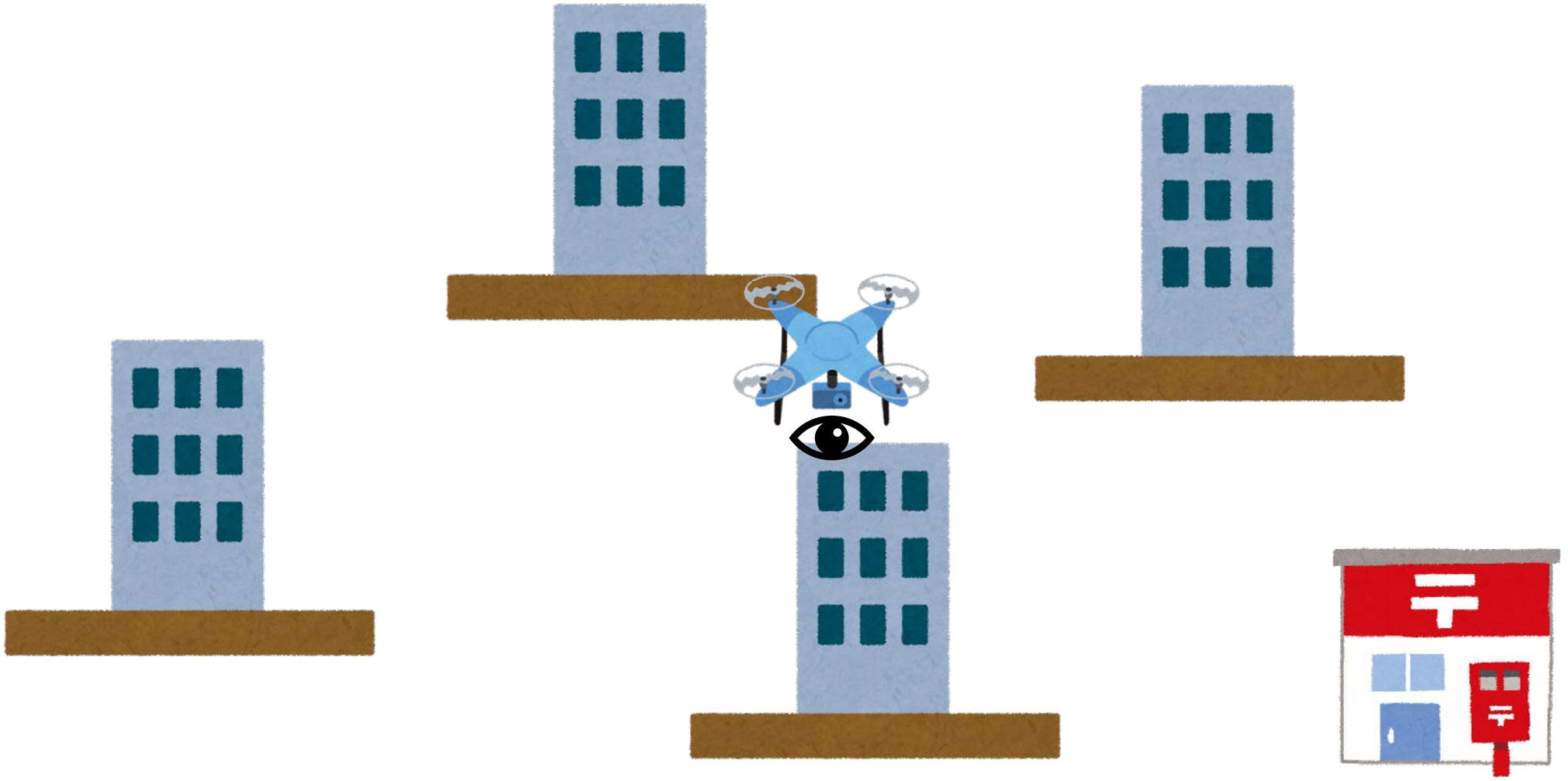
Usual control loop



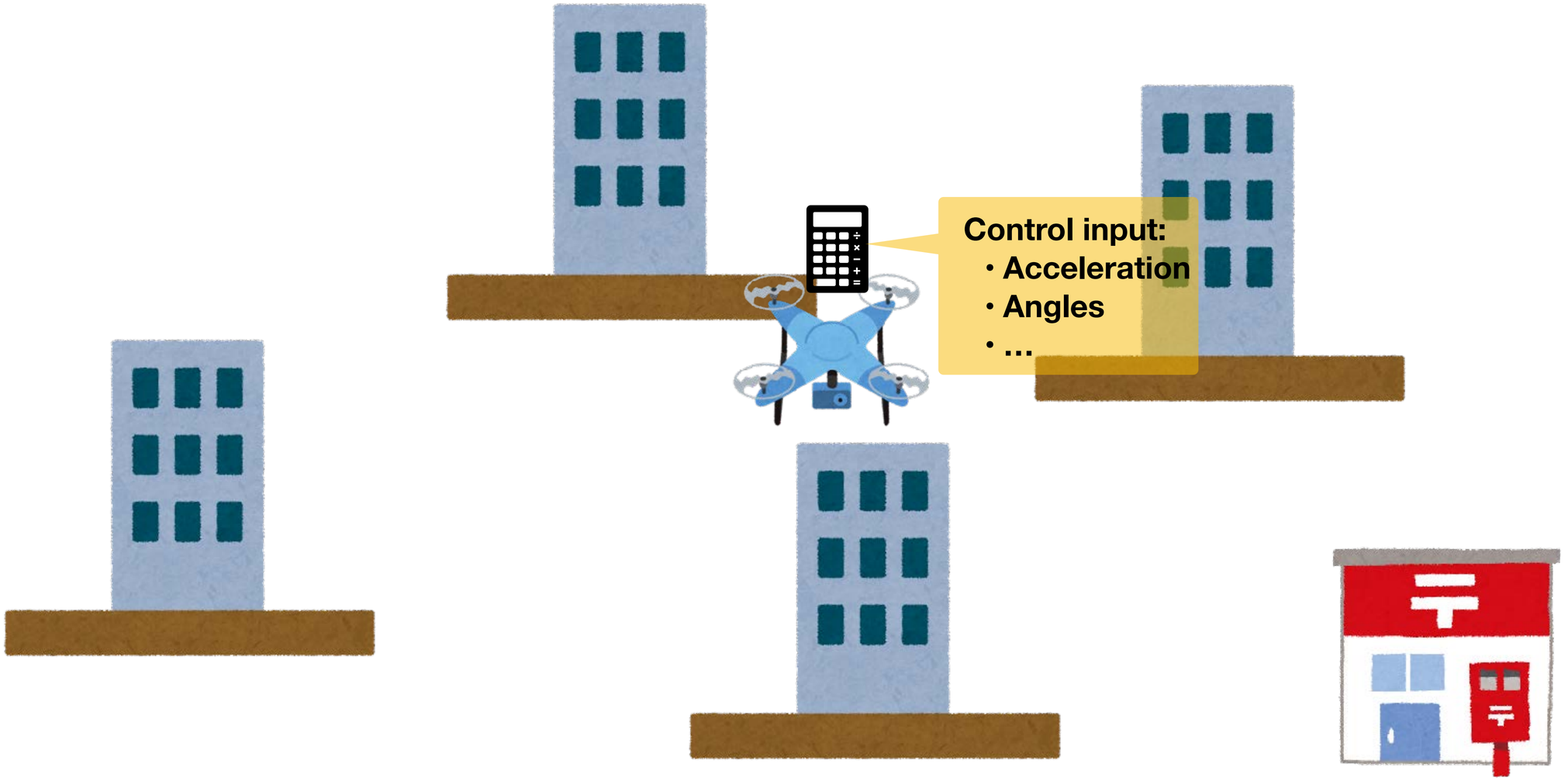
Usual control loop



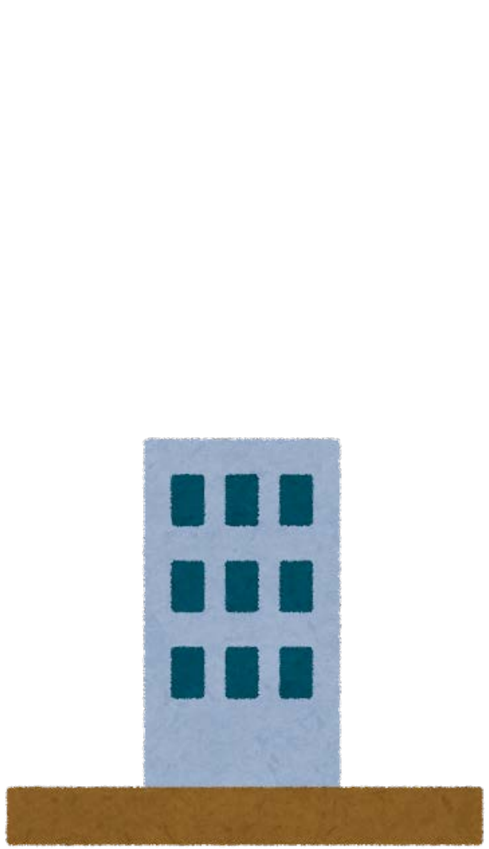
Usual control loop

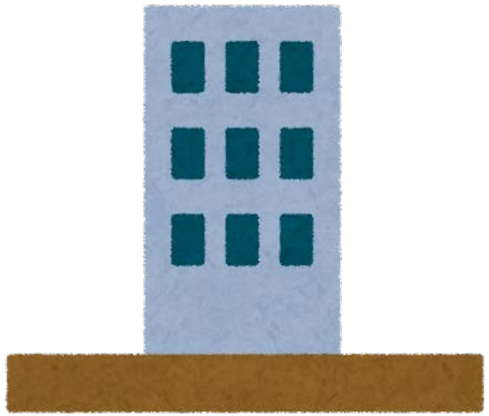


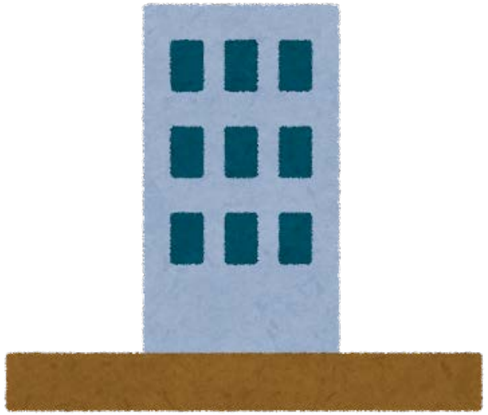
Usual control loop

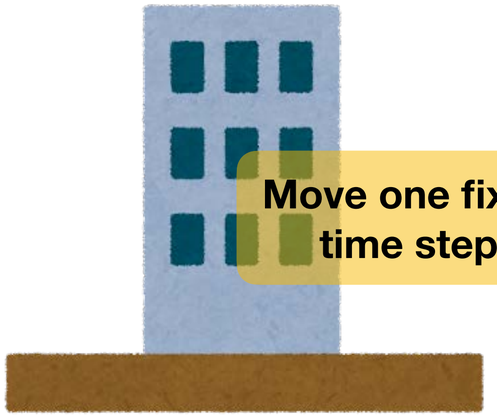


Usual control loop

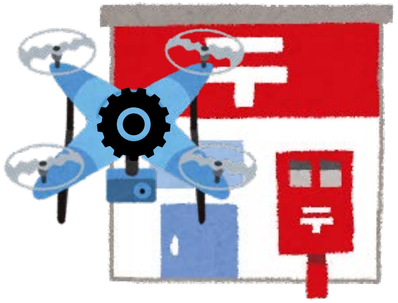




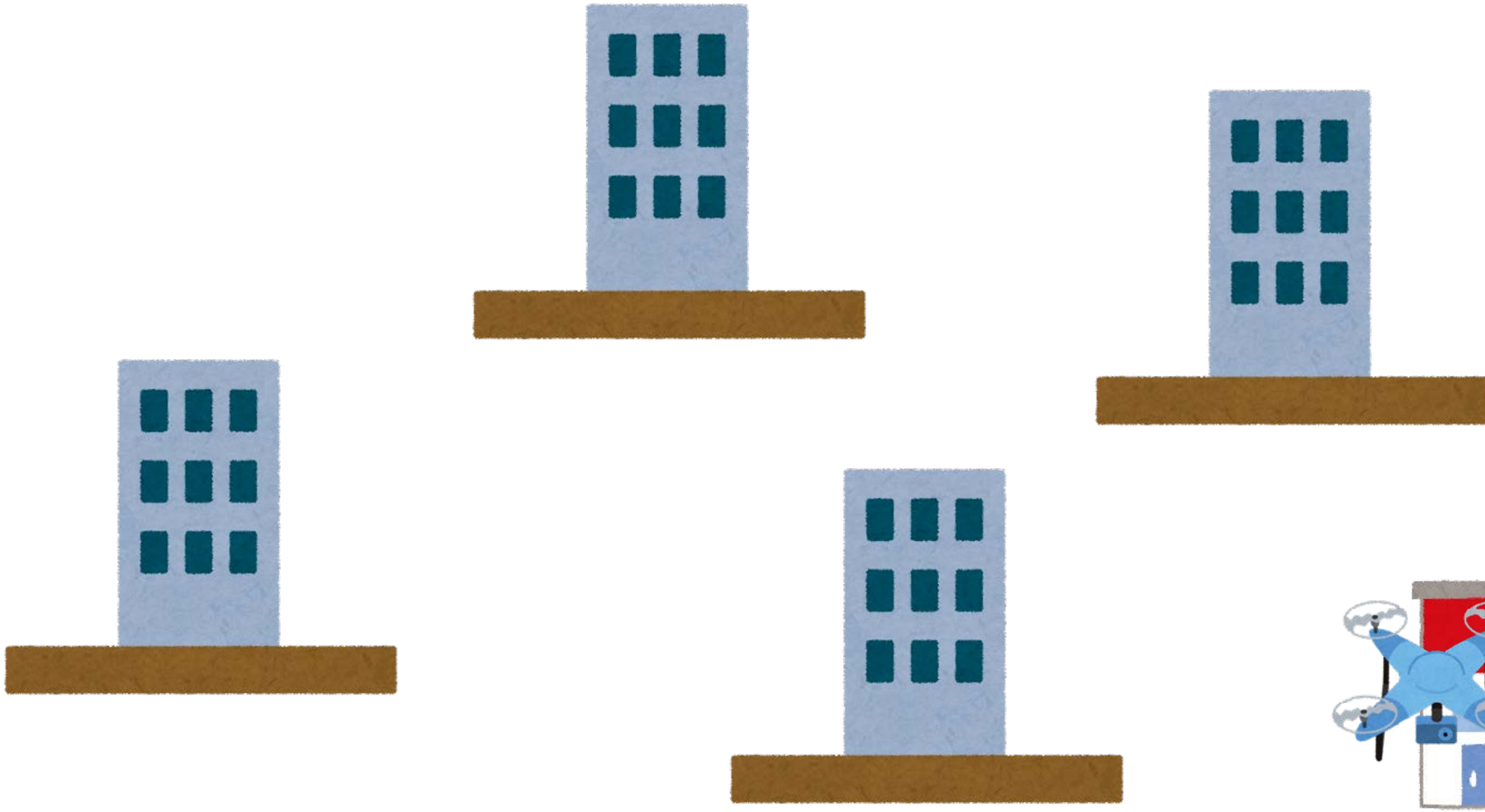




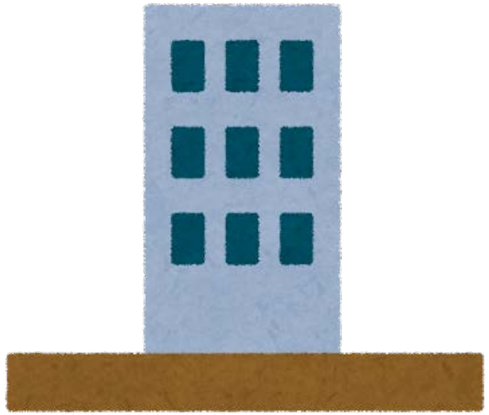
Move one fixed time step

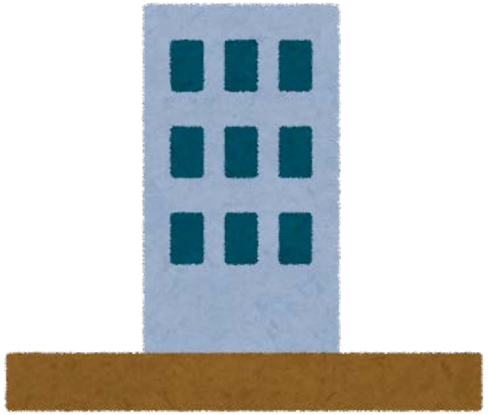


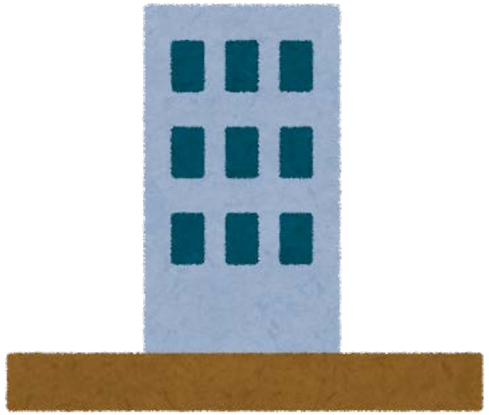
Usual control loop

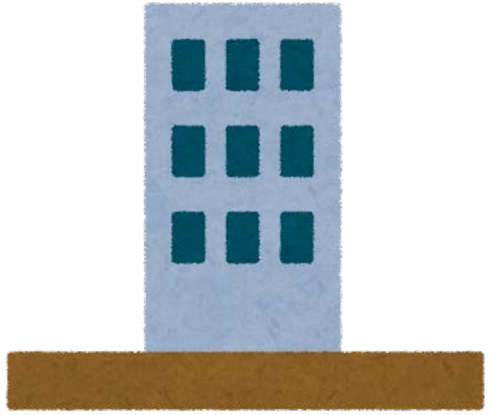


Self-triggered control loop

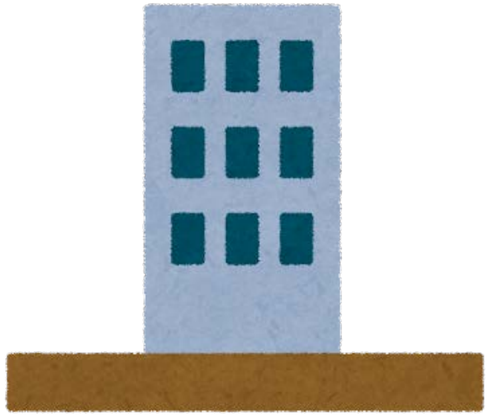


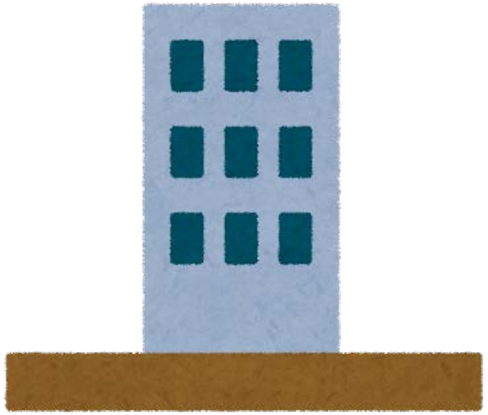


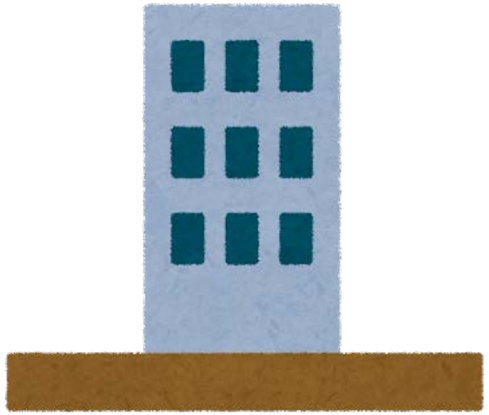


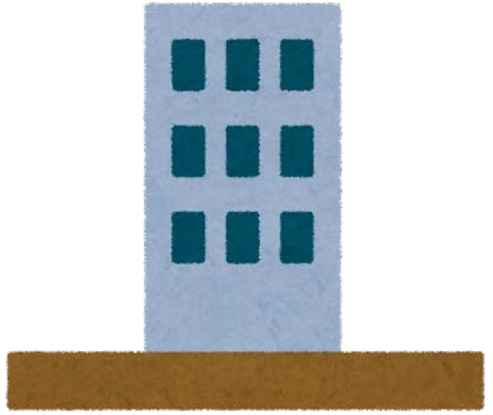


Self-triggered control loop



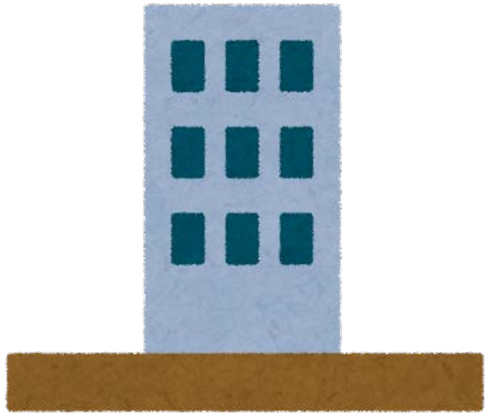


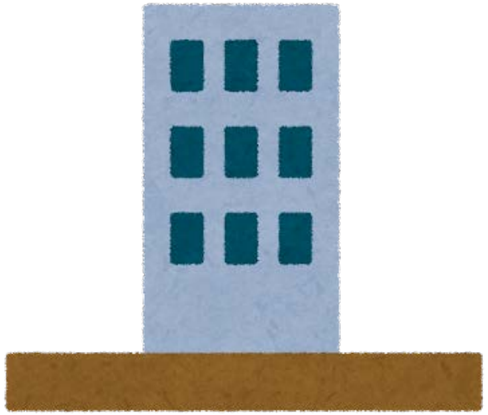


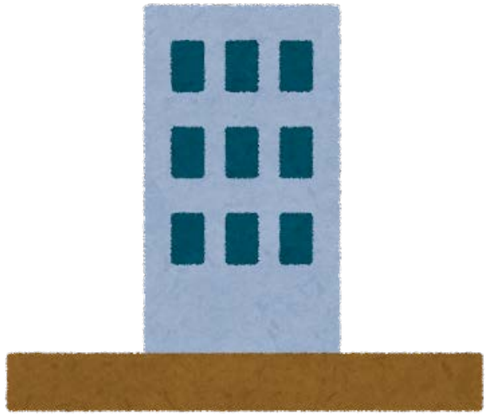


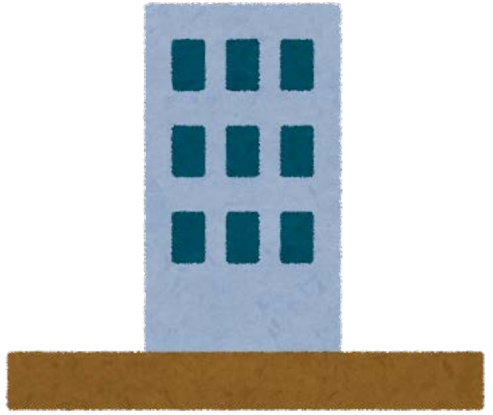
Self-triggered control loop

Control input:
- **Acceleration**
- **Angles**
- **Length of the control input**

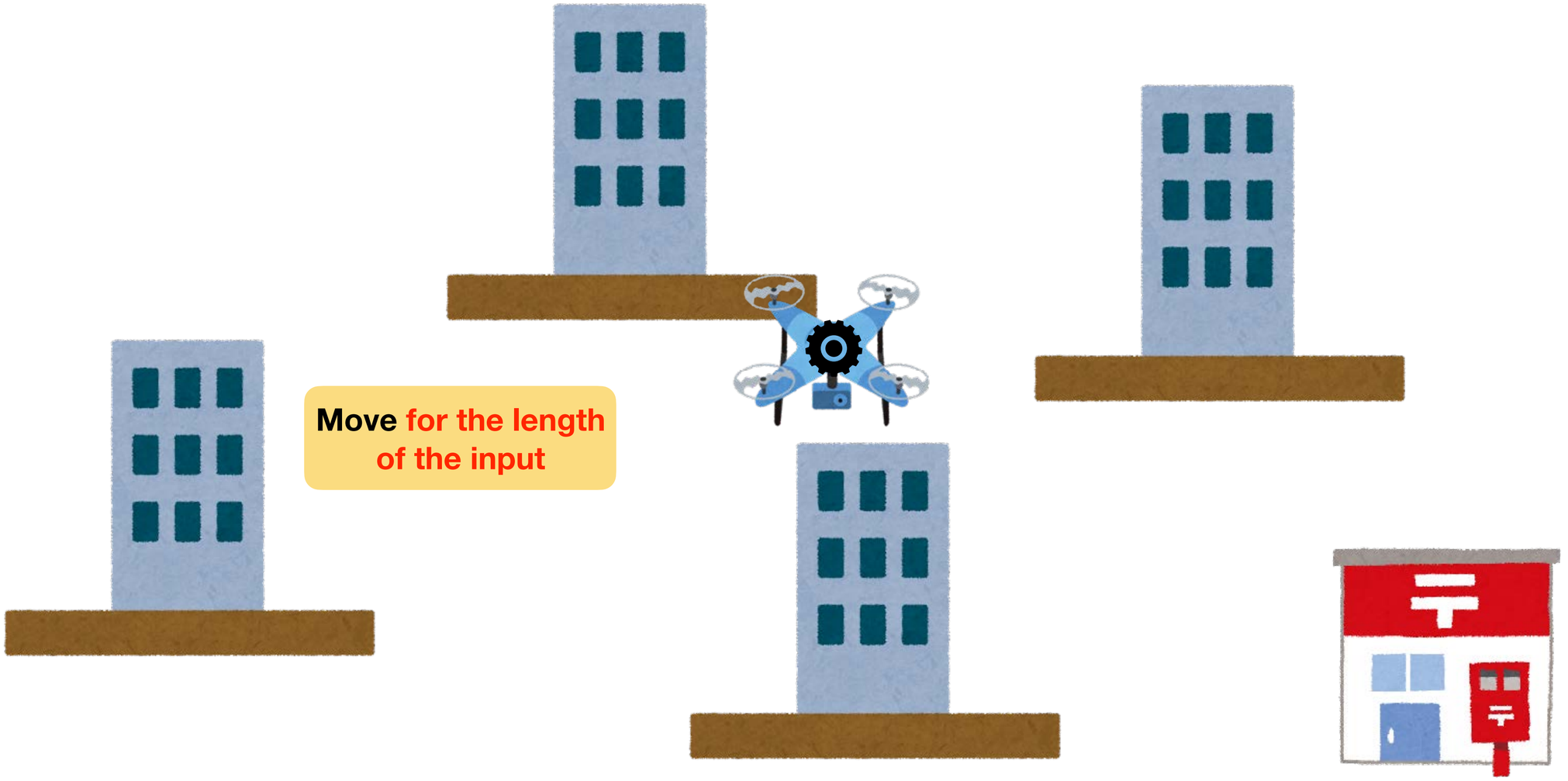




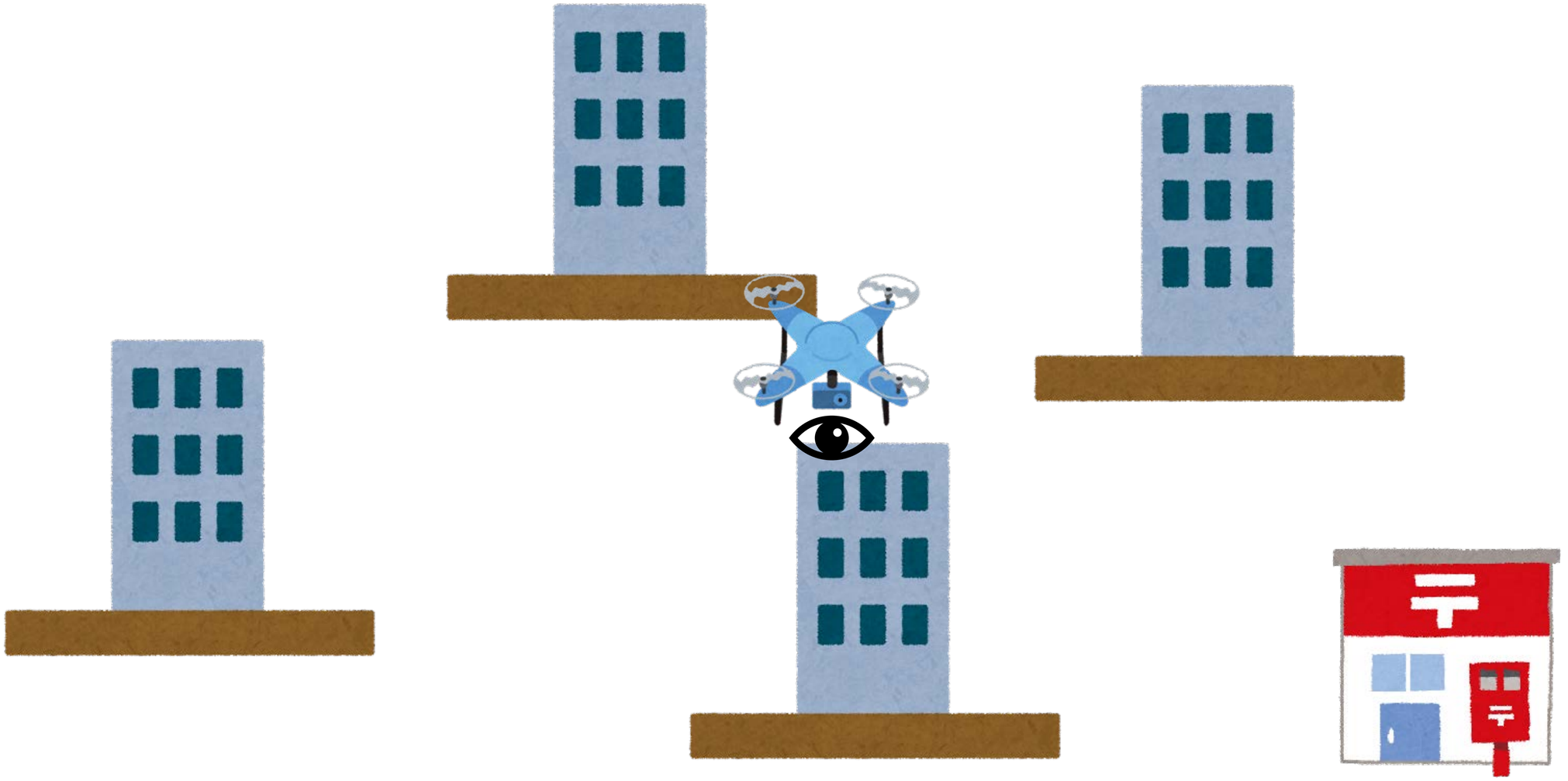




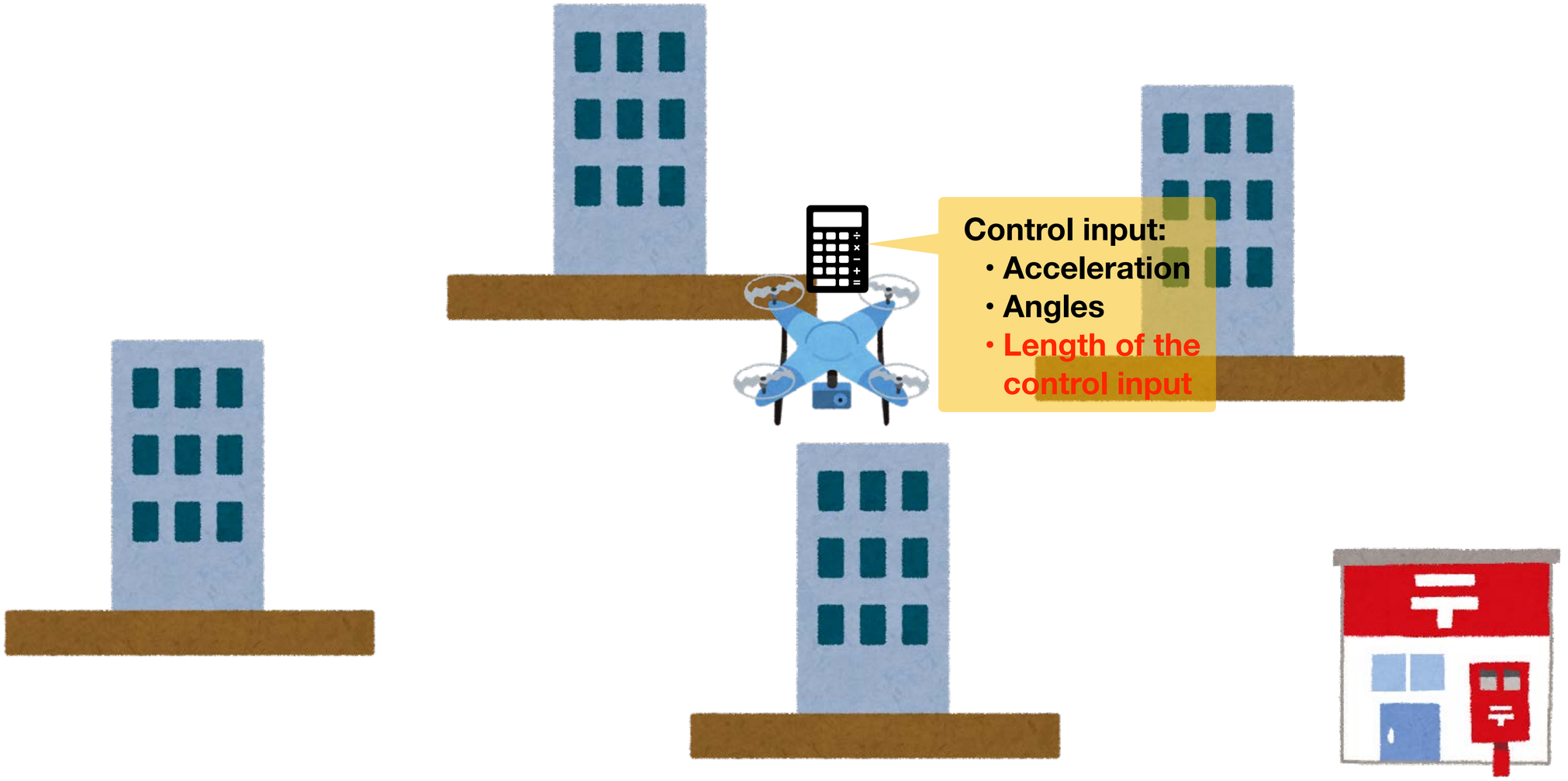
Self-triggered control loop



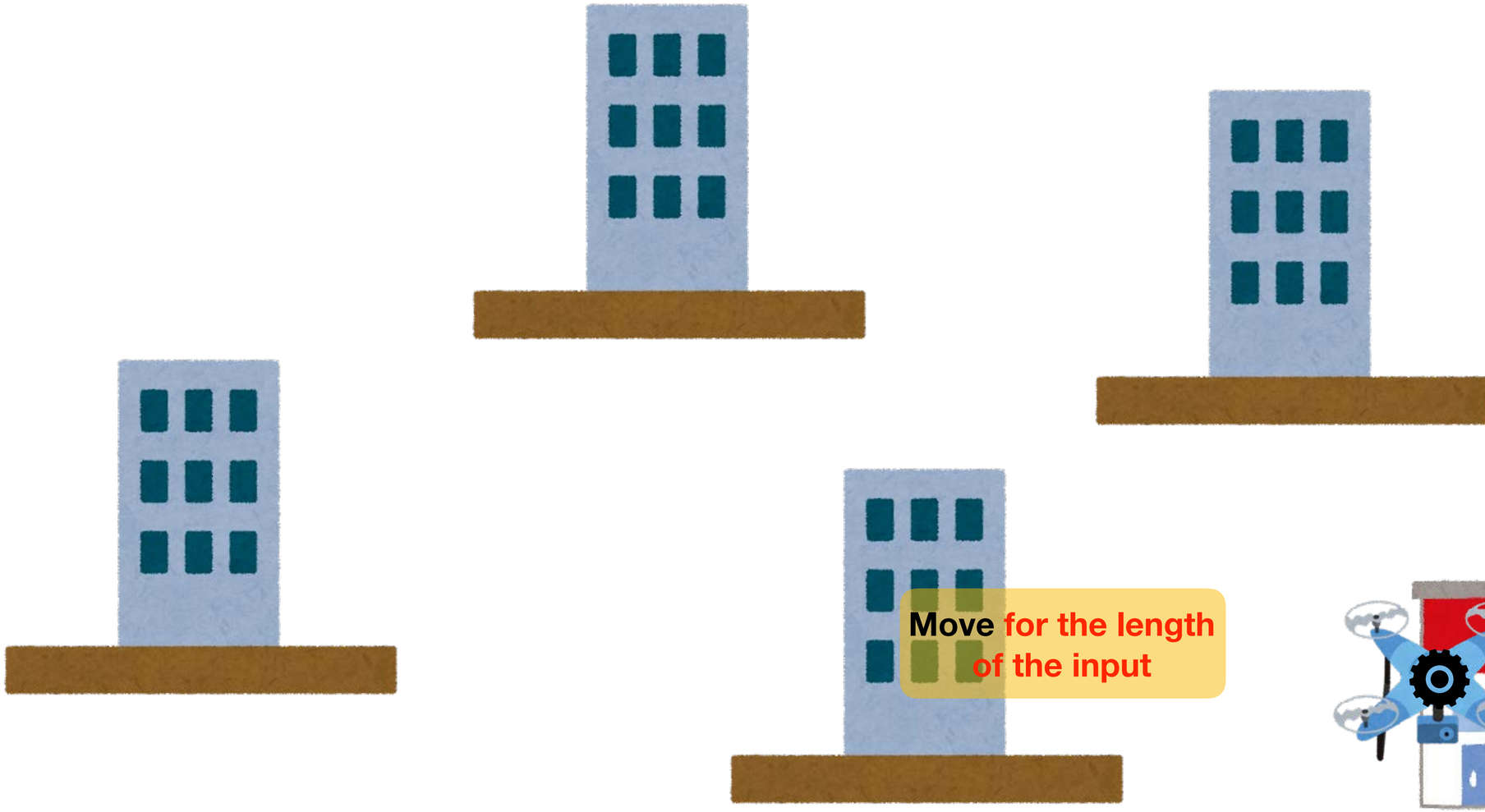
Self-triggered control loop



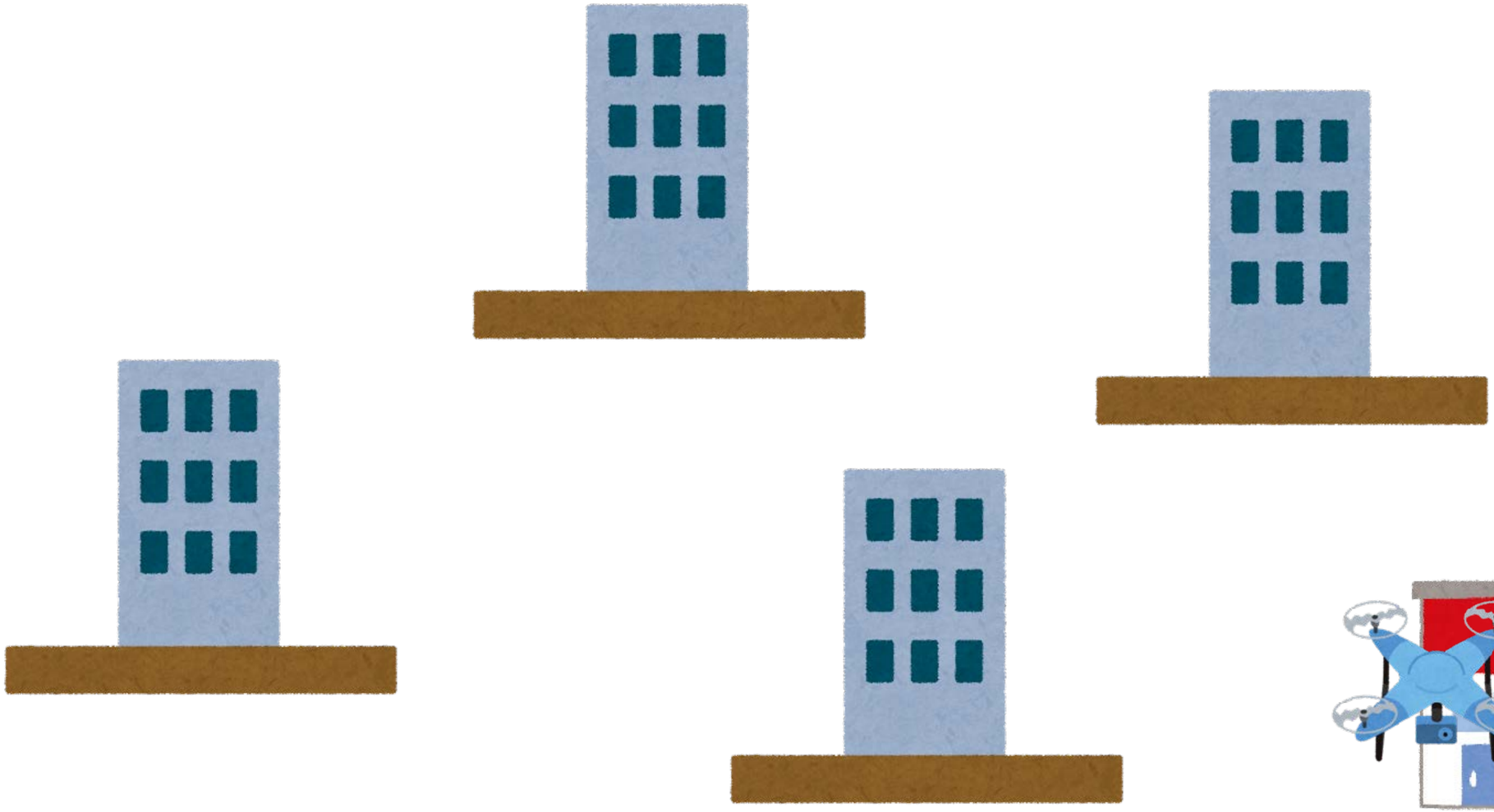
Self-triggered control loop



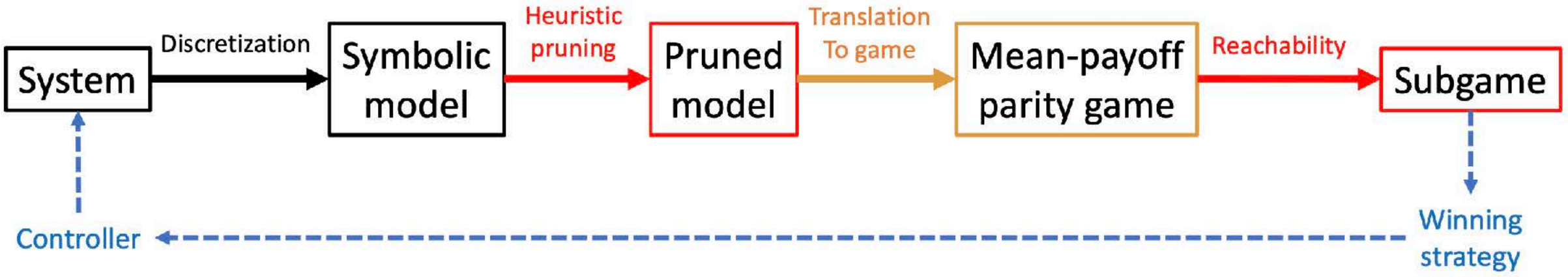
Self-triggered control loop



Self-triggered control loop



Overview of the method



Comparison with related work

Methods	Systems			Specifications
	Time	Type	Self-Triggered	
Zamani et al., IEEE-TAC 2012	Continuous	Deterministic Non-linear	No	Reach-avoid
Hashimoto et al., L-CSS 2019	Discrete	Deterministic Non-linear	Yes	Reach-avoid
Ours	Continuous	Non-deterministic Non-linear	Yes	Right-recursive LTL

Continuous-time non-deterministic system

$$\Sigma = \left(X, X_{in}, U, \mathcal{U}, \xi^{\rightarrow}, \xi^{\leftarrow}\right)$$

