A Compositional Framework for Non-Convex Sequential Decision Problems

Matthew Hale

AFOSR DSCT Review
August 28, 2024

AFOSR YIP Grant #FA9550-23-1-0120

Project title: A Morse-Theoretic Approach to Non-Convex Optimization

Georgia Institute of Technology

This talk is about joint work

- Everything in this talk is joint with

James Fairbanks (UF MAE)

Tyler Hanks (UF CISE)

Decision problems often have temporal coupling

- Example #1: State space control systems

$$\begin{aligned} \underset{u(0),\ldots,u(T)}{\text{minimize}} \; & \sum_{t=0}^{T} \ell\big(x(t), u(t)\big) \\ \text{subject to } \; & x(t+1) = f\big(x(t), u(t)\big) \\ & x(0) = x_0 \end{aligned}$$

- Example #2: Markov decision processes

 →

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o

$$P_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_a = a)$$

- Example #3: Various classes of games

1 / 2	*L*	*M*	*R*
T	$8, 8$	$0, 0$	$1, 9^*$
M	$0, 0$	$5^*, 5^*$	$0, 0$
B	$9^*, 1$	$0, 0$	$3^*, 3^*$

Decision problems often have temporal coupling

- Example #1: State space control systems

$$\boxed{\begin{aligned} &\underset{u(0),\dots,u(T)}{\text{minimize}} \sum_{t=0}^{T} \ell\big(x(t), u(t)\big) \\ &\text{subject to } x(t+1) = f\big(x(t), u(t)\big) \\ &x(0) = x_0 \end{aligned}}$$

Today's talk will focus on this!

- Example #2: Markov decision processes

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o

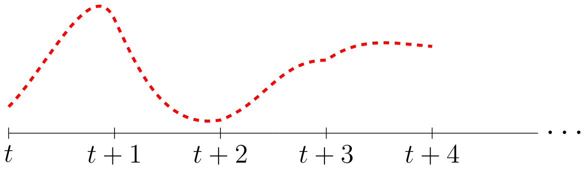
$$P_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_a = a)$$

- Example #3: Various classes of games

1 / 2	*L*	*M*	*R*
T	8, 8	0, 0	1, 9*
M	0, 0	5*, 5*	0, 0
B	9*, 1	0, 0	3*, 3*

We will examine MPC for nonlinear systems

- Model-predictive control (MPC) optimizes inputs over a finite lookahead window

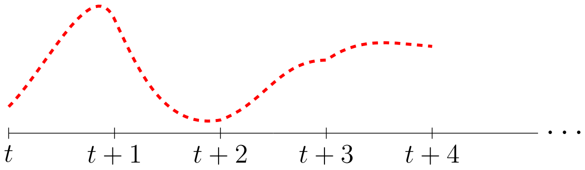


- A standard MPC problem formulation is then

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{k=t}^{t+N} \ell\big(x(k), u(k)\big) \\ \text{subject to} & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \quad k = t, t+1, \ldots, t+N \end{array}$$

We will examine MPC for nonlinear systems

- Model-predictive control (MPC) optimizes inputs over a finite lookahead window



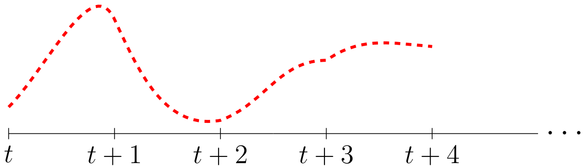
- A standard MPC problem formulation is then

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{k=t}^{t+N} \ell\big(x(k), u(k)\big) \\
\text{subject to} & x(k+1) = f\big(x(k), u(k)\big) \\
 & g\big(x(k), u(k)\big) \le 0 \quad k = t, t+1, \ldots, t+N
\end{array}
$$

- A decision at one time parameterizes the constraints at the next time

We will examine MPC for nonlinear systems

- Model-predictive control (MPC) optimizes inputs over a finite lookahead window



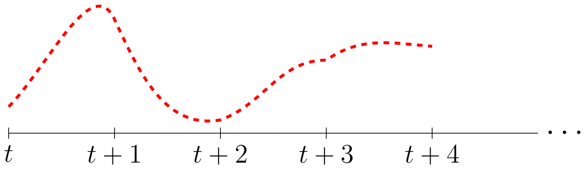
- A standard MPC problem formulation is then

$$\begin{aligned} \text{minimize} \quad & \sum_{k=t}^{t+N} \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \quad k = t, t+1, \ldots, t+N \end{aligned}$$

