Last Time:

- ☐ formalizing safe sets
- ☐ computing safety filters

Lecture 4

EAIS S'25

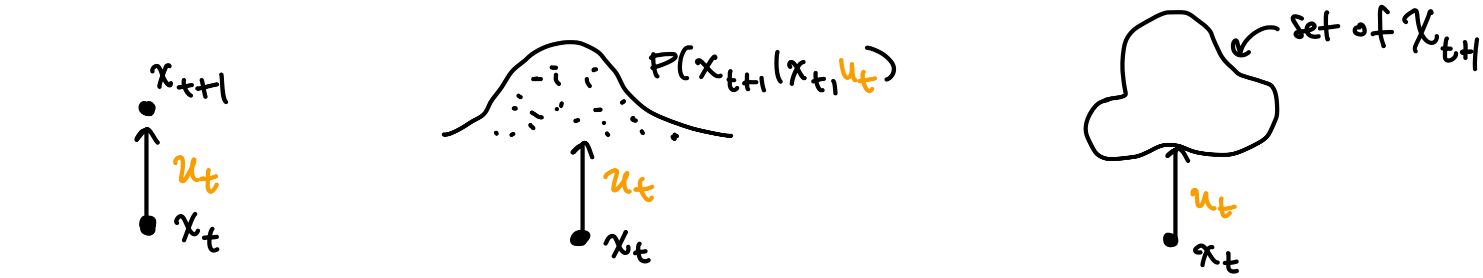
Andrea Bajcsy

This Time:

- ☐ robustifying safety - why?
- ☐ zero-sum dynamic games
- ☐ dynamic programming for games

Robustifying Safety

So far, we have assumed that our dynamical system perfectly evolves via $\dot{x} = f(x,u)$ with no uncertainty.
This isn't realistic for many real-world scenarios (e.g. friction!)



No uncertainty | Probabilistic | Non-deterministic

There are two ways to model uncertainty:

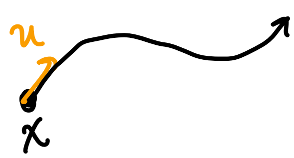
1) probabilistic uncertainty (i.e. "I have observed data")

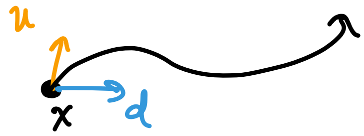
2) non-derministic uncertainty (i.e "I have minimal additional info.")

How should we handle the design of our safety filter (+ analysis) to handle uncertainty?

Typically, we do this via modelling another "input" that influences the state evolution:

BEFORE: → NOW:





$\dot{x} = f(x, u)$

$\dot{x} = f(x, u, d)$

"robot" "ego"

"disturbance"

In robust safety, we take a non-deterministic view of uncertainty and assume that $d \in \mathcal{D}$ is chosen from some bounded set and we want our robot to be <u>ROBUST</u> to the <u>WORST POSSIBLE</u> sequence of d's!

<u>ROBUST BACKWARDS REACHABLE TUBE</u> (BRT) of a set $\mathcal{F} \subset \mathcal{X}$ and dynamical system $\dot{x} = f(x, u, d)$ is:

$$BRT(t) := \Big\{ x \in \mathcal{X} : \underbrace{\forall u(\cdot) \in \mathbb{U}_t^T}_{\text{for all things the robot could do}},\ \underbrace{\exists\, d(\cdot) \in \mathbb{D}_t^T}_{\text{there is something the disturbance could do} \ldots},\ \underbrace{\mathbf{x}_{x,t}^{u,d}(\tau) \in \mathcal{F}}_{\text{that leads the robot to failure}} \text{ for some time } \tau \in [t, T] \Big\}$$

This is the set of all starting states from which no matter the controller's effort, the disturbance can push system into $\mathcal{F}$.