Where:

- $X \subseteq \mathbb{R}^n$ bounded convex (state space)
- $X_{in} \subseteq X$ (initial states)
- $U \subseteq \mathbb{R}^m$ bounded convex (input space)
- $\mathcal{U} \subseteq \left\{(T, u) \mid T > 0 \wedge u : [0,T] \to U\right\}$ (signal space)
- $\xi^{\rightarrow} : \mathbb{R}^n \times \mathcal{U} \times \mathbb{R}_{\geq 0} \to \mathscr{P}(\mathbb{R}^n)$ (forward dynamic)
- $\xi^{\leftarrow} : \mathbb{R}^n \times \mathcal{U} \times \mathbb{R}_{\geq 0} \to \mathscr{P}(\mathbb{R}^n)$ (backward dynamic)

Example: non-holonomic robot

With equations:

$$\dot{x}(t) = (1 + \lambda(t)) \,.\, v \,.\, \cos(\theta(t))$$
$$\dot{y}(t) = (1 + \lambda(t)) \,.\, v \,.\, \sin(\theta(t))$$
$$\dot{\theta}(t) = \omega(t)$$

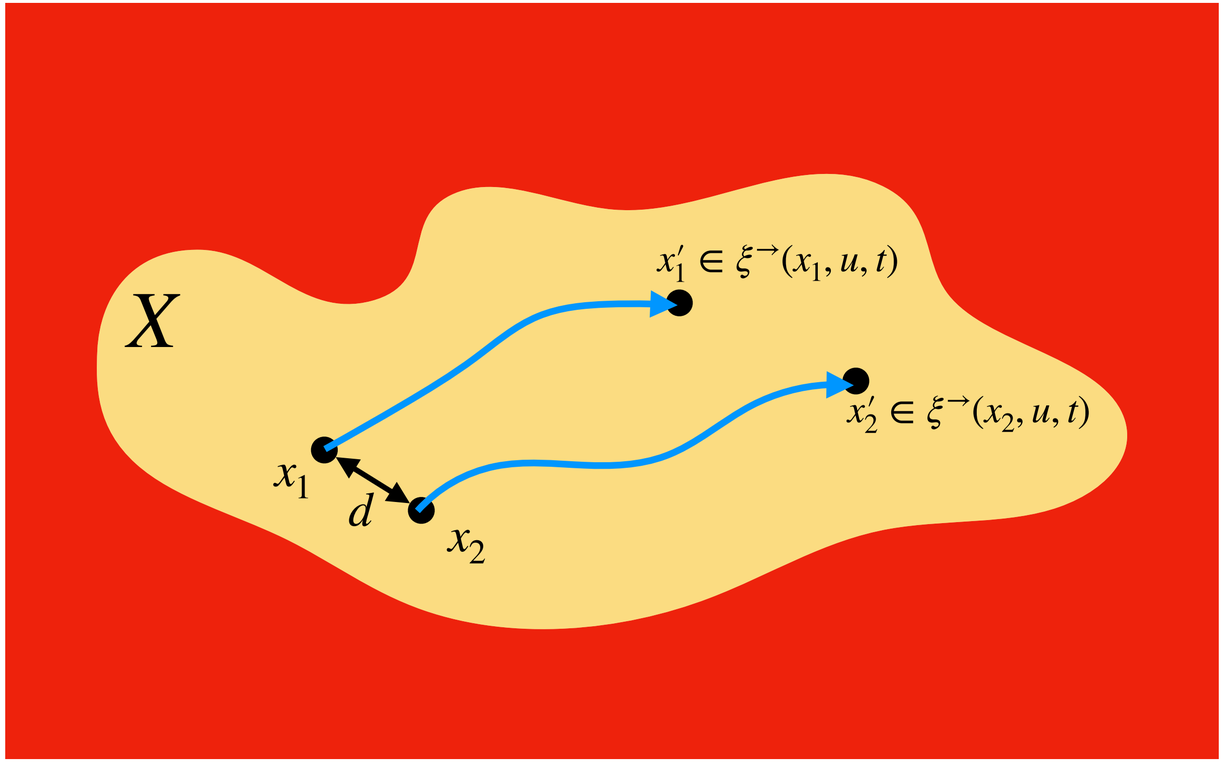
Where:

- v is constant speed
- ω is the control input (steering angle speed)
- λ is non-deterministic ranging over $\left[-\overline{\lambda}, \overline{\lambda} \right]$

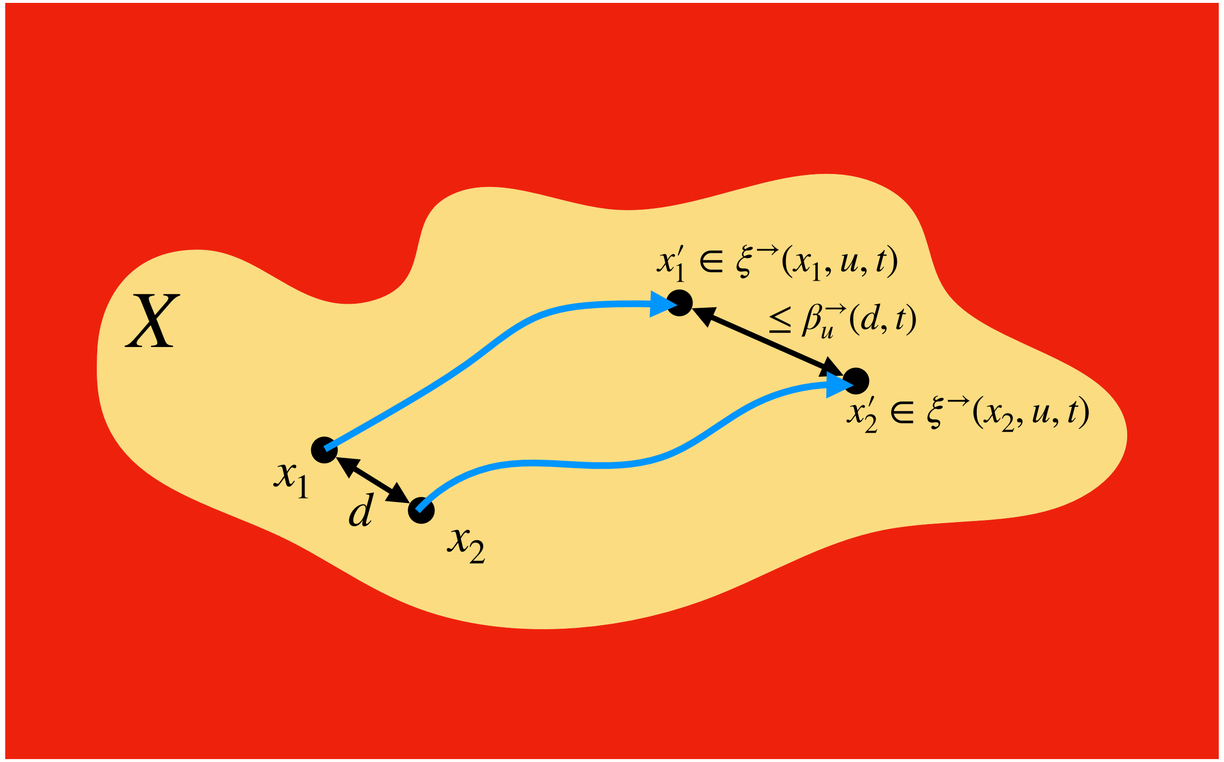
As a system:

- $X \subseteq \mathbb{R}^3$ bounded convex
- $U \subseteq \mathbb{R}$ bounded convex
- $\xi^{\rightarrow} : \mathbb{R}^3 \times \mathcal{U} \times \mathbb{R}_{\geq 0} \to \mathcal{P}(\mathbb{R}^3)$
  $\xi^{\rightarrow}(x, y, \theta, \omega, t) = \Big\{ (x', y', \theta') \mid \exists \lambda : [0,t] \to \left[-\overline{\lambda}, \overline{\lambda} \right] .$
  $\exists \phi : [0,t] \to \mathbb{R}^3$. sol. for ω and λ with
  $\phi(0) = (x, y, \theta) \wedge \phi(t) = (x', y', \theta') \Big\}$
- $\xi^{\leftarrow} : \mathbb{R}^n \times \mathcal{U} \times \mathbb{R}_{\geq 0} \to \mathcal{P}(\mathbb{R}^n)$
  $\xi^{\rightarrow}(x, y, \theta, \omega, t) = \Big\{ (x', y', \theta') \mid \exists \lambda : [0,t] \to \left[-\overline{\lambda}, \overline{\lambda} \right] .$
  $\exists \phi : [0,t] \to \mathbb{R}^3$. sol. for ω and λ with
  $\phi(t) = (x, y, \theta) \wedge \phi(0) = (x', y', \theta') \Big\}$

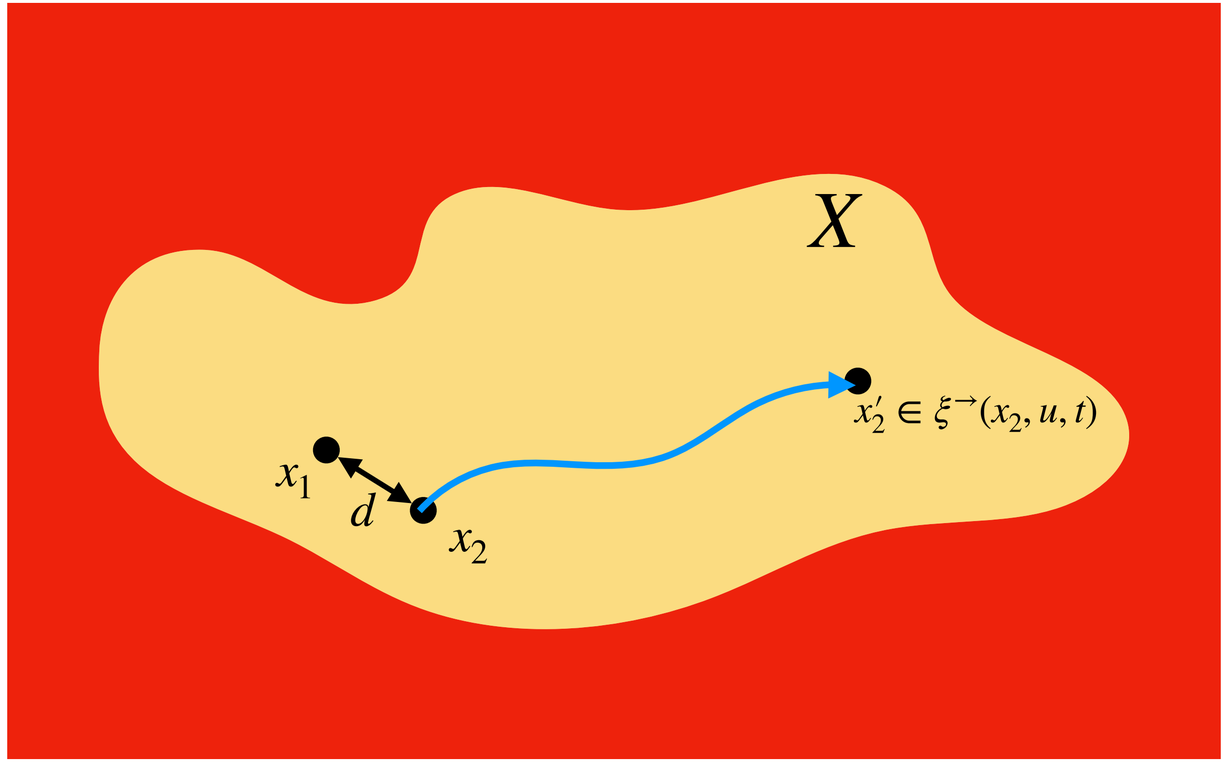
Conditions on systems (I)



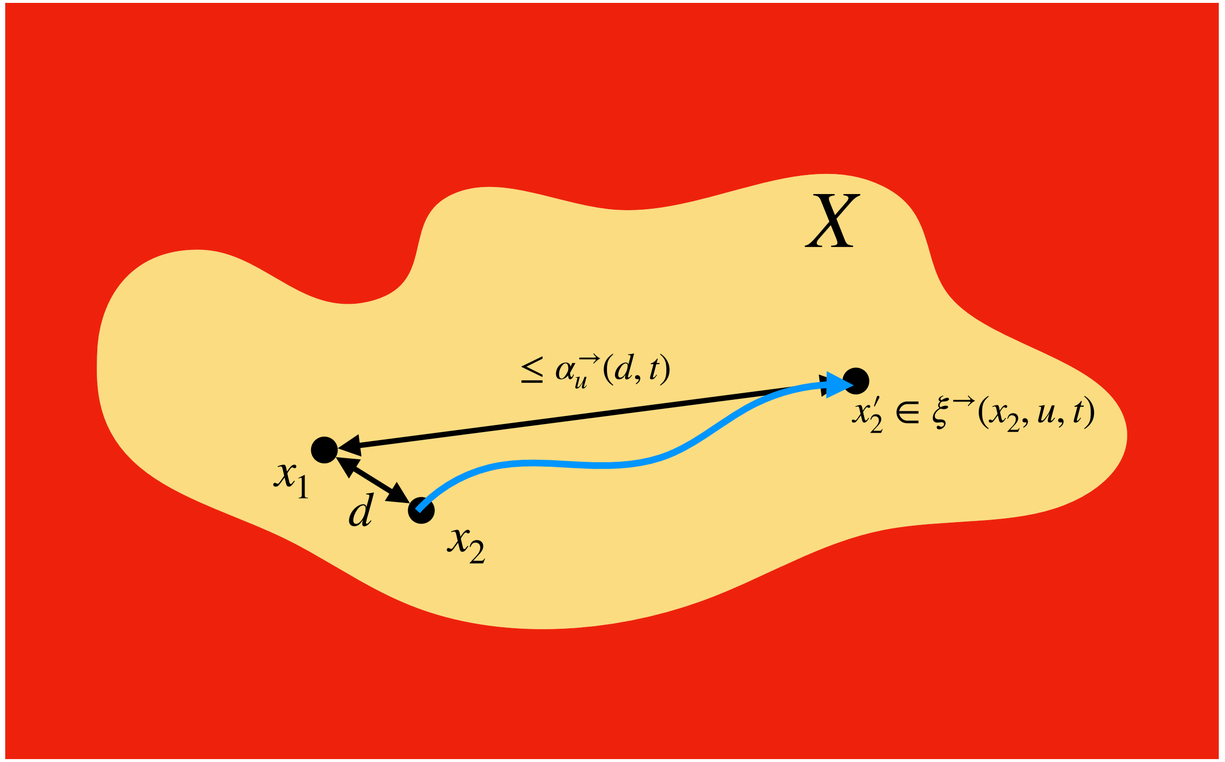
Conditions on systems (I)



Conditions on systems (I)



Conditions on systems (I)



Runs and trajectories

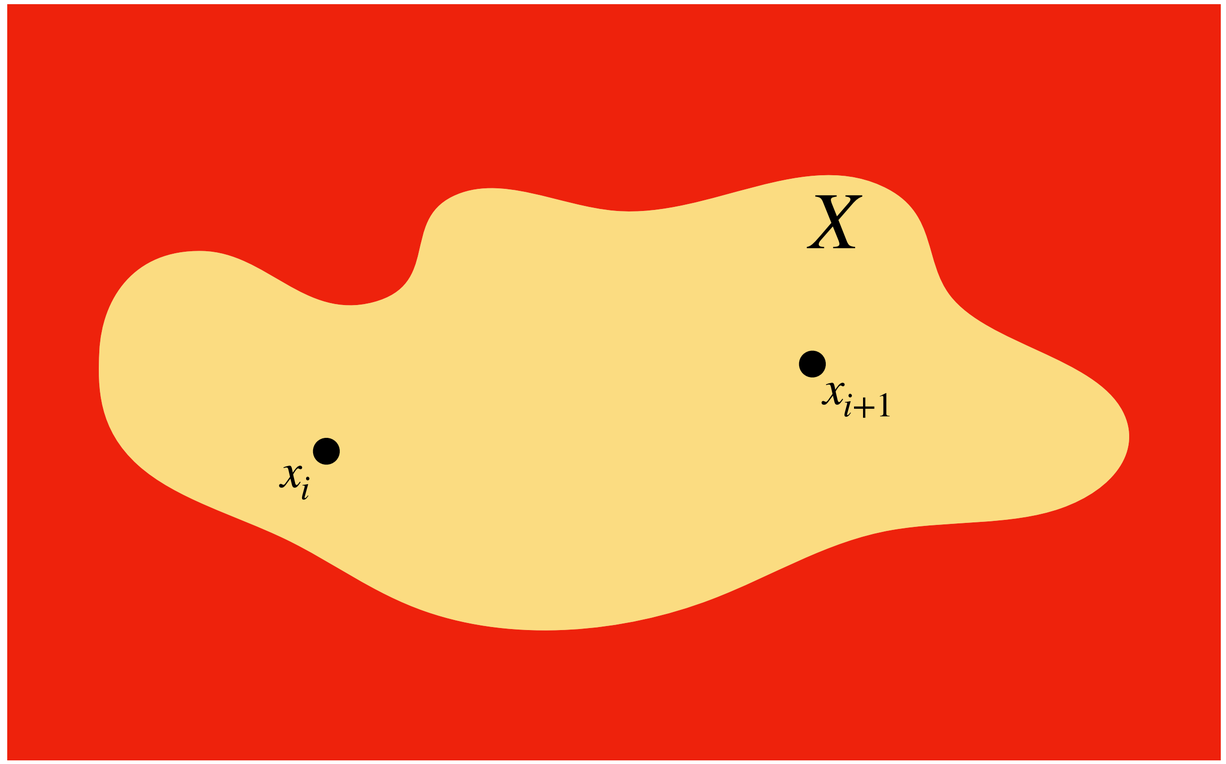
Run:

$$x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$$

Runs and trajectories

Run:

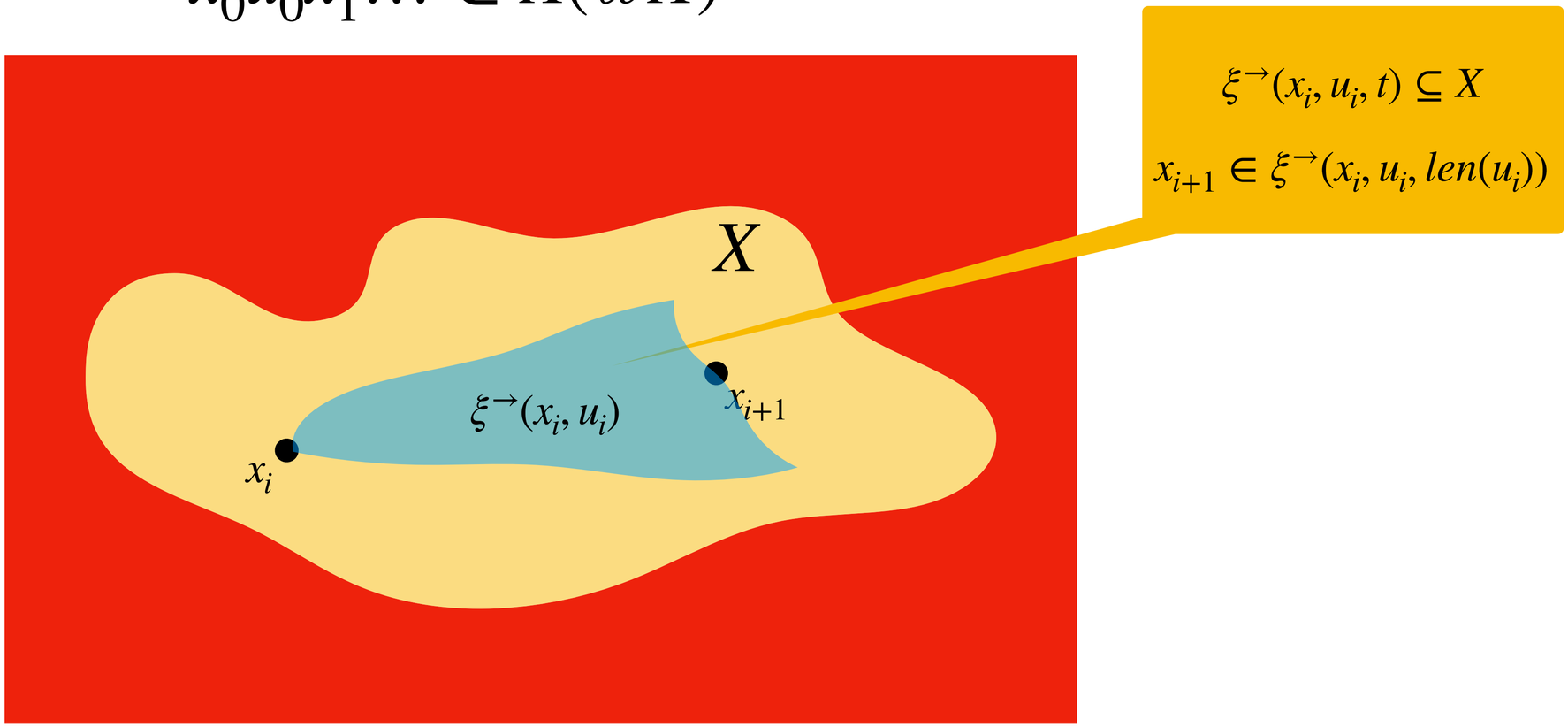
$$x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$$



Runs and trajectories

Run:

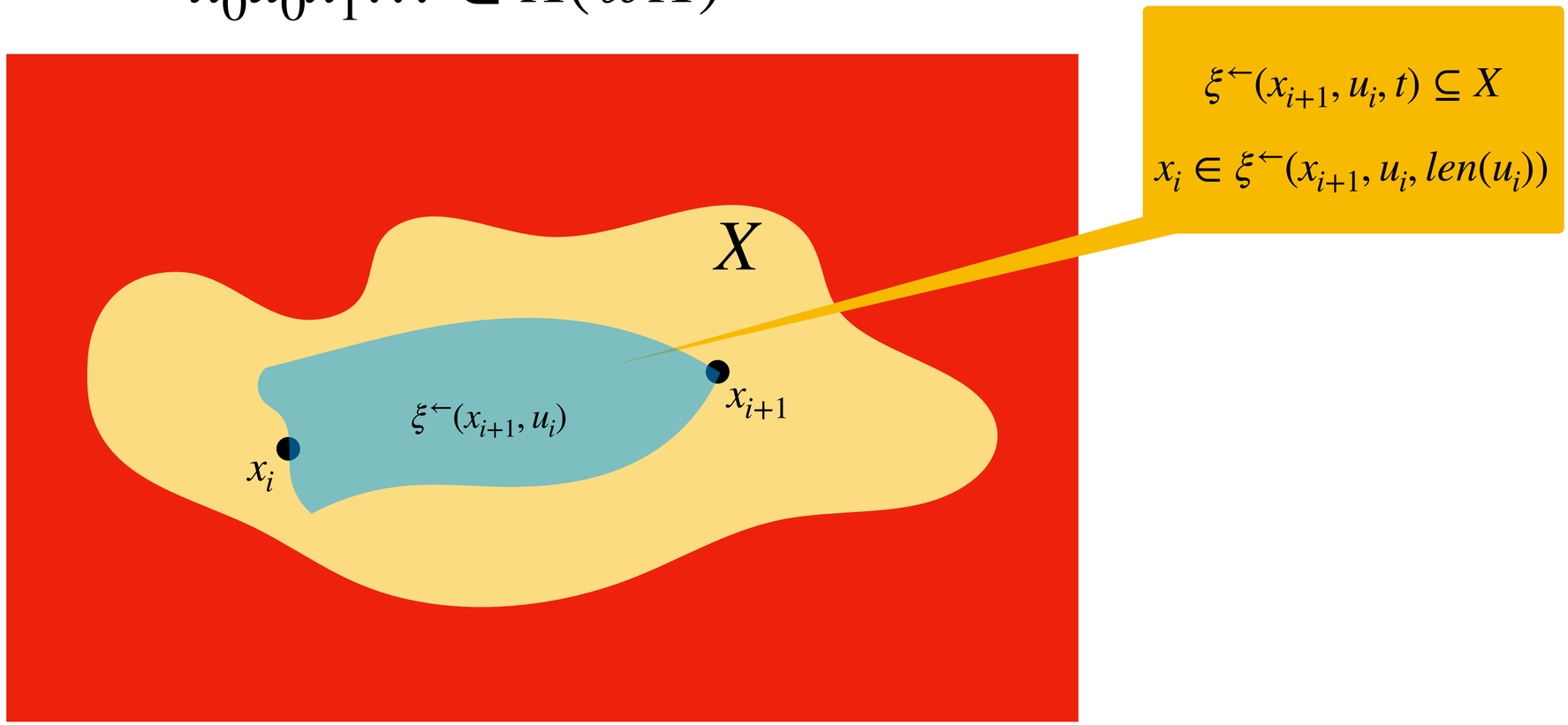
$$x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$$



Runs and trajectories

Run:

$$x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$$



Runs and trajectories

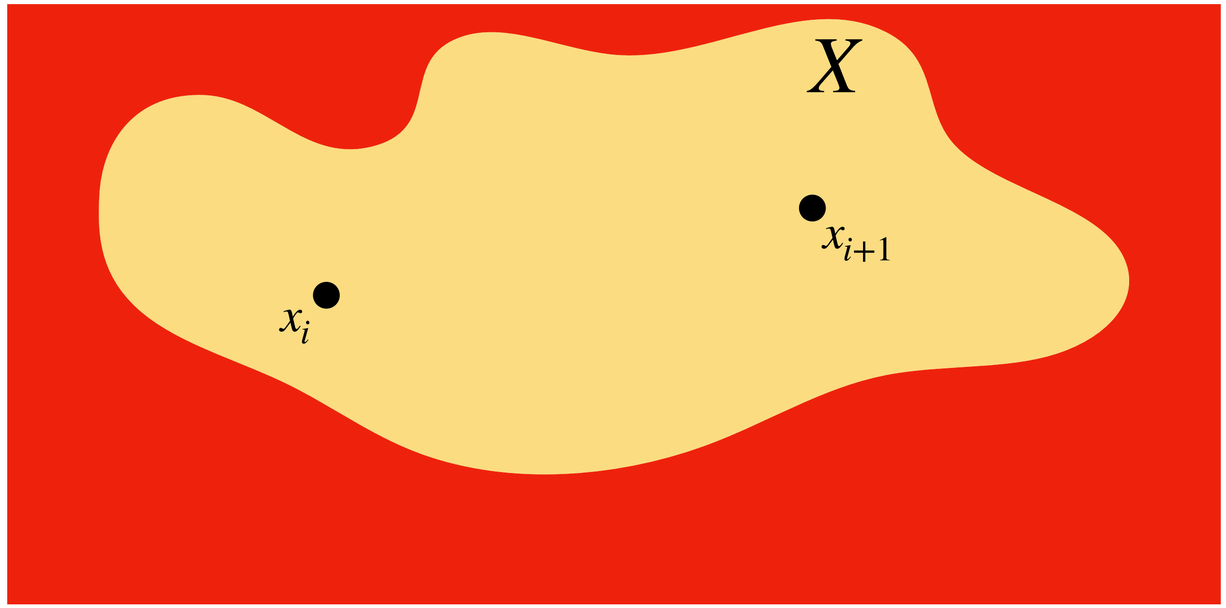
Trajectory: given a run $x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$, an associated trajectory:

$$\sigma : \mathbb{R}_{\geq 0} \to X$$

Runs and trajectories

Trajectory: given a run $x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$, an associated trajectory:

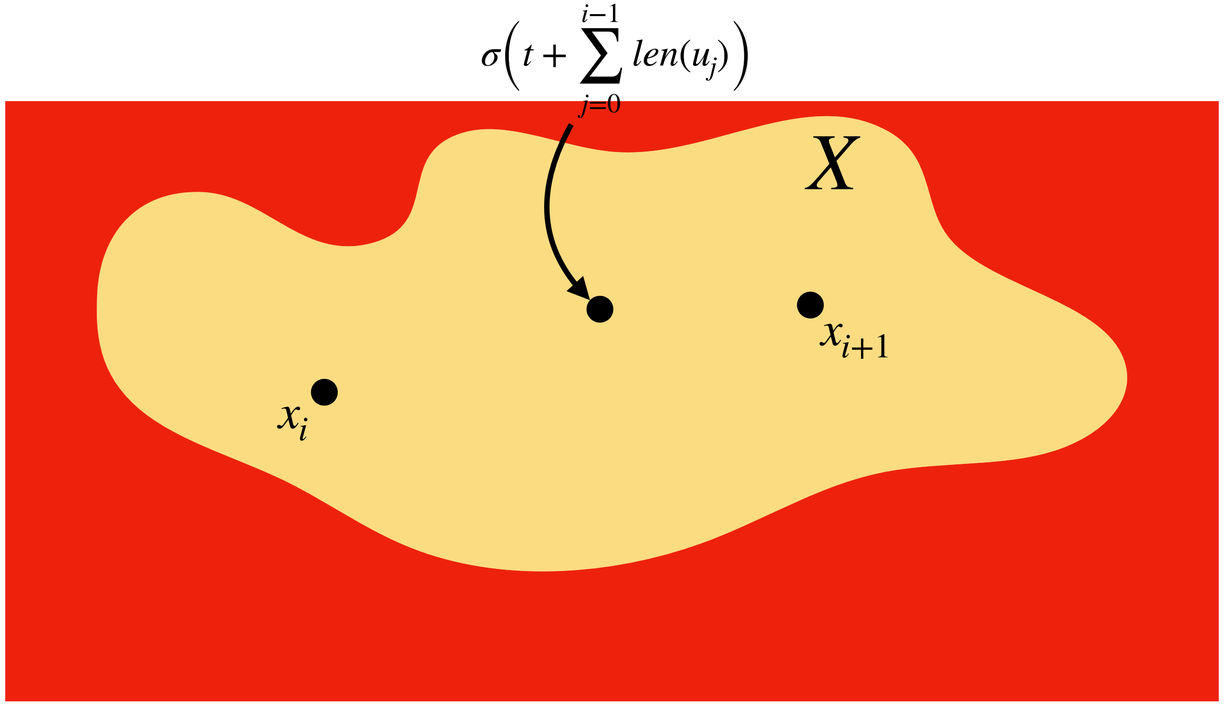
$$\sigma : \mathbb{R}_{\geq 0} \to X$$



Runs and trajectories

Trajectory: given a run $x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$, an associated trajectory:

$$\sigma : \mathbb{R}_{\geq 0} \to X$$



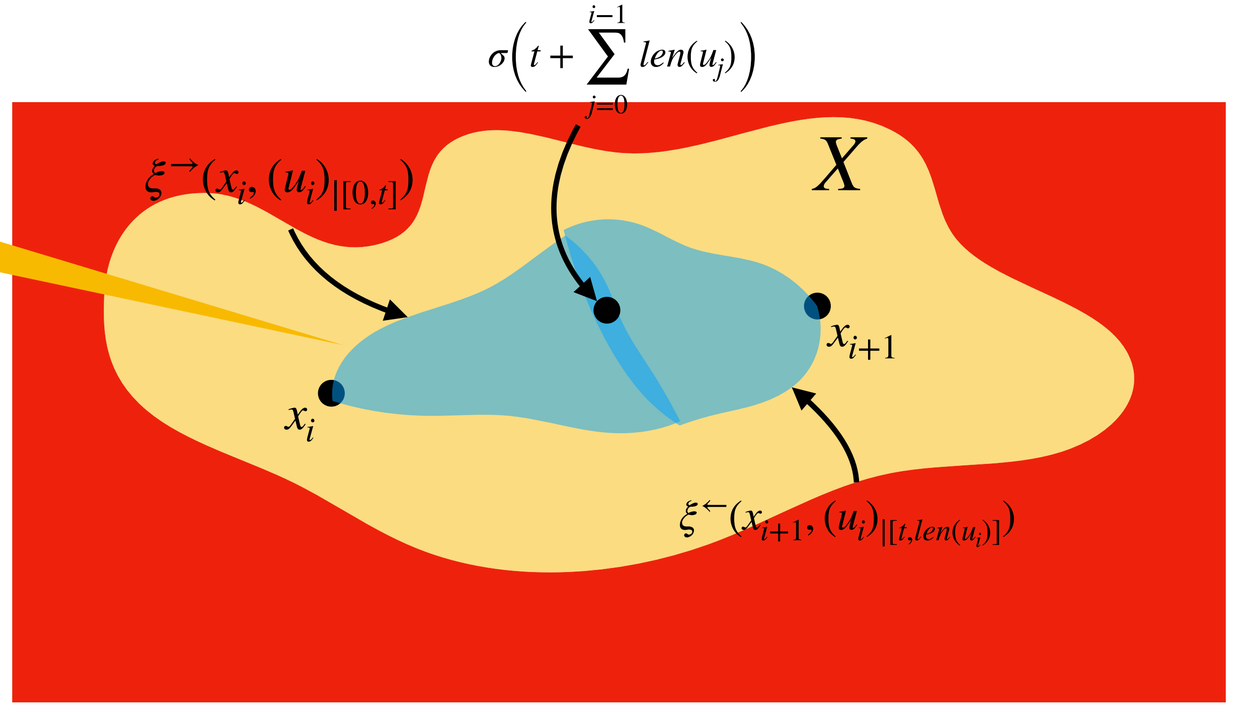
Runs and trajectories

Trajectory: given a run $x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)^\omega$, an associated trajectory:

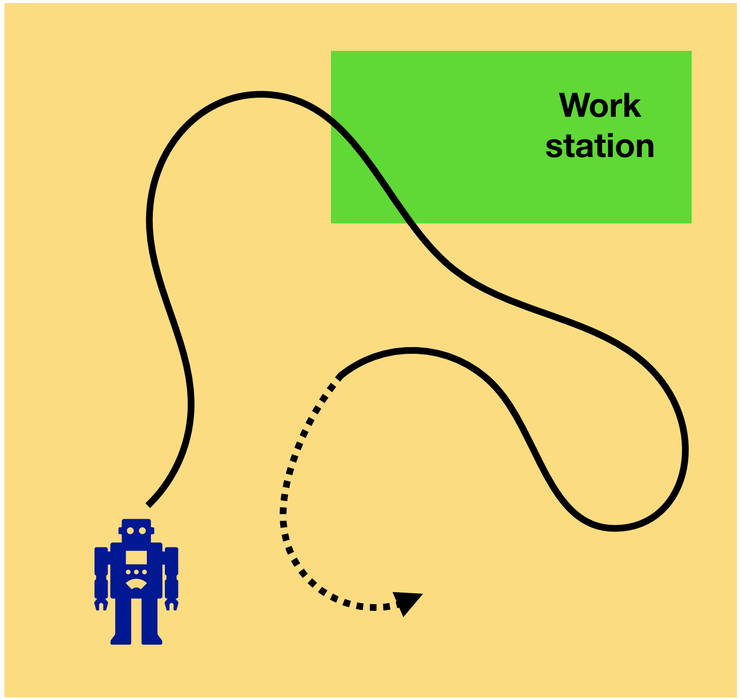
$$\sigma : \mathbb{R}_{\geq 0} \to X$$

$$\sigma\left(t + \sum_{j=0}^{i-1} len(u_j)\right) \in$$

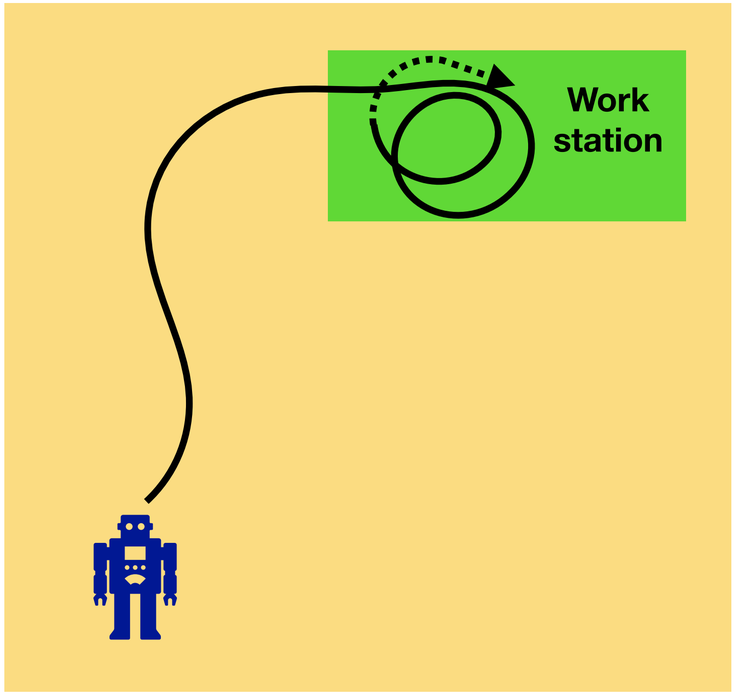
$$\xi^{\rightarrow}(x_i, u_i, t) \cap \xi^{\leftarrow}(x_{i+1}, u_i, len(u_i) - t)$$



Specifications, informally

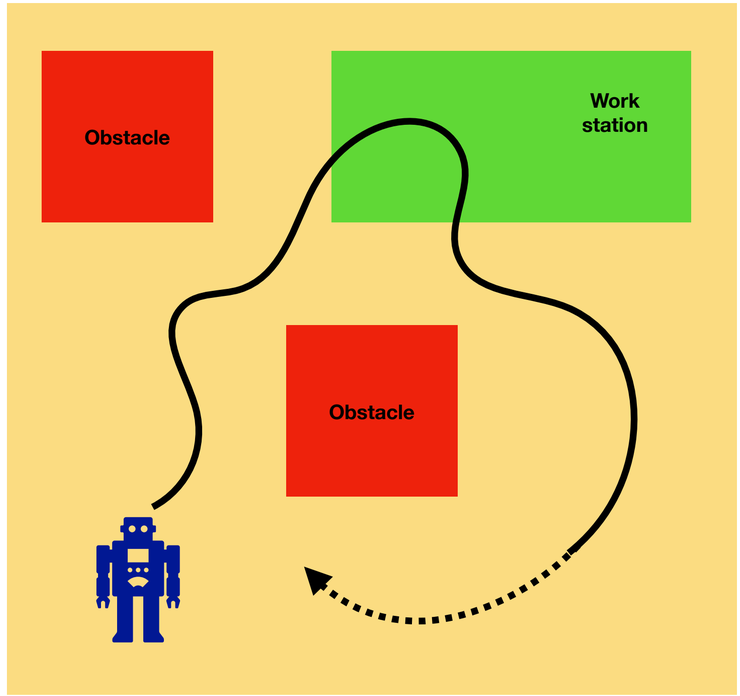


Reach
“*At some point*
we enter the desired zone**”**

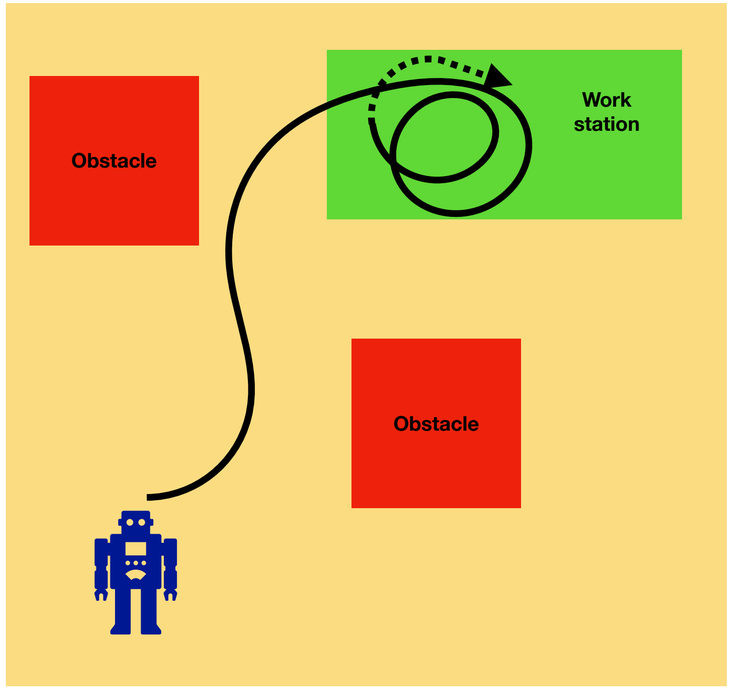


Better reach
“*At some point*
we enter the desired zone
and we stay in it ***indefinitely*****”**

Specifications, informally

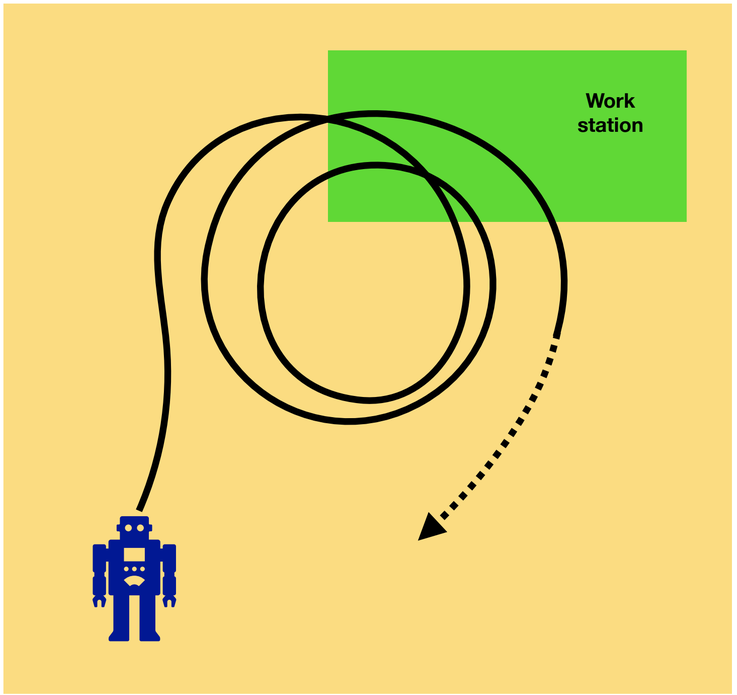


Reach-avoid
“*At some point*
we enter the desired zone
while avoiding obstacles
***at all times*”**

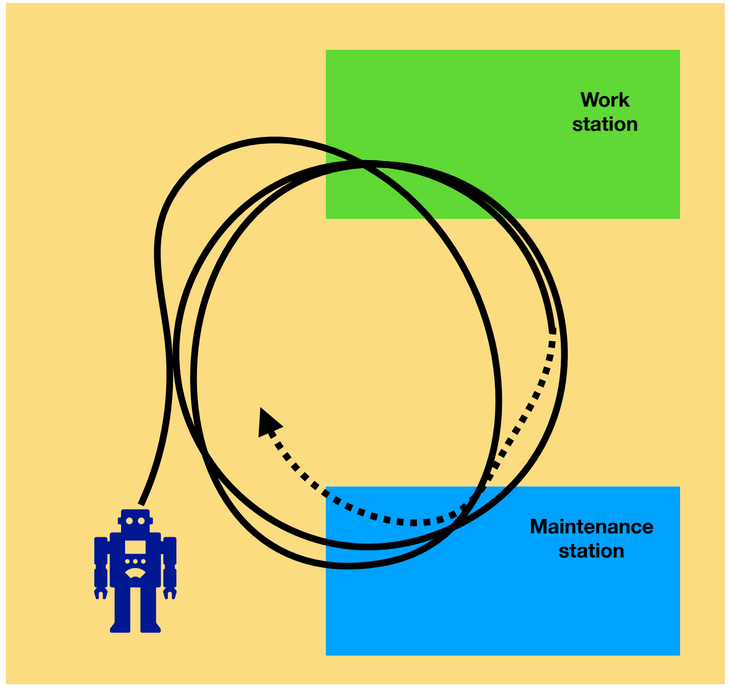


Better reach-avoid
“*At some point*
we enter the desired zone
and we stay in it ***indefinitely***
while avoiding obstacles
***at all times*”**

Specifications, informally



Recurrent reach
“Visit the desired zone
infinitely often”



Double recurrent reach
“Navigate between
two desired zones
infinitely often”

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

State formula

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Atomic propositions

$$P : X \to \mathscr{P}(AP)$$

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Path formulas

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Difference with LTL: no negation

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Until: "ϕ holds until Φ holds, and Φ must hold at some point"
Difference with LTL: state formula on the left only

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Diamond: "ϕ holds at some point"

Definable: $\top \mathcal{U} \phi$

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Box: "ϕ holds at all times"

~~Definable: $\neg(\top \mathcal{U} \neg\phi)$~~

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

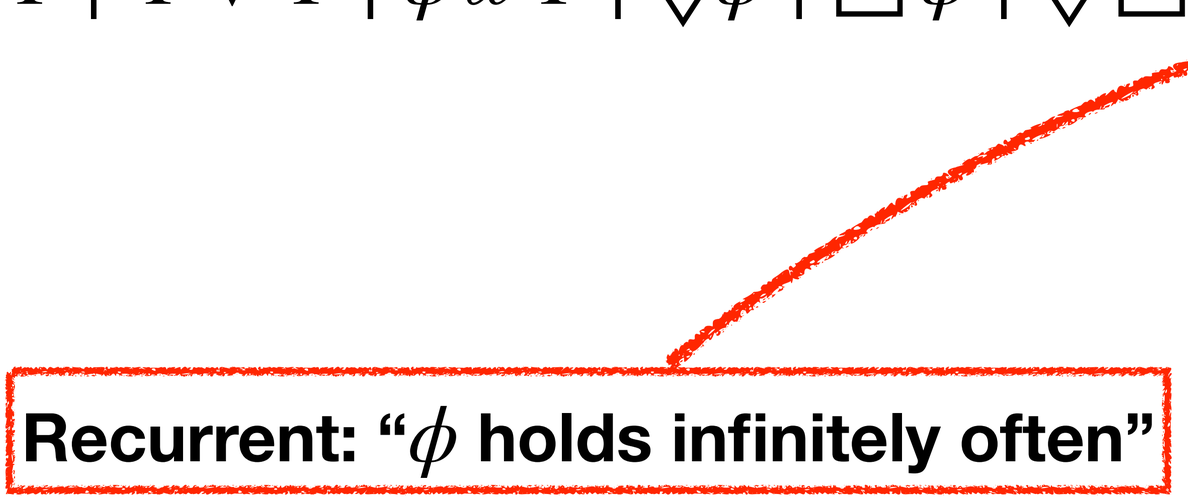
Ultimately: "ϕ holds at all times after some point"

Definable: $\top \mathcal{U} (\Box\phi)$

Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$



Specifications, formally: right-recursive LTL

$$\phi ::= \top \mid p \in AP \mid \neg\phi \mid \phi \vee \phi$$

$$\Phi ::= \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \phi \mathcal{U} \Phi \mid \Diamond\phi \mid \Box\phi \mid \Diamond\Box\phi \mid \Box\Diamond\phi$$

Difference with LTL: no arbitrary alternations of modalities?