- A decision at one time parameterizes the constraints at the next time
- Once $u^*(k)$ is computed and applied, we get the state $x(k+1) = f\big(x(k), u^*(k)\big)$
- That state is in the next constraint: $x(k+2) = f\Big(f\big(x(k), u^*(k)\big), u(k+1)\Big)$

We will examine MPC for nonlinear systems

- Model-predictive control (MPC) optimizes inputs over a finite lookahead window



- A standard MPC problem formulation is then

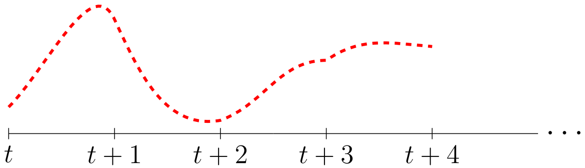
Can be non-convex for non-linear systems!

$$\left\{\begin{array}{ll} \text{minimize} & \displaystyle\sum_{k=t}^{t+N} \ell\big(x(k), u(k)\big) \\ \text{subject to} & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \quad k = t, t+1, \ldots, t+N \end{array}\right.$$

- A decision at one time parameterizes the constraints at the next time
- Once $u^*(k)$ is computed and applied, we get the state $x(k+1) = f\big(x(k), u^*(k)\big)$
- That state is in the next constraint: $x(k+2) = f\Big(f\big(x(k), u^*(k)\big), u(k+1)\Big)$

We will examine MPC for nonlinear systems

- Model-predictive control (MPC) optimizes inputs over a finite lookahead window



- A standard MPC problem formulation is then

Can be non-convex for non-linear systems!

$$\begin{aligned} \text{minimize} \quad & \sum_{k=t}^{t+N} \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \le 0 \quad k = t, t+1, \ldots, t+N \end{aligned}$$

- A decision at one time parameterizes the constraints at the next time
- Once $u^*(k)$ is computed and applied, we get the state $x(k+1) = f\big(x(k), u^*(k)\big)$
- That state is in the next constraint: $x(k+2) = f\Big(f\big(x(k), u^*(k)\big), u(k+1)\Big)$

Question for this talk

How can we model this temporal coupling in MPC?

We will use category theory to answer this question

- Why category theory?

We will use category theory to answer this question

- Why category theory?
    1. It's a natural language for composition

We will use category theory to answer this question

- Why category theory?
  1. It's a natural language for composition
  2. Abstraction can offer new insights and extensions

We will use category theory to answer this question

- Why category theory?
  1. It's a natural language for composition
  2. Abstraction can offer new insights and extensions
  3. It offers an ecosystem to plug into

E.g., other categories:

Dynam

Gph

Man

We will use category theory to answer this question

- Why category theory?
  1. It's a natural language for composition
  2. Abstraction can offer new insights and extensions
  3. It offers an ecosystem to plug into

E.g., other categories:

Dynam

Gph

Man

- What does it take to make a category?
  - We need (i) objects, (ii) morphisms, and (iii) a way to compose morphisms
  - E.g., in Set the objects are sets and the morphisms are total functions

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "g \circ f"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

Sets are red
Functions are blue

We can draw from classic work of Rockafellar

- In 1970, Rockafellar proposed "bifunctions" for convex problems[1]
- We replace

$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \qquad \text{with} \qquad \begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \leq y_1 \\ & h(x) = y_2 \end{aligned}$$

[1] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

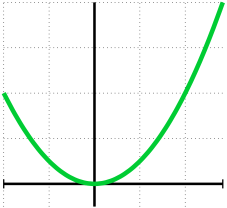
We can draw from classic work of Rockafellar

- In 1970, Rockafellar proposed "bifunctions" for convex problems[1]
- We replace

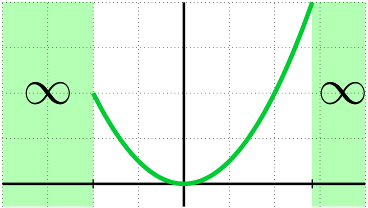
$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \le \color{red}{0} \\ & h(x) = \color{red}{0} \end{aligned} \qquad \text{with} \qquad \begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \le \color{red}{y_1} \\ & h(x) = \color{red}{y_2} \end{aligned}$$

- Then with $y = (y_1^T, y_2^T)^T$ we form the bifunction

$$B(x,y) = \begin{cases} f(x) & g(x) \le y_1, h(x) = y_2 \\ \infty & \text{otherwise} \end{cases}$$



becomes



[1] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

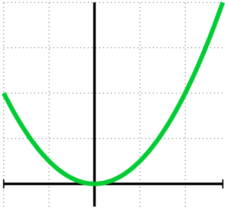
We can draw from classic work of Rockafellar