The way we will formulate an "optimal control" problem whose solution represents this unsafe set will be via:

$$\underbrace{\text{zero-sum}}_{\text{there is a winner + loser}}\ \ \underbrace{\text{dynamic}}_{\text{game evolves over time}}\ \ \underbrace{\text{games}}_{\text{result/outcome depends on 2+ players/inputs}}$$

Our "game" formulation could look something like this:

$$V(x,t) \equiv \max_{u(\cdot) \in \mathbb{U}_t^T} \; \min_{d(\cdot) \in \mathbb{D}_t^T} J(x, u(\cdot), d(\cdot))$$

$$\text{s.t. } \dot{x}(\tau) = f(x(\tau), u(\tau), d(\tau)) \quad \forall \tau \in [t, T]$$

$$u(\tau) \in \mathcal{U}$$

$$d(\tau) \in \mathcal{D}$$

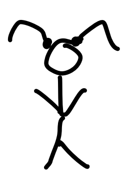
Here, the robot (u) is trying to maximize the objective $J(\cdot,\cdot,\cdot)$ and the other player (d) is minimizing. In other words, the robot is optimizing the <u>worst-case objective</u>

Importance of Information Patterns

When we have 2 players reacting to each other, their optimal strategy will depend on what information they each have access to.

Example Suppose there are 2 boxes, each with 2 slots. each slot contains prize money. Player A (You ü) wants to maximize prize money while Player B (competition organizer) wants to minimize Player A's prize money.

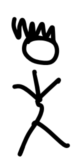
Box 1		Box 2	
2,000	10	1	6,000
s1	s2	s1	s2



Player A

$u \in [\text{Box } 1, \text{Box } 2]$

you choose <u>Box</u>

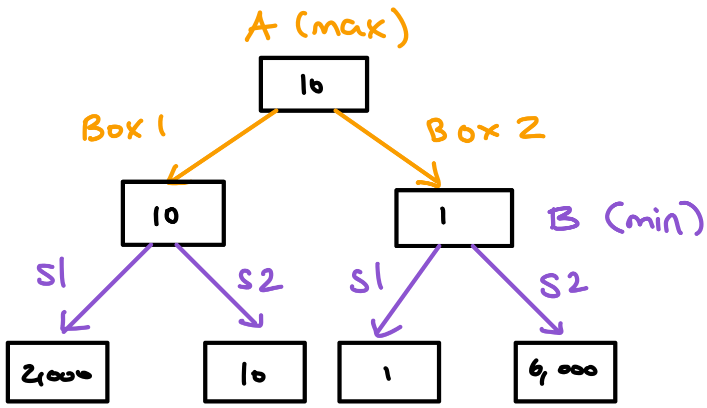


Player B

$d \in [\text{slot } 1, \text{slot } 2]$

they choose <u>slot</u>

Suppose player A goes first:



Box 1

2,000	10
s1	s2

Box 2

1	6,000
s1	s2

Best outcome for A is to pick Box 1 & get Rew = $10.

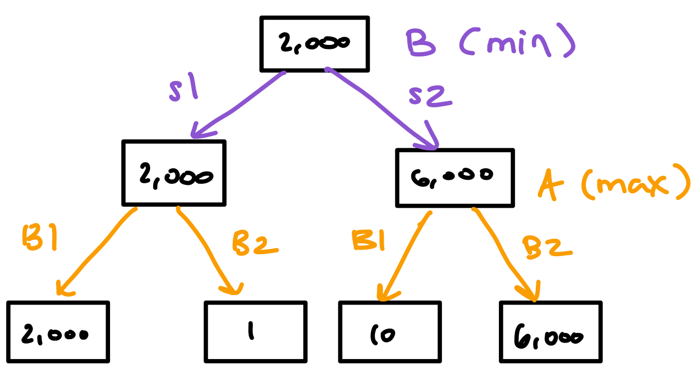
Mathematically:

$$\overline{\max_{u\in[B1,B2]}}\left(\min_{d\in[s1,s2]} J(u,d)\right) = 10$$

"computed 2nd" but "plays first"

"computed first" but "plays 2nd" ⇒ "best response" strategy

Suppose Player B goes first:



Box 1

2,000	10
s1	s2

Box 2

1	6,000
s1	s2

Best strategy for player B is to pick slot 1 and pay player A a reward of \$ 2,000.

$$\min_{d \in [S1, S2]} \left(\max_{u \in [B1, B2]} J(u, d) \right) = \$\,2{,}000$$

This phenomenon we just saw can be stated via

<u>minimax inequality:</u>

$$\max_{a} \left(\min_{b} J(a,b) \right) \leq \min_{b} \left(\max_{a} J(a,b) \right)$$

<u>Von Neumann 1928</u>: equal when A, B are compact, convex sets + $J(\cdot, b)$ is <u>concave</u> for fixed b

$J(a, \cdot)$ is <u>convex</u> for fixed a

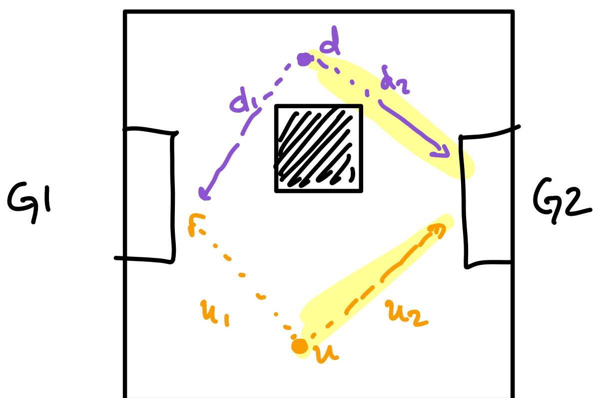
In dynamic games the outcome depends on <u>WHEN</u> and <u>WITH WHAT INFORMATION</u> each player decides their inputs

just talked about this (order of play)

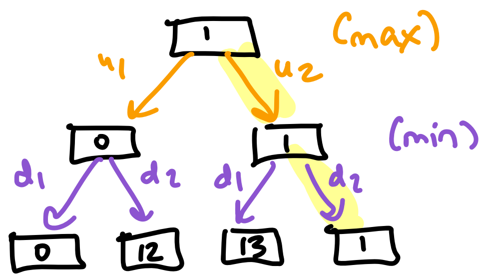
NOW.

<u>OPEN-LOOP INFO PATTERN:</u>

$$\max_{u(\cdot) \in \mathbb{U}_t^T} \; \min_{d(\cdot) \in \mathbb{D}_t^T} J(x, u(\cdot), d(\cdot))$$



Here u wants to reach G1 <u>OR</u> G2 without being intercepted by d. BUT d wants to intercept u.



} time [0,T]

} time [0,T]

J ↪ "dist btwn. agents"

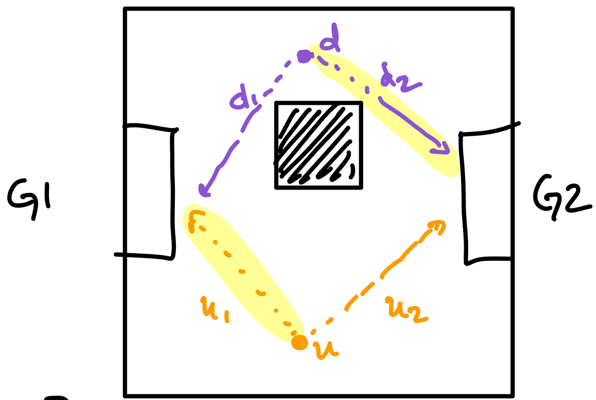
u(·) "declares" entire signal and then d(·) gets to see this and <u>intercepts</u>!

⊛ OVERLY PESSIMISTIC!