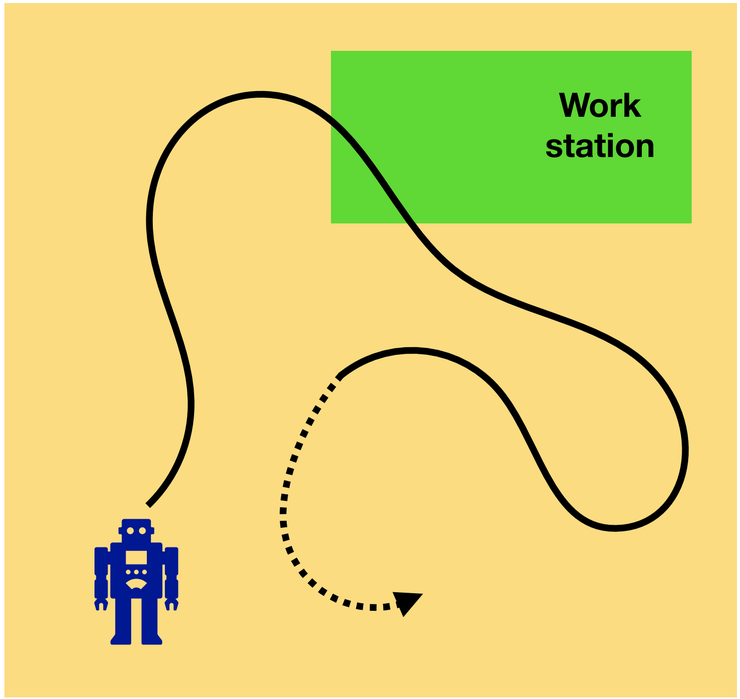
$$\Diamond\Diamond\phi \equiv \Diamond\phi$$

$$\Box\Box\phi \equiv \Box\phi$$

$$\Diamond\Box\Diamond\phi \equiv \Box\Diamond\phi$$

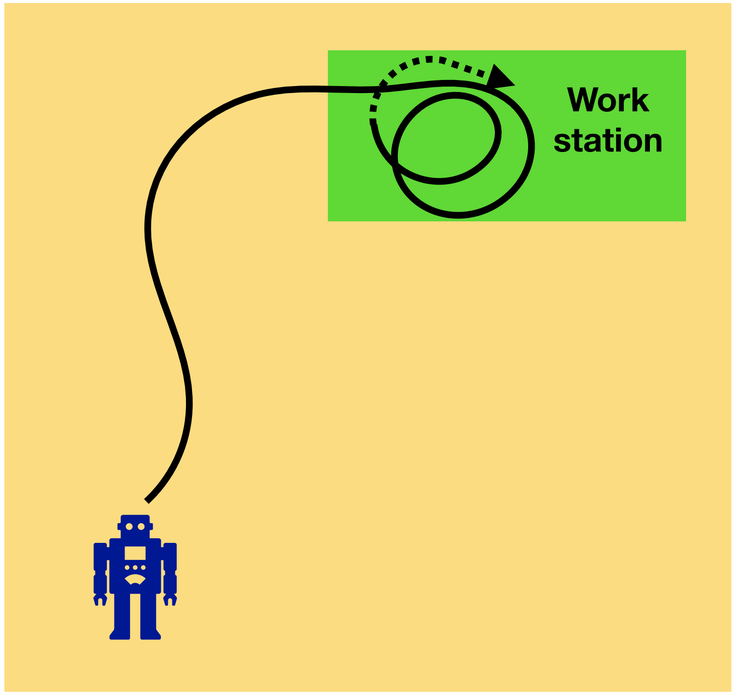
$$\Box\Diamond\Box\phi \equiv \Diamond\Box\phi$$

Specifications, formally



Reach

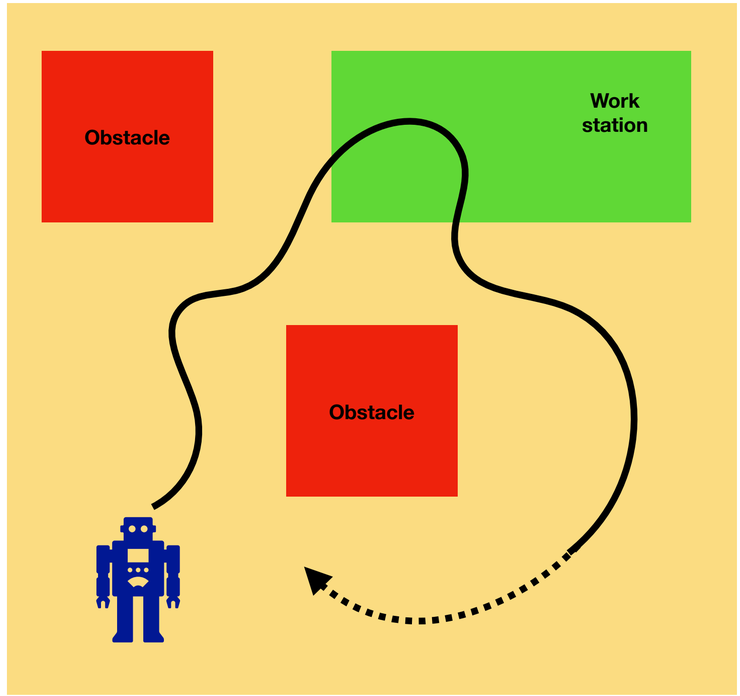
$\Diamond$**work_station**



Better reach

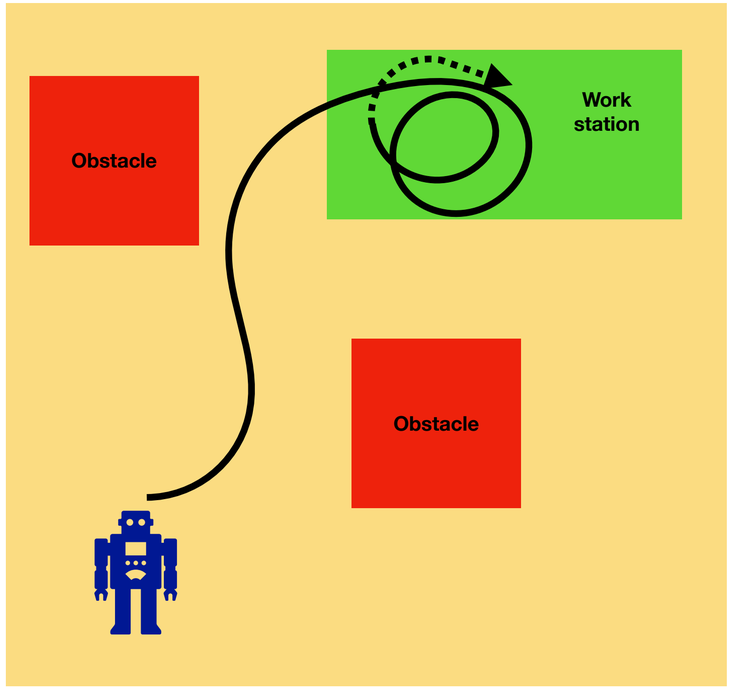
$\Diamond \Box$ **work_station**

Specifications, formally



Reach-avoid

$\Diamond$**work_station** $\wedge$ $\Box$ $\neg$**obstacle**

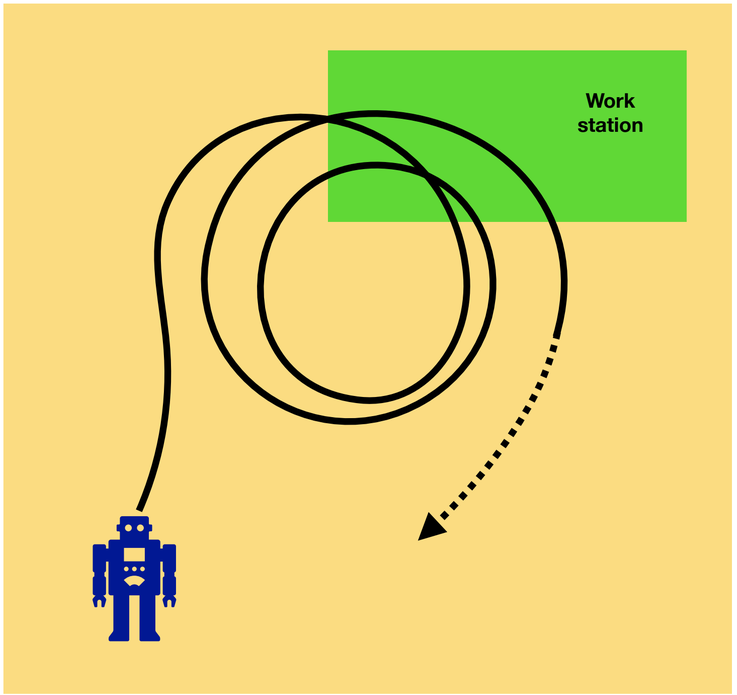


Better reach-avoid

$\Diamond$ $\Box$ **work_station** $\wedge$ $\Box$ $\neg$**obstacle**

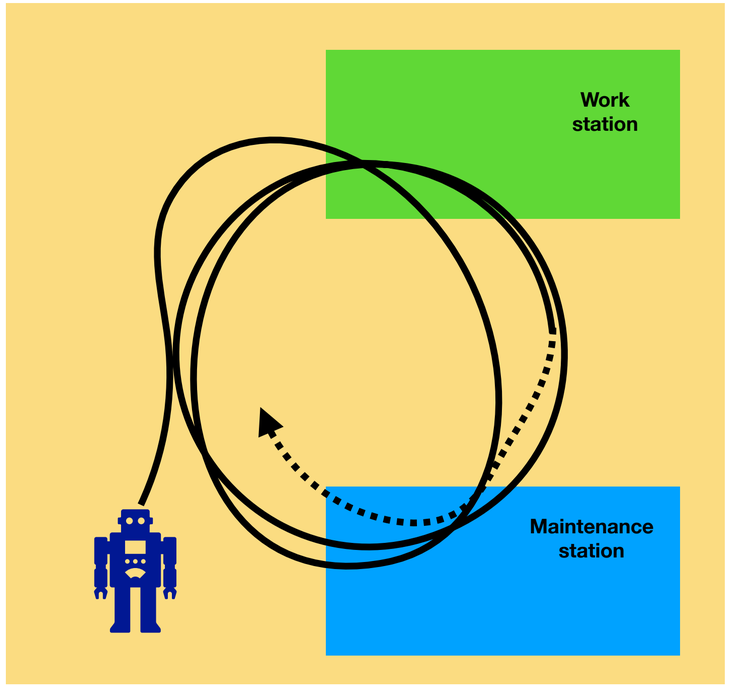
$(\neg$**obstacle**$)$ $\mathcal{U}$ $(\Box$ **work_station**$)$

Specifications, formally



Recurrent reach

$\Box \Diamond$**work_station**



Double recurrent reach

$\Box \Diamond$**work_station** $\wedge \Box \Diamond$**maintenance_station**

Semantics

State formula:

$x \vDash \top$ **always holds**

$x \vDash p \in AP$ **iff** $p \in P(x)$

$x \vDash \neg\phi$ **iff** $x \nvDash \phi$

$x \vDash \phi_1 \vee \phi_2$ **iff** $x \vDash \phi_1$ **or** $x \vDash \phi_1$

Path formula:

$\sigma \vDash \phi \mathcal{U} \Phi$ **iff** $\exists t \,.\, \sigma_{|t} \vDash \Phi \wedge$
$\forall t' \geq t \,.\, \sigma(t') \vDash \phi$

$\sigma \vDash \Diamond \phi$ **iff** $\exists t \,.\, \sigma(t) \vDash \phi$

$\sigma \vDash \Box \phi$ **iff** $\forall t \,.\, \sigma(t) \vDash \phi$

$\sigma \vDash \Diamond \Box \phi$ **iff** $\exists t \,.\, \forall t' \geq t \,.\, \sigma(t') \vDash \phi$

$\sigma \vDash \Box \Diamond \phi$ **iff** $\forall t \,.\, \exists t' \geq t \,.\, \sigma(t') \vDash \phi$

Controller and controlled runs

Controller of the system Σ: partial function:

$$C : \ FRun(\Sigma) \to \mathcal{U}$$

Controlled run: $x_0 u_0 x_1 \ldots \in X(\mathcal{U}X)$ such that for all i:

$$C(x_0 u_0 x_1 \ldots u_{i-1} x_i) = u_i$$

Controlled trajectory: trajectory associated with a controlled run

Control problem

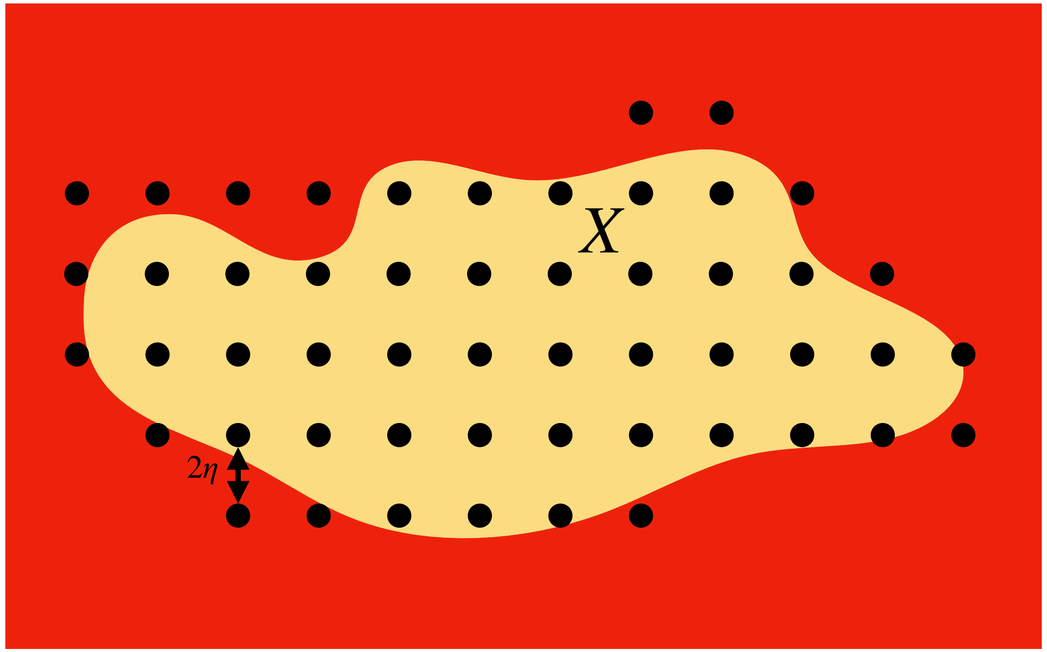
Data:

- Continuous-time non-deterministic system Σ
- Function $P : X \to \mathscr{P}(AP)$
- Right-recursive formula Φ
- Real $\nu > 0$

Output: controller C for Σ such that:

- **(Non-blocking)** every finite controlled run can be extended into an infinite one
- **(Valid)** every controlled trajectory σ satisfies Φ
- **(Self-triggered)** $\liminf_{h \to \infty} \frac{1}{h} \sum_{i=0}^{h} len(u_i) \geq \nu$

Symbolic model: discretised states, inputs

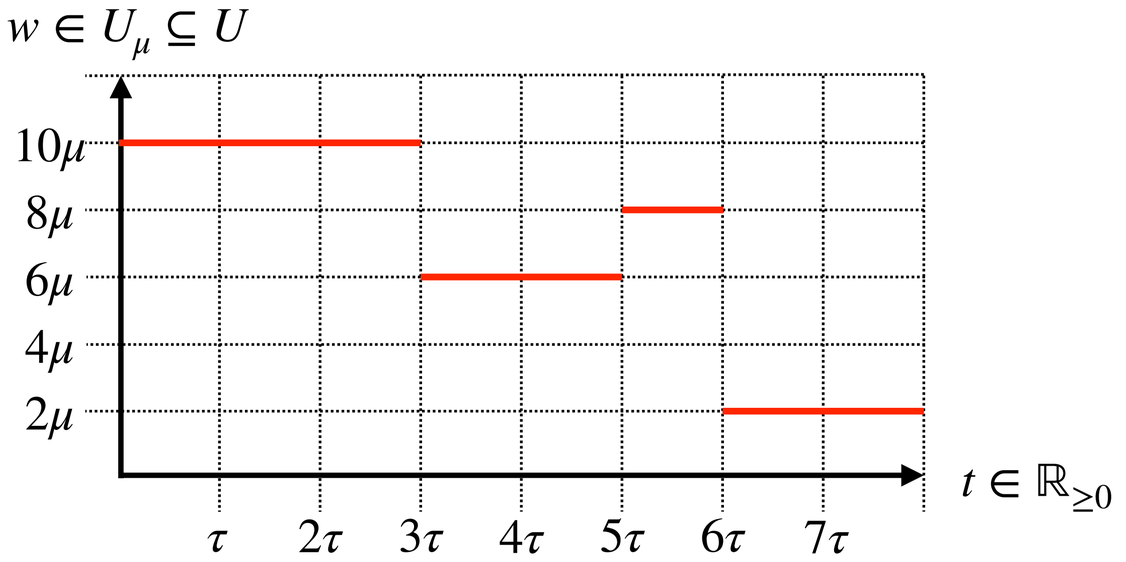


$$[X]_\eta = \{q \in (2\eta\mathbb{Z})^n \mid \mathscr{B}(q,\eta) \cap X \neq \emptyset\}$$

Discretised states

$$U_\mu = \{w \in (2\mu\mathbb{Z})^m \mid w \in U\}$$

Discretised input values



$\mathscr{U}_{\tau,\mu} \subseteq \mathscr{U}$ **piecewise constant in** U_μ

Discretised input signals

Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$

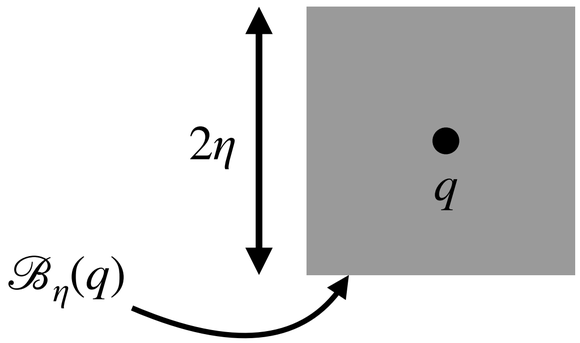
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$

q

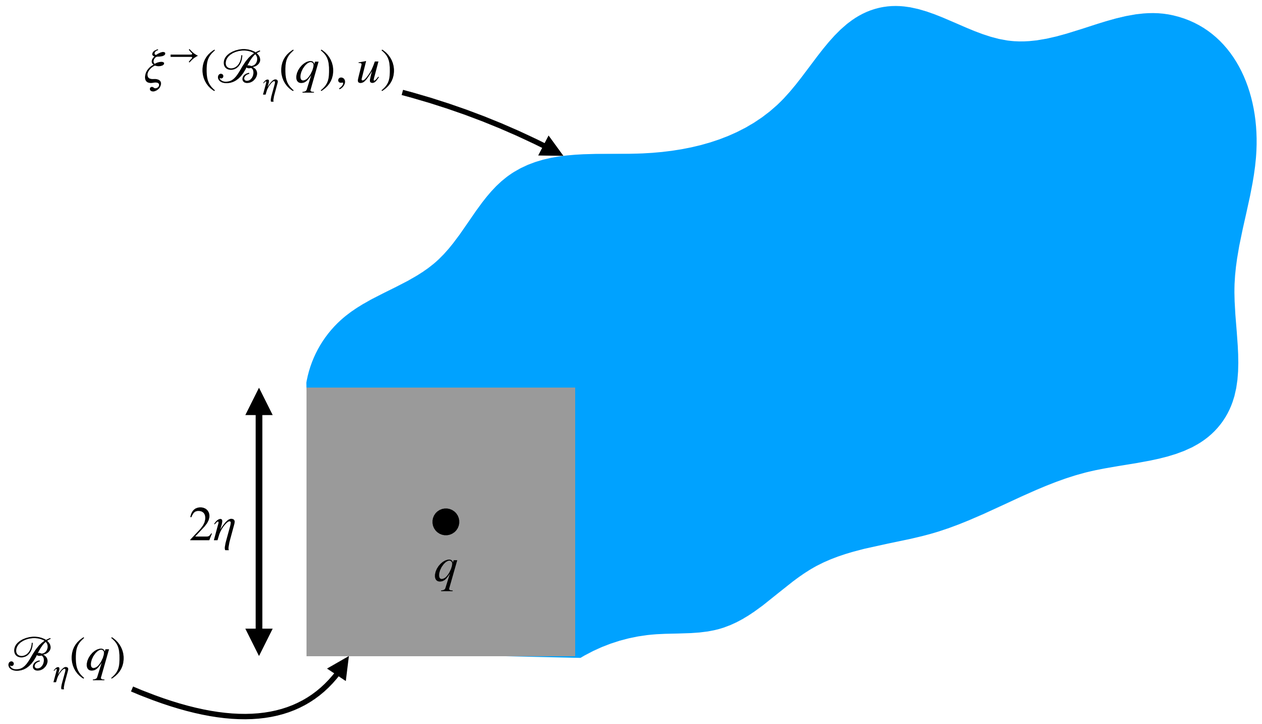
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



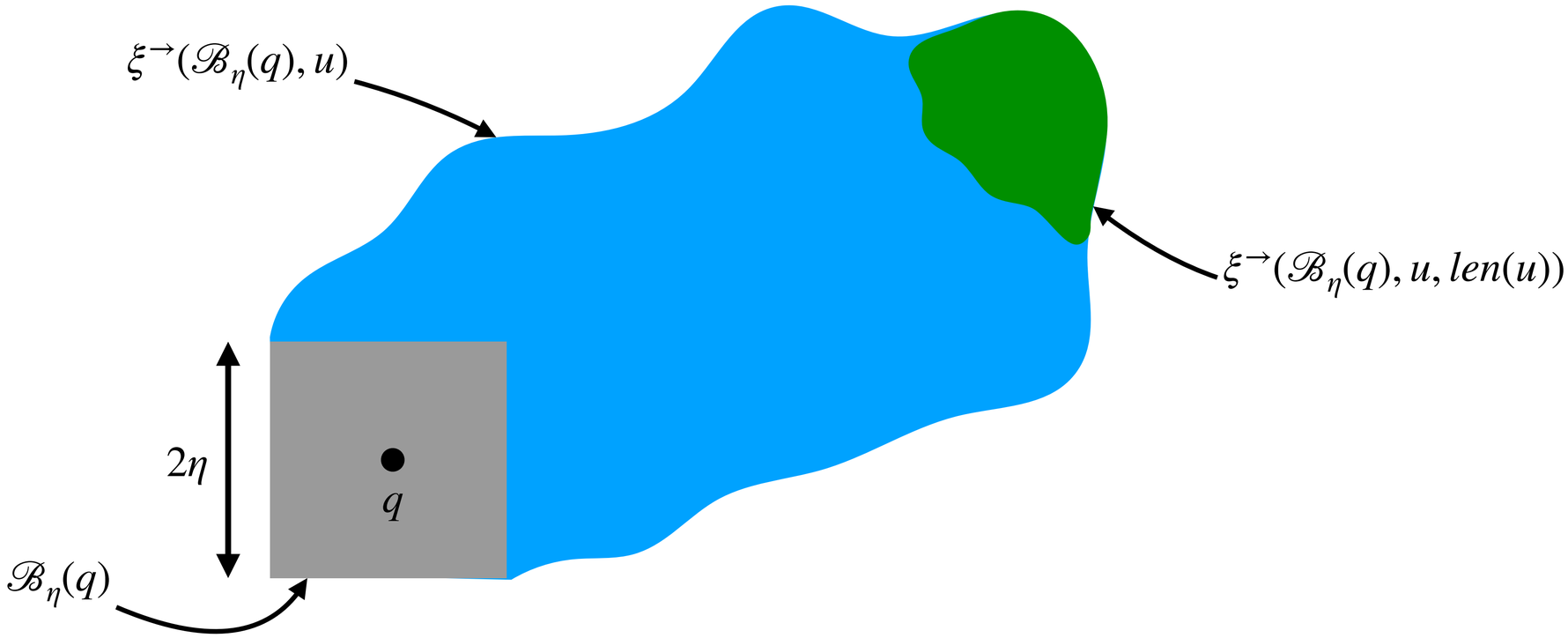
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



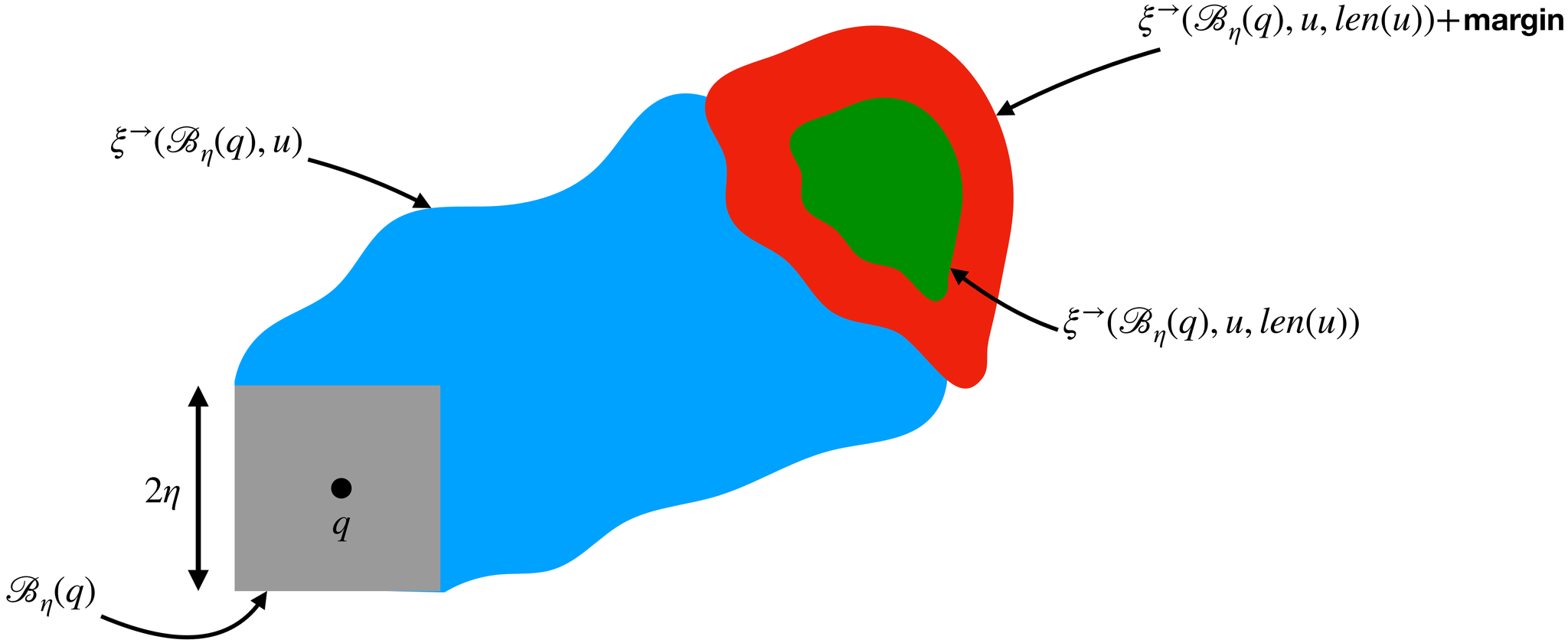
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



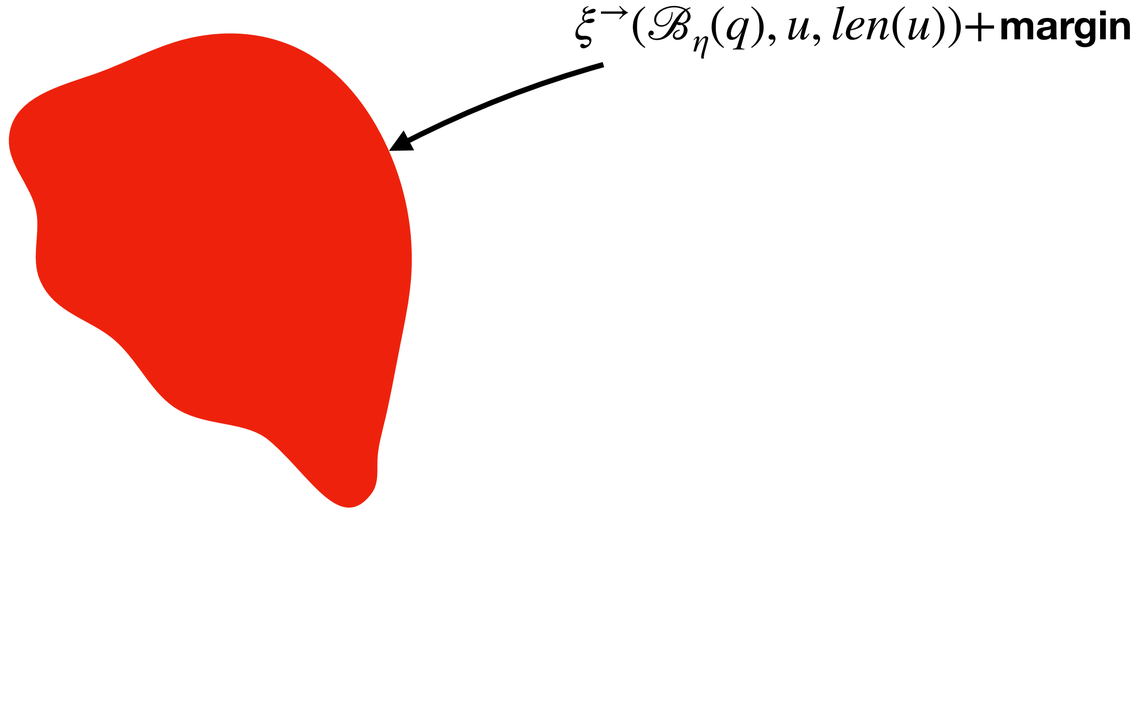
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



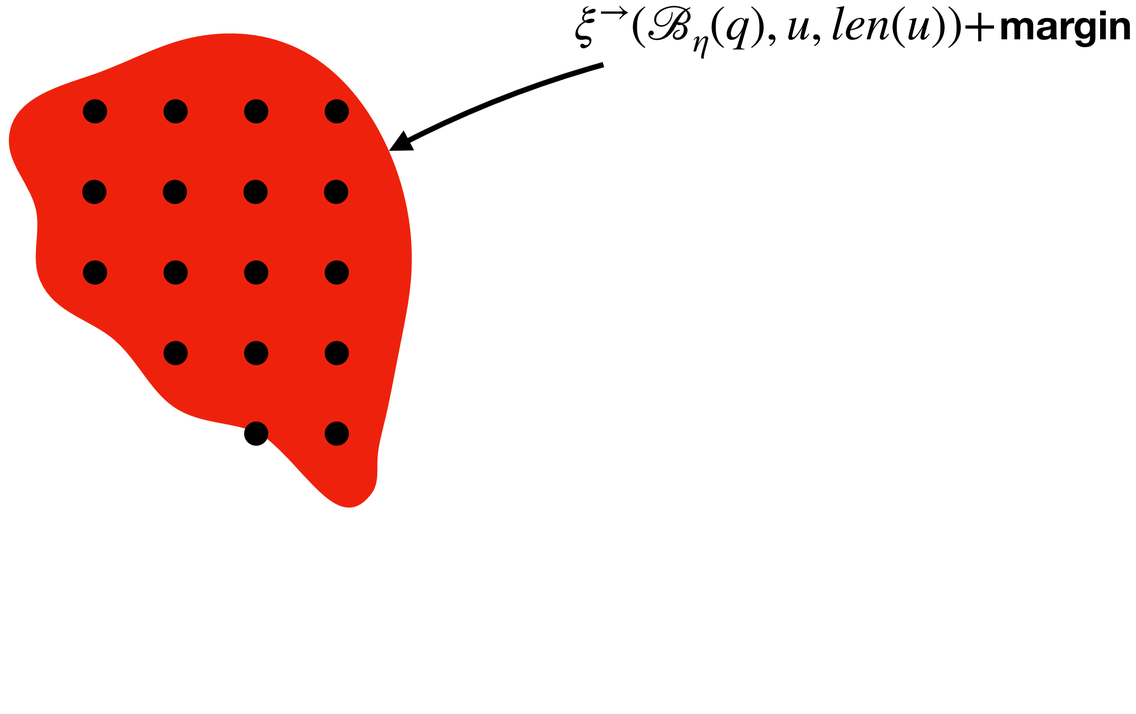
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



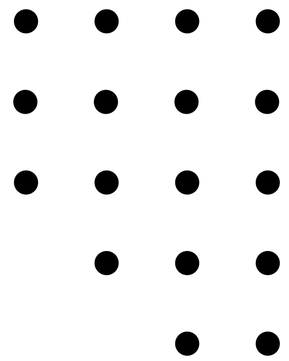
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



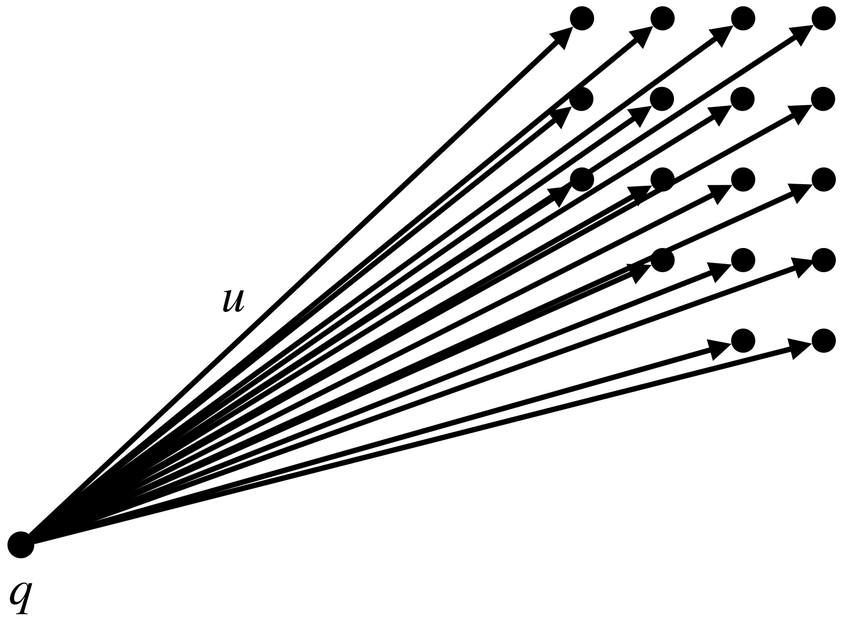
Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



Symbolic model: transitions

Transitions: $\delta \subseteq [X]_\eta \times \mathcal{U}_{\tau,\mu} \times [X]_\eta$



Soundness of the discretisation

Under some assumptions (existence of β)

Assume given a controller $\widetilde{C}$ for the symbolic model.

We can construct a controller C for the system as follows:

- Given a finite run $x_0 u_0 x_1 \ldots x_i$ of the system such that u_i is discretised, we can build a finite run $q_0 u_0 \ldots q_i$ of the symbolic model with q_i “close” to x_i
- Define $C(x_0 u_0 \ldots x_i) = \widetilde{C}(q_0 u_0 \ldots q_i)$

Key argument: alternating simulation

Given:

- two transition models $M_1 = (Q_1, Q_{1,\text{init}}, \mathscr{V}, \Delta_1)$, $M_2 = (Q_2, Q_{2,\text{init}}, \mathscr{V}, \Delta_2)$,
- a function $d : Q_1 \times Q_2 \to \mathbb{R}_{\geq 0}$ (not necessarily a metric),
- a precision $\epsilon \in \mathbb{R}_{\geq 0}$

we say that M_1 alternating ϵ-approximately simulate M_2 if:

- $\forall q_1 \in Q_{1,\text{init}} \,.\, \exists q_2 \in Q_{2,\text{init}} \,.$ such that $d(q_1, q_2) \leq \epsilon$
- $\forall (q_1, q_2) \in Q_1 \times Q_2 \,.$ such that $d(q_1, q_2) \leq \epsilon$,
  and $\forall u \in \mathscr{V} \,.$ such that $\exists (q_2, u, q_2') \in \Delta_2$
  - $\exists (q_1, u, q_1') \in \Delta_1$
  - $\forall (q_1, u, q_1') \in \Delta_1 \,.\, \exists (q_2, u, q_2') \in \Delta_2 \,.$ such that $d(q_1', q_2') \leq \epsilon$

Key argument: alternating simulation

Given:

- two transition models $M_1 = (Q_1, Q_{1,\text{init}}, \mathscr{V}, \Delta_1)$, $M_2 = (Q_2, Q_{2,\text{init}}, \mathscr{V}, \Delta_2)$,
- a function $d : Q_1 \times Q_2 \to \mathbb{R}_{\geq 0}$ (not necessarily a metric),
- a precision $\epsilon \in \mathbb{R}_{\geq 0}$

we say that M_1 alternating ϵ-approximately simulate M_2 if:

- $\forall q_1 \in Q_{1,\text{init}} \,.\, \exists q_2 \in Q_{2,\text{init}} \,.$ such that $d(q_1, q_2) \leq \epsilon$
- $\forall (q_1, q_2) \in Q_1 \times Q_2 \,.$ such that $d(q_1, q_2) \leq \epsilon$,
  and $\forall u \in \mathscr{V} \,.$ such th
  - $\exists (q_1, u, q_1') \in \Delta_1$
  - $\forall (q_1, u, q_1') \in \Delta_1 \,.\, \exists (q_2, u, q_2') \in \Delta_2 \,.$ such that $d(q_1', q_2') \leq \epsilon$

The system has more moves in M_1 than in M_2

Key argument: alternating simulation

Given:

- two transition models $M_1 = (Q_1, Q_{1,\text{init}}, \mathscr{V}, \Delta_1)$, $M_2 = (Q_2, Q_{2,\text{init}}, \mathscr{V}, \Delta_2)$,
- a function $d : Q_1 \times Q_2 \to \mathbb{R}_{\geq 0}$ (not necessarily a metric),
- a precision $\epsilon \in \mathbb{R}_{\geq 0}$

we say that M_1 alternating ϵ-approximately simulate M_2 if:

- $\forall q_1 \in Q_{1,\text{init}} \,.\, \exists q_2 \in Q_{2,\text{init}} \,.$ such that $d(q_1, q_2) \leq \epsilon$
- $\forall (q_1, q_2) \in Q_1 \times Q_2 \,.$ such that $d(q_1, q_2) \leq \epsilon$,
  and $\forall u \in \mathscr{V} \,.$ such th
  - $\exists (q_1, u, q_1') \in \Delta_1$
  - $\forall (q_1, u, q_1') \in \Delta_1 \,.\, \exists (q_2, u, q_2') \in \Delta_2 \,.$ such that $d(q_1', q_2') \leq \epsilon$

The system has more moves in M_1 than in M_2

The environment has more moves in M_2 than in M_1

Key argument: alternating simulation

Given:

- two transition models $M_1 = (Q_1, Q_{1,\text{init}}, \mathscr{V}, \Delta_1)$, $M_2 = (Q_2, Q_{2,\text{init}}, \mathscr{V}, \Delta_2)$,
- a function $d : Q_1 \times Q_2 \to \mathbb{R}_{\geq 0}$ (not necessarily a metric),
- a precision $\epsilon \in \mathbb{R}_{\geq 0}$

we say that M_1 alternating ϵ-approxima

- $\forall q_1 \in Q_{1,\text{init}} \,.\, \exists q_2 \in Q_{2,\text{init}} \,.$ such that $d(q_1, q_2) \leq \epsilon$
- $\forall (q_1, q_2) \in Q_1 \times Q_2 \,.$ such that $d(q_1, q_2) \leq \epsilon$,
  and $\forall u \in \mathscr{V} \,.$ such th
  - $\exists (q_1, u, q_1') \in \Delta_1$
  - $\forall (q_1, u, q_1') \in \Delta_1 \,.\, \exists (q_2, u, q_2') \in \Delta_2 \,.$ such that $d(q_1', q_2') \leq \epsilon$

Any controller in M_2 can be translated into a controller in M_1

The system has more moves in M_1 than in M_2

The environment has more moves in M_2 than in M_1

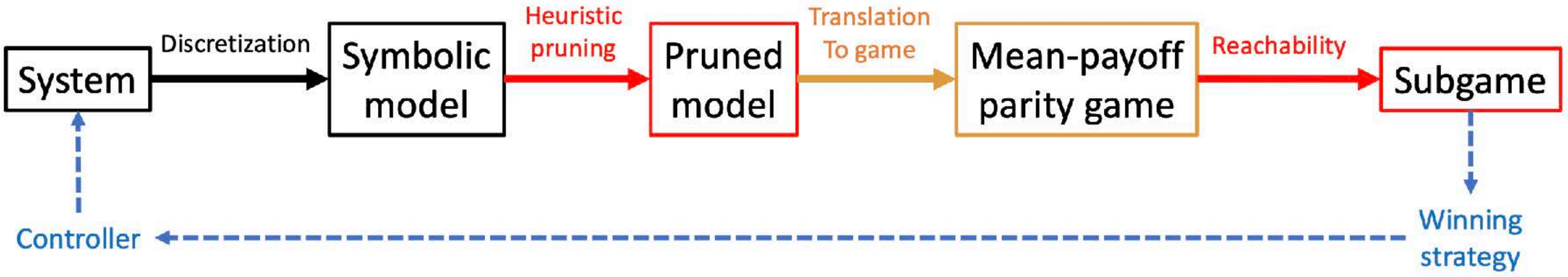
How to use that?

How to use that?