- In 1970, Rockafellar proposed "bifunctions" for convex problems[1]
- We replace

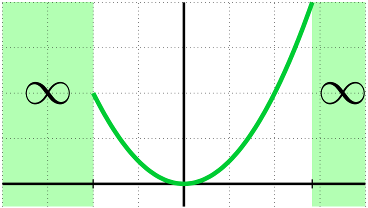
$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \le {\color{red}0} \\ & h(x) = {\color{red}0} \end{aligned} \qquad \text{with} \qquad \begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \le {\color{red}y_1} \\ & h(x) = {\color{red}y_2} \end{aligned}$$

- Then with $y = (y_1^T, y_2^T)^T$ we form the bifunction

$$B(x,y) = \begin{cases} f(x) & g(x) \le y_1, h(x) = y_2 \\ \infty & \text{otherwise} \end{cases}$$



becomes



- Bifunctions have an associative composition law! It is inf-multiplication, i.e.,

$$\big(B_1 \circ B_2\big)(x,z) = \inf_{\color{blue}y} \big[B_1(x,{\color{blue}y}) + B_2({\color{blue}y},z)\big]$$

[1] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

Major changes are required to make this useful to us

- We can take objects as Euclidean spaces, e.g., $\mathbb{R}^m$ and $\mathbb{R}^n$
- A morphism is a bifunction $B : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$

$$\mathbb{R}^m \xrightarrow{\quad B \quad} \mathbb{R}^n$$

Major changes are required to make this useful to us

- We can take objects as Euclidean spaces, e.g., $\mathbb{R}^m$ and $\mathbb{R}^n$
- A morphism is a bifunction $B : \mathbb{R}^m \nrightarrow \mathbb{R}^n$

$$\mathbb{R}^m \xrightarrow{\quad B \quad} \mathbb{R}^n$$

- Composition via $\circ$ takes in $B_1 : \mathbb{R}^m \nrightarrow \mathbb{R}^n$ and $B_2 : \mathbb{R}^n \nrightarrow \mathbb{R}^p$ and gives back

$$\big(B_1 \circ B_2\big)(x, z) = \inf_y \big[B_1(x, y) + B_2(y, z)\big]$$

$$\begin{tikzcd}
\mathbb{R}^m \arrow[r, "B_1"] \arrow[dr, "B_1 \circ B_2"'] & \mathbb{R}^n \arrow[d, "B_2"] \\
& \mathbb{R}^p
\end{tikzcd}$$

Major changes are required to make this useful to us

- We can take objects as Euclidean spaces, e.g., $\mathbb{R}^m$ and $\mathbb{R}^n$
- A morphism is a bifunction $B : \mathbb{R}^m \nrightarrow \mathbb{R}^n$

$$\mathbb{R}^m \xrightarrow{\quad B \quad} \mathbb{R}^n$$

- Composition via $\circ$ takes in $B_1 : \mathbb{R}^m \nrightarrow \mathbb{R}^n$ and $B_2 : \mathbb{R}^n \nrightarrow \mathbb{R}^p$ and gives back

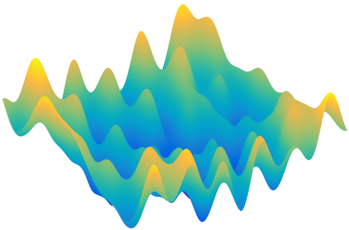
$$\big(B_1 \circ B_2\big)(x, z) = \inf_y \big[B_1(x, y) + B_2(y, z)\big]$$

$$\begin{tikzcd} \mathbb{R}^m \arrow[r, "B_1"] \arrow[dr, "B_1 \circ B_2"'] & \mathbb{R}^n \arrow[d, "B_2"] \\ & \mathbb{R}^p \end{tikzcd}$$

- We **will not** use $\inf$-multiplication for composition!