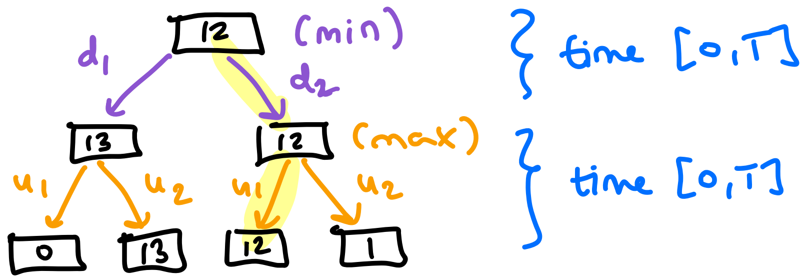
let's swap the order of play again, but we are playing over continuous signals (or sequences of decisions):

OPEN-LOOP INFO PATTERN:

$$\min_{d(\cdot)\in \mathbb{D}_t^T} \max_{u(\cdot)\in \mathbb{U}_t^T} J(x, u(\cdot), d(\cdot))$$



$d(\cdot)$ "declares" entire signal and then $u(\cdot)$ gets to see this and picks the other goal!



J ↺ "dist btwn. agents"

(*) OVERLY OPTIMISTIC!

CLOSED-LOOP INFORMATION PATTERNS

The above formulation is not suitable for many practical systems. We would like the controller (u) to ADAPT over time as system evolves, BUT respect the fact that @ any time t, we only have information up to time t.

We can model this by solving for feedback policies:

$$V(x,t) \equiv \max_{\pi_u} \min_{\pi_d} J(x, \pi_u(\cdot), \pi_d(\cdot))$$

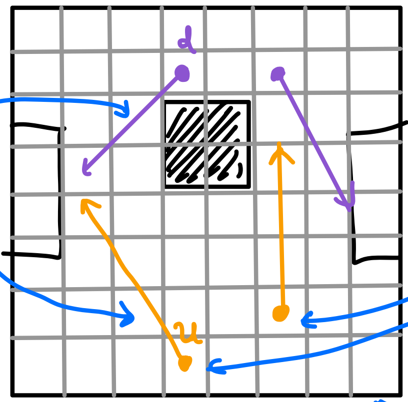
$\pi_d : X \to D$

$\pi_u : X \to U$

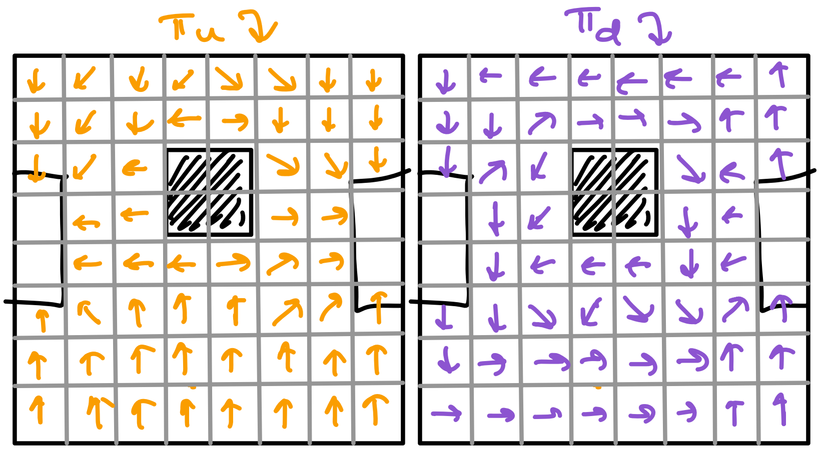
Before $\left(\max_{u(\cdot) \in \mathbb{U}_t^T} \; \min_{d(\cdot) \in \mathbb{D}_t^T} \cdots \right)$

Now $\left(\max_{\pi_u} \; \min_{\pi_d} \cdots \right)$

signal from t to T



2 different initial states for player u



from any initial state (denoted by • icon), each agent has to commit to entire control signal or sequence that they cannot change in future.

$\Rightarrow$ this is $\boxed{\text{open-loop}}$

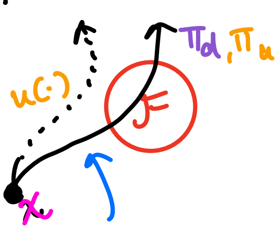
for each initial state, the players commit to a policy - or a map from each state to their action.