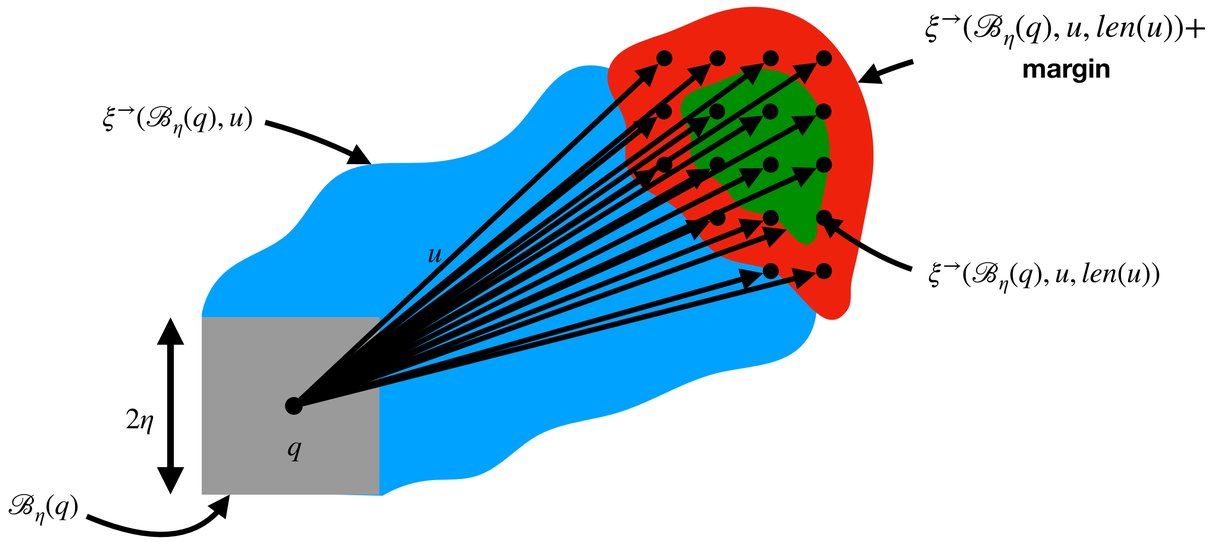
Theorem:

If the function β exists, then by taking $d(x, q) = \|x - q\|$,
the system Σ alternating η-approximately simulate the symbolic model.

Overview of the method



Keeping track of the atomic propositions



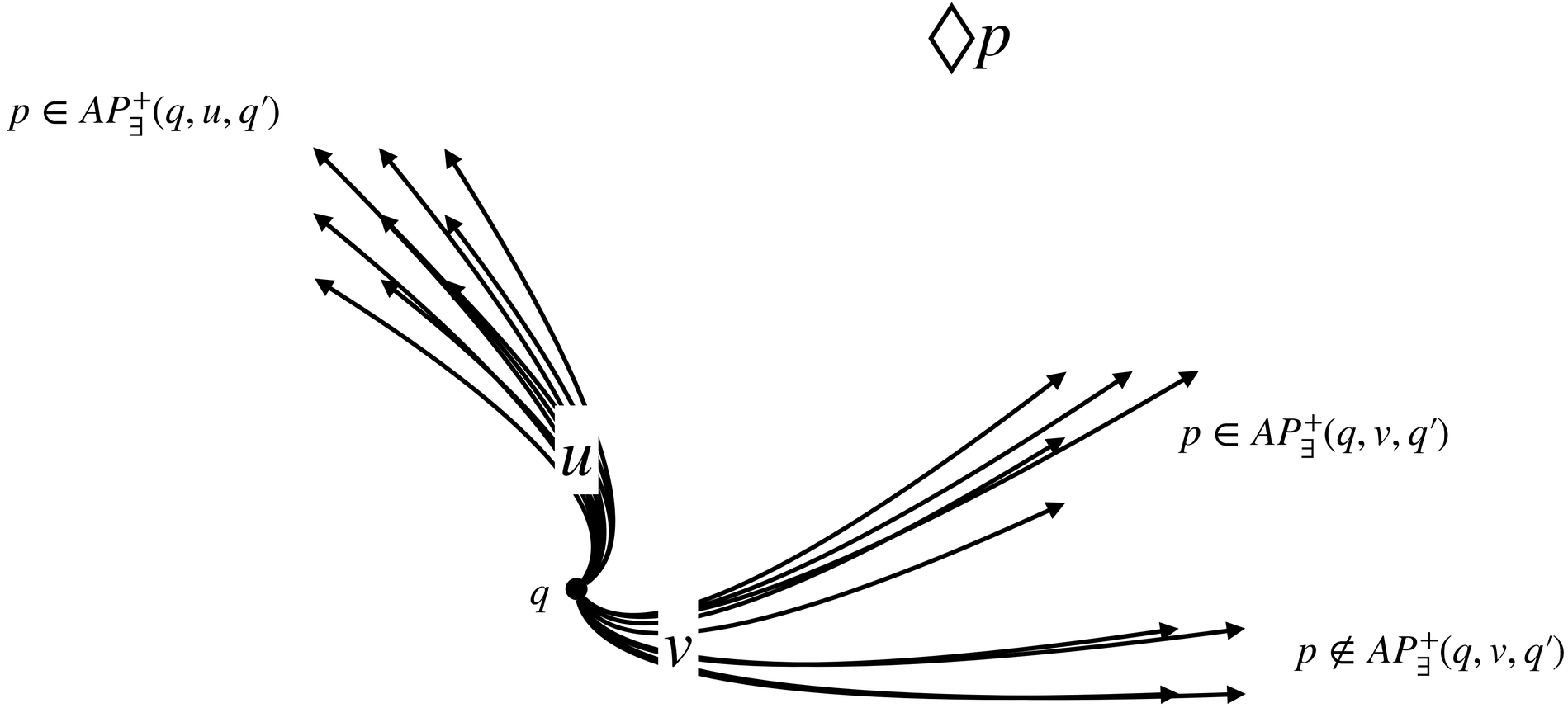
$$AP^{+}_{\forall}(q,u,q') \subseteq \Big\{ p \mid \forall \sigma \in \textbf{Traj}(xux') \,. \; (x \sim q \land x' \sim q') \implies \forall t \,.\, p \in P(\sigma(t)) \Big\}$$

$$AP^{-}_{\forall}(q,u,q') \subseteq \Big\{ p \mid \forall \sigma \in \textbf{Traj}(xux') \,. \; (x \sim q \land x' \sim q') \implies \forall t \,.\, p \notin P(\sigma(t)) \Big\}$$

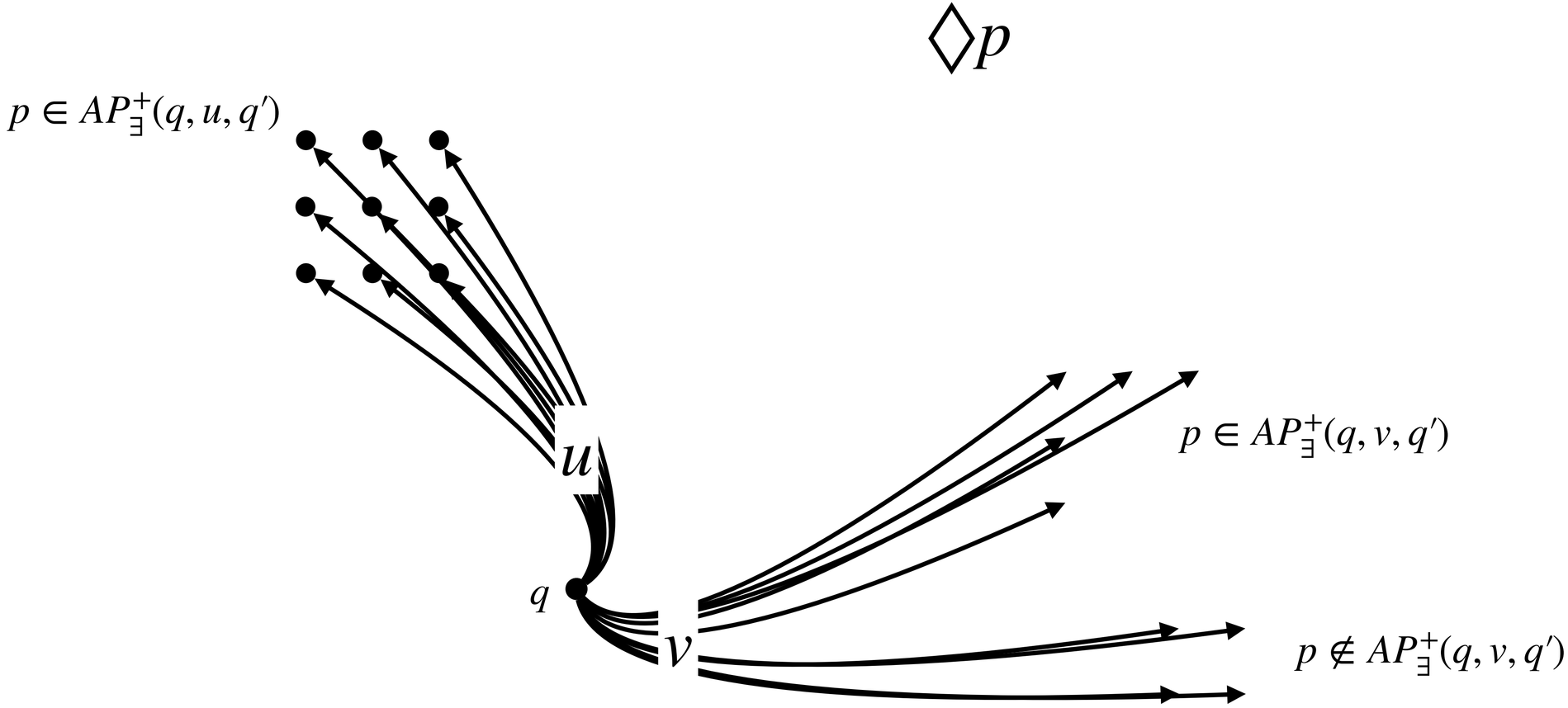
$$AP^{+}_{\exists}(q,u,q') \subseteq \Big\{ p \mid \forall \sigma \in \textbf{Traj}(xux') \,. \; (x \sim q \land x' \sim q') \implies \exists t \,.\, p \in P(\sigma(t)) \Big\}$$

$$AP^{-}_{\exists}(q,u,q') \subseteq \Big\{ p \mid \forall \sigma \in \textbf{Traj}(xux') \,. \; (x \sim q \land x' \sim q') \implies \exists t \,.\, p \notin P(\sigma(t)) \Big\}$$

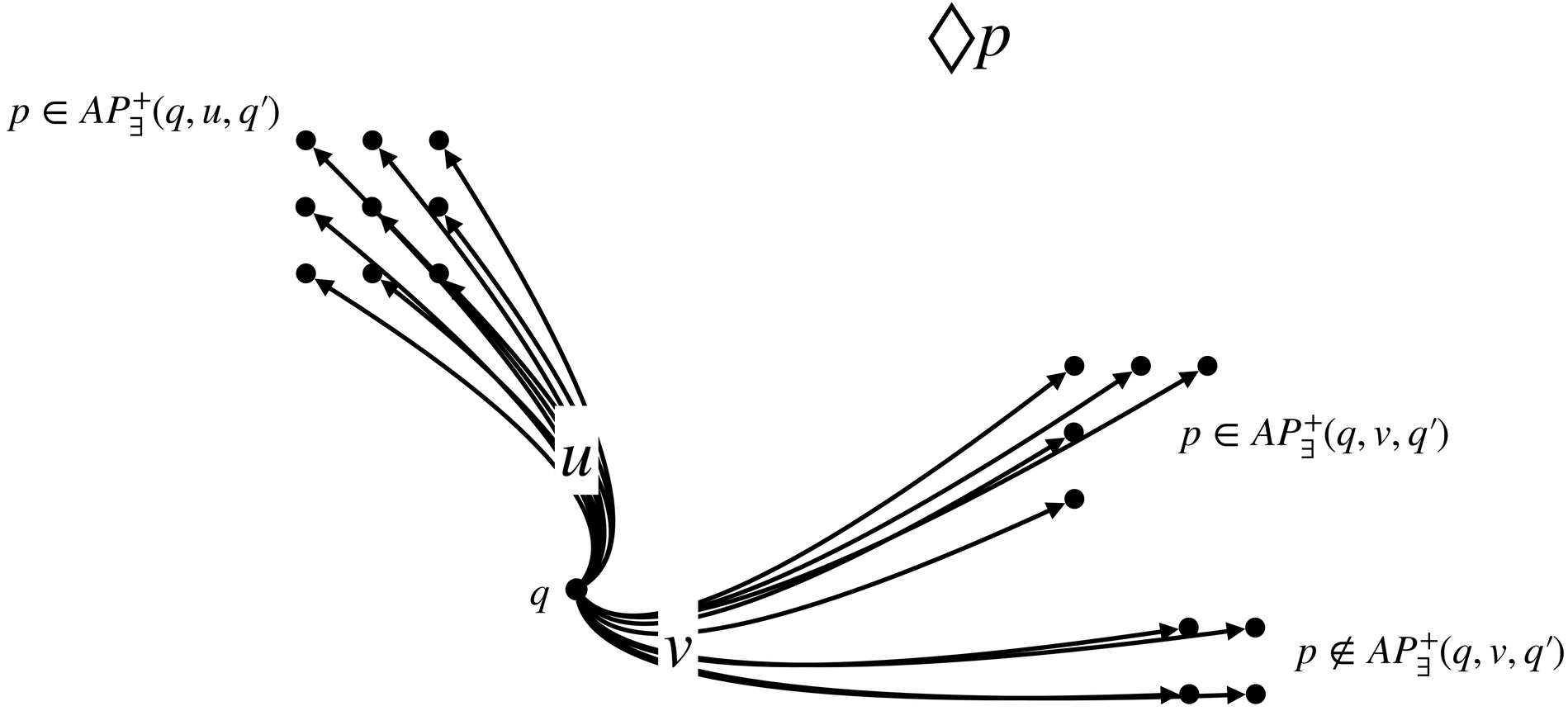
Pruning the symbolic model



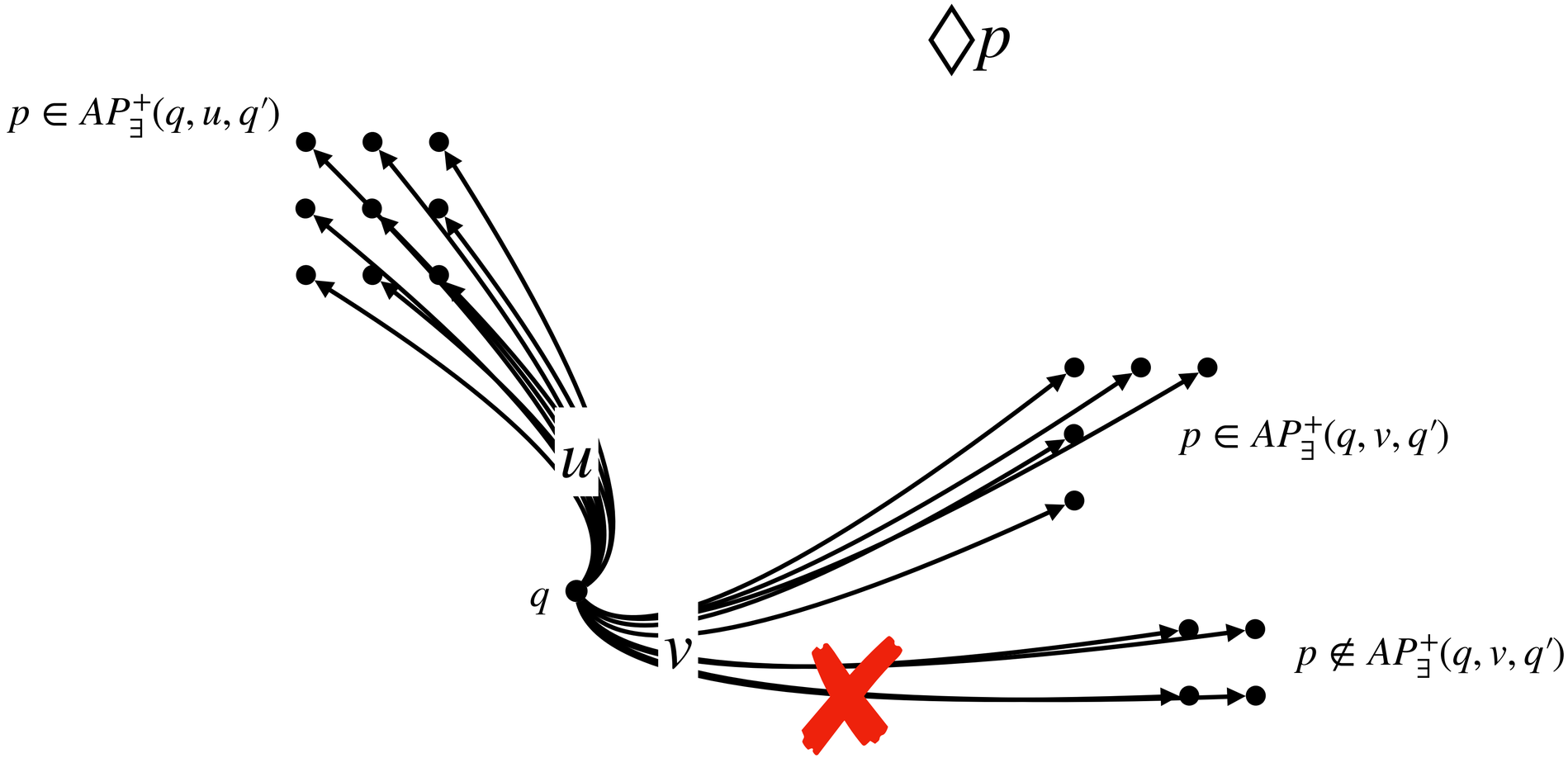
Pruning the symbolic model



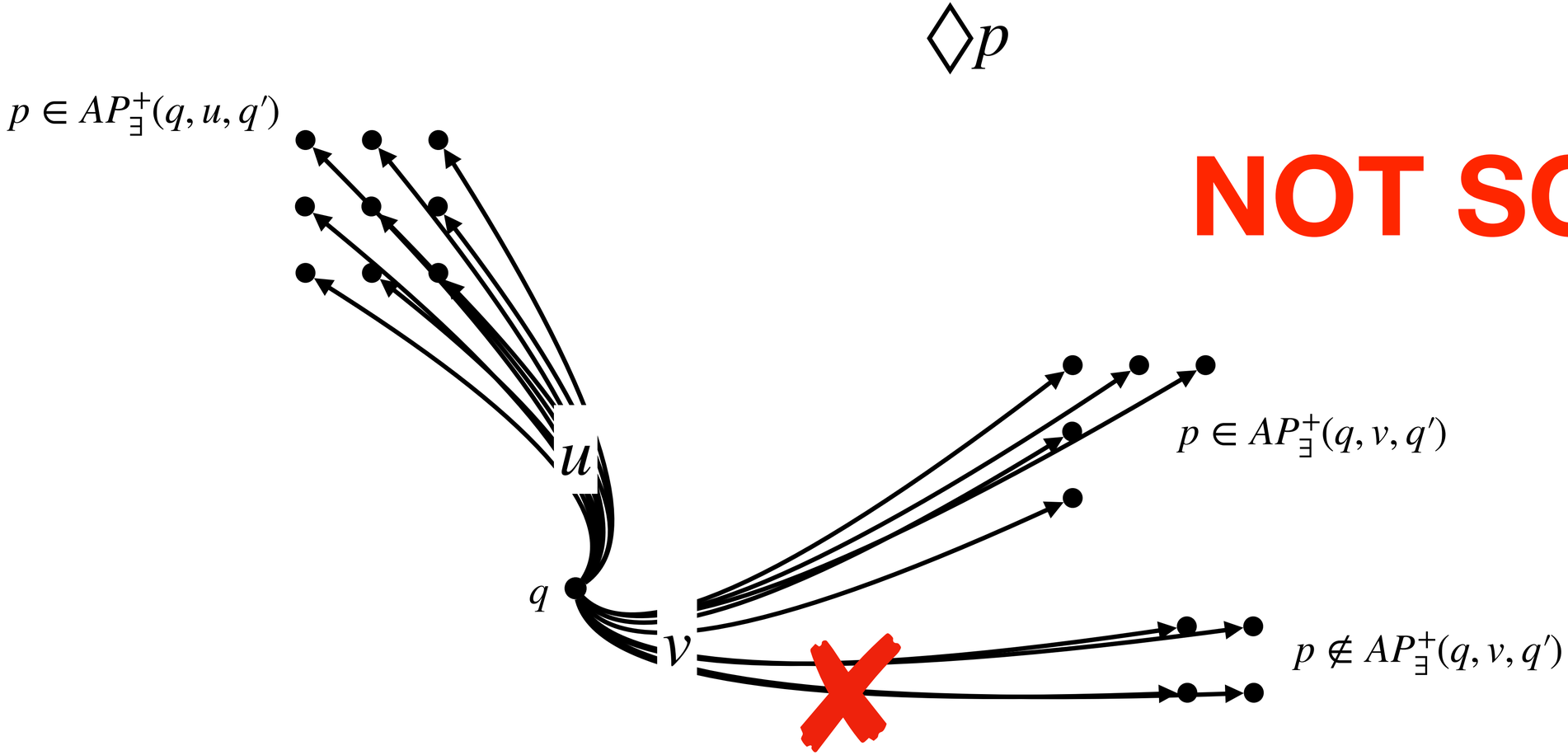
Pruning the symbolic model



Pruning the symbolic model

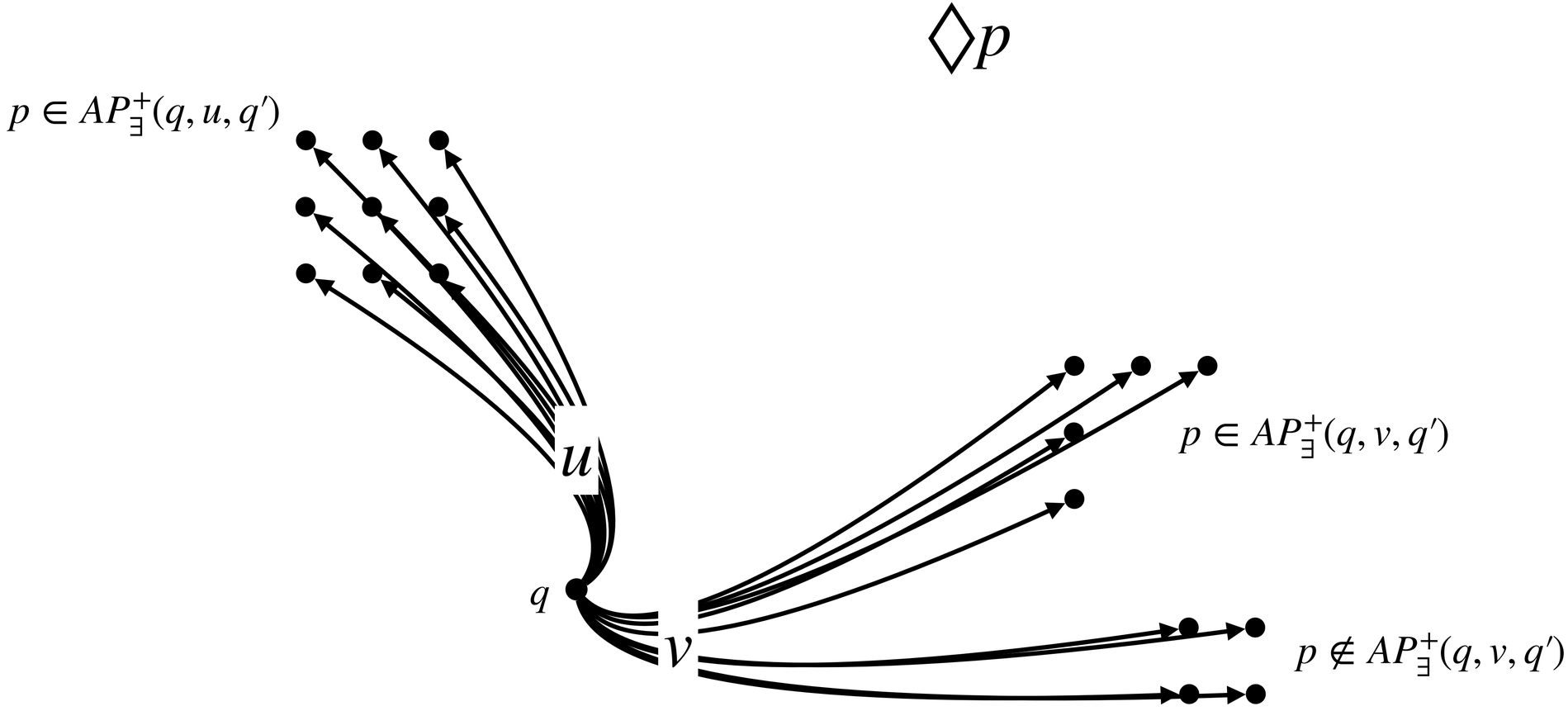


Pruning the symbolic model

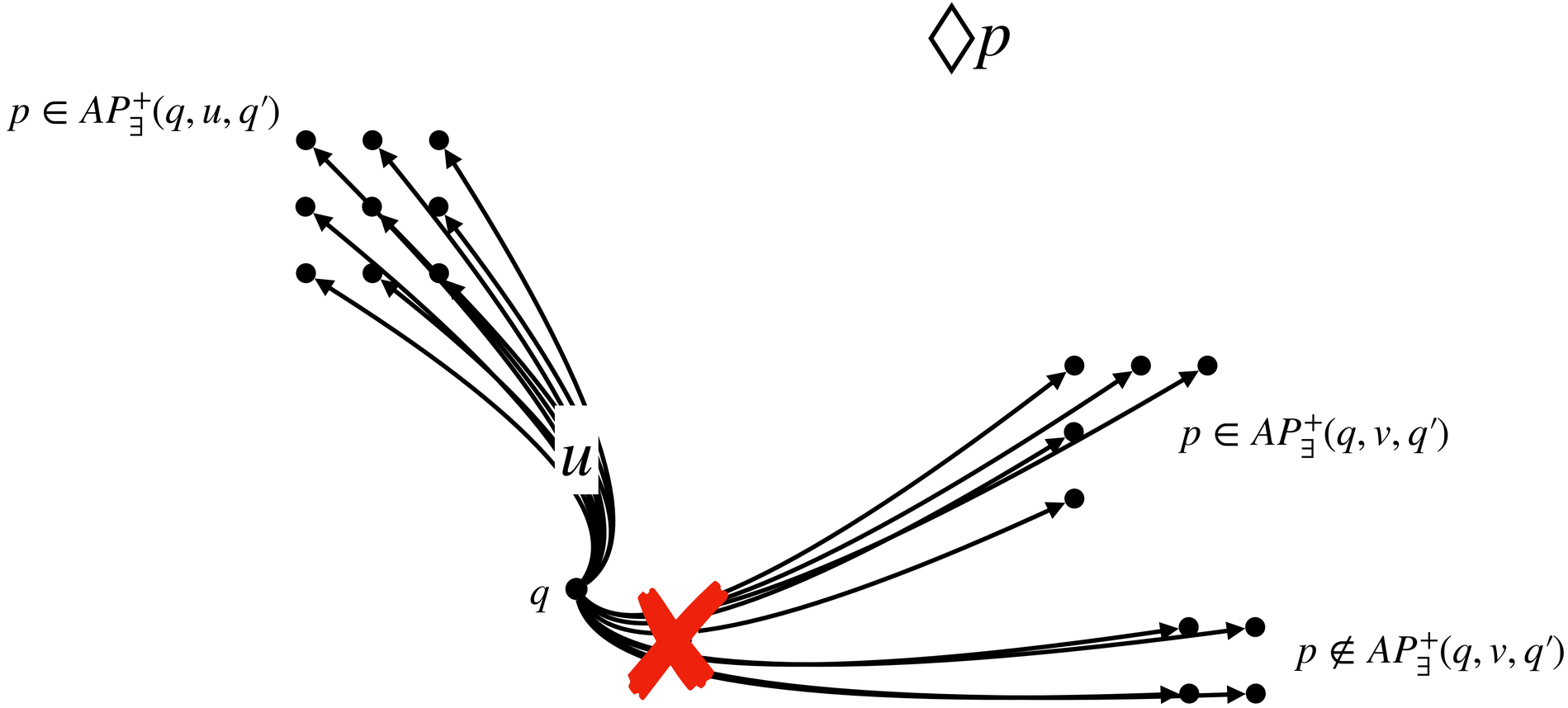


NOT SOUND

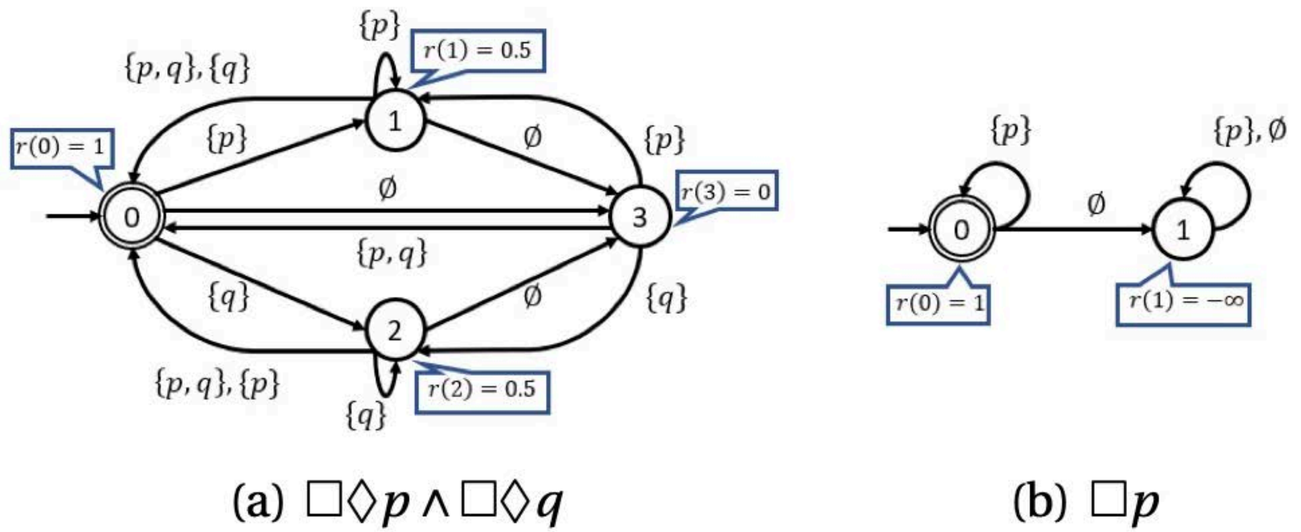
Pruning the symbolic model



Pruning the symbolic model



In general



Idea:

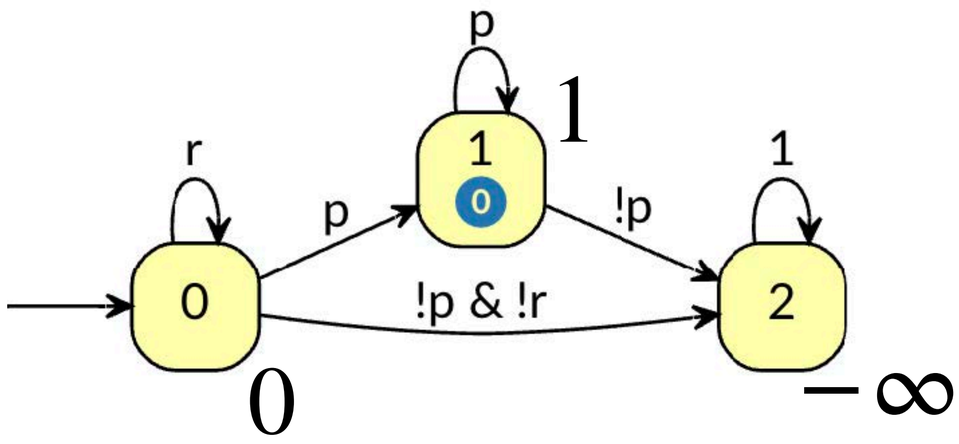
- we translate Φ into a Büchi automaton with reward
- we cut some paths that do not maximise the reward

Getting a Büchi automaton (I)

Idea:

- Given a LTL formula, there are tools for computing an "equivalent" automaton
- Some can already give rewards

$r \mathcal{U} \square p$

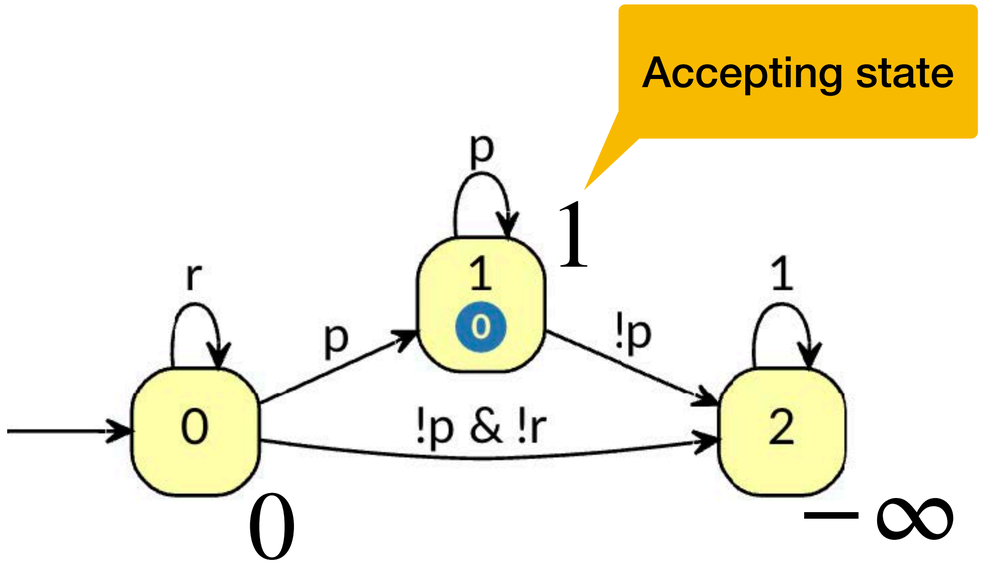


Getting a Büchi automaton (I)

Idea:

- Given a LTL formula, there are tools for computing an "equivalent" automaton
- Some can already give rewards

$r \mathcal{U} \square p$

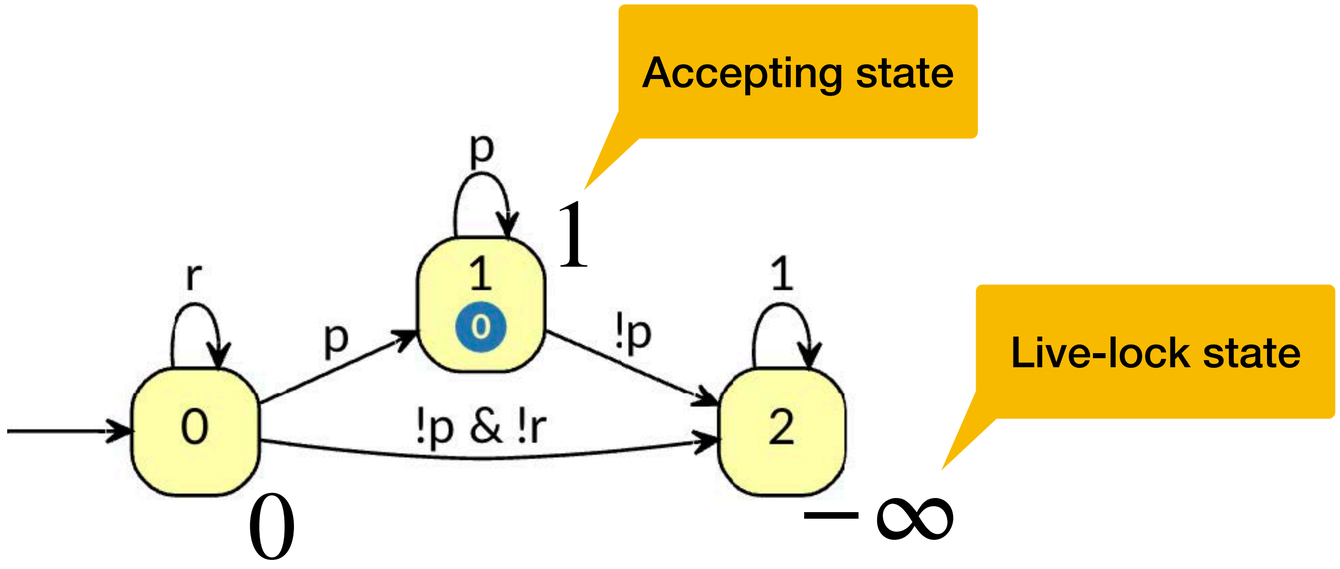


Getting a Büchi automaton (I)

Idea:

- Given a LTL formula, there are tools for computing an "equivalent" automaton
- Some can already give rewards

$r \mathcal{U} \square p$

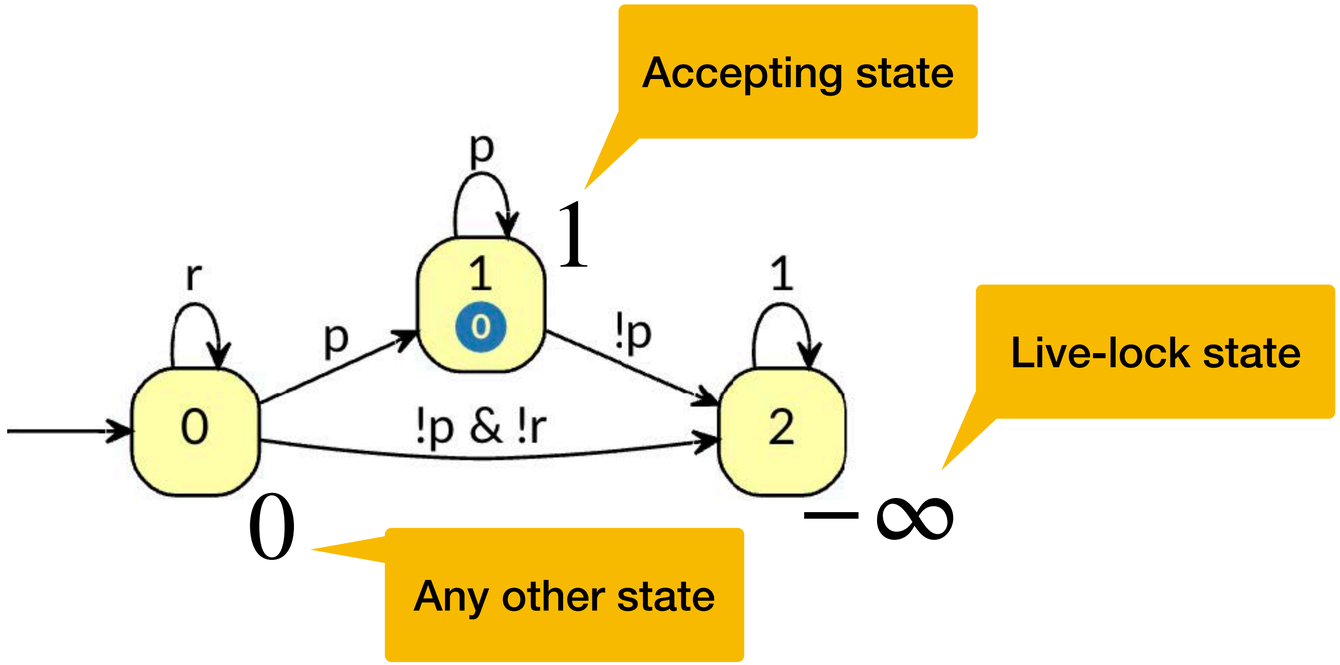


Getting a Büchi automaton (I)

Idea:

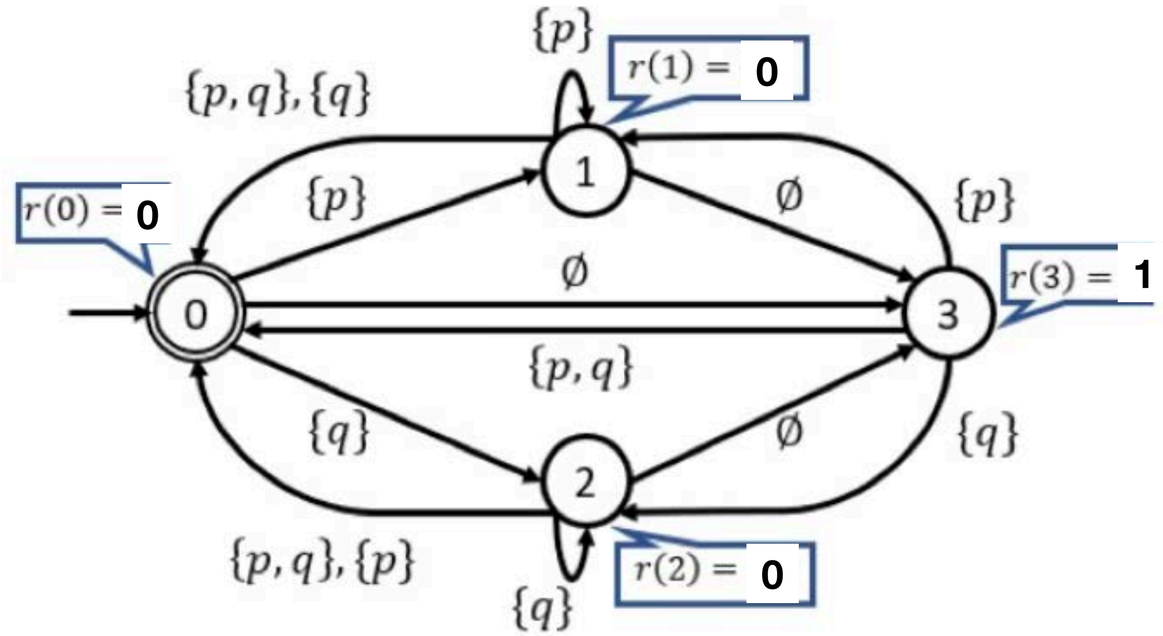
- Given a LTL formula, there are tools for computing an "equivalent" automaton
- Some can already give rewards

$r \mathcal{U} \square p$



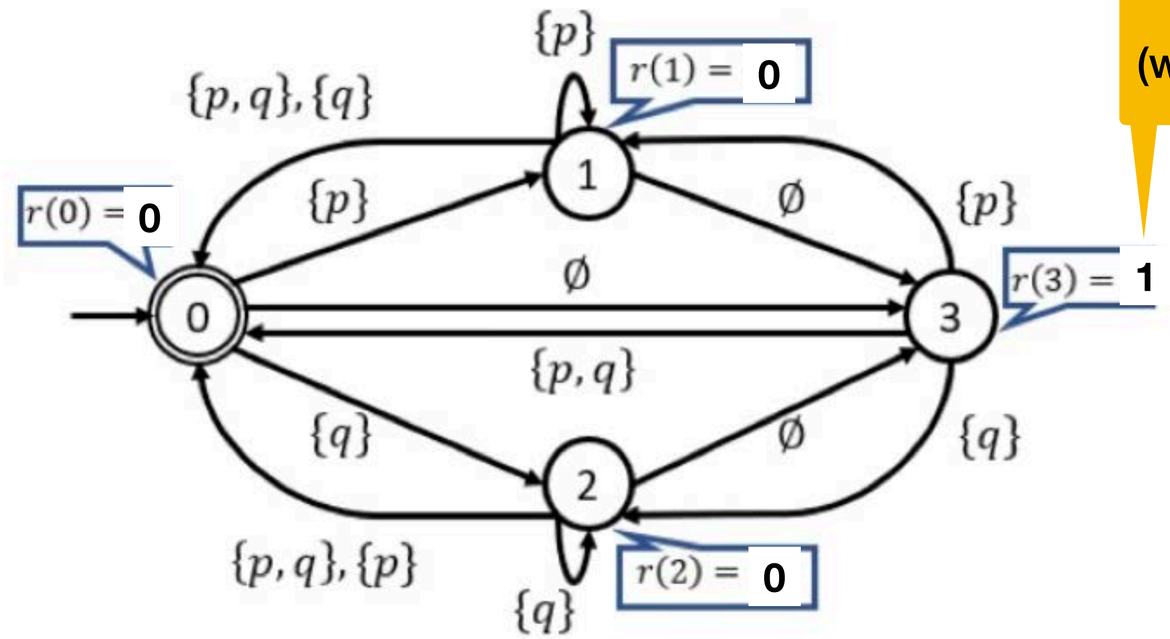
Getting a Büchi automaton (II)