Optimality is non-convex problems is inherently local

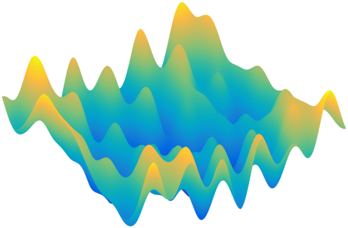
- Non-convex problems are (almost always) solved to local optimality



- We need to model local optimality in a composition law

Optimality is non-convex problems is inherently local

- Non-convex problems are (almost always) solved to local optimality

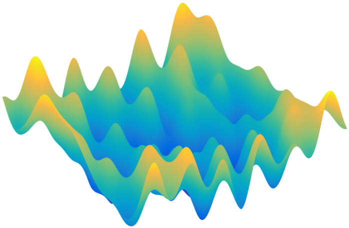


- We need to model local optimality in a composition law
- We will consider systems with polynomial dynamics and costs

  $\implies$ we have polynomial optimization problems!

Optimality is non-convex problems is inherently local

- Non-convex problems are (almost always) solved to local optimality



- We need to model local optimality in a composition law
- We will consider systems with polynomial dynamics and costs
  $\implies$ we have polynomial optimization problems!
- For polynomial f, g, and h:

$$\begin{aligned} \text{(locally) minimize } & f(x) \text{ (starting from } x_0\text{)} \\ \text{subject to } & h(x) = 0 \\ & g(x) \leq 0 \end{aligned}$$

- We want (i) local optimality and (ii) feasibility

Modeling question

Which local optimum?

We use negative gradient flows to model optimization algorithms

- The region $\mathscr{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\}$ is a Nash manifold with corners

Assumption #1: Morse property

The objective f is a stratified Morse function on $\mathscr{F}$.

[2] D.G.C. Handron, "Generalized Billiard Paths and Morse Theory for Manifolds with Corners", *Topology and its Applications*, 126 (2002), pp. 83-118.

We use negative gradient flows to model optimization algorithms

- The region $\mathscr{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\}$ is a Nash manifold with corners

Assumption #1: Morse property

The objective f is a stratified Morse function on $\mathscr{F}$.

- The projected negative gradient field[2] $-\tilde{\nabla} f$ models optimization algorithms

[2] D.G.C. Handron, "Generalized Billiard Paths and Morse Theory for Manifolds with Corners", *Topology and its Applications*, 126 (2002), pp. 83-118.

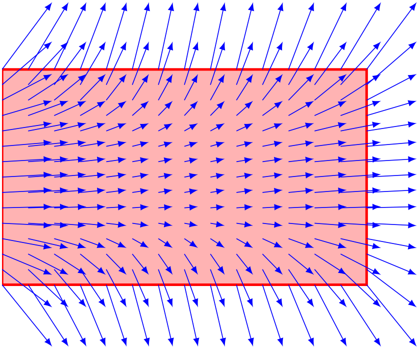
We use negative gradient flows to model optimization algorithms

- The region $\mathscr{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\}$ is a Nash manifold with corners

Assumption #1: Morse property

The objective f is a stratified Morse function on $\mathscr{F}$.

- The projected negative gradient field[2] $-\tilde{\nabla} f$ models optimization algorithms



[2] D.G.C. Handron, "Generalized Billiard Paths and Morse Theory for Manifolds with Corners", *Topology and its Applications*, 126 (2002), pp. 83-118.

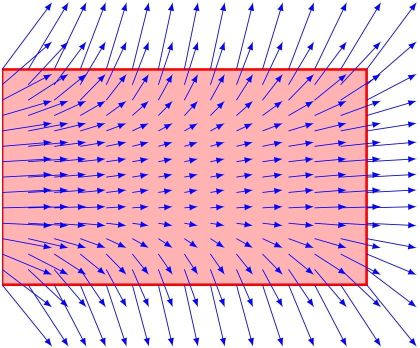
We use negative gradient flows to model optimization algorithms

- The region $\mathscr{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\}$ is a Nash manifold with corners

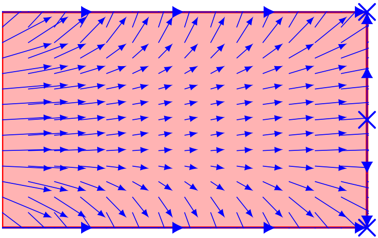
Assumption #1: Morse property

The objective f is a stratified Morse function on $\mathscr{F}$.

- The projected negative gradient field[2] $-\tilde{\nabla} f$ models optimization algorithms



becomes



[2] D.G.C. Handron, "Generalized Billiard Paths and Morse Theory for Manifolds with Corners", *Topology and its Applications*, 126 (2002), pp. 83-118.