$\Rightarrow$ this is $\boxed{\text{closed-loop}}$

(*) NOTE: for this illustrative example, since u and d are 2 separate agents, the state space is $\mathcal{X} := \mathcal{X}_u \times \mathcal{X}_d$ and policies map from where BOTH players are, to actions:

$$\pi_u(x_u, x_d) \to u \quad , \quad \pi_d(x_u, x_d) \to d$$

Bringing this back to safety analysis, we just have to choose a objective function which lets us remember the closest we ever got to failure:



BEFORE (single-agent)

NOW (game)

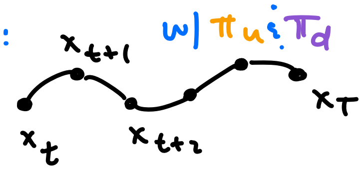
$$J(x, \pi_u, \pi_d, t) := \underbrace{\min_{\tau \in [t,T]} \ell\left(\mathrm{x}_{x,t}^{\pi_u \pi_d}(\tau)\right)}_{\text{minimum future distance influenced by } u \text{ and } d}$$

We can apply the game-theoretic principle of optimality called the <u>Tenet of Transition</u> by Rufus Isaacs:

"If play proceeds from one position (state) to a second, and V is thought of as known to the second, then it is determined at the first by demanding that players optimize (i.e. make minimax) the increment of V during the transition."

$\boxed{\text{example}}$ (discrete-time)

trajectory:



$$V_t(x_t) := \min_{\pi_u(x)} \max_{\pi_d(x)} \min_{\tau \in \{t,\dots T\}} \ell(x_\tau^{u,d})$$

$$= \min_{u^t} \max_{d^t} \dots \min_{u^T} \max_{d^T} \min_{\tau \in \{t,\dots T\}} \ell(x_\tau^{u,d})$$

$$= \min_{u^t} \max_{d^t} \min \Big\{ \ell(x_t), \underbrace{\min_{u^{t+1}} \max_{d^{t+1}} \dots \min_{u^T} \max_{d^T} \min_{s \in \{t+1,\dots T\}} \ell(x_s^{u,d})}_{:= V_{t+1}(f(x_t,u_t,d_t)) \text{ by Tenet of Transition}} \Big\}$$

$$= \min_{u^t} \max_{d^t} \min \Big\{ \ell(x_t), V_{t+1}(f(x_t, u_t, d_t)) \Big\}$$

$$= \min \Big\{ \ell(x_t), \min_{u^t} \max_{d^t} V_{t+1}(f(x_t, u_t, d_t)) \Big\}$$

Applying this to the continuous or discrete-time zero sum safety critical games we formulated above, we get two key equations that allow us to extend dynamic programming tools to the robust (dynamic game) setting:

b/c it's a game!

Hamilton-Jacobi-Isaacs Variational Inequality (HJ-VI)

$$\min\left\{ \ell(x) - V(x,t),\ \frac{\partial V}{\partial t} + \max_{u \in \mathcal{U}} \boxed{\min_{d \in \mathcal{D}}} \frac{\partial V}{\partial x} \cdot f(x,u,d) \right\} = 0$$

$$V(x,T) = \ell(x)$$

add this & solve "subgame" over just instantaneous actions

ROBUST SAFETY BACKUP (discrete-time):

$$V_t(x_t) = \min\left\{ \ell(x_t),\ \max_{u_t} \min_{d_t} V_{t+1}(f(x_t, u_t, d_t)) \right\}$$

$$V_T(x_T) = \ell(x)$$

SUMMARY

Continuous-Time

Problem

$$V(x,t) := \max_{\pi_u(x)} \min_{\pi_d(x)} \min_{\tau \in [t,T]} \ell(x_\tau^{u,d}(\tau))$$

$\min_{\pi_d(x)}$ for robust

Dynamic Programming "Backup"

$$\min\left\{ \ell(x) - V(x,t),\ \frac{\partial V(x,t)}{\partial t} + \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \frac{\partial V(x,t)}{\partial x} \cdot f(x,u,d) \right\} = 0$$

$$V(x,T) = \ell(x)$$

$\min_{d \in \mathcal{D}}$ for robust

Discrete-Time

Problem

$$V_t(x_t) := \max_{\pi_u(x)} \min_{\pi_d(x)} \min_{\tau \in \{t,\dots,T\}} \ell(x_\tau^{u,d})$$