$(\Box\Diamond p) \wedge (\Box\Diamond q)$



Getting a Büchi automaton (II)

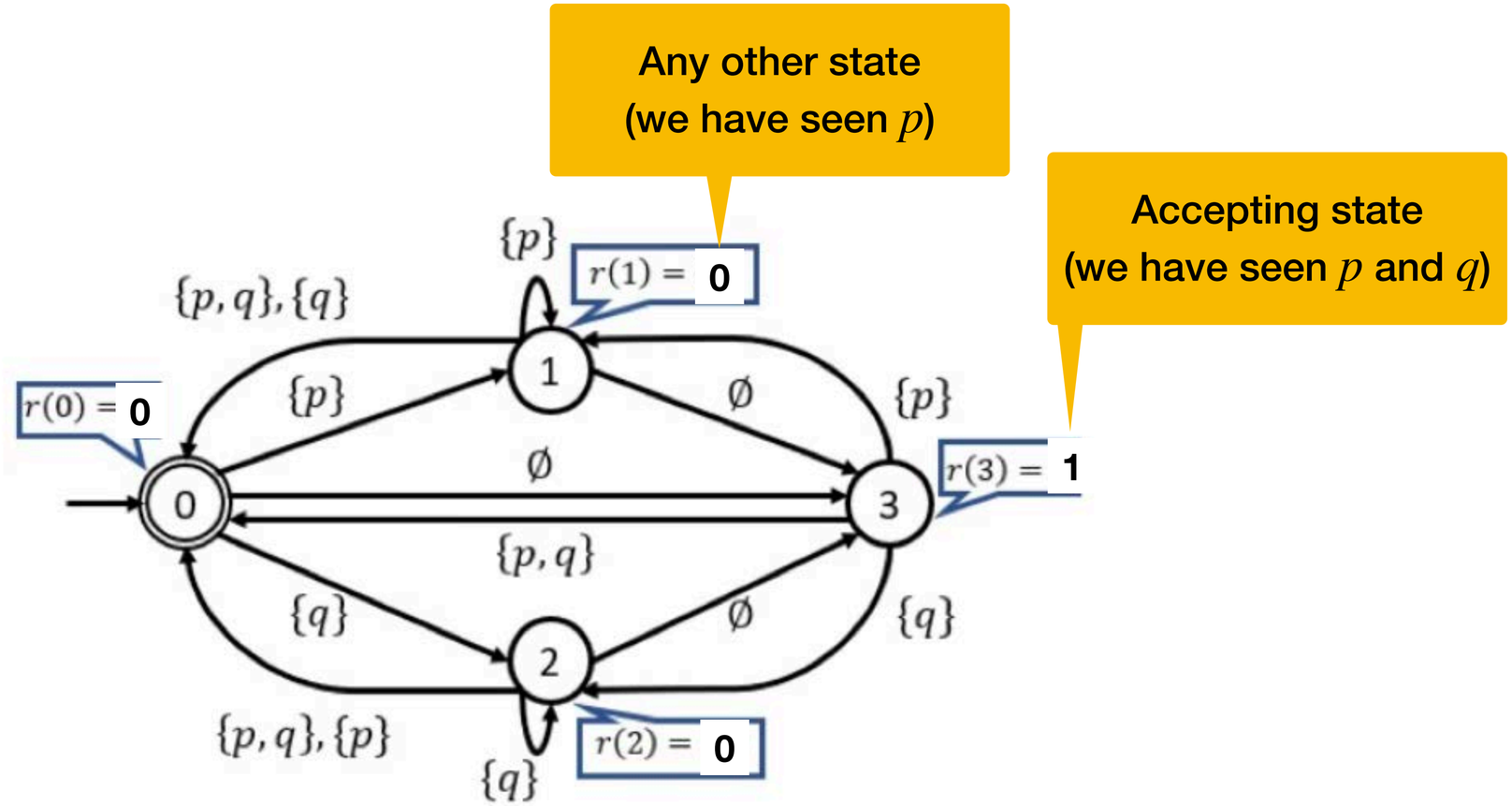
$$(\Box \Diamond p) \wedge (\Box \Diamond q)$$



Accepting state
(we have seen p and q)

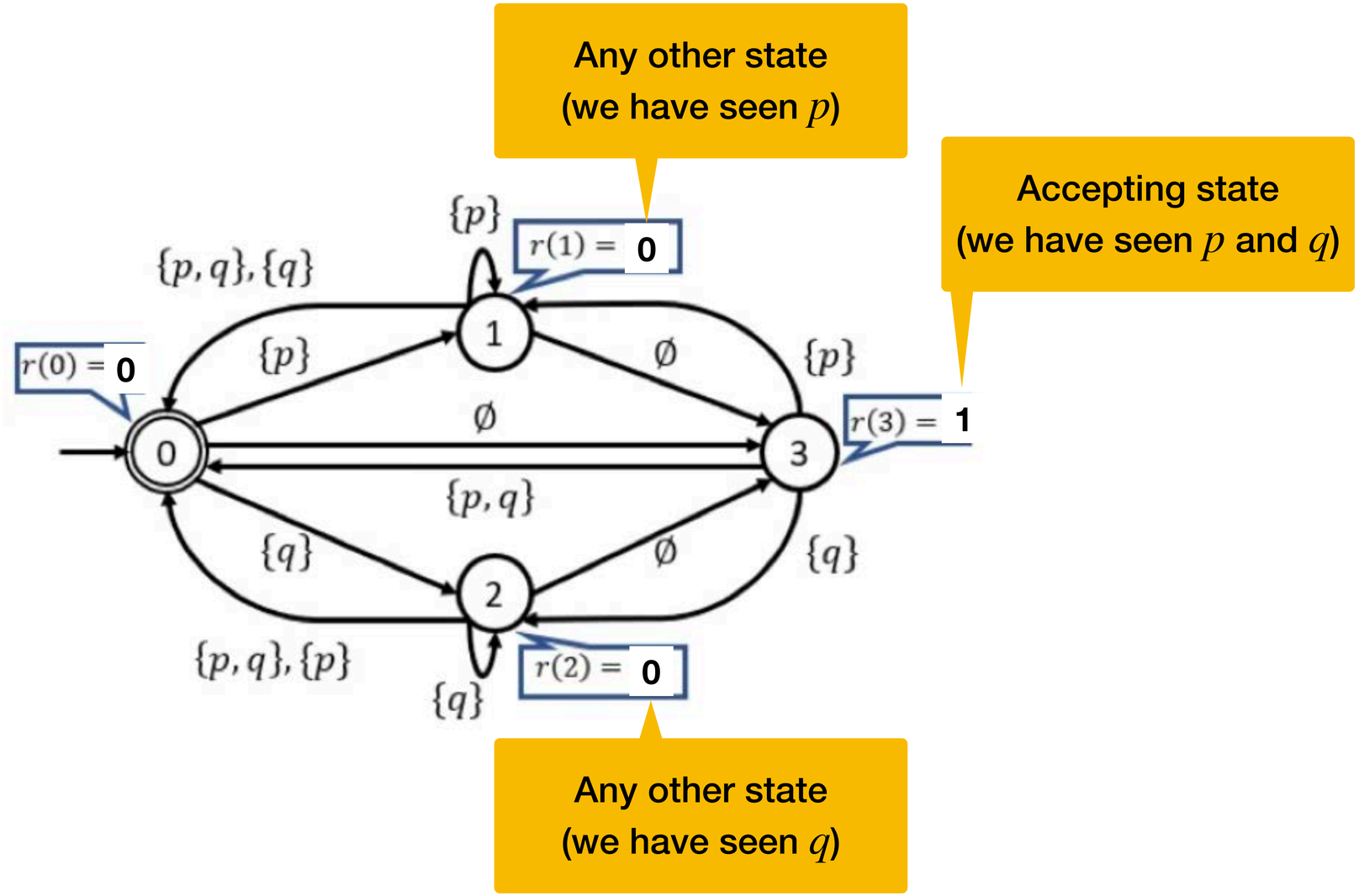
Getting a Büchi automaton (II)

$$(\square \lozenge p) \wedge (\square \lozenge q)$$



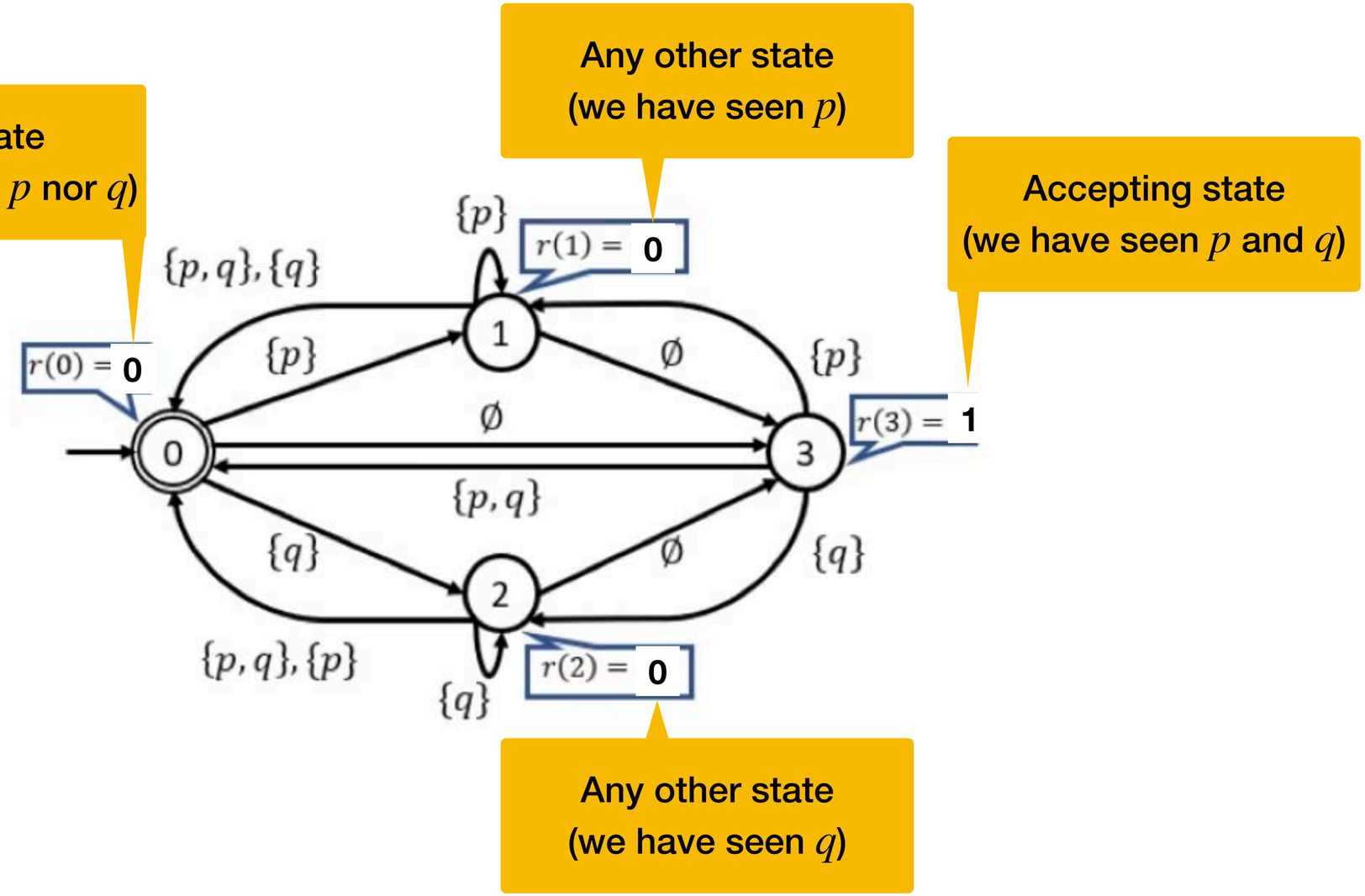
Getting a Büchi automaton (II)

$$(\square \lozenge p) \wedge (\square \lozenge q)$$



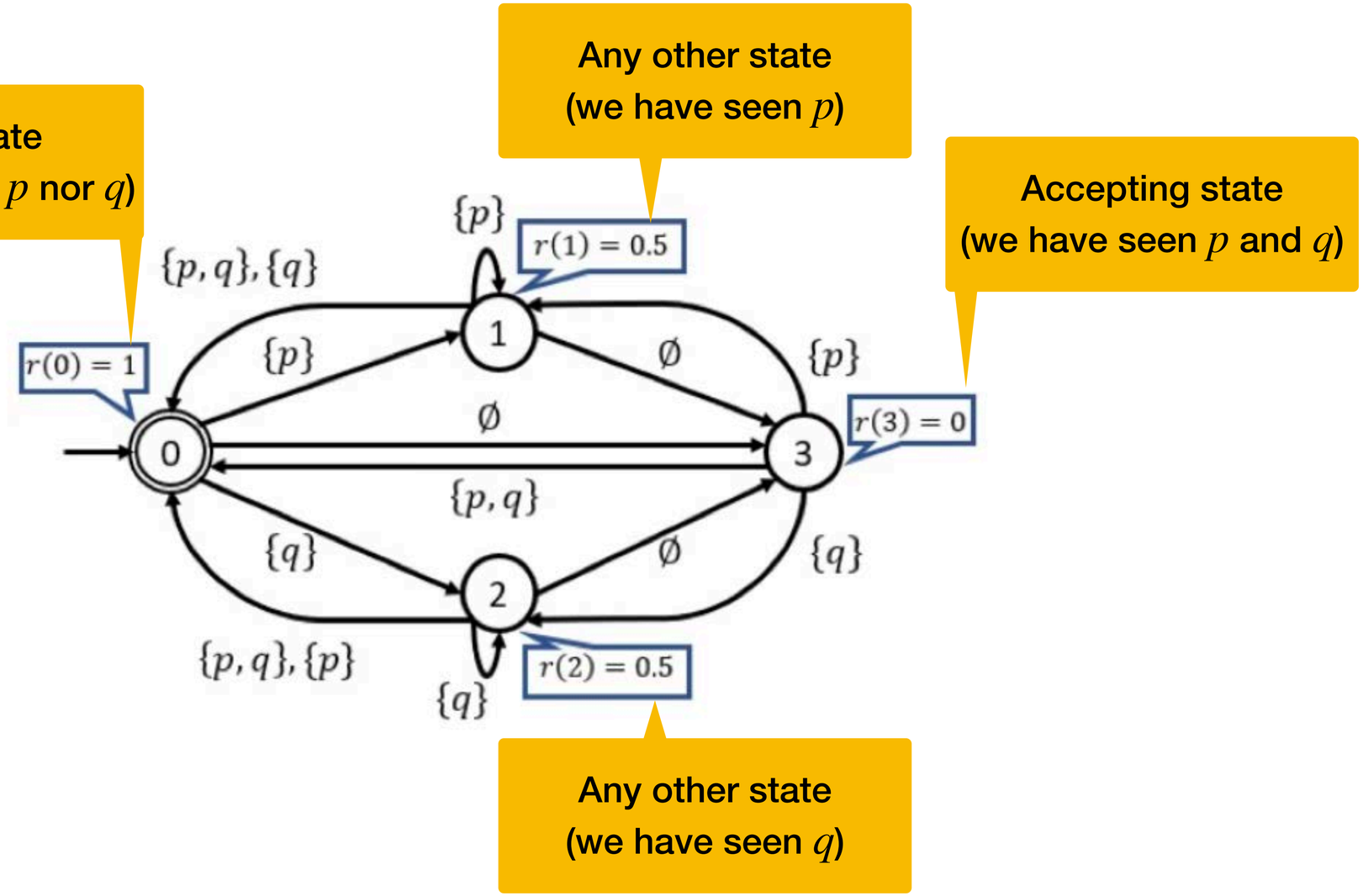
Getting a Büchi automaton (II)

$$(\square \lozenge p) \wedge (\square \lozenge q)$$

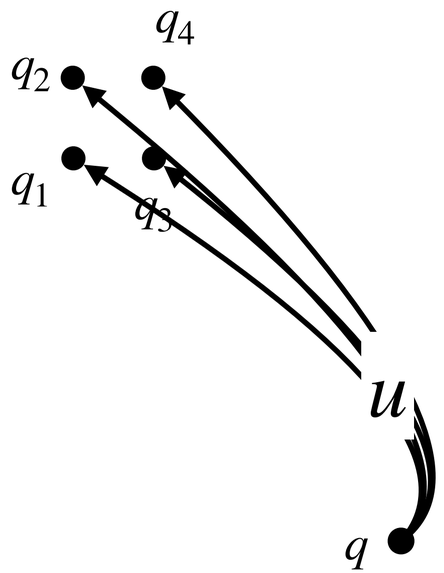


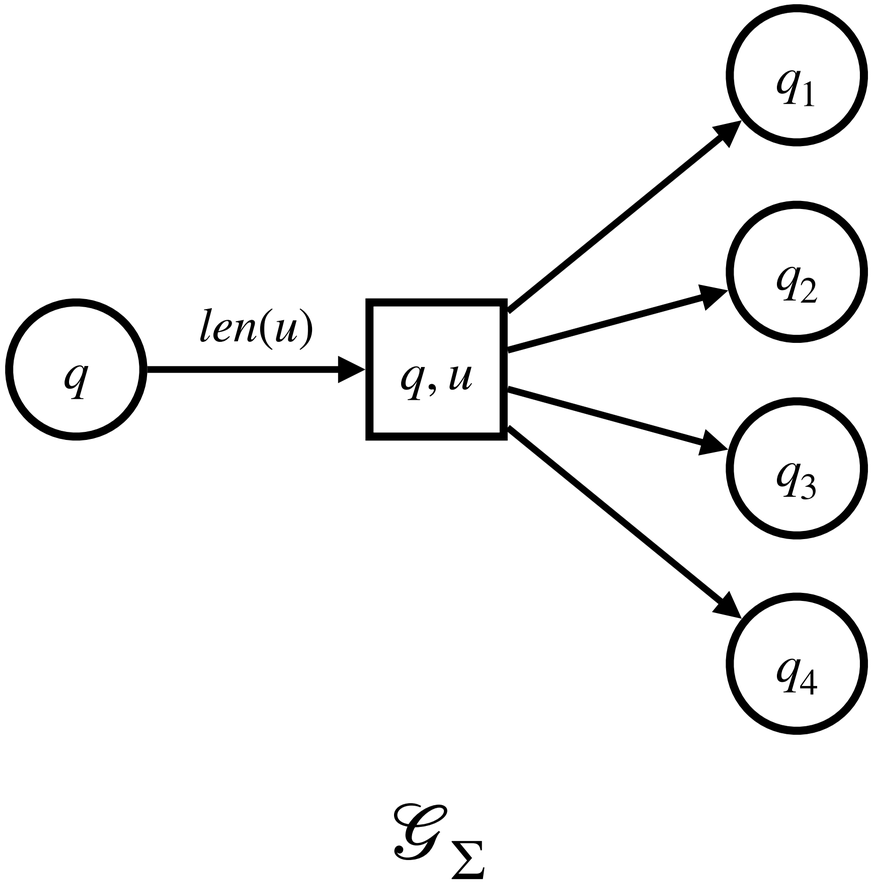
Getting a Büchi automaton (II)

$$(\square \lozenge p) \wedge (\square \lozenge q)$$



Mean payoff games from symbolic model





$\mathcal{G}_\Sigma$

Mean payoff games, in general

$$\mathcal{G} = \left(X_s, X_e, Tr_s, Tr_e, w\right)$$

Where:

- X_s finite set (system states)
- X_e finite set (environment states)
- $Tr_s \subseteq X_s \times X_e$ (system transitions)
- $Tr_e \subseteq X_e \times X_s$ (environment transitions)
- $w : Tr_s \to \mathbb{R}_{\geq 0}$ (weight function)

Strategies, winning condition

Strategy for system: $S_s : (X_s \times X_e)^* \times X_s \to X_e$ such that

If $S_s(s_1 e_1 \dots s_n) = e_n$ then $(s_n, e_n) \in Tr_s$

Strategy for environment: $S_e : (X_s \times X_e)^+ \to X_s$ such that

If $S_e(s_1 e_1 \dots s_n e_n) = s_{n+1}$ then $(e_n, s_{n+1}) \in Tr_e$

Two strategies generate a unique run ρ_s for every state $s \in X_s$

System wins the game at s if it has a strategy S_s such that for all strategy S_e of the environment, ($\rho_s = s_1 e_1 \dots$):

$$\liminf_{h \to \infty} \frac{1}{h} \sum_{i=1}^{h} w(s_i, e_i) \geq \nu$$

Calculability

The following is decidable

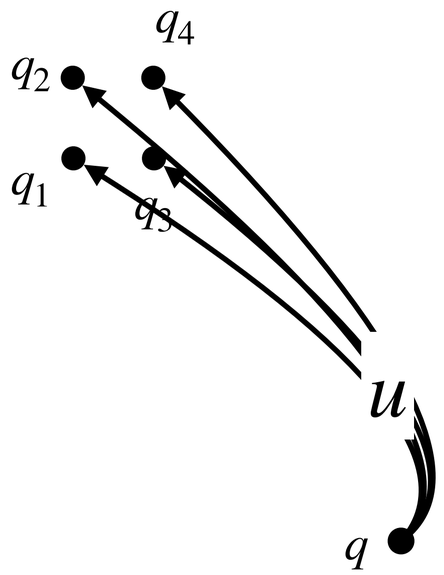
Data:

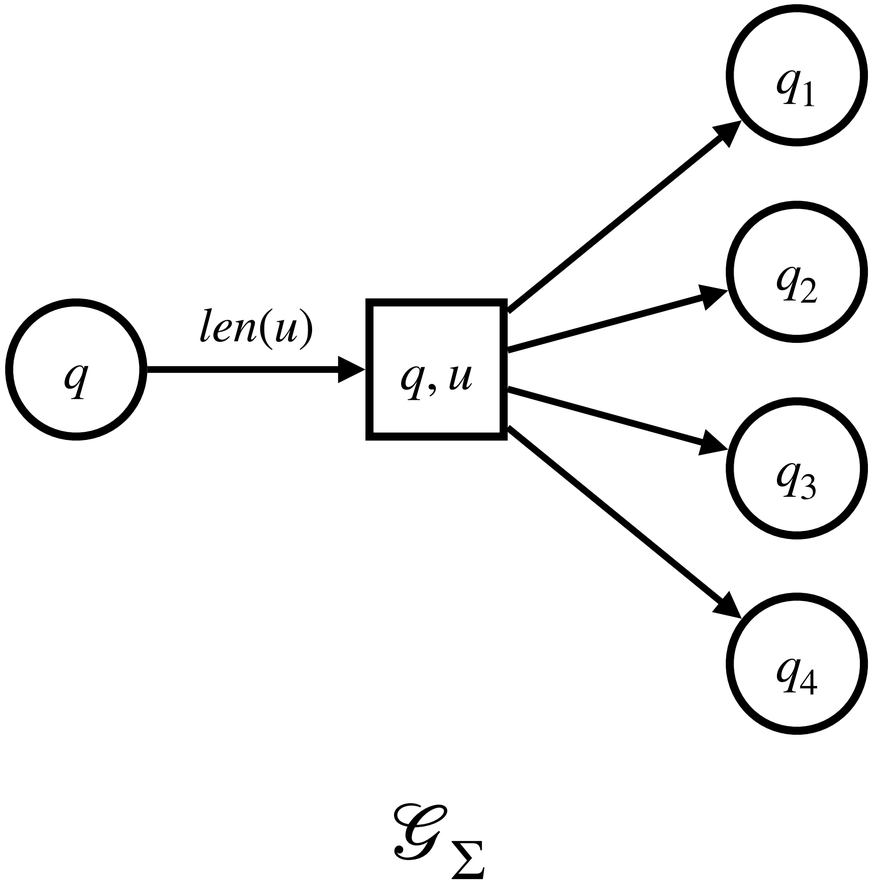
- Mean payoff game $\mathscr{G}$
- Threshold ν

Output:

- Sets of winning states for system and environment
- Strategies

Inserting the specification, diamond case

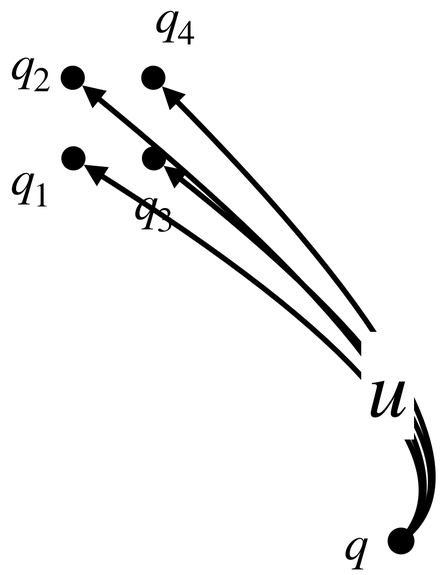


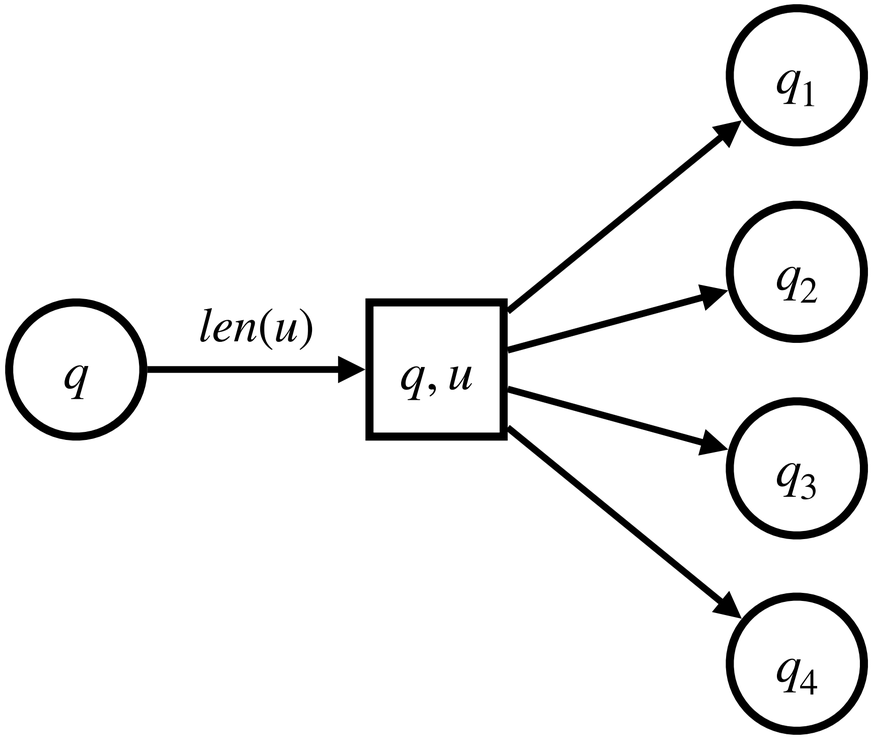


$\mathcal{G}_\Sigma$

Inserting the specification, diamond case

$\Diamond p$

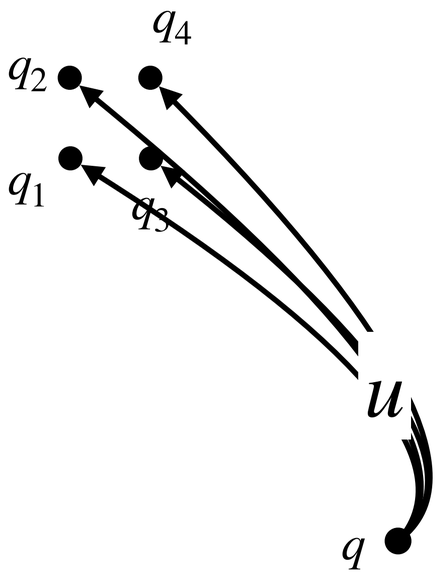


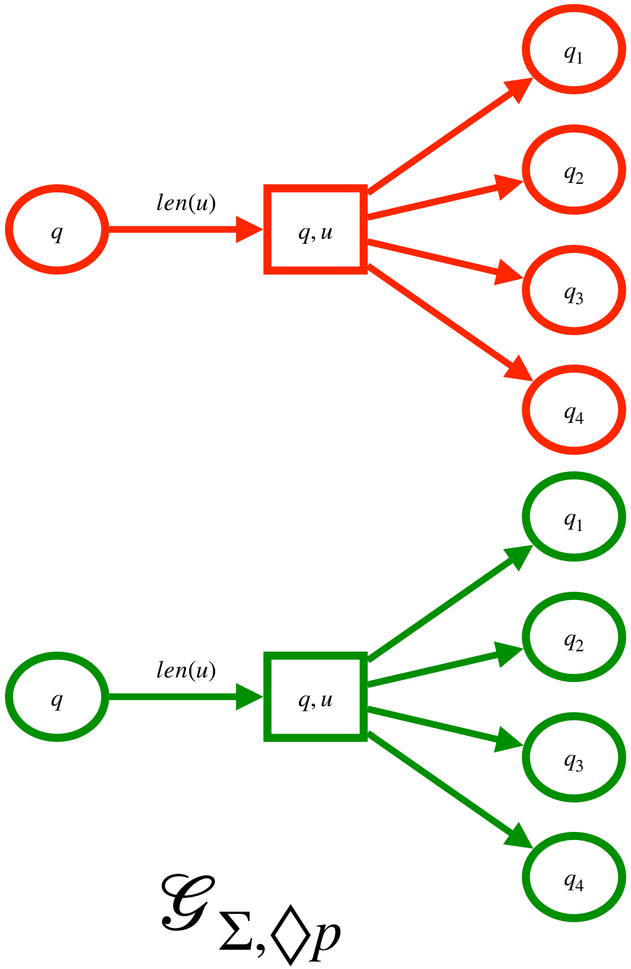


$\mathcal{G}_\Sigma$

Inserting the specification, diamond case

$\Diamond p$

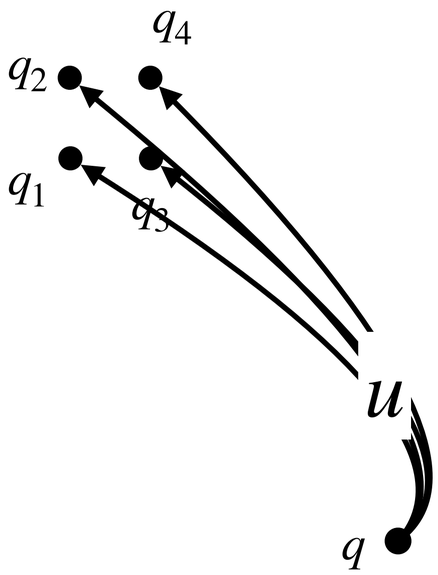


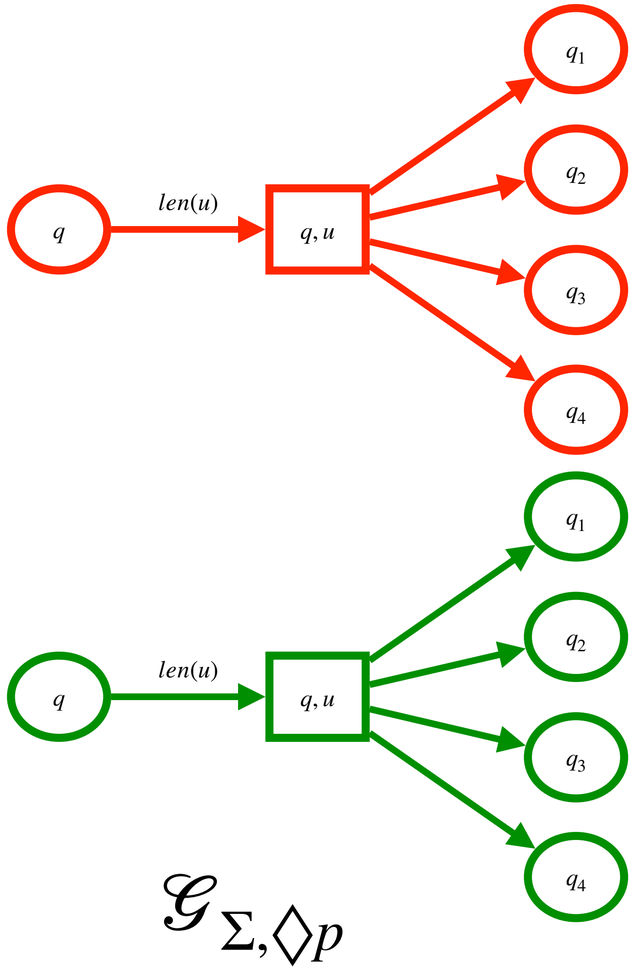


$\mathcal{G}_{\Sigma,\Diamond p}$

Inserting the specification, diamond case

$\Diamond p$





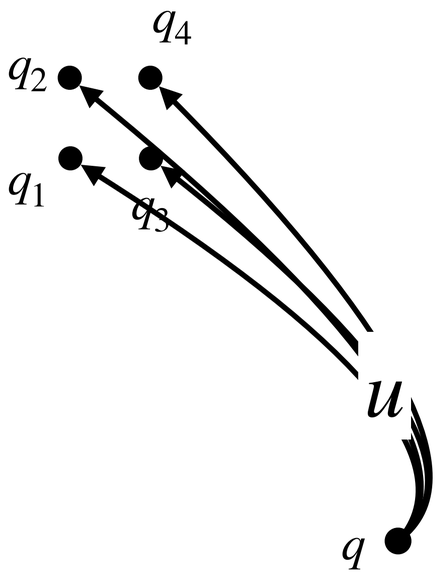
$\mathcal{G}_{\Sigma,\Diamond p}$

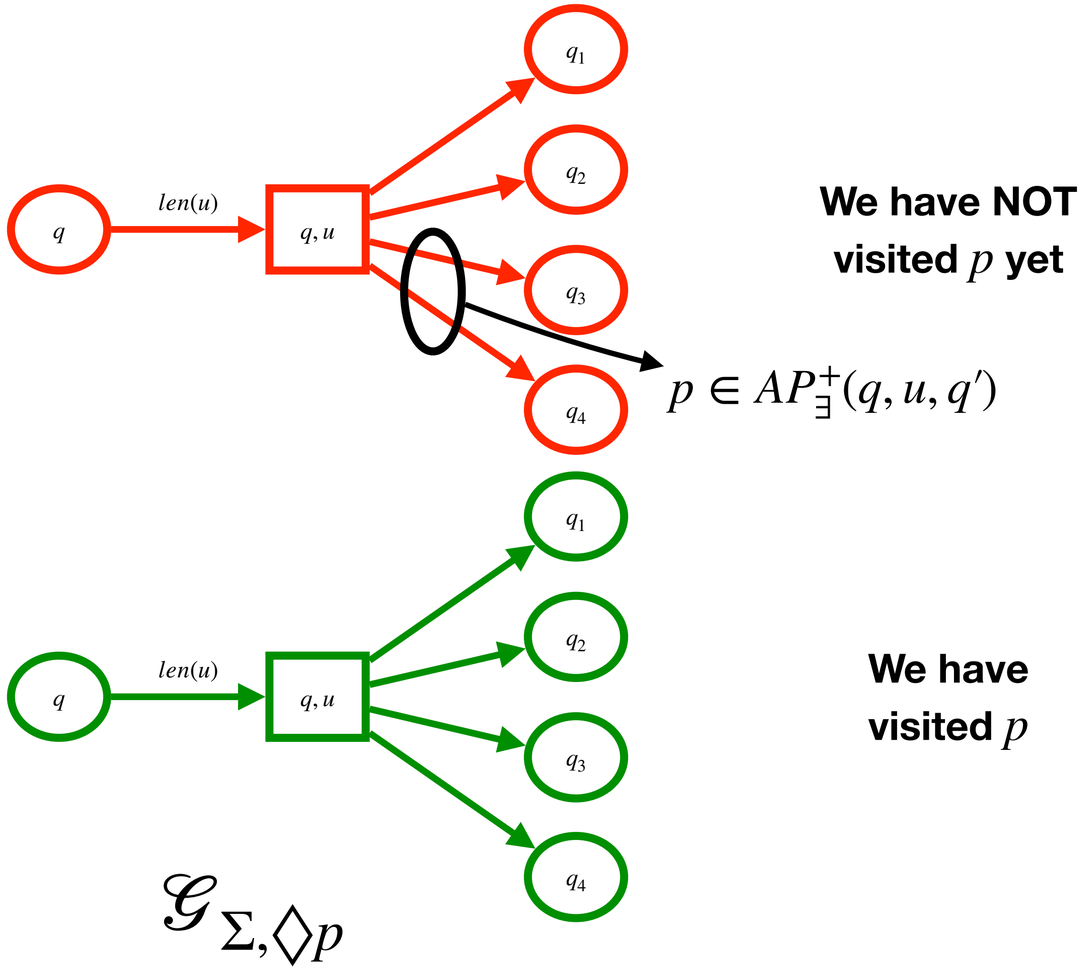
We have NOT visited p yet

We have visited p

Inserting the specification, diamond case

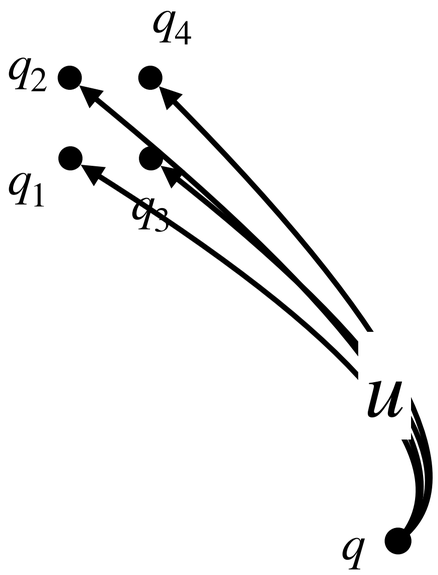
$\Diamond p$

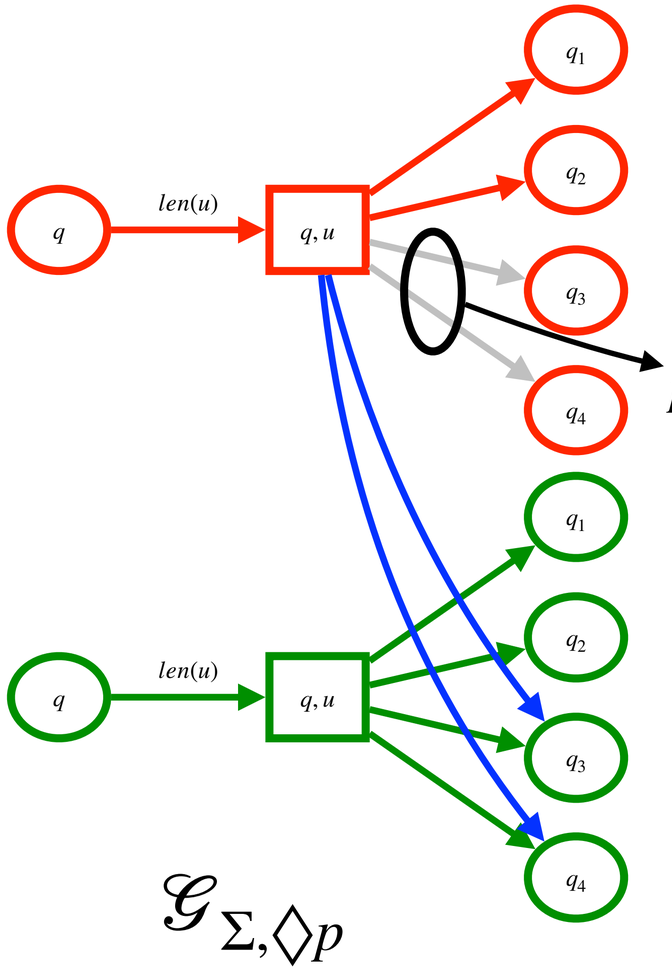




Inserting the specification, diamond case

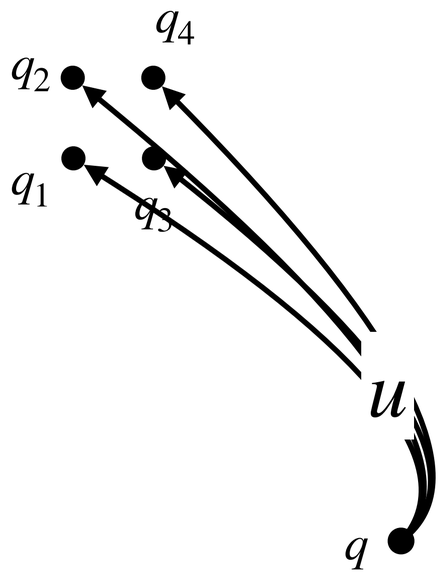
$\Diamond p$

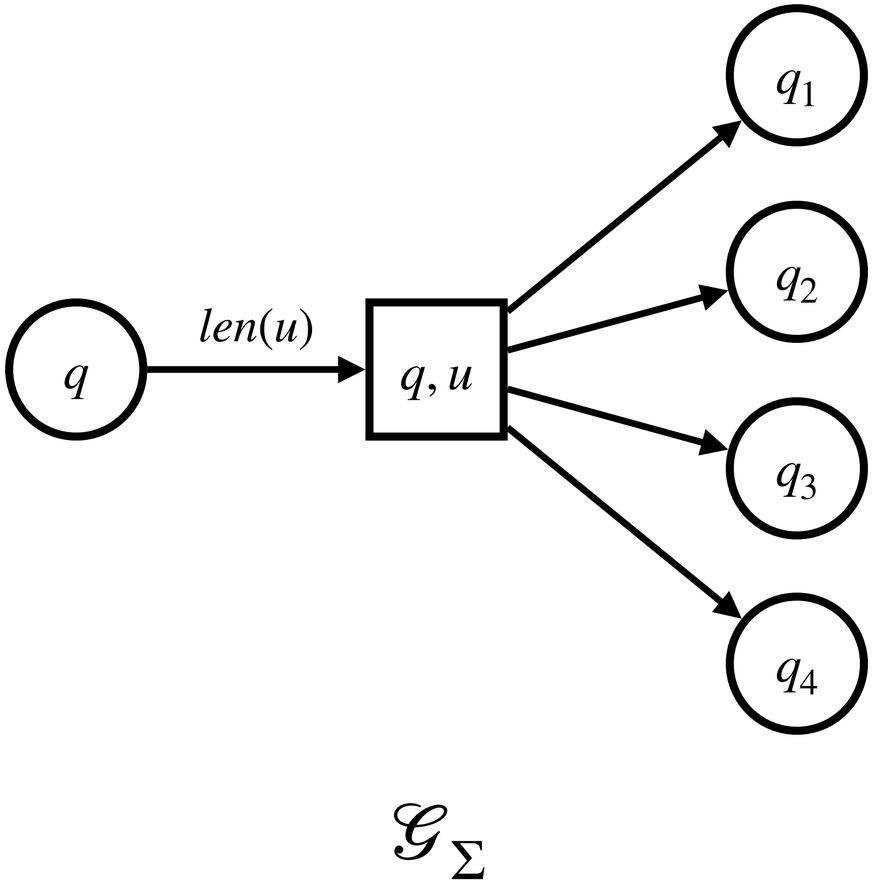




$\mathcal{G}_{\Sigma,\Diamond p}$

Inserting the specification, recurrent case

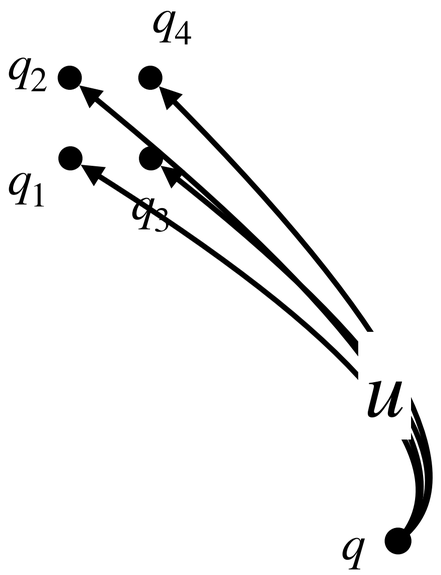


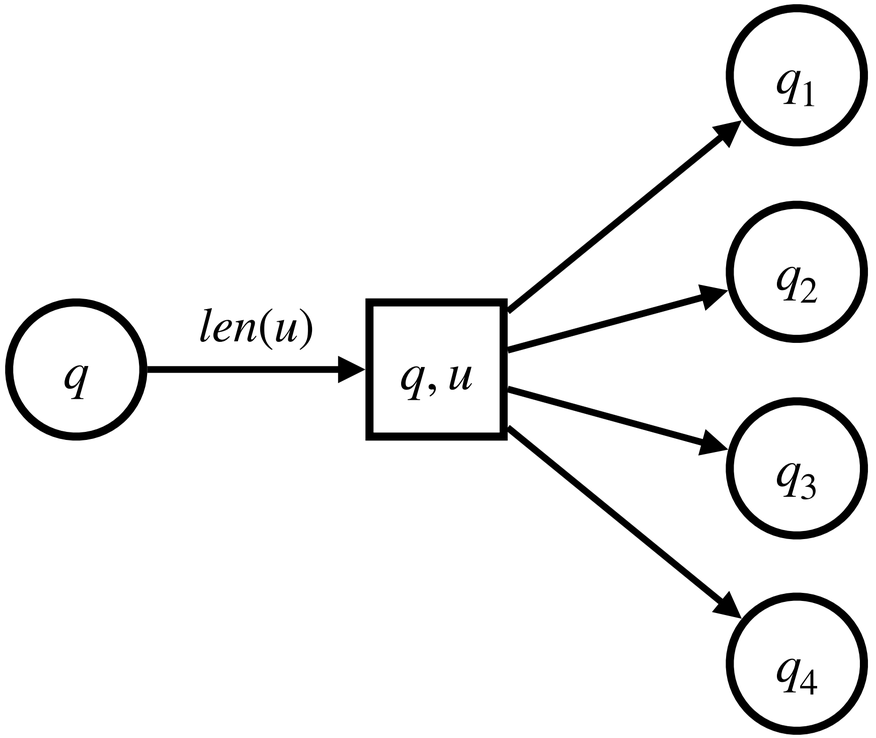


$\mathcal{G}_\Sigma$

Inserting the specification, recurrent case

$\square \Diamond p$

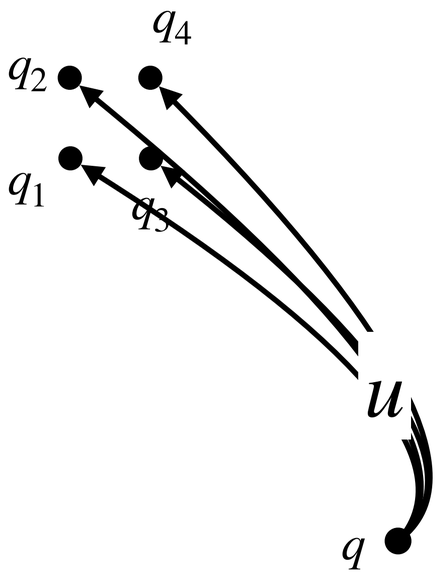


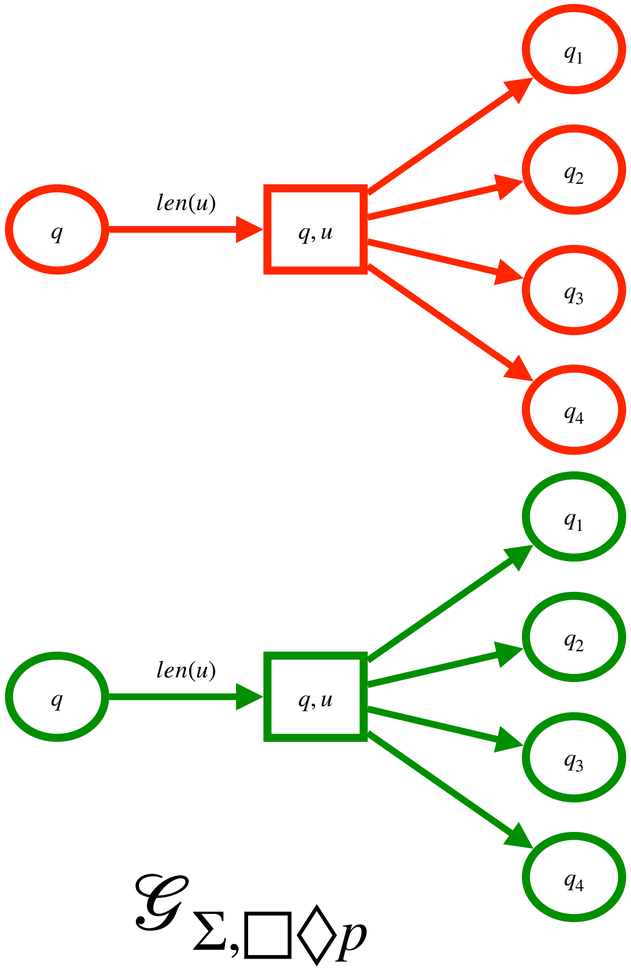


$\mathcal{G}_\Sigma$

Inserting the specification, recurrent case

$\square \Diamond p$

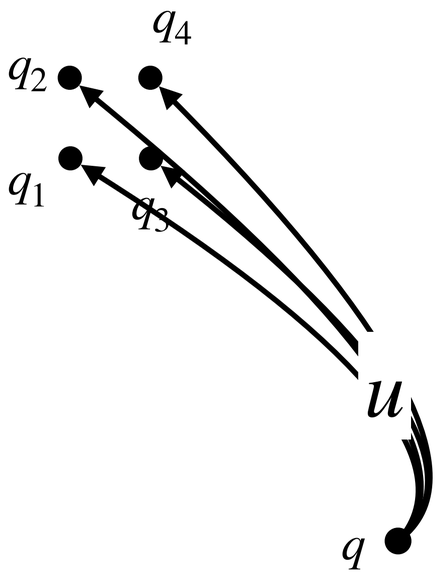


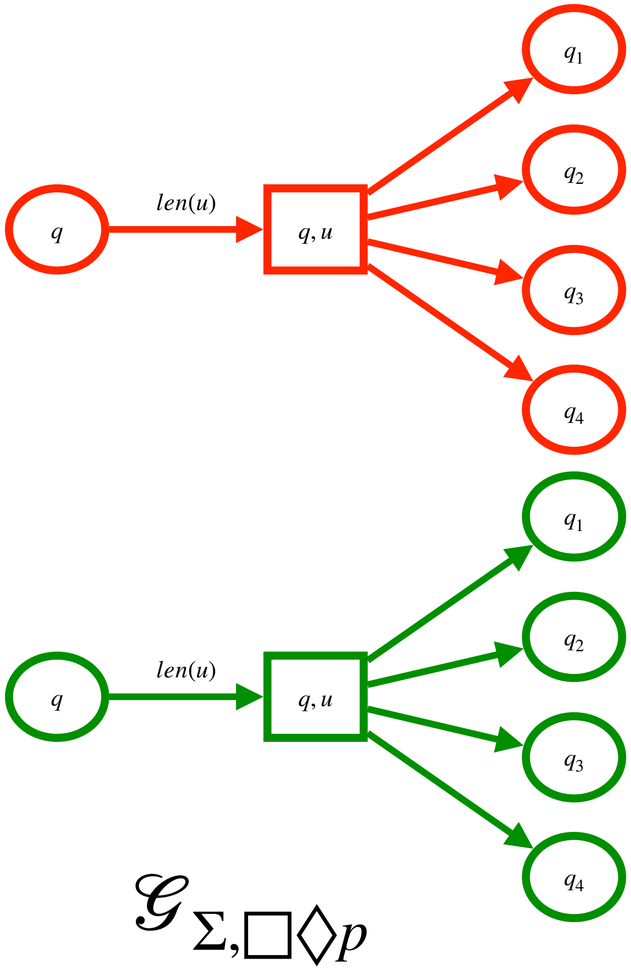


$\mathcal{G}_{\Sigma,\square\Diamond p}$

Inserting the specification, recurrent case

$\square\Diamond p$





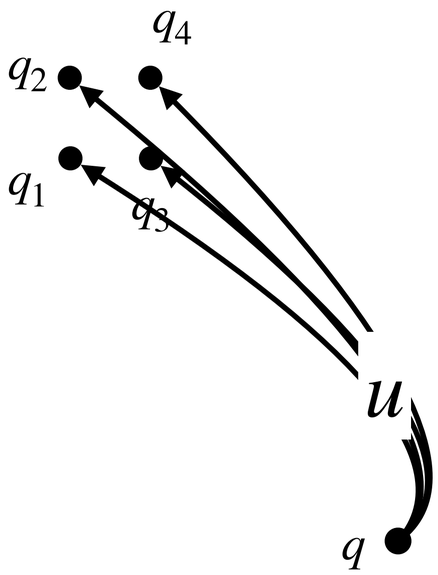
$\mathcal{G}_{\Sigma,\square\Diamond p}$

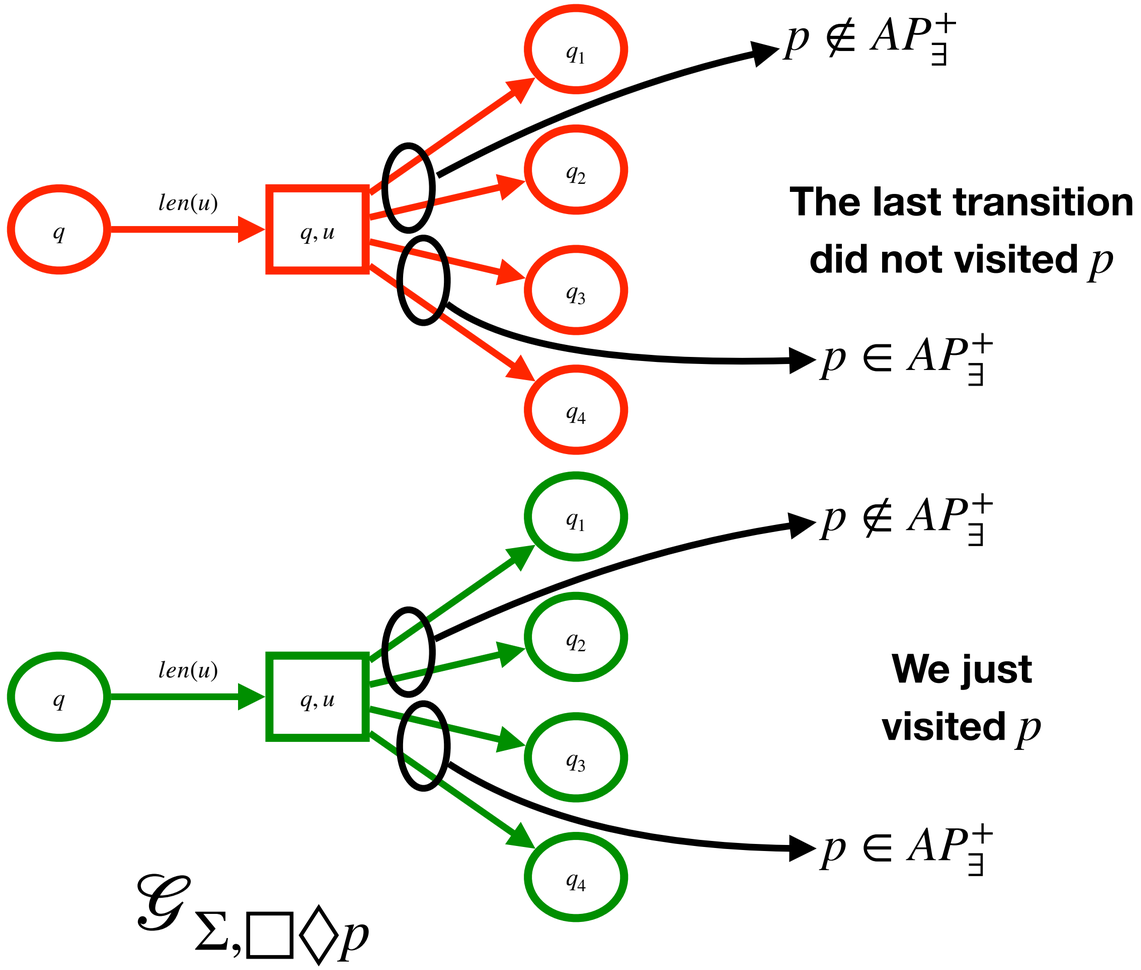
The last transition did not visited p

We just visited p

Inserting the specification, recurrent case

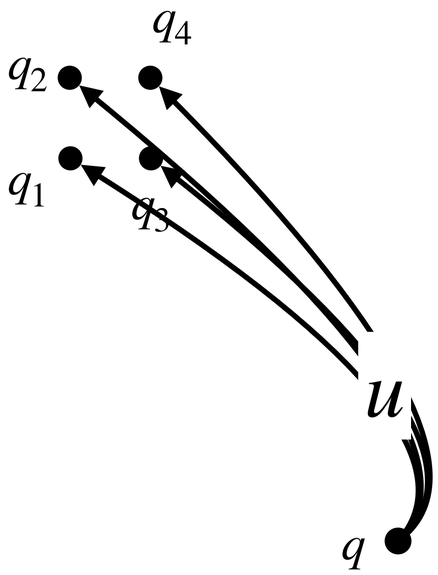
$\square \Diamond p$

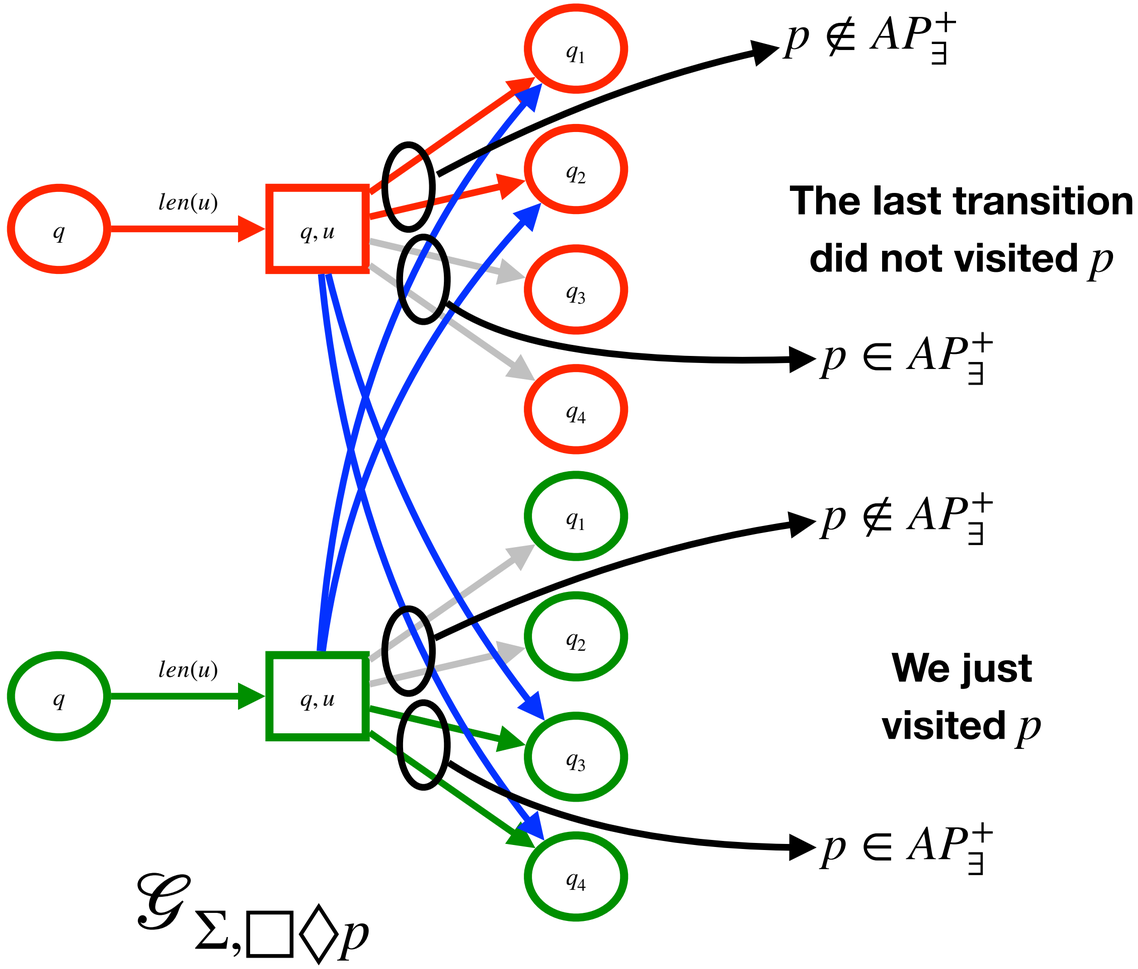




Inserting the specification, recurrent case

$\square \Diamond p$





Mean payoff parity games

$$\mathcal{G} = \left(X_s, X_e, Tr_s, Tr_e, w, p\right)$$

Where:

- $(X_s, X_e, Tr_s, Tr_e, w)$ mean payoff game
- $p : X_s \to \{0,1,\ldots,2n-1,2n\}$ (parity function)

Strategies, winning condition

Strategy for system: $S_s : (X_s \times X_e)^* \times X_s \to X_e$ such that

If $S_s(s_1 e_1 \ldots s_n) = e_n$ then $(s_n, e_n) \in Tr_s$

Strategy for environment: $S_e : (X_s \times X_e)^+ \to X_s$ such that

If $S_e(s_1 e_1 \ldots s_n e_n) = s_{n+1}$ then $(e_n, s_{n+1}) \in Tr_e$

Two strategies generate a unique run ρ_s for every state $s \in X_s$

System wins the game at s if it has a strategy S_s such that for all strategy S_e of the environment, ($\rho_s = s_1 e_1 \ldots$):

$$\liminf_{h \to \infty} \frac{1}{h} \sum_{i=1}^{h} w(s_i, e_i) \geq \nu \text{ and } \max \left\{ b \mid |\{i \mid p(s_i) = b\}| = \infty \right\} \text{ is even}$$

Calculability

The following is decidable

Data:

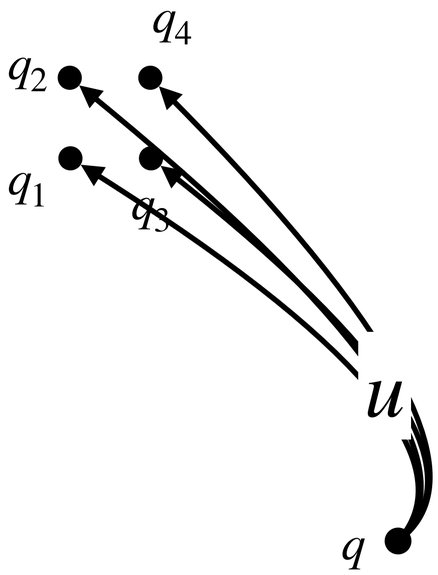
- Mean payoff parity game $\mathcal{G}$
- Threshold ν

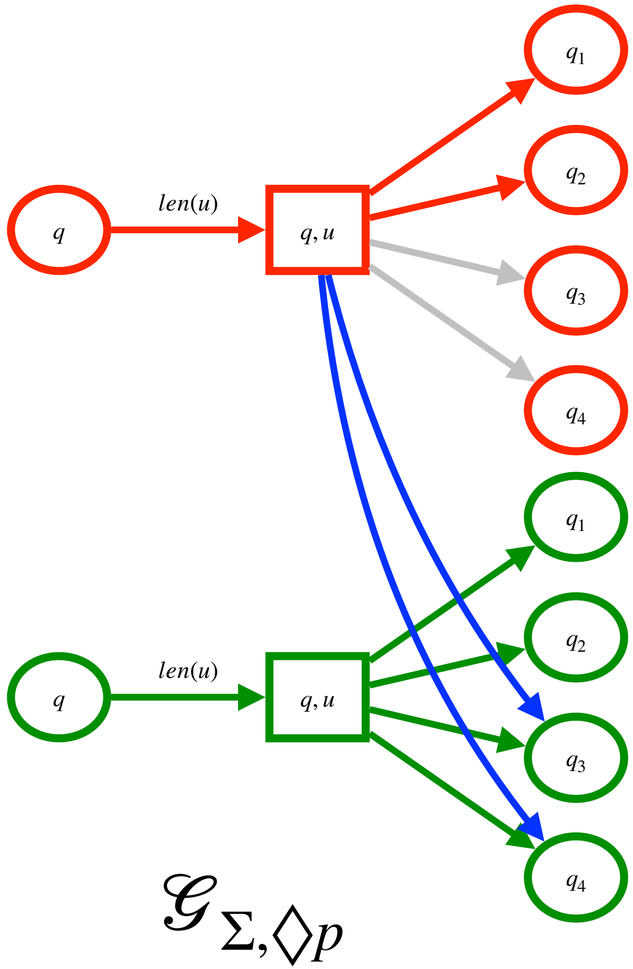
Output:

- Sets of winning states for system and environment
- Strategies

Mean payoff parity game for diamond

$\Diamond p$





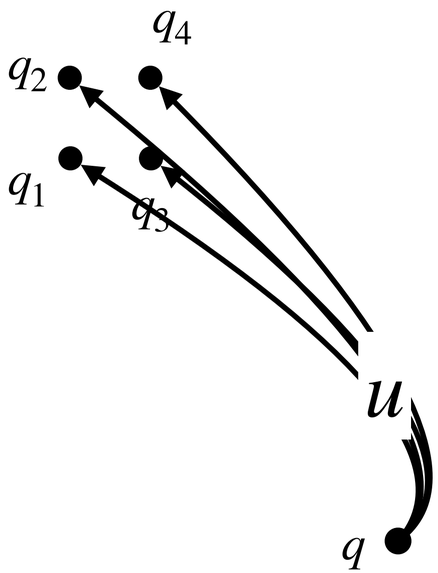
$\mathcal{G}_{\Sigma,\Diamond p}$

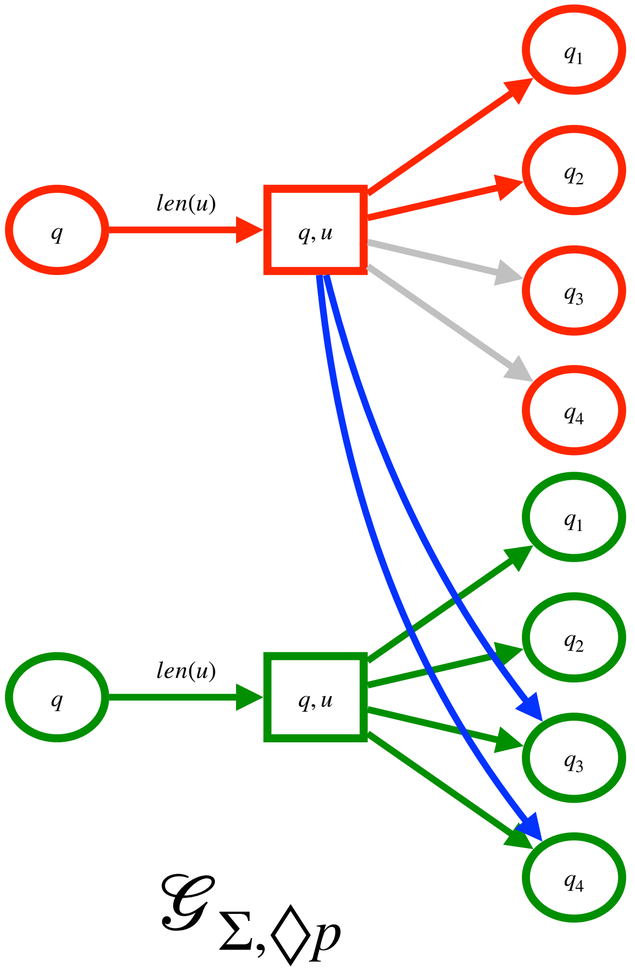
We have NOT visited p yet

We have visited p

Mean payoff parity game for diamond

$\Diamond p$





$\mathcal{G}_{\Sigma,\Diamond p}$

We have NOT visited p yet

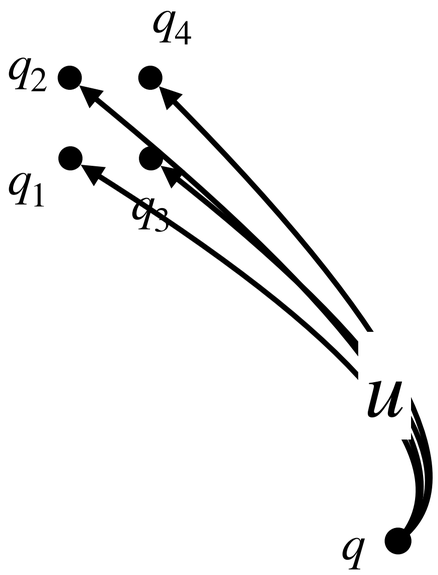
PARITY 1

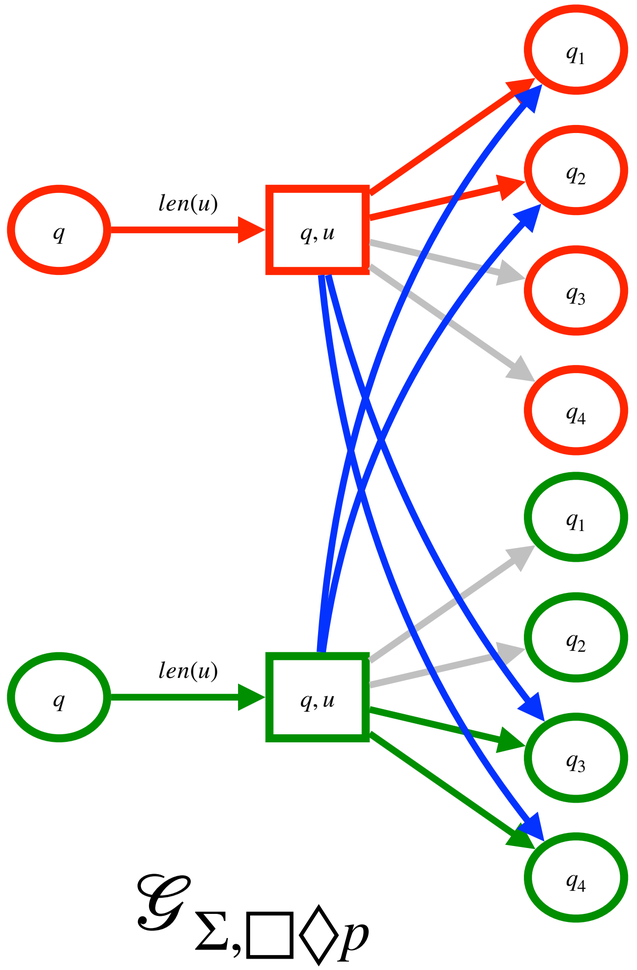
We have visited p

PARITY 0

Mean payoff parity game for recurrent

$\square\Diamond p$





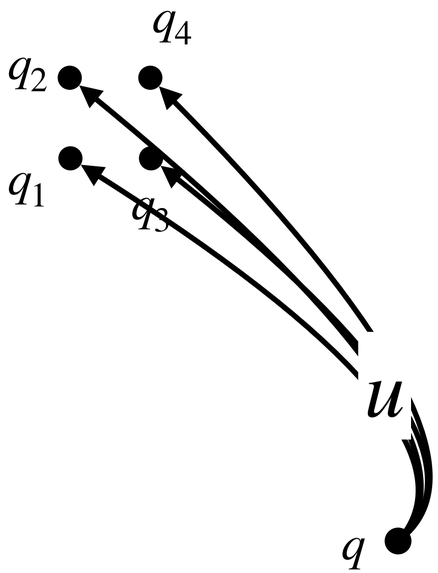
$\mathcal{G}_{\Sigma,\square\Diamond p}$

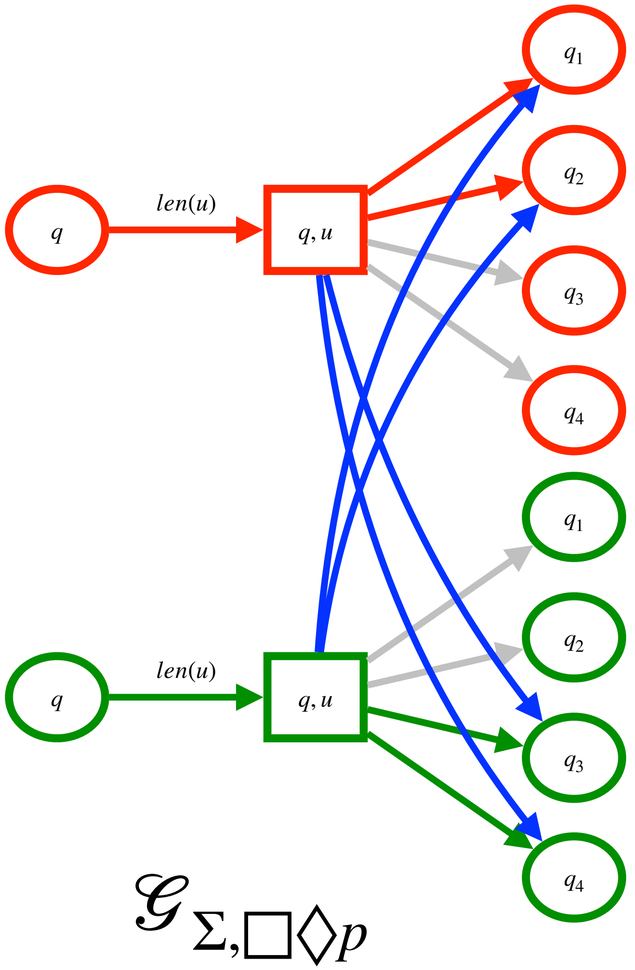
The last transition did not visited p

We just visited p

Mean payoff parity game for recurrent

$\square\Diamond p$





$\mathcal{G}_{\Sigma,\square\Diamond p}$

The last transition did not visited p

PARITY 1

We just visited p

PARITY 2

In general

Theorem:

Given a symbolic model S and a right-recursive LTL specification Φ, we can effectively build a game $\mathcal{G}_{S,\Phi}$ such that from a winning strategy for the system in $\mathcal{G}_{S,\Phi}$ we can effectively construct a controller for S solving our problem

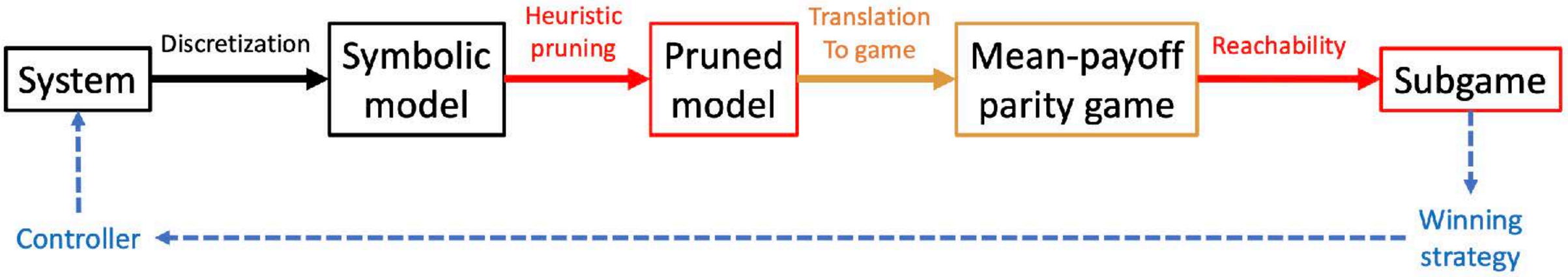
Strategy for system: $S_s : (X_s \times X_e)^* \times X_s \to X_e$ such that

If $S_s(s_1 e_1 \dots s_n) = e_n$ then $(s_n, e_n) \in Tr_s$

In $\mathcal{G}_{S,\Phi}$, s_i are states of the symbolic model and e_i are pairs (q, u) of a state of the symbolic model and of a signal, we can define:

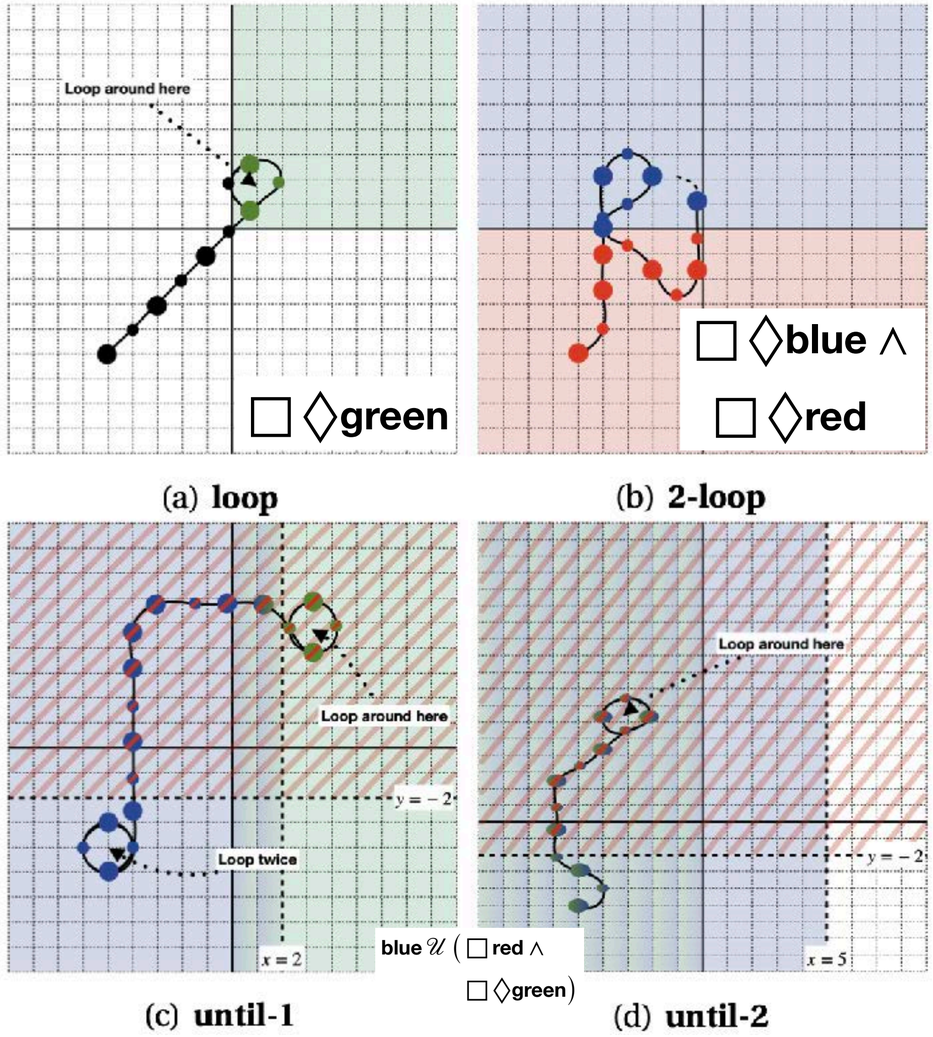
$$\widetilde{C}\left(q_1 u_1 q_2 \dots u_n q_{n+1}\right) = \pi_2\left(S_s\left(q_1 (q_1, u_1) \dots (q_n, u_n) q_{n+1}\right)\right)$$

Overview of the method



Some experiments

spec.	no pre-comp.	reach	prune + reach	no pre-comp.	reach	prune + reach
loop	202	36	33	82272	12866	4920
2-loop	16131	853	157	658176	33780	4497
until-1	timeout	2351	211*	709506	35107	12853
until-2	timeout	timeout	964	1181964	69744	39339



(a) **loop** (b) **2-loop**

(c) **until-1** (d) **until-2**

Monotony and optimality properties with KeYmaera X

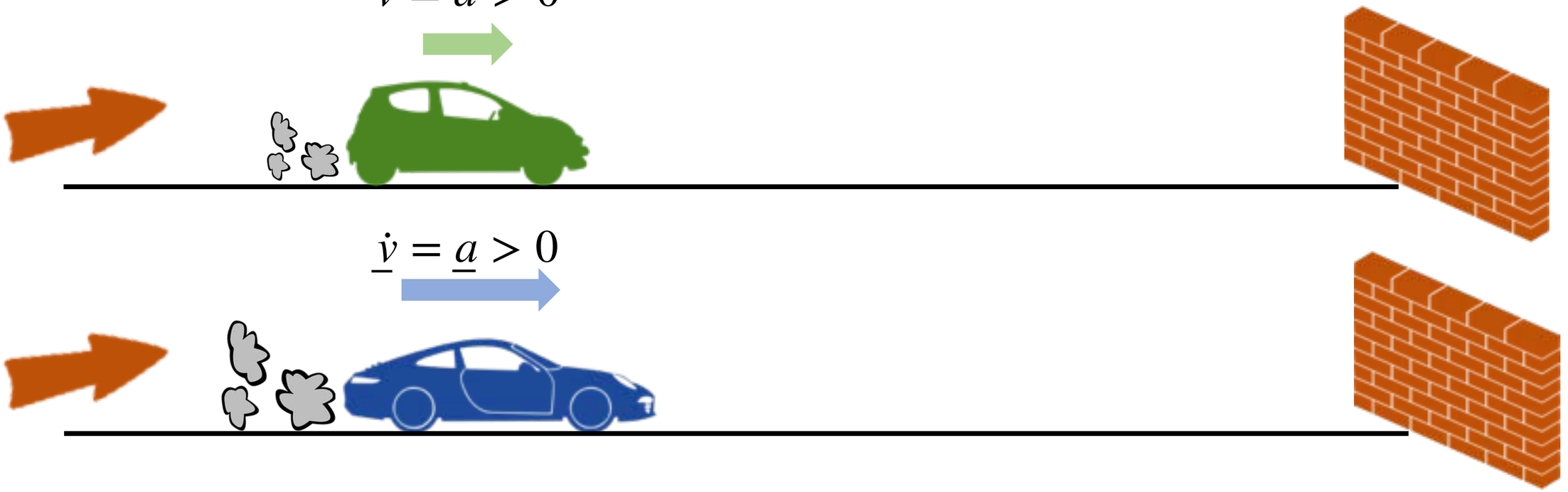
Collaborations

Joint work with:
- from Tokyo: Ichiro Hasuo, Akihisa Yamada, David Sprunger, and Shin-ya Katsumata (more recently, James Haydon)
- from France —> Denmark: Juraj Kolčák
- from the US: Brandon Bohrer

Initiated by discussions with Kenji Kamijo, Yoshiyuki Shinya, and Takamasa Suetomi from Mazda Motor Corporation

Sources:
- J. Kolčák, I. Hasuo, J. Dubut, S. Katsumata, D. Sprunger, A. Yamada, Relational Differential Dynamic Logic. TACAS'20.
- some implementations on GitHub

Title of the slide



When $\overline{a} < \underline{a}$ which vehicle crash harder?

How would you prove it?

How would you prove it?

“At all times, the car that accelerates more has a bigger speed.”

How would you prove it?

~~**“At all times, the car that accelerates more has a bigger speed.”**~~

The blue car crashes before!

Elementary proof

Consider the following easy dynamics:

$$\underline{\dot{x}} = \underline{v}$$
$$\underline{\dot{v}} = \underline{a}$$

$$\overline{\dot{x}} = \overline{v}$$
$$\overline{\dot{v}} = \overline{a}$$

Elementary proof

Consider the following easy dynamics:

$$\underline{\dot{x}} = \underline{v}$$
$$\underline{\dot{v}} = \underline{a}$$

$$\overline{\dot{x}} = \overline{v}$$
$$\overline{\dot{v}} = \overline{a}$$

Solving the equations:

$$\underline{v} = \underline{a} \,.\, \underline{t} + \underline{v}_0$$

$$\underline{x} = \frac{\underline{a}}{2} \,.\, \underline{t}^2 + \underline{v}_0 \,.\, \underline{t}$$

$$\overline{v} = \overline{a} \,.\, \overline{t} + \overline{v}_0$$

$$\overline{x} = \frac{\overline{a}}{2} \,.\, \overline{t}^2 + \overline{v}_0 \,.\, \overline{t}$$

Elementary proof

Consider the following easy dynamics:

$$\underline{\dot{x}} = \underline{v} \qquad \underline{\dot{v}} = \underline{a}$$

$$\overline{\dot{x}} = \overline{v} \qquad \overline{\dot{v}} = \overline{a}$$

Solving the equations:

$$\underline{v} = \underline{a} \,.\, \underline{t} + \underline{v}_0 \qquad \overline{v} = \overline{a} \,.\, \overline{t} + \overline{v}_0$$

$$\underline{x} = \frac{\underline{a}}{2} \,.\, \underline{t}^2 + \underline{v}_0 \,.\, \underline{t} \qquad \overline{x} = \frac{\overline{a}}{2} \,.\, \overline{t}^2 + \overline{v}_0 \,.\, \overline{t}$$

The time at which the vehicles reach the position x is:

$$\underline{t}(x) = \frac{\sqrt{\underline{v}_0^2 + 2\underline{a}x} - \underline{v}_0}{\underline{a}} \qquad \overline{t}(x) = \frac{\sqrt{\overline{v}_0^2 + 2\overline{a}x} - \overline{v}_0}{\overline{a}}$$

Elementary proof

Consider the following easy dynamics:

$$\underline{\dot{x}} = \underline{v} \qquad \underline{\dot{v}} = \underline{a}$$

$$\overline{\dot{x}} = \overline{v} \qquad \overline{\dot{v}} = \overline{a}$$

Solving the equations:

$$\underline{v} = \underline{a} \,.\, \underline{t} + \underline{v}_0 \qquad \overline{v} = \overline{a} \,.\, \overline{t} + \overline{v}_0$$

$$\underline{x} = \frac{\underline{a}}{2} \,.\, \underline{t}^2 + \underline{v}_0 \,.\, \underline{t} \qquad \overline{x} = \frac{\overline{a}}{2} \,.\, \overline{t}^2 + \overline{v}_0 \,.\, \overline{t}$$

The time at which the vehicles reach the position x is:

$$\underline{t}(x) = \frac{\sqrt{\underline{v}_0^2 + 2\underline{a}x} - \underline{v}_0}{\underline{a}} \qquad \overline{t}(x) = \frac{\sqrt{\overline{v}_0^2 + 2\overline{a}x} - \overline{v}_0}{\overline{a}}$$

The speed at position x is:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

Elementary proof

Consider the following easy dynamics:

$$\dot{\underline{x}} = \underline{v} \qquad \dot{\underline{v}} = \underline{a}$$

$$\dot{\overline{x}} = \overline{v} \qquad \dot{\overline{v}} = \overline{a}$$

Solving the equations:

$$\underline{v} = \underline{a} \,.\, \underline{t} + \underline{v}_0 \qquad \overline{v} = \overline{a} \,.\, \overline{t} + \overline{v}_0$$

$$\underline{x} = \frac{\underline{a}}{2} \,.\, \underline{t}^2 + \underline{v}_0 \,.\, \underline{t} \qquad \overline{x} = \frac{\overline{a}}{2} \,.\, \overline{t}^2 + \overline{v}_0 \,.\, \overline{t}$$

The time at which the vehicles reach the position x is:

$$\underline{t}(x) = \frac{\sqrt{\underline{v}_0^2 + 2\underline{a}x} - \underline{v}_0}{\underline{a}} \qquad \overline{t}(x) = \frac{\sqrt{\overline{v}_0^2 + 2\overline{a}x} - \overline{v}_0}{\overline{a}}$$

The speed at position x is:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

If $\underline{a} \leq \overline{a}$ and $\underline{v}_0 \leq \overline{v}_0$ then $\underline{v}(x) \leq \overline{v}(x)$ and the blue car crashes harder!