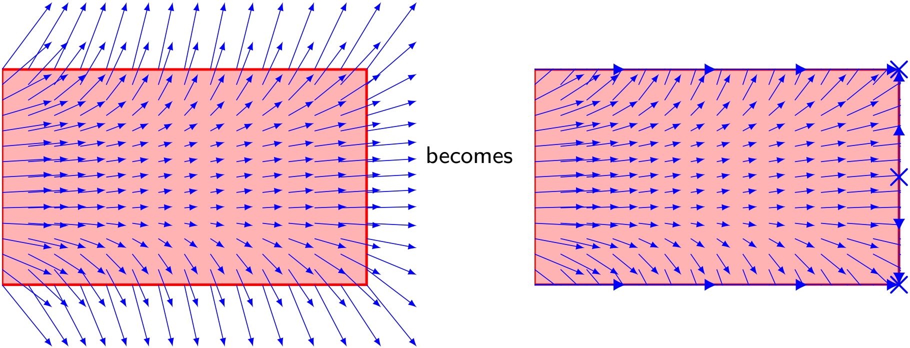
We use negative gradient flows to model optimization algorithms

- The region $\mathscr{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0\}$ is a Nash manifold with corners

Assumption #1: Morse property

The objective f is a stratified Morse function on $\mathscr{F}$.

- The projected negative gradient field[2] $-\tilde{\nabla} f$ models optimization algorithms



- Flows follow $-\nabla f$ as much as possible while keeping $\mathscr{F}$ forward-invariant.

[2] D.G.C. Handron, "Generalized Billiard Paths and Morse Theory for Manifolds with Corners", *Topology and its Applications*, 126 (2002), pp. 83-118.

Now we can define local $\inf$-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

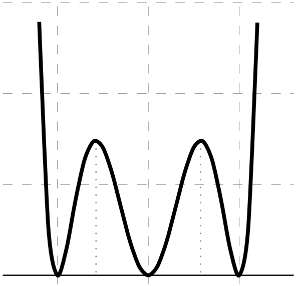
Now we can define local inf-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

- For example:



[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

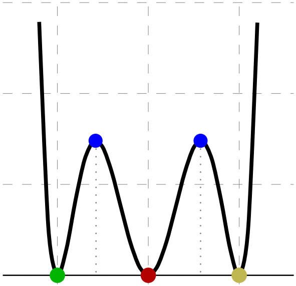
Now we can define local inf-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

- For example:



[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

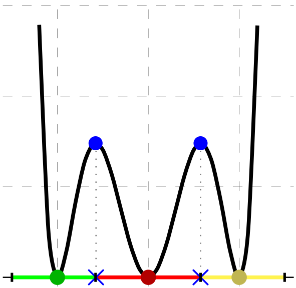
Now we can define local inf-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

- For example:



[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

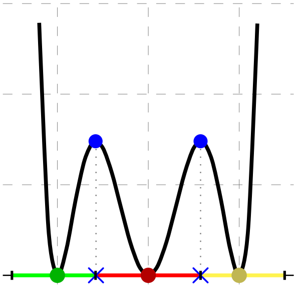
Now we can define local inf-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

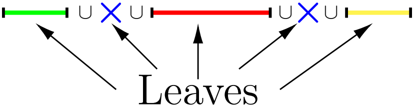
Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

- For example:



- The foliation is



[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

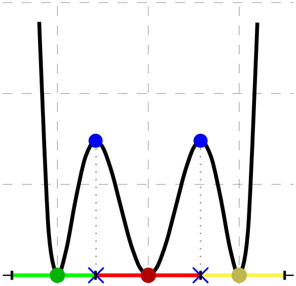
Now we can define local inf-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

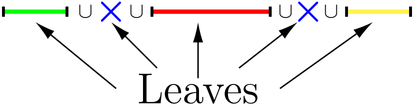
Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

- For example:



- The foliation is



- Then local inf-multiplication is inf-multiplication over a leaf
- Our composition law is

$$
\begin{aligned}
\left(B_1 \circ_{y_0} B_2\right)(x,z) &= \operatorname*{local\,min}_{y,y_0} \left[B_1(x,y) + B_2(y,z)\right] \\
&= \min_{y \in \mathscr{L}(y_0)} \left[B_1(x,y) + B_2(y,z)\right]
\end{aligned}
$$

[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

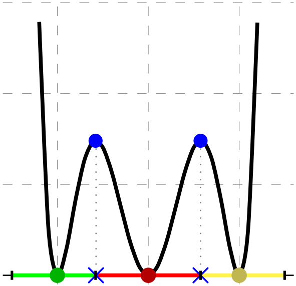
Now we can define local $\inf$-multiplication

- We need the stable foliation of $\mathscr{F}$ with respect to $-\tilde{\nabla} f$