$\min_{\pi_d(x)}$ for robust

Dynamic Programming "Backup"

$$V_t(x_t) = \min\left\{ \ell(x_t),\ \max_{u_t \in \mathcal{U}} \min_{d_t \in \mathcal{D}} V_{t+1}(f(x_t, u_t, d_t)) \right\}$$

$$V_T(x_T) = \ell(x)$$

$\min_{d_t \in \mathcal{D}}$ for robust

Numerical comparison of Robust vs. Non-Robust Unsafe Set

Recall: state is (x, y, θ) of Dubins' car

dynamics are
$$\dot{x} = \underbrace{v}_{1.5}\cos\theta + d_x \qquad d \in [-0.5, 0.5] \equiv \mathcal{D}$$
$$\dot{y} = v\sin\theta + d_y$$
$$\dot{\theta} = u \qquad u \in [-0.5, 0.5] \equiv \mathcal{U}$$

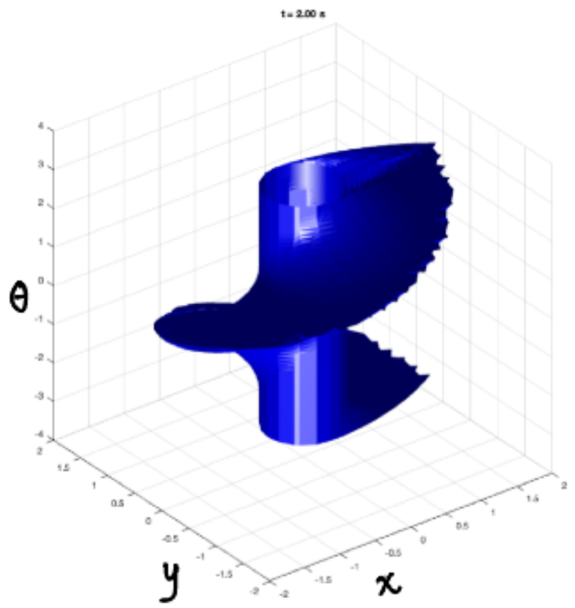
θ

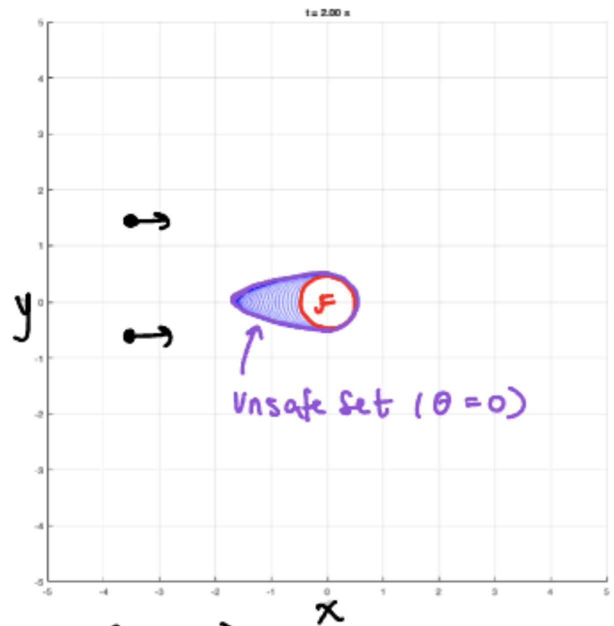
(x, y)

failure is cylinder @ origin with radius 0.3

Non-Robust BRT

$$\max_{u} \min_{\tau \in [t, T]} \ell(x(\tau))$$





Robust BRT

$$\max_{\pi_u} \min_{\pi_d} \min_{\tau \in [t, T]} \ell(x(\tau))$$