Elementary proof

Consider the following easy dynamics:

$$\dot{\underline{x}} = \underline{v} \qquad \dot{\overline{x}} = \overline{v}$$
$$\dot{\underline{v}} = \underline{a} \qquad \dot{\overline{v}} = \overline{a}$$

Solving the equations:

$$\underline{v} = \underline{a} \,.\, \underline{t} + \underline{v}_0 \qquad \overline{v} = \overline{a} \,.\, \overline{t} + \overline{v}_0$$

$$\underline{x} = \frac{\underline{a}}{2} \,.\, \underline{t}^2 + \underline{v}_0 \,.\, \underline{t} \qquad \overline{x} = \frac{\overline{a}}{2} \,.\, \overline{t}^2 + \overline{v}_0 \,.\, \overline{t}$$

The time at which the vehicles reach the position x is:

$$\underline{t}(x) = \frac{\sqrt{\underline{v}_0^2 + 2\underline{a}x} - \underline{v}_0}{\underline{a}} \qquad \overline{t}(x) = \frac{\sqrt{\overline{v}_0^2 + 2\overline{a}x} - \overline{v}_0}{\overline{a}}$$

The speed at position x is:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

If $\underline{a} \leq \overline{a}$ and $\underline{v}_0 \leq \overline{v}_0$ then $\underline{v}(x) \leq \overline{v}(x)$ and the blue car crashes harder!

KeYmaera X in a nutshell

- A Hoare-triples-style syntax to formalise properties of hybrid system
- A sequent calculus to implement proofs of those properties
- A tool: KeYmaeraX

Ref: A. Platzer's group http://symbolaris.com

Hybrid programs and Hoare triples

Every execution of the program α starting from a state satisfying the property Γ, satisfies the property P

$$\Gamma \vdash [\alpha]\, P$$

Γ, P: sets of first order formulae of real arithmetic

α: hybrid program

$$\begin{aligned}
\alpha ::= \; & x := e \quad \text{(assignement)} \\
| \; & ?P \quad \text{(conditional)} \\
| \; & \alpha + \alpha \quad \text{(non-deterministic choice)} \\
| \; & \alpha; \alpha \quad \text{(sequential composition)} \\
| \; & \alpha^\star \quad \text{(iteration)} \\
| \; & \dot{x} = f(x) \;\&\; Q \quad \text{(continuous dynamics)} \\
| \; & \ldots
\end{aligned}$$

Invariants

$$\frac{\Gamma \vdash \mathbf{Inv} \quad \mathbf{Inv} \vdash [\alpha]\,\mathbf{Inv} \quad \mathbf{Inv} \vdash P}{\Gamma \vdash [\alpha]\,P}(\mathbf{Inv})$$

Invariants

$$\frac{\Gamma \vdash \mathbf{Inv} \quad \mathbf{Inv} \vdash [\alpha]\,\mathbf{Inv} \quad \mathbf{Inv} \vdash P}{\Gamma \vdash [\alpha]\,P}(\mathbf{Inv})$$

To prove the statement $\Gamma \vdash [\alpha]\,P$

Invariants

It is enough to find a set/property Inv such that:

$$\frac{\Gamma \vdash \mathbf{Inv} \quad \mathbf{Inv} \vdash [\alpha]\,\mathbf{Inv} \quad \mathbf{Inv} \vdash P}{\Gamma \vdash [\alpha]\,P}(\mathbf{Inv})$$

To prove the statement $\Gamma \vdash [\alpha]\,P$

Invariants

It is enough to find a set/property Inv such that:

- **holds initially**

$$\frac{\Gamma \vdash \mathbf{Inv} \qquad \mathbf{Inv} \vdash [\alpha]\,\mathbf{Inv} \qquad \mathbf{Inv} \vdash P}{\Gamma \vdash [\alpha]\,P}(\mathbf{Inv})$$

To prove the statement $\Gamma \vdash [\alpha]\,P$

Invariants

It is enough to find a set/property Inv such that:

- **holds initially**
- **implies the post-condition**

$$\frac{\Gamma \vdash \textbf{Inv} \qquad \textbf{Inv} \vdash [\alpha]\,\textbf{Inv} \qquad \textbf{Inv} \vdash P}{\Gamma \vdash [\alpha]\,P}\,(\textbf{Inv})$$

To prove the statement $\Gamma \vdash [\alpha]\,P$

Invariants

It is enough to find a set/property Inv such that:

- **holds initially**
- **implies the post-condition**
- **is preserved by the program**

$$\frac{\Gamma \vdash \mathbf{Inv} \qquad \mathbf{Inv} \vdash [\alpha]\,\mathbf{Inv} \qquad \mathbf{Inv} \vdash P}{\Gamma \vdash [\alpha]\,P}\,(\mathbf{Inv})$$

To prove the statement $\Gamma \vdash [\alpha]\,P$

Loop invariants

$$\frac{\Gamma \vdash \mathbf{Inv} \quad \mathbf{Inv} \vdash [\alpha]\,\mathbf{Inv} \quad \mathbf{Inv} \vdash P}{\Gamma \vdash [\alpha^\star]\,P}(\mathbf{LI})$$

Differential invariants?

$$\dot{\mathsf{x}} = \mathsf{e} \;\&\; Q \quad \simeq \quad \left(?Q; \mathsf{x} := \mathsf{x} + dt \,.\, \mathsf{e}\right)^{\star}; ?Q$$

$$\frac{\Gamma, Q \vdash \mathsf{Inv} \quad \mathsf{Inv}, Q \vdash \mathsf{Inv}(\mathsf{x} \leftarrow \mathsf{x} + dt \,.\, \mathsf{e}) \quad \mathsf{Inv} \vdash P}{\Gamma \vdash [\dot{\mathsf{x}} = \mathsf{e} \;\&\; Q]P} \text{ (dtI)}$$

Assume that $P = \mathsf{Inv} \equiv f = 0$. We want something to ensure:

$$f(\omega) = 0 \Rightarrow f(\omega + dt \,.\, \mathsf{e}(\omega)) = 0$$

It is enough to require that f is constant along the dynamics, that is, if ψ is a solution of $\dot{\mathsf{x}} = \mathsf{e}$, then $K : t \mapsto f(\psi(t))$ is constant, that is, its derivative is zero.

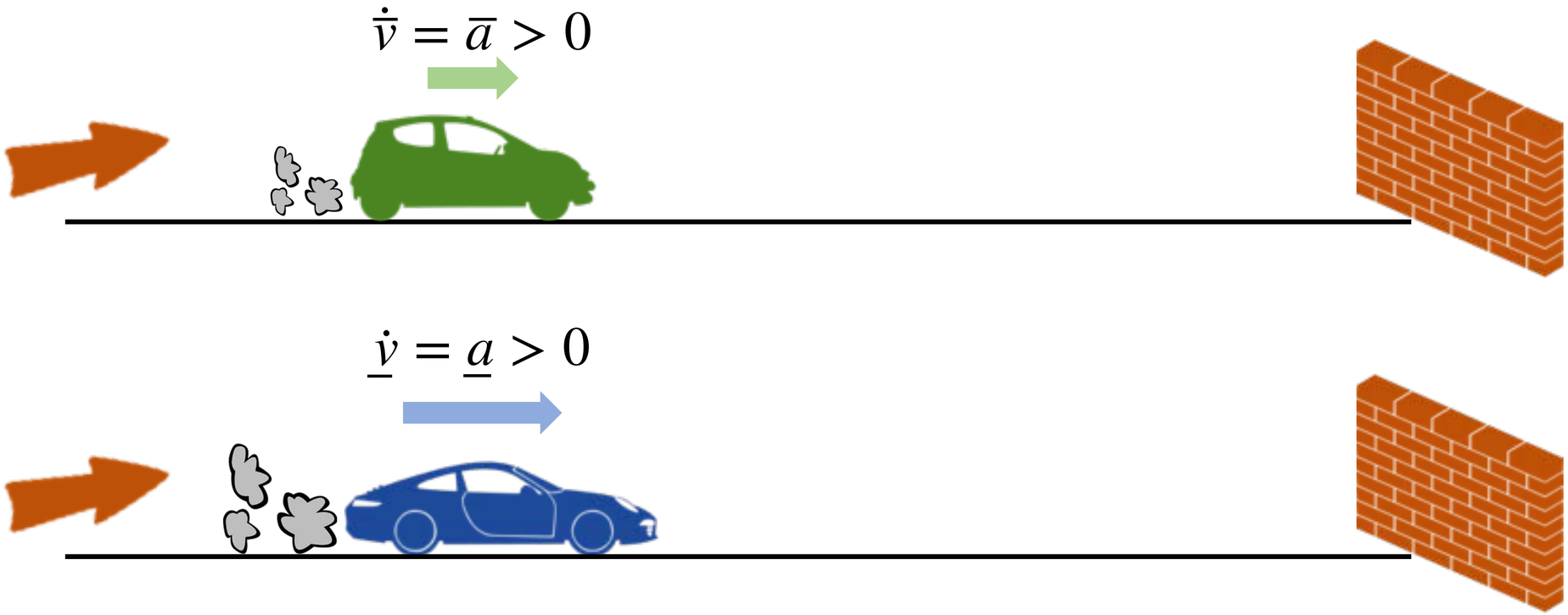
$$\dot{K}(t) = \sum_{x \in \mathsf{X}} \frac{\partial f}{\partial x}(\psi(t)) \,.\, \dot{\psi}(t) = \sum_{x \in \mathsf{X}} \frac{\partial f}{\partial x}(\psi(t)) \,.\, \mathsf{e}_x(\psi(t))$$

So it is enough that the function $\mathscr{L}_{\mathsf{e}} f = \sum_{x \in \mathsf{X}} \frac{\partial f}{\partial x} \,.\, \mathsf{e}_x$ to be zero along the dynamics.

Differential invariants

$$\frac{\Gamma, Q \vdash f = 0 \quad \Gamma \vdash [\dot{\mathbf{x}} = \mathbf{e} \,\&\, Q]\, \mathscr{L}_{\mathbf{e}} f = 0}{\Gamma \vdash [\dot{\mathbf{x}} = \mathbf{e} \,\&\, Q]\, f = 0}(\mathbf{DI})$$

Basic example: simplified ISO26262

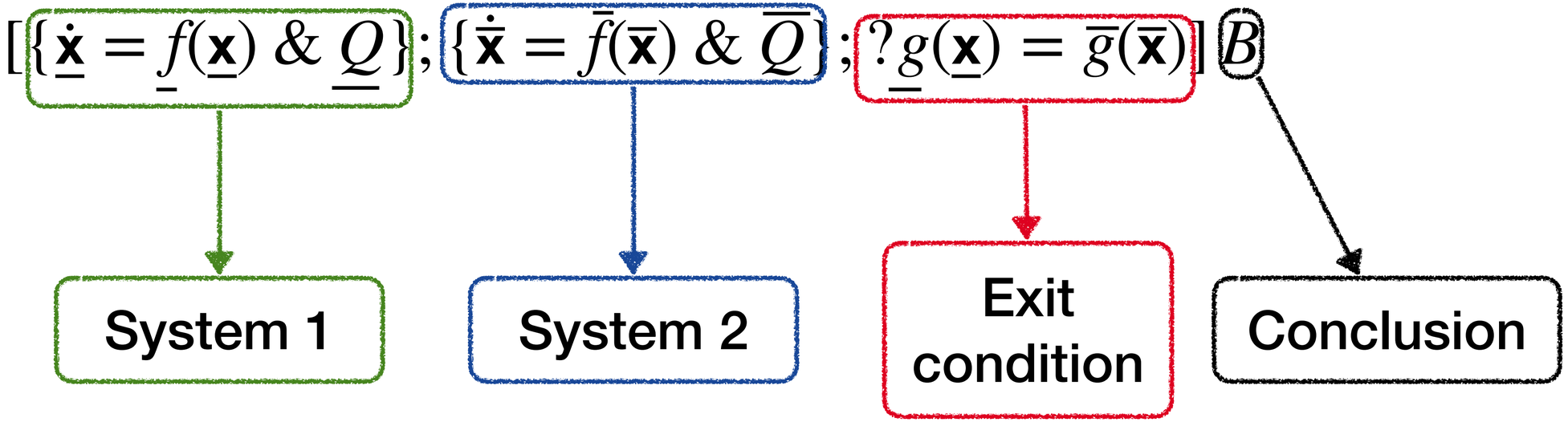


When $\overline{a} < \underline{a}$ which vehicle crash harder?

Simplified ISO26262 as Hoare triple

$$\underline{v} \geq \overline{v}, \underline{a} \geq \overline{a} \vdash [\{\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x})]\, \underline{v} \geq \overline{v}$$

Relational formulae



Relational invariants

$$\frac{\Gamma, \underline{Q}, \overline{Q} \vdash \textbf{Inv} \quad \textbf{Inv} \vdash [\underline{\delta}; \overline{\delta}; ?E]\,\textbf{Inv} \quad \textbf{Inv}, E \vdash B}{\Gamma \vdash [\underline{\delta}; \overline{\delta}; ?E]\,B}(\textbf{RI})$$

$$\underline{\delta} \equiv \underline{\dot{\mathbf{x}}} = \underline{f}(\underline{\mathbf{x}}) \;\&\; \underline{Q}$$

$$\overline{\delta} \equiv \overline{\dot{\mathbf{x}}} = \overline{f}(\overline{\mathbf{x}}) \;\&\; \overline{Q}$$

$$E \equiv \underline{g}(\underline{\mathbf{x}}) = \overline{g}(\overline{\mathbf{x}})$$

Relational invariant, for ISO26262

We know that:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad\qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

So:

$$\frac{\underline{v}(x)^2 - \underline{v}_0^2}{2\underline{a}} = x = \frac{\overline{v}(x)^2 - \overline{v}_0^2}{2\overline{a}}$$

And then:

$$R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$$

is a relational invariant.

Relational invariant, for ISO26262

We know that:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad\qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

So:

$$\frac{\underline{v}(x)^2 - \underline{v}_0^2}{2\underline{a}} = x = \frac{\overline{v}(x)^2 - \overline{v}_0^2}{2\overline{a}}$$

And then:

$$R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$$

is a relational invariant.

That is, one has to prove the following statements:

- $\underline{a} \leq \overline{a}, \underline{v} \geq \underline{v}_0, \overline{v} \geq \overline{v}_0, R \vdash \overline{v} \geq \underline{v}$
- $\underline{v} = \underline{v}_0, \overline{v} = \overline{v}_0 \vdash R$
- $R \vdash [\{\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x}]\, R$

Relational invariant, for ISO26262

We know that:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

So:

$$\frac{\underline{v}(x)^2 - \underline{v}_0^2}{2\underline{a}} = x = \frac{\overline{v}(x)^2 - \overline{v}_0^2}{2\overline{a}}$$

And then:

$$R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$$

is a relational invariant.

That is, one has to prove the following statements:

- $\underline{a} \leq \overline{a}, \underline{v} \geq \underline{v}_0, \overline{v} \geq \overline{v}_0, R \vdash \overline{v} \geq \underline{v}$
- $\underline{v} = \underline{v}_0, \overline{v} = \overline{v}_0 \vdash R$
- $R \vdash [\{\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x}]\, R$

The invariant implies the property (easy proof)

Relational invariant, for ISO26262

We know that:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

So:

$$\frac{\underline{v}(x)^2 - \underline{v}_0^2}{2\underline{a}} = x = \frac{\overline{v}(x)^2 - \overline{v}_0^2}{2\overline{a}}$$

And then:

$$R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$$

is a relational invariant.

That is, one has to prove the following statements:

- $\underline{a} \leq \overline{a}, \underline{v} \geq \underline{v}_0, \overline{v} \geq \overline{v}_0, R \vdash \overline{v} \geq \underline{v}$
- $\underline{v} = \underline{v}_0, \overline{v} = \overline{v}_0 \vdash R$
- $R \vdash [\{\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x}]\, R$

The invariant holds initially (easy proof)

Relational invariant, for ISO26262

We know that:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

So:

$$\frac{\underline{v}(x)^2 - \underline{v}_0^2}{2\underline{a}} = x = \frac{\overline{v}(x)^2 - \overline{v}_0^2}{2\overline{a}}$$

And then:

$$R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$$

is a relational invariant.

That is, one has to prove the following statements:

- $\underline{a} \leq \overline{a}, \underline{v} \geq \underline{v}_0, \overline{v} \geq \overline{v}_0, R \vdash \overline{v} \geq \underline{v}$
- $\underline{v} = \underline{v}_0, \overline{v} = \overline{v}_0 \vdash R$
- $R \vdash [\{\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x}]\, R$

The invariant is preserved by the dynamics (easy proof?)

Problem to tackle

We want a method that:

- does not require the solutions of the differential equations in any way
- transforms the two dynamics into one unique, so that we can use known methods from differential invariants.

Two systems into one?

What we have now:

- one system on $\underline{x}, \underline{v}$
- one system on $\overline{x}, \overline{v}$

that take different times to arrive at a particular position.

What we want:

- one system on $\underline{x}, \underline{v}, \overline{x}, \overline{v}$ such that the positions are synchronised, that is, at all time t:

$$\underline{x}(t) = \overline{x}(t)$$

Time stretching

Crucial idea: reparametrise the time of $\overline{x}, \overline{v}$

Time stretch function: derivable function $k : \mathbb{R} \longrightarrow \mathbb{R}$ with

$$\dot{k} > 0$$

Reparamatrised dynamics: let $\dot{x} = f(x)$ be a differential equation.
Its reparametrisation by k is $\dot{x} = \dot{k}(t) \,.\, f(x)$.

Proposition:

x is a solution of $\dot{x} = f(x)$ iff

$x \circ k$ is a solution of $\dot{x} = \dot{k}(t) \,.\, f(x)$.

Is it OK?

Proposition:

Fix some initial condition x_0. Then

$$\{x(t) \mid x \text{ sol. of } \dot{x} = f(x) \text{ at } x_0\} = \{x(t) \mid x \text{ sol. of } \dot{x} = \dot{k}(t) \,.\, f(x) \text{ at } x_0\}.$$

Why is it OK then?

→ we do not care about « at time t, the vehicle is at position x with speed v »

→ we care about « at position x, the vehicle has speed v »

Which reparametrisation for ISO26262?

Fix $\underline{v}_0, \overline{v}_0$ and note $(\underline{\psi}_x, \underline{\psi}_v)$ the solution of $\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}$ at $\underline{x} = 0, \underline{v} = \underline{v}_0$, $(\overline{\psi}_x, \overline{\psi}_v)$ the solution of $\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}$ at $\overline{x} = 0, \overline{v} = \overline{v}_0$.

We have a time stretch function $k : \mathbb{R} \longrightarrow \mathbb{R}$ such that for every x

$$k(\underline{t}(x)) = \overline{t}(x)$$

given by:

$$k(t) = \frac{\sqrt{at^2 + 2\underline{v}_0\overline{a}t + \overline{v}_0^2} - \overline{v}_0}{\overline{a}}$$

Which reparametrisation for ISO26262?

Fix $\underline{v}_0, \overline{v}_0$ and note $(\underline{\psi}_x, \underline{\psi}_v)$ the solution of $\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}$ at $\underline{x} = 0, \underline{v} = \underline{v}_0$, $(\overline{\psi}_x, \overline{\psi}_v)$ the solution of $\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}$ at $\overline{x} = 0, \overline{v} = \overline{v}_0$.

We have a time stretch function $k : \mathbb{R} \longrightarrow \mathbb{R}$ such that for every x

$$k(\underline{t}(x)) = \overline{t}(x)$$

So we want to look at:

$$\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}, \dot{\overline{x}} = \dot{k}(t) \,.\, \overline{v}, \dot{\overline{v}} = \dot{k}(t) \,.\, \overline{a}$$

But we have:

$$\overline{\psi}_x(k(t)) = \underline{\psi}_x(t)$$

Then:

$$\dot{k}(t) \,.\, \overline{\psi}_v(k(t)) = \underline{\psi}_v(t)$$

So we want to look at:

$$\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}, \dot{\overline{x}} = \frac{\underline{v}}{\overline{v}} \,.\, \overline{v}, \dot{\overline{v}} = \frac{\underline{v}}{\overline{v}} \,.\, \overline{a}$$

In general, for relational formulae

In general, from

$$\dot{\underline{x}} = \underline{f}(\underline{x})$$

and

$$\dot{\overline{x}} = \overline{f}(\overline{x})$$

under the exit condition

$$\underline{g}(\underline{x}) = \overline{g}(\overline{x})$$

we consider:

$$\dot{\underline{x}} = \underline{f}(\underline{x}), \dot{\overline{x}} = \frac{\mathscr{L}_{\underline{f}}\, \underline{g}}{\mathscr{L}_{\overline{f}}\, \overline{g}} \,.\, \overline{f}(\overline{x})$$

Synchronisation rule

$$\frac{\Gamma, \underline{Q}, \overline{Q} \vdash E \quad \Gamma \vdash [\underline{\delta}]\, \mathscr{L}_{\underline{f}}\, \underline{g} > 0 \quad \Gamma \vdash [\overline{\delta}]\, \mathscr{L}_{\overline{f}}\, \overline{g} > 0 \quad \Gamma \vdash [\delta]\, B}{\Gamma \vdash [\underline{\delta}; \overline{\delta}; ?E]\, B}(\textbf{Syn})$$

$$\underline{\delta} \equiv \dot{\underline{\mathbf{x}}} = \underline{f}(\underline{\mathbf{x}}) \;\&\; \underline{Q}$$

$$\overline{\delta} \equiv \dot{\overline{\mathbf{x}}} = \overline{f}(\overline{\mathbf{x}}) \;\&\; \overline{Q}$$

$$E \equiv \underline{g}(\underline{\mathbf{x}}) = \overline{g}(\overline{\mathbf{x}})$$

$$\delta \equiv \dot{\underline{\mathbf{x}}} = \underline{f}(\underline{\mathbf{x}}), \dot{\overline{\mathbf{x}}} = \frac{\mathscr{L}_{\underline{f}}\, \underline{g}}{\mathscr{L}_{\overline{f}}\, \overline{g}} . \overline{f}(\overline{\mathbf{x}}) \;\&\; \underline{Q} \wedge \overline{Q}$$

Relational invariant, for ISO26262

We know that:

$$\underline{v}(x) = \sqrt{\underline{v}_0^2 + 2\underline{a}x} \qquad \overline{v}(x) = \sqrt{\overline{v}_0^2 + 2\overline{a}x}$$

So:

$$\frac{\underline{v}(x)^2 - \underline{v}_0^2}{2\underline{a}} = x = \frac{\overline{v}(x)^2 - \overline{v}_0^2}{2\overline{a}}$$

And then:

$$R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$$

is a relational invariant.

That is, one has to prove the following statements:

- $\underline{a} \leq \overline{a}, \underline{v} \geq \underline{v}_0, \overline{v} \geq \overline{v}_0, R \vdash \overline{v} \geq \underline{v}$
- $\underline{v} = \underline{v}_0, \overline{v} = \overline{v}_0 \vdash R$
- $R \vdash [\{\dot{\underline{x}} = \underline{v}, \dot{\underline{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x}]\, R$

The invariant is preserved by the dynamics (easy proof?)

Relational invariant, for ISO26262

Let's prove the following statement with the (**Syn**) rule:

$$R \vdash [\{\underline{\dot{x}} = \underline{v}, \underline{\dot{v}} = \underline{a}\}; \{\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}\}; ?\underline{x} = \overline{x}]\, R$$

with $R \equiv \overline{a}(\underline{v}^2 - \underline{v}_0^2) = \underline{a}(\overline{v}^2 - \overline{v}_0^2)$, that is:

- $\underline{v} > 0, \underline{a} \geq 0 \vdash [\underline{\dot{x}} = \underline{v}, \underline{\dot{v}} = \underline{a}]\, \underline{v} > 0$ (easy with differential invariant)
- $\overline{v} > 0, \overline{a} \geq 0 \vdash [\dot{\overline{x}} = \overline{v}, \dot{\overline{v}} = \overline{a}]\, \overline{v} > 0$ (easy with differential invariant)
- $R \vdash [\{\underline{\dot{x}} = \underline{v}, \underline{\dot{v}} = \underline{a}, \dot{\overline{x}} = \frac{\underline{v}}{\overline{v}} \cdot \overline{v}, \dot{\overline{v}} = \frac{\underline{v}}{\overline{v}} \cdot \overline{a}\}]\, R$

using differential invariant rule:

$$R \vdash [\{\underline{\dot{x}} = \underline{v}, \underline{\dot{v}} = \underline{a}, \dot{\overline{x}} = \frac{\underline{v}}{\overline{v}} \cdot \overline{v}, \dot{\overline{v}} = \frac{\underline{v}}{\overline{v}} \cdot \overline{a}\}]\, 2\overline{a}\underline{v}\underline{a} = 2\underline{a}\overline{v}\frac{\underline{v}}{\overline{v}}\overline{a}$$

Summary so far

Guideline:

- start with two independent systems and compare them under some conditions
- synchronise them by reparametrising one of them using the (**Syn**) rule
- use usual invariant techniques from dL

Case studies:

- monotonicity properties
- abstraction

What is a good safety distance? The French law

On a French highway, when driving at 130km/h with normal conditions, the law states a safety distance of 73m.

Explanation:
- 73m = 2s at 130km/h ~ 36.5m/s
- 2s >> usual reaction time (~1s)
- So with 73m distance, a driver has time to react and brake at least as much as the vehicle in front of them to avoid collision

Why this is not a good definition:
- Assume that every car can brake the same ways
- Does not take comfort into account: if the car in front of you brake hard or collide, you have to brake hard too

In practice (2 lines = 91m):

What is a good safety distance? The maths

Step 1: forgetting about the reaction time, what is the minimal distance with the front car so that I can safely and comfortably brake

Assumptions:

- Car in front has velocity v_f and brakes at maximal possible rate $b_{\max}$
- I am at velocity v
- Here, braking comfortably means with braking rate $< b_{\text{comf}}$

Then, *a* safe distance is given by:

$$d_{\text{safe}} = \frac{v^2}{2b_{\text{comf}}} - \frac{v_f^2}{2b_{\max}}$$

Is this optimal?

What is a good safety distance? The maths

Step 1: forgetting about the reaction time, what is the minimal distance with the front car so that I can safely and comfortably brake

Assumptions:

- Car in front has velocity v_f and brakes at maximal possible rate $b_{\max}$
- I am at velocity v
- Here, braking comfortably means with braking rate < b_{comf}

Then, *a* safe distance is given by:

$$d_{\text{safe}} = \frac{v^2}{2b_{\text{comf}}} - \frac{v_f^2}{2b_{\max}}$$

For $v_f = v = 36.5 m/s$
$b_{max} = 8 m/s^2$
$b_{\text{comf}} = 4 m/s^2$
$d_{\text{safe}} = 83 m$

Is this optimal?

Optimality

Optimality here means: braking at maximal comfortable rate is the comfortable strategy that makes the distance to full stop the smallest

In KeYmaera X:

$$\Gamma \vdash [(\dot{x} = v, \dot{v} = -b_{\min} \;\&\; v \geq 0)\,;(a := *; ?(-b_{\min} \leq a < 0)\,;(\dot{y} = u, \dot{u} = a \;\&\; u \geq 0))^*]\; (v = 0 \land u = 0) \Rightarrow x \leq y$$

Which is provable using our Sync rule!

In total, we have formal guarantees of safety and of optimality.

What is a good safety distance? The maths II

Step 2: incorporating the reaction time, what is the minimal distance for which I cannot do whatever I want otherwise I may not be able to react on time to safely and comfortably brake

Assumptions:
- Same as before
- Reaction time is ρ, and maximal possible acceleration is $a_{\max}$

Then, the RSS distance is given by:

$$d_{\text{RSS}} = v\rho + \frac{a_{\max}\rho^2}{2} + \frac{(v + a_{\max}\rho)^2}{2b_{\text{comf}}} - \frac{v_f^2}{2b_{\max}}$$

What is a good safety distance? The maths II

Step 2: incorporating the reaction time, what is the minimal distance for which I cannot do whatever I want otherwise I may not be able to react on time to safely and comfortably brake

Assumptions:
- Same as before
- Reaction time is ρ, and maximal possible acceleration is $a_{\max}$

Then, the RSS distance is given by:

$$d_{\text{RSS}} = v\rho + \frac{a_{\max}\rho^2}{2} + \frac{(v + a_{\max}\rho)^2}{2b_{\text{comf}}} - \frac{v_f^2}{2b_{\max}}$$

For $a_{\max} = 0m/s^2$
$\rho = 1s$
$d_{RSS} = 120m$

A safe and comfortable controller

At every control loop, do:

- If the distance is bigger than the RSS distance, do whatever you want
- If the distance is smaller than the RSS distance, brake with maximal comfortable rate

This has been formalise in KeYmaera X and formally proved safe (even before the RSS paper)

No optimality of any sort has been formally proved

Optimality of the RSS distance

It can be formulated as follows:

- The RSS controller is safe and comfortable, and
- If you replace the RSS distance by a smaller distance in the previous controller, then there is a scenario in which you cannot safely and comfortably brake

Claimed in the RSS paper, but never formally proved

Future research: can we prove it using the Sync rule?

Optimality of the controller

It can be formulated as follows:

- The RSS controller is safe and comfortable, and
- Any safe and comfortable controller is not doing any better than this controller

It is not clear what “any better” would mean here:

- More progress
- Faster
- More comfortable

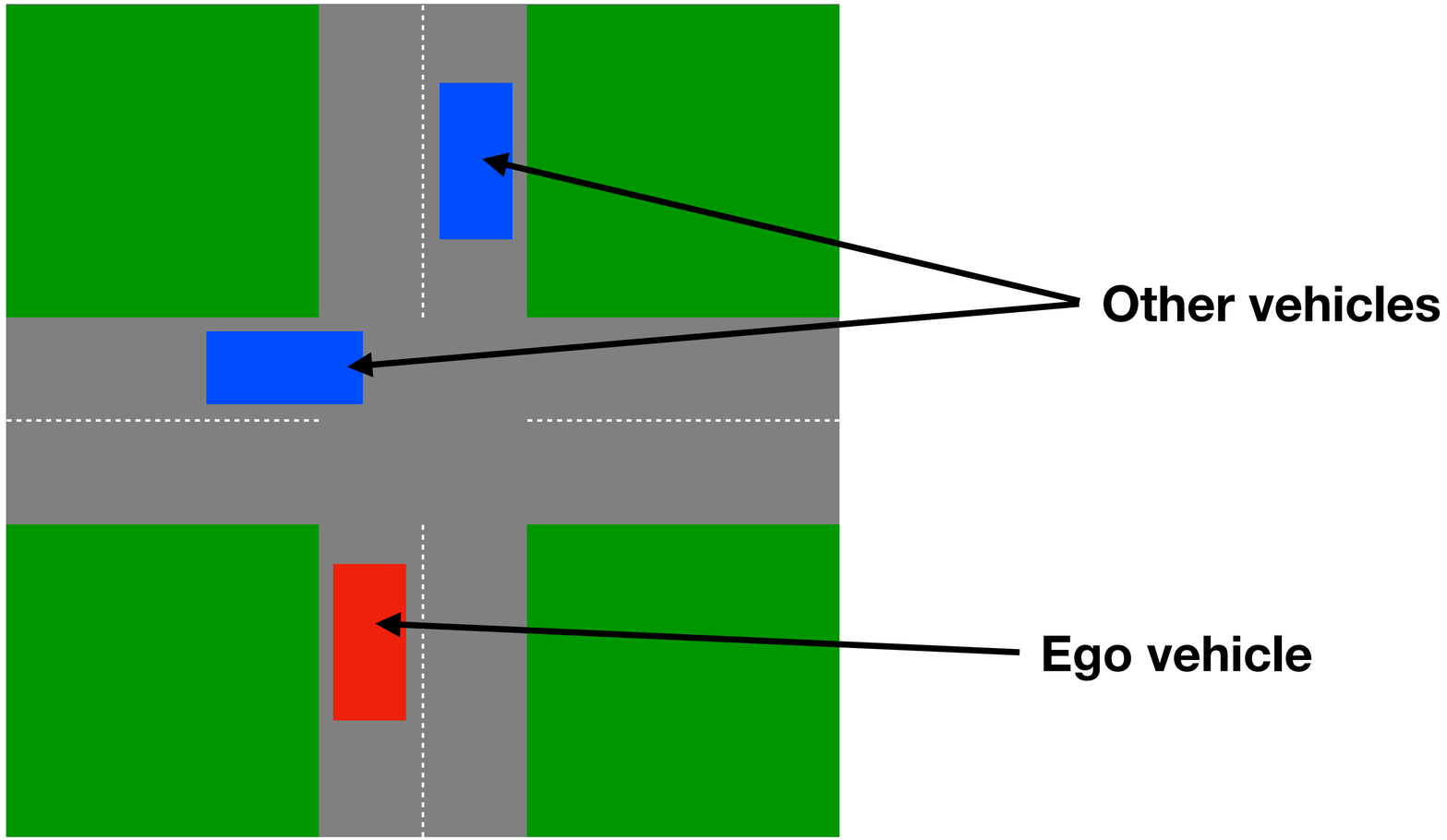
Whatever the measure of “any better”, the RSS controller is not optimal.

Future research:

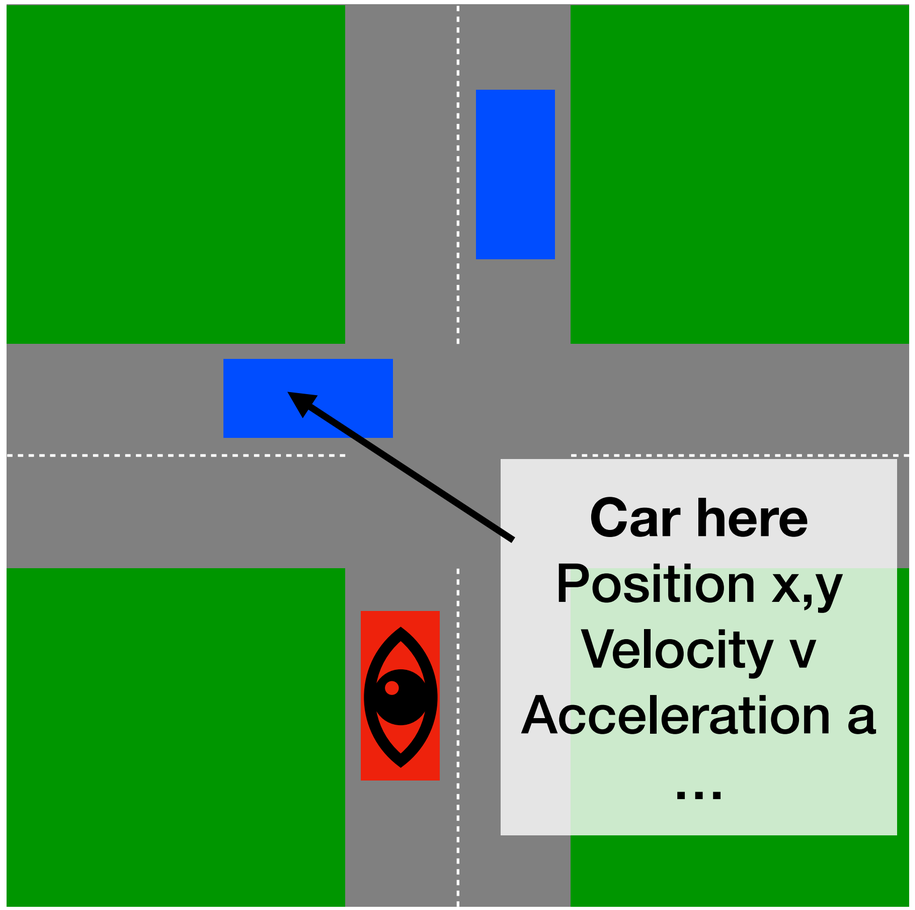
- **What is an optimal controller?**
- **How can we prove its optimality?**

Decision making using game theory

Decision making?

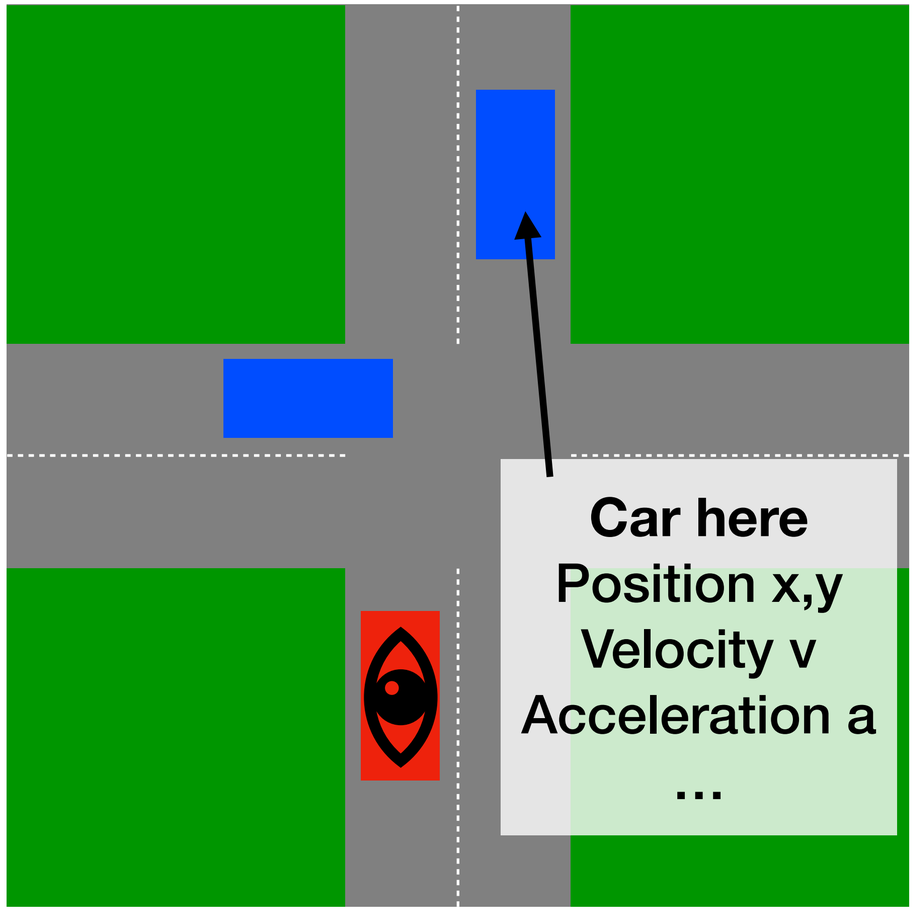


Decision making?



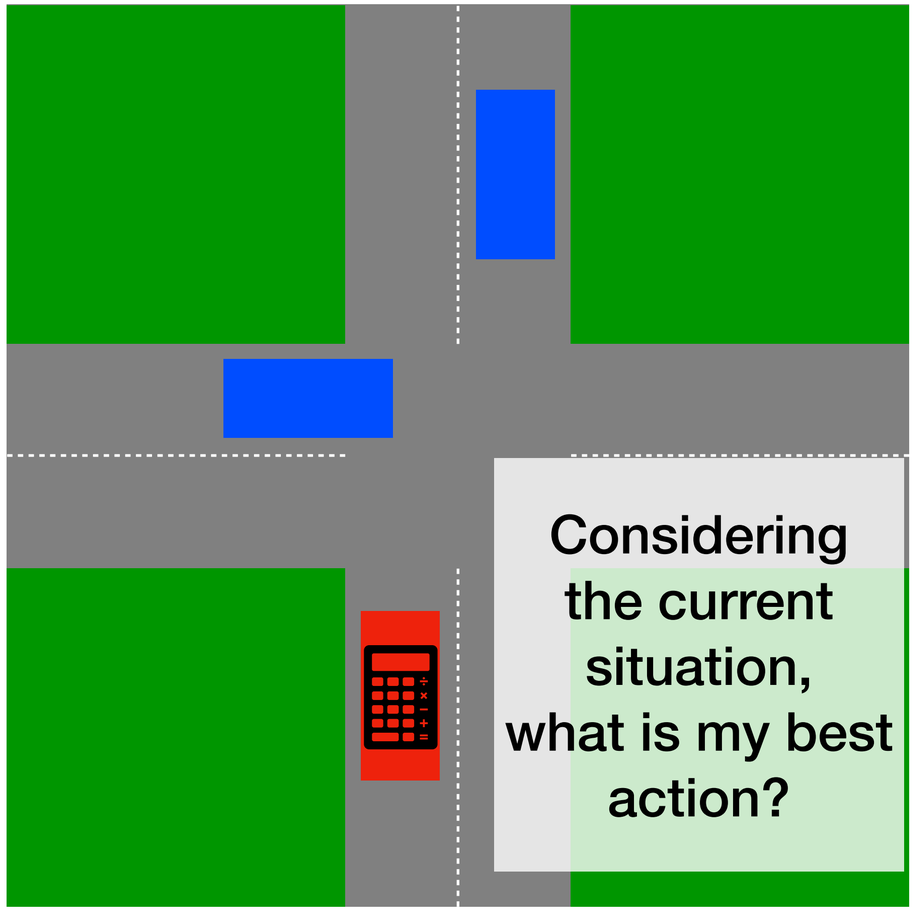
Step 1: Observation

Decision making?



Step 1: Observation

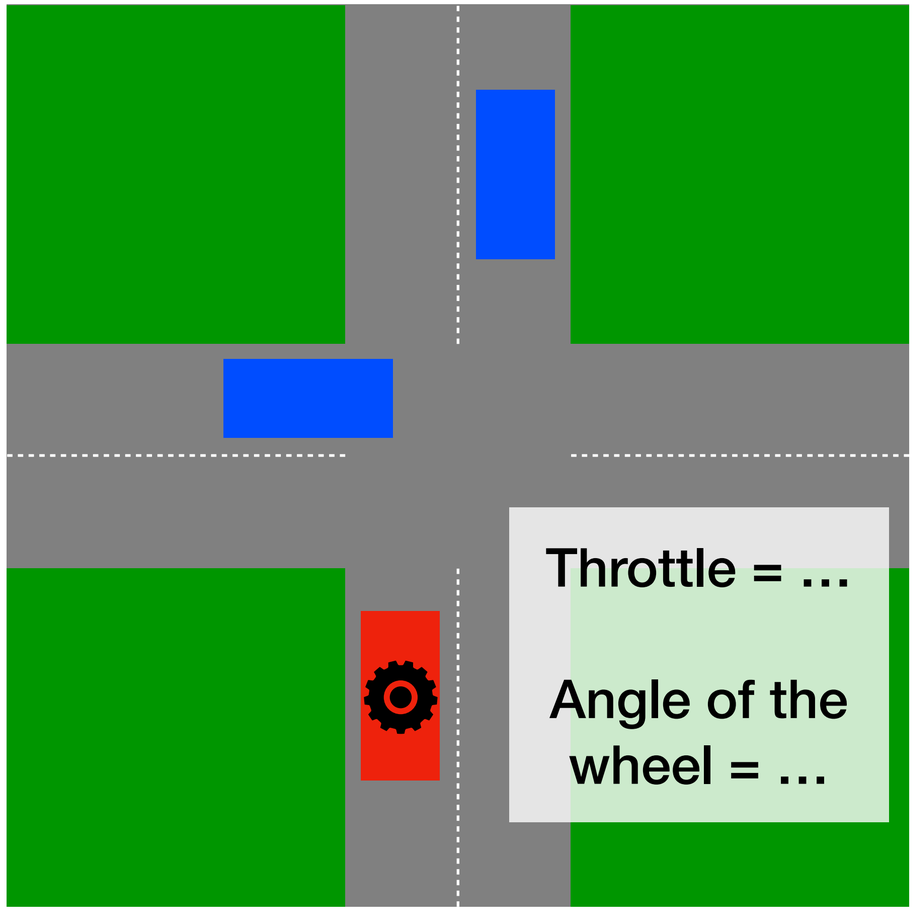
Decision making?



Step 1: Observation

Step 2: Computation

Decision making?

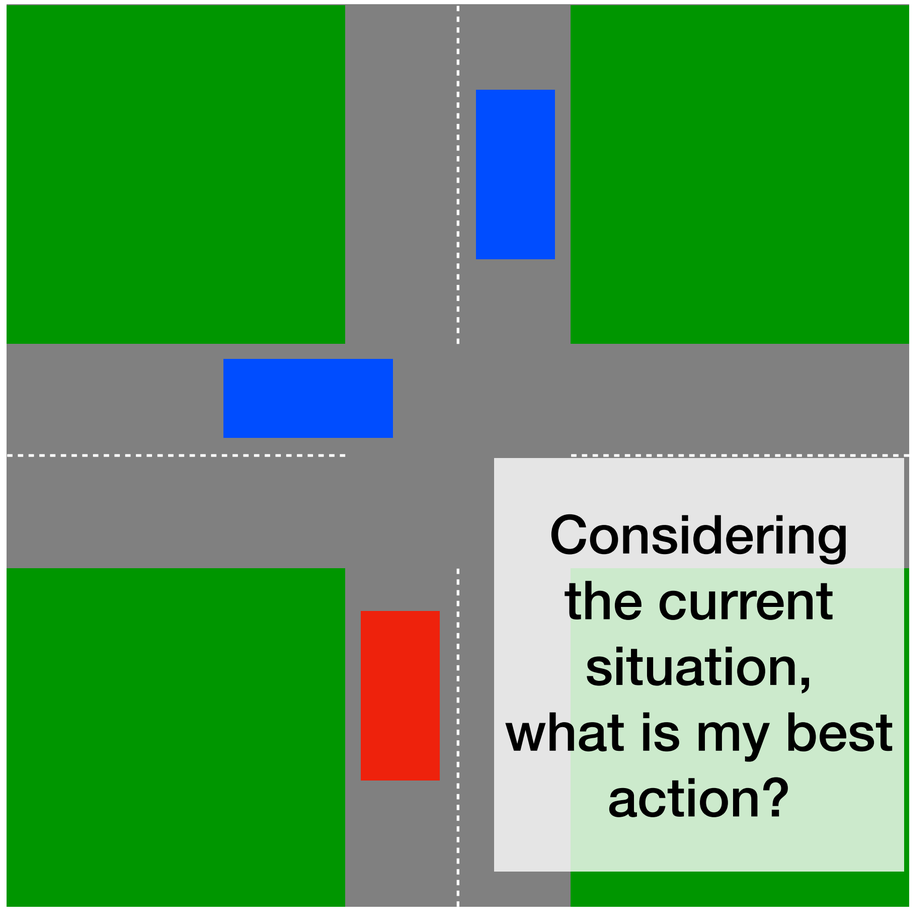


Step 1: Observation

Step 2: Computation

Step 3: Actuation

Decision making?

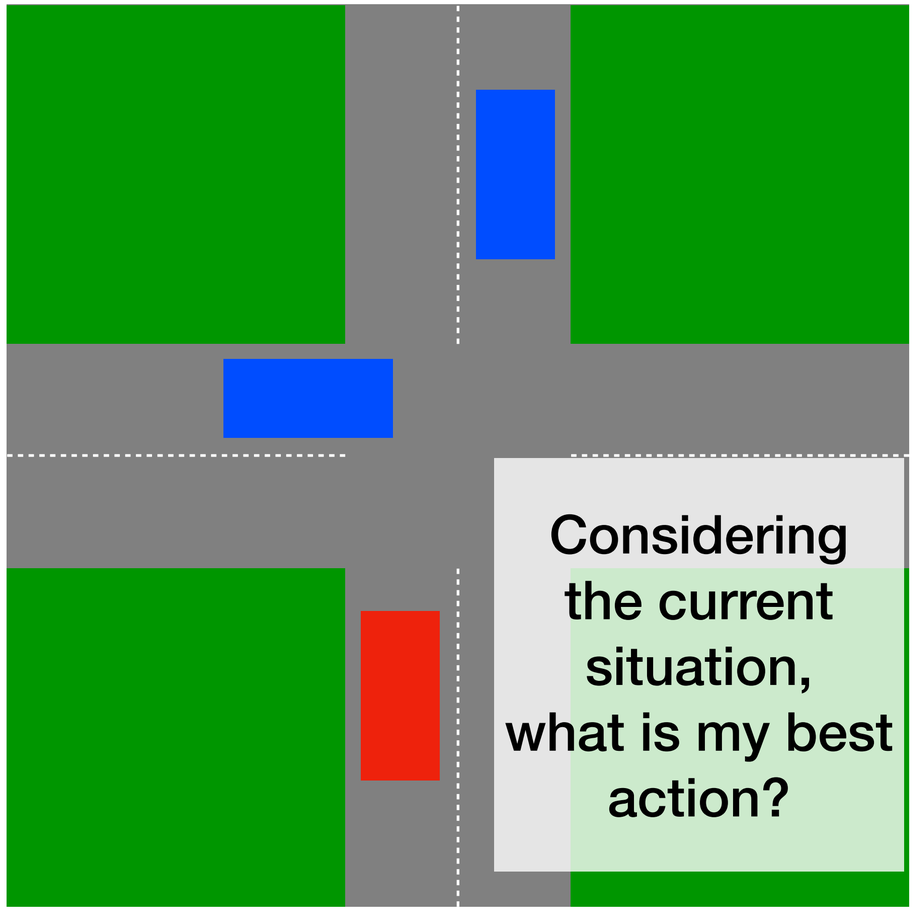


Step 1: Observation

Step 2: Computation

Step 3: Actuation

Decision making?

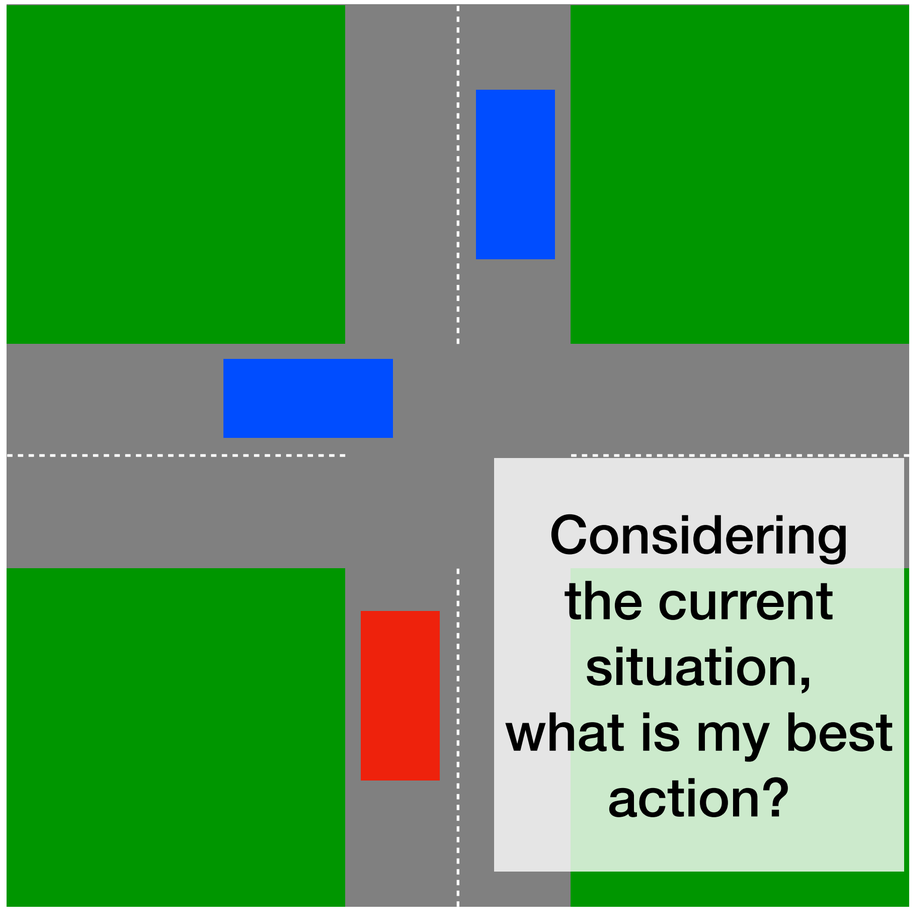


Step 2: Computation

Two main methods:

- Learning methods
- Game theoretic methods

Decision making?



Step 2: Computation

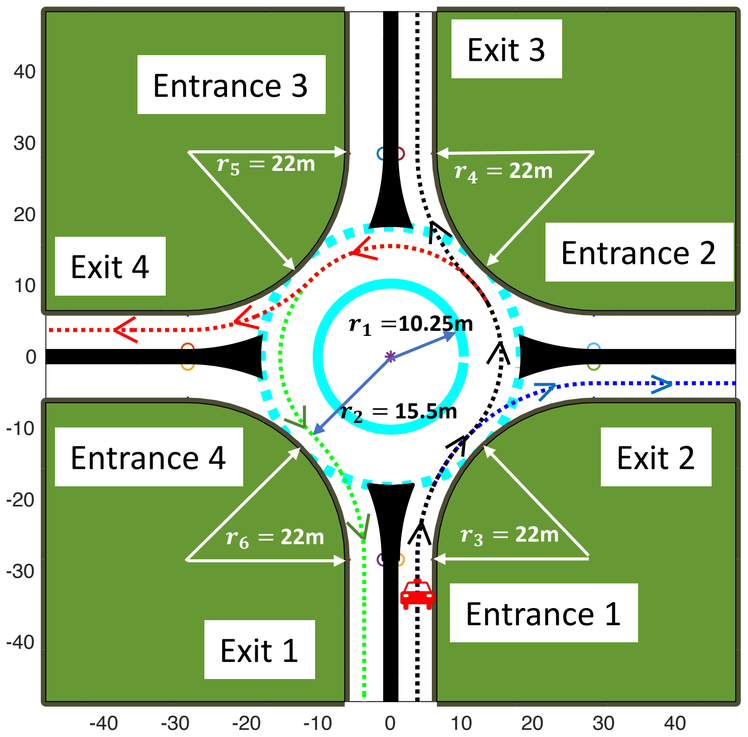
Two main methods:

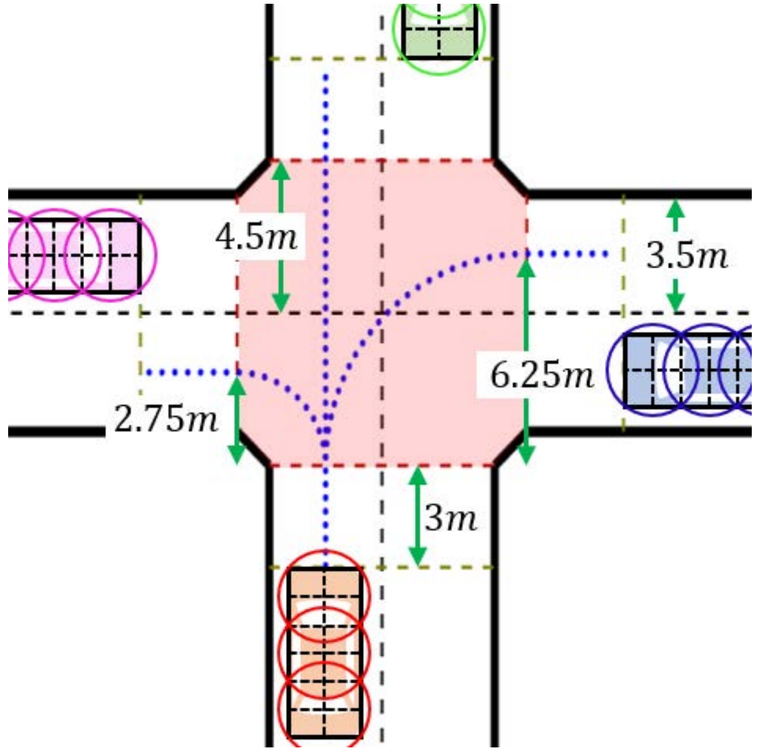
- Learning methods
- Game theoretic methods

Li et al., “Game Theoretic Modeling of Vehicle Interactions at Unsignalized Intersections and Application to Autonomous Vehicle Control” IEEE-ACC2018.

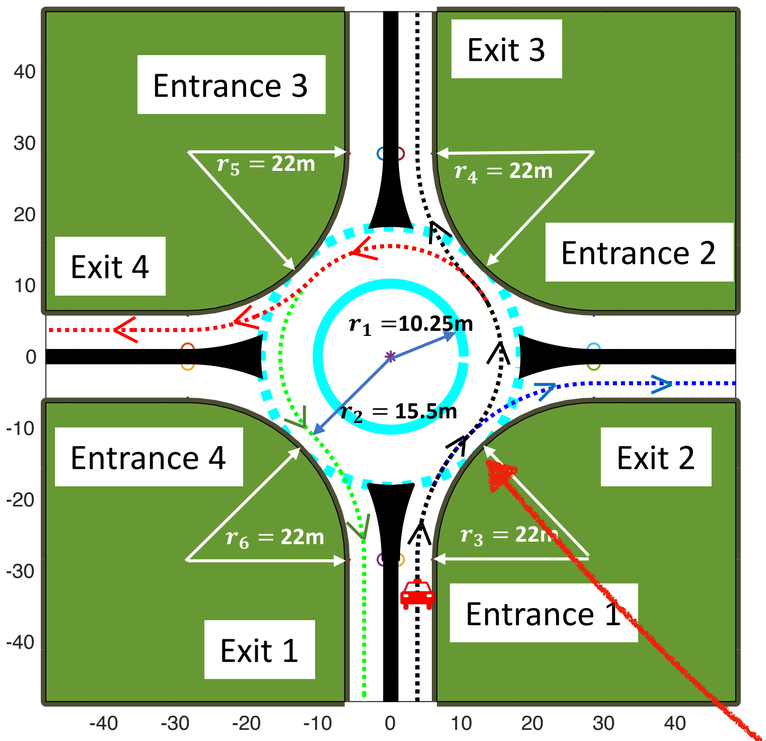
Tian et al., “Adaptive Game-Theoretic Decision Making for Autonomous Vehicle Control at Roundabouts” IEEE-CDC2018.

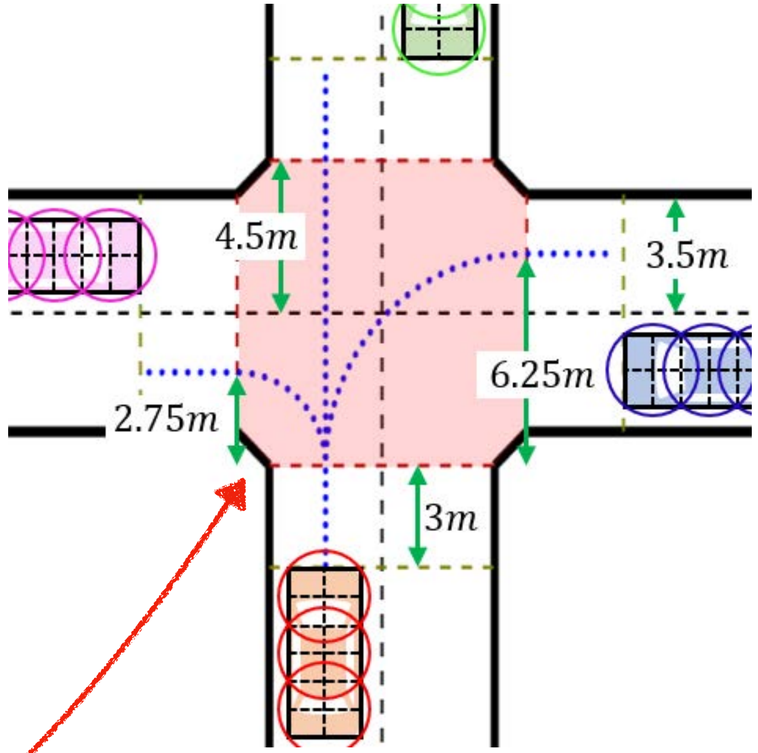
Which scenarios?





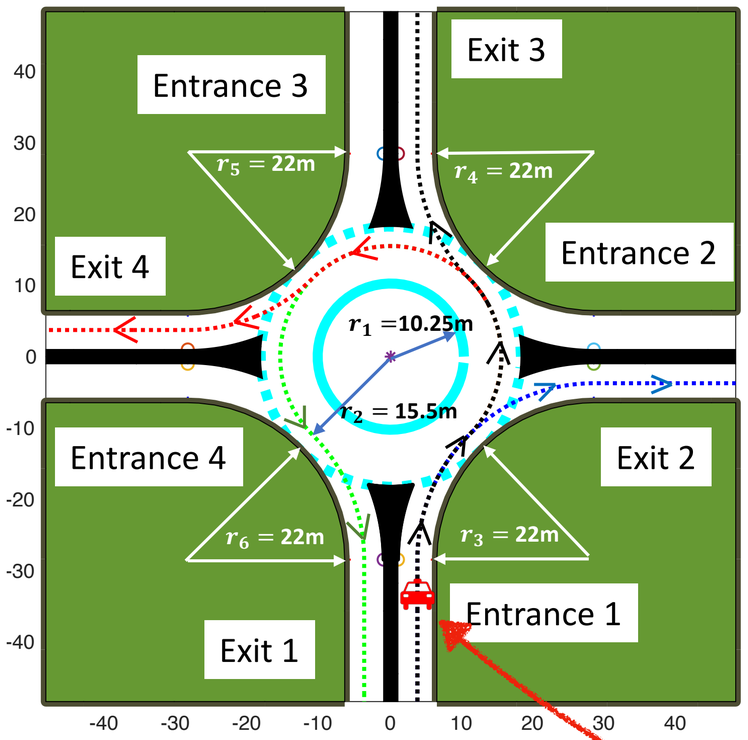
Which scenarios?

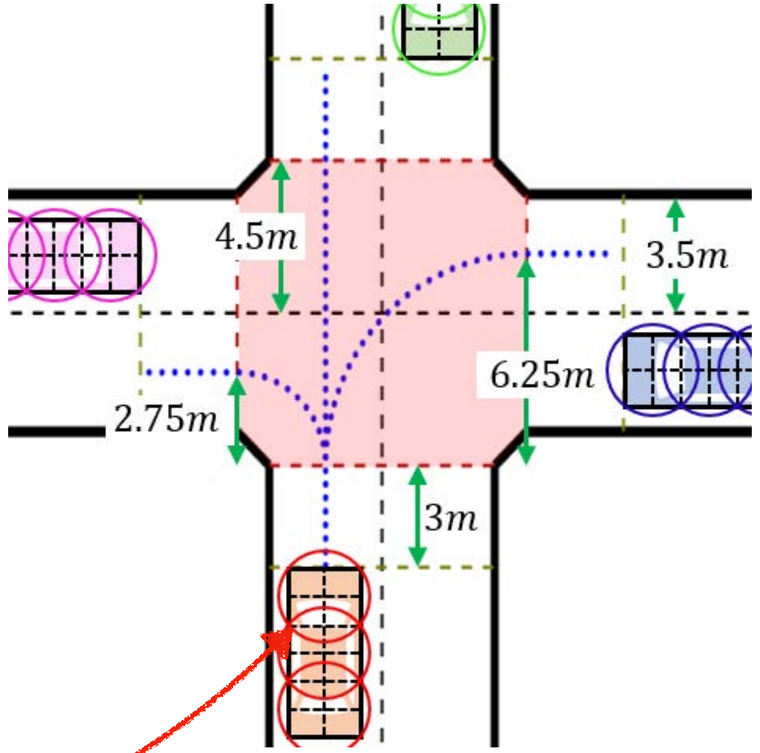




Fixed intersection-like road situations

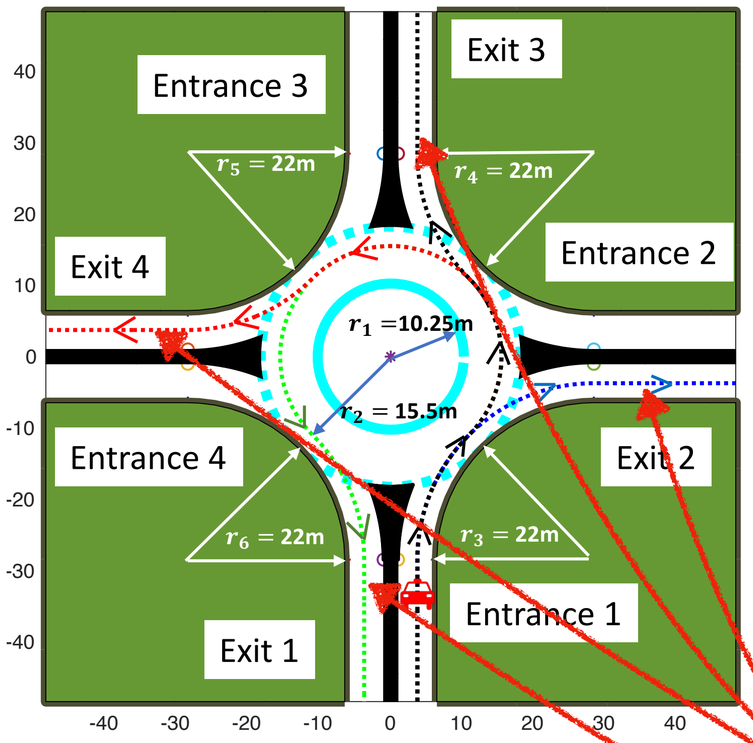
Which scenarios?

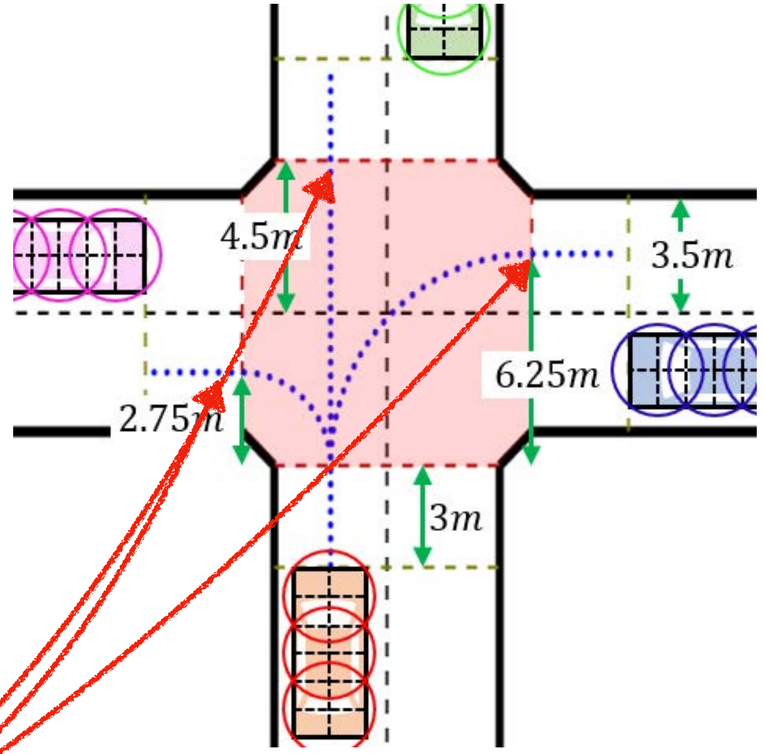




Autonomous vehicles « playing » with each other or with others types of vehicles

Which scenarios?





Their paths are fixed

NextStep$_i$: conf $*$ a $\longrightarrow$ conf

Best move?

The ego vehicle must choose its acceleration in such a way that:

- it follows its path,
- it optimises its time in the intersection,
- it does not collide with other vehicles,
- it respects the law,
- …

Since we have several things to optimise, we have to think about trade-off.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$
$$\text{with } \alpha_i + \beta_i = 1$$

Cost function

$$\mathbf{Cost}^i(\mathbf{conf}_1, \ldots, \mathbf{conf}_n) = \alpha_i \,.\, \mathbf{Cost}^i_{\mathbf{velo}} + \beta_i \,.\, \mathbf{Cost}^i_{\mathbf{safe}}$$
$$\text{with } \alpha_i + \beta_i = 1$$

The cost of a situation (either observed or predicted)
The lower, the better.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$

with $\alpha_i + \beta_i = 1$

Configurations of the vehicle $1, \ldots, n$ given by:

- Their positions (cartesian, polar coordinates)
- Their velocities
- Their situations in the intersection (entering, inside, exiting)

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$

with $\alpha_i + \beta_i = 1$

A cost function that describes how close to the limit velocity the vehicle is.
The lower, the closer.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$

with $\alpha_i + \beta_i = 1$

A cost function that describes how safe the vehicle is
Roughly speaking: how closed to the other vehicle it is
The lower, the safer.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$
$$\text{with } \alpha_i + \beta_i = 1$$

Coefficient that describes how much the vehicle cares about the optimising its velocity compared to its safety.
Roughly speaking: its an “aggressiveness” coefficient

Receding horizon cost

$$\textbf{HCost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n, (a^i_{j,s})_{1 \leq j \leq n, 0 \leq s \leq h}) = \sum_{s=0}^{h} \delta^h \,.\, \textbf{Cost}^i(\textbf{conf}_{1,s}, \ldots, \textbf{conf}_{n,s})$$

with:

$$\text{conf}_{j,0} = \text{conf}_j$$

$$\text{conf}_{j,s+1} = \text{NextStep}^i_j(\text{conf}_{j,s}, a^i_{j,s})$$

Receding horizon cost

$$\mathbf{HCost}^i(\mathbf{conf}_1, \ldots, \mathbf{conf}_n, (a^i_{j,s})_{1\leq j\leq n, 0\leq s\leq h}) = \sum_{s=0}^{h} \delta^h \,.\, \mathbf{Cost}^i(\mathbf{conf}_{1,s}, \ldots, \mathbf{conf}_{n,s})$$

with:

$$\mathrm{conf}_{j,0} = \mathrm{conf}_j$$

$$\mathrm{conf}_{j,s+1} = \mathrm{NextStep}^i_j(\mathrm{conf}_{j,s}, a^i_{j,s})$$

Accumulated cost of the all upcoming situations when the initial situation is $\mathrm{conf}_1, \ldots, \mathrm{conf}_n$ and if the vehicle i predicts the accelerations of all vehicles are given by $(a^i_{j,s})_{1\leq j\leq n, 0\leq s\leq h}$ and that the vehicle j follows the path given by $\mathrm{NextStep}^i_j$

Best global move = Nash equilibrium

A **game**:

- A set of **players** $P = \{1, \ldots, n\}$
  - Ex: the vehicles
- Each player i has a set of **possible moves** Γ_i
  - Ex: an acceleration profile $(a_s)_{0 \leq s \leq h}$
- Each player has a cost function it wants to minimise, of type:

$$H_i : \Gamma_1 \times \ldots \times \Gamma_n \to \mathbb{R}$$

  - Ex: the accumulated costs

What does it mean for the players to conjunctly optimise their cost?

$\Rightarrow$ best possible response: a move m_i for every player such that for any other move m_i':

$$H_i(m_1, \ldots, m_n) \leq H_i(m_1, \ldots, m_i', \ldots, m_n)$$

Nash equilibrium

Priority order and backward induction

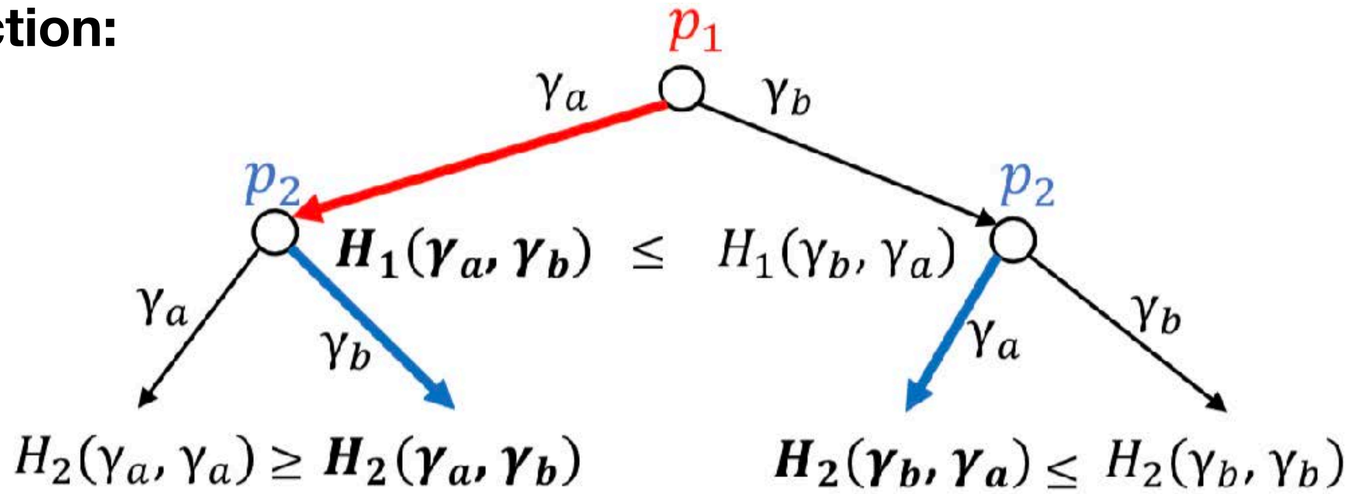
How to enforce the existence of a Nash equilibrium and compute it?

Idea: order the players, the smallest one chooses first, the second smallest chooses second, …

Assume: a total order $\preceq$ on P

Ex: $i \preceq j$ if i is more aggressive than j, or if the law tells that i has priority over j

Do backward induction:



Decision making procedure

Algorithm Decision making of the ego vehicle

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
    $X_j :=$ observe(j, t);
    NextStep$_j :=$ initial_path(j);
    HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order;
while I am still in the intersection **do**
    $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
    return $a_{\text{ego},0}$ as control input;
    $t := t +$ time_step;
    $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
    $X_j :=$ observe(j, t);
    if some $\widehat{X}_j$ are not close to X_j **then**
        for all $j \in N$ **do**
            NextStep$_j :=$ update_path(j);
            HCost$_j :=$ update_cost(j);
        end for;
        $\preceq :=$ update_order;
    end if
    $N :=$ update_neighbors;
end while

Decision making procedure

Algorithm Decision making of the ego vehicle

1: $t := 0$;
2: $N :=$ initial_neighbors;
3: **for all** $j \in N$ **do**
4: $X_j :=$ observe(j, t);
5: NextStep$_j :=$ initial_path(j);
6: HCost$_j :=$ initial_cost(j);
7: **end for**;
8: $\preceq :=$ initial_order;
9: **while** I am still in the intersection **do**
10: $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
11: **return** $a_{\text{ego},0}$ as control input;
12: $t := t +$ time_step;
13: $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
14: $X_j :=$ observe(j, t);
15: **if** some $\widehat{X}_j$ are not close to X_j **then**
16: **for all** $j \in N$ **do**
17: NextStep$_j :=$ update_path(j);
18: HCost$_j :=$ update_cost(j);
19: **end for**;
20: $\preceq :=$ update_order;
21: **end if**
22: $N :=$ update_neighbors;
23: **end while**

Initialization phase

Restrict the set of vehicles you are considering

Observe their current configuration

Guess their path

Guess their cost function

Initialise the order of the vehicles

Decision making procedure

Algorithm Decision making of the ego vehicle

```
1:  t := 0;
2:  N := initial_neighbors;
3:  for all j ∈ N do
4:      X_j := observe(j, t);
5:      NextStep_j := initial_path(j);
6:      HCost_j := initial_cost(j);
7:  end for;
8:  ≼ := initial_order;
9:  while I am still in the intersection do
10:     (a_{j,s})_{j,s} := nash_equilibrium(HCost_j, ≼);
11:     return a_{ego,0} as control input;
12:     t := t + time_step;
13:     X̂_j := NextStep_j(X_j, a_{j,0});
14:     X_j := observe(j, t);
15:     if some X̂_j are not close to X_j then
16:         for all j ∈ N do
17:             NextStep_j := update_path(j);
18:             HCost_j := update_cost(j);
19:         end for;
20:         ≼ := update_order;
21:     end if
22:     N := update_neighbors;
23: end while
```

Initialise the order of the vehicles

Order the vehicles using the right-of-way:

- Vehicles already in the intersection have more priority.
- Vehicles on your left-hand-side have more priority.
- …

Decision making procedure

Algorithm Decision making of the ego vehicle

```
1:  t := 0;
2:  N := initial_neighbors;
3:  for all j ∈ N do
4:      X_j := observe(j, t);
5:      NextStep_j := initial_path(j);
6:      HCost_j := initial_cost(j);
7:  end for;
8:  ⪯ := initial_order;
9:  while I am still in the intersection do
10:     (a_{j,s})_{j,s} := nash_equilibrium(HCost_j, ⪯);
11:     return a_{ego,0} as control input;
12:     t := t + time_step;
13:     X̂_j := NextStep_j(X_j, a_{j,0});
14:     X_j := observe(j, t);
15:     if some X_j are not close to X_j then
16:         for all j ∈ N do
17:             NextStep_j := update_path(j);
18:             HCost_j := update_cost(j);
19:         end for;
20:         ⪯ := update_order;
21:     end if
22:     N := update_neighbors;
23: end while
```

Main loop

Compute the best responses from your guesses

Your best response is your new control input

Compute your guess of the next configurations

Observe the real next configurations

In case your guesses are far from reality, make some updates

Decision making procedure

Algorithm Decision making of the ego vehicle

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
    $X_j :=$ observe(j, t);
    NextStep$_j :=$ initial_path(j);
    HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order;
while I am still in the intersection **do**
    $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
    return $a_{\text{ego},0}$ as control input;
    $t := t +$ time_step;
    $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
    $X_j :=$ observe(j, t);
    if some $\widehat{X}_j$ are not close to X_j **then**
        for all $j \in N$ **do**
            NextStep$_j :=$ update_path(j);
            HCost$_j :=$ update_cost(j);
        end for;
        $\preceq :=$ update_order;
    end if
    $N :=$ update_neighbors;
end while

In case your guesses are far from reality, make some updates

Compute the order that fit the situation the most:

- For every possible order, compute:

$$(a_{j,s}) = \text{nash\_equilibrium}(\text{HCost}_j, \preceq)\,.$$

- Then compute the corresponding predicted configuration:

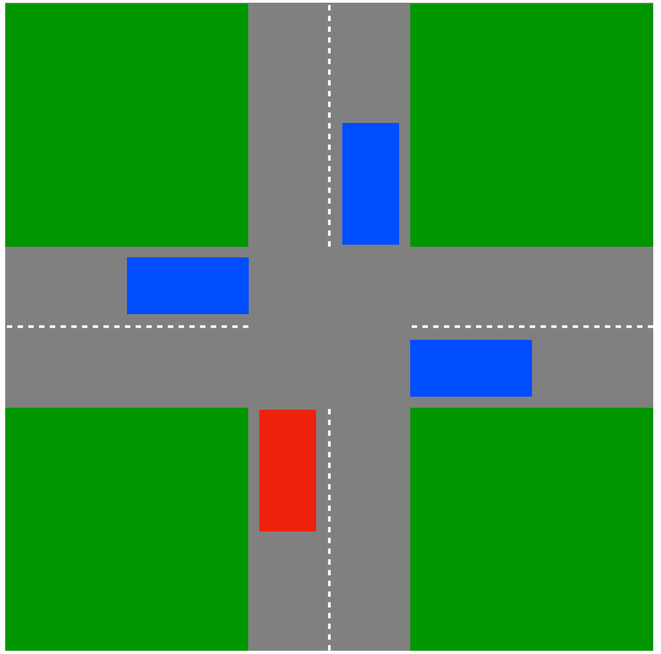
$$\widehat{X}_j(\preceq) = \text{NextStep}_j(X_j, a_{j,0})\,.$$

- Choose $\preceq$ whose predictions are the closest to the observation X_j.

Dealing with deadlocks

A **dead-lock**: a situation where all the vehicles are waiting for others to take a decision.

In our case: symmetric situations where every vehicles are stopped at the entrance of the intersection.



How to solve it?

⇒ theoretically insolvable with deterministic systems

⇒ add probabilities: when a deadlock is detected, take a decision with some probability

Behaviour of the adversaries

Algorithm Decision making

1: $t := 0$;
2: $N :=$ initial_neighbors;
3: **for all** $j \in N$ **do**
4: $\quad X_j :=$ observe(j, t);
5: $\quad$ NextStep$_j :=$ initial_path(j);
6: $\quad$ HCost$_j :=$ initial_cost(j);
7: **end for**;
8: $\preceq :=$ initial_order;
9: **while** I am still in the intersection **do**
10: $\quad (a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
11: $\quad$ **return** $a_{\text{ego},0}$ as control input;
12: $\quad t := t +$ time_step;
13: $\quad \widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
14: $\quad X_j :=$ observe(j, t);
15: $\quad$ **if** some $\widehat{X}_j$ are not close to X_j **then**
16: $\quad\quad$ **for all** $j \in N$ **do**
17: $\quad\quad\quad$ NextStep$_j :=$ update_path(j);
18: $\quad\quad\quad$ HCost$_j :=$ update_cost(j);
19: $\quad\quad$ **end for**;
20: $\quad\quad \preceq :=$ update_order;
21: $\quad$ **end if**
22: $\quad N :=$ update_neighbors;
23: **end while**

ego/angelic

Right of way

Fitting

Behaviour of the adversaries

Algorithm Decision making

```
1:  t := 0;
2:  N := initial_neighbors;
3:  for all j ∈ N do
4:      X_j := observe(j, t);
5:      NextStep_j := initial_path(j);
6:      HCost_j := initial_cost(j);
7:  end for;
8:  ⪯ := initial_order;
9:  while I am still in the intersection do
10:     (a_{j,s})_{j,s} := nash_equilibrium(HCost_j, ⪯);
11:     return a_{ego,0} as control input;
12:     t := t + time_step;
13:     X̂_j := NextStep_j(X_j, a_{j,0});
14:     X_j := observe(j, t);
15:     if some X̂_j are not close to X_j then
16:         for all j ∈ N do
17:             NextStep_j := update_path(j);
18:             HCost_j := update_cost(j);
19:         end for;
20:         ⪯ := update_order;
21:     end if
22:     N := update_neighbors;
23: end while
```

Demonic

I have priority

I do not care

Behaviour of the adversaries

Algorithm Decision making

```
1:  t := 0;
2:  N := initial_neighbors;
3:  for all j ∈ N do
4:      X_j := observe(j, t);
5:      NextStep_j := initial_path(j);
6:      HCost_j := initial_cost(j);
7:  end for;
8:  ≼ := initial_order;
9:  while I am still in the intersection do
10:     (a_{j,s})_{j,s} := nash_equilibrium(HCost_j, ≼);
11:     return a_{ego,0} as control input;
12:     t := t + time_step;
13:     X̂_j := NextStep_j(X_j, a_{j,0});
14:     X_j := observe(j, t);
15:     if some X̂_j are not close to X_j then
16:         for all j ∈ N do
17:             NextStep_j := update_path(j);
18:             HCost_j := update_cost(j);
19:         end for;
20:         ≼ := update_order;
21:     end if
22:     N := update_neighbors;
23: end while
```

Intermediate

I have priority

Fitting

Behaviour of the adversaries

Algorithm Random decision making

1: **while** I am still in the intersection **do**
2: **choose randomly** an acceleration a
3: **return** a as control input;
4: **end while**

Irrational

Simulation results

Roundabout

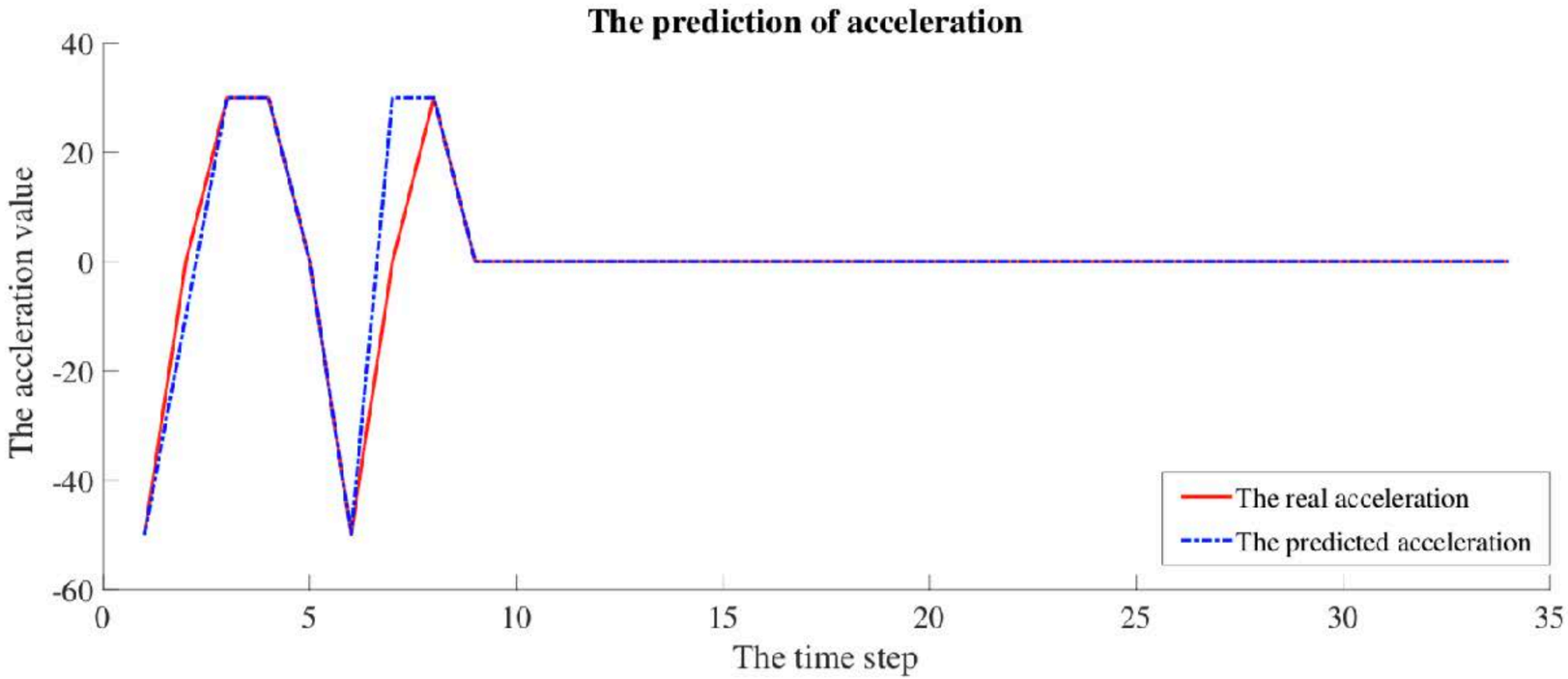
Case	Collision rate(%)	Min dist.(m)	Avg. Total time(s)
4	0	14.49	10.4
5	0	9.81	12.0
6	0	8.94	13.3
7	0	8.90	14.4
8	0	8.93	15.1

Unsignalized intersection

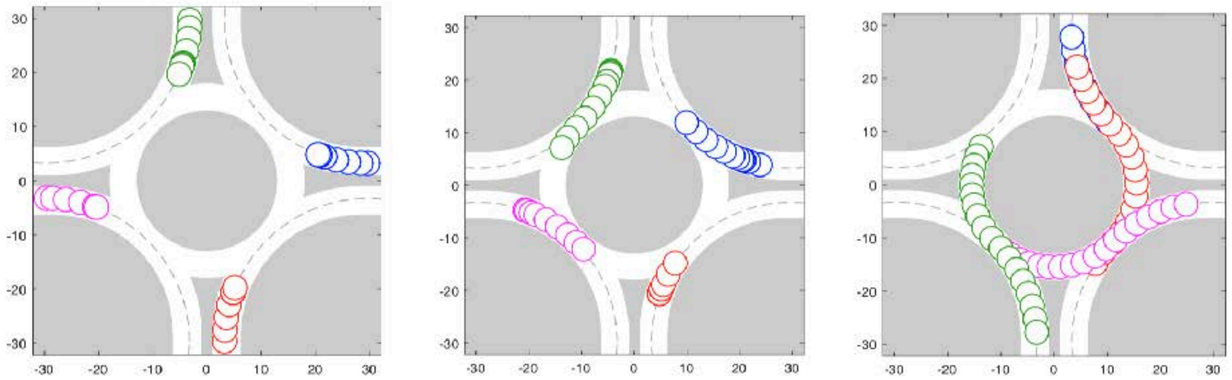
Case	Collision rate(%)	Congestion rate(%)	Avg. Total time steps
1	0	0	56.87 (5.687s)
2	0	0.2	53.98 (5.398s)
3	0	0	59.09 (5.909s)
4	0.4	4.0	91.88 (9.988s)
1'	0	0.5	55.43 (5.543s)
2'	0	1.4	50.58 (5.058s)
3'	0	9.4	55.81 (5.581s)
4'	1.1	14.3	75.82 (7.582s)

1: four angelic
2: three angelic + one demonic
3: four intermediate
4: three intermediate + one irrational

Simulation results



Simulation results



(a) time step ≤ 6 (b) $7 \leq$ time step ≤ 15 (c) time step ≥ 16