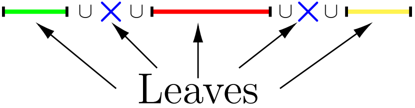
Definition #3: Stable foliation[3]

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the stationary points of $-\tilde{\nabla} f$. Then $\mathscr{F} = \bigcup_{p_i \in \mathcal{P}} W^s(p_i)$

- For example:



- The foliation is



- Then local $\inf$-multiplication is $\inf$-multiplication over a leaf
- Our composition law is

$$\begin{aligned}\big(B_1 \circ_{y_0} B_2\big)(x,z) &= \operatorname*{local\,min}_{y,y_0} \big[B_1(x,y) + B_2(y,z)\big] \\ &= \min_{y \in \mathscr{L}(y_0)} \big[B_1(x,y) + B_2(y,z)\big]\end{aligned}$$

- We'd like to have

$$\begin{tikzcd} \mathbb{R}^m \arrow[r, "B_1"] \arrow[dr, "B_1 \circ_{y_0} B_2"'] & \mathbb{R}^n \arrow[d, "B_2"] \\ & \mathbb{R}^p \end{tikzcd}$$

[3] R. Thom, Sur une partition en cellules associée à une fonction sur une variété, C. R. Acad. Sci. Paris 228 (1949), 973–975.

We now have a category!

Definition #4: The category C

We define C such that

- Ob(C): objects are pointed Euclidean spaces, e.g, $(\mathbb{R}^n, x_0)$
- Mor(C): morphisms are algebraic bifunctions, i.e., polynomial B where

$$B(x,y) = \begin{cases} f(x) & g(x) \leq y_1, h(x) = y_2 \\ \infty & \text{otherwise} \end{cases}$$

- Composition uses local $\inf$-multiplication
- For $B_1 : (\mathbb{R}^m, x_0) \nrightarrow (\mathbb{R}^n, y_0)$ and $B_2 : (\mathbb{R}^n, y_0) \nrightarrow (\mathbb{R}^p, z_0)$, we have

$$\left(B_1 \circ_{y_0} B_2\right)(x,z) = \operatorname*{local\,min}_{y, y_0} \left[B_1(x,y) + B_2(y,z)\right]$$

We now have a category!

Definition #4: The category C

We define C such that

- Ob(C): objects are pointed Euclidean spaces, e.g, $(\mathbb{R}^n, x_0)$
- Mor(C): morphisms are algebraic bifunctions, i.e., polynomial B where

$$B(x,y) = \begin{cases} f(x) & g(x) \le y_1, h(x) = y_2 \\ \infty & \text{otherwise} \end{cases}$$

- Composition uses local $\inf$-multiplication
- For $B_1 : (\mathbb{R}^m, x_0) \nrightarrow (\mathbb{R}^n, y_0)$ and $B_2 : (\mathbb{R}^n, y_0) \nrightarrow (\mathbb{R}^p, z_0)$, we have

$$\big(B_1 \circ_{y_0} B_2\big)(x,z) = \operatorname*{local\,min}_{y, y_0} \Big[B_1(x,y) + B_2(y,z)\Big]$$

$$\begin{tikzcd} (\mathbb{R}^m, x_0) \arrow[r, "B_1"] \arrow[rr, bend right, "B_1 \circ B_2"'] & (\mathbb{R}^n, y_0) \arrow[r, "B_2"] & (\mathbb{R}^p, z_0) \end{tikzcd}$$

We now have a category!

Definition #4: The category C

We define C such that

- Ob(C): objects are pointed Euclidean spaces, e.g, $(\mathbb{R}^n, x_0)$
- Mor(C): morphisms are algebraic bifunctions, i.e., polynomial B where

$$B(x,y) = \begin{cases} f(x) & g(x) \leq y_1, h(x) = y_2 \\ \infty & \text{otherwise} \end{cases}$$

- Composition uses local inf-multiplication
- For $B_1 : (\mathbb{R}^m, x_0) \nrightarrow (\mathbb{R}^n, y_0)$ and $B_2 : (\mathbb{R}^n, y_0) \nrightarrow (\mathbb{R}^p, z_0)$, we have

$$\big(B_1 \circ_{y_0} B_2\big)(x,z) = \operatorname*{local\,min}_{y, y_0} \Big[B_1(x,y) + B_2(y,z)\Big]$$

$$\begin{tikzcd} (\mathbb{R}^m, x_0) \arrow[r, "B_1"] \arrow[rr, bend right, "B_1 \circ B_2"'] & (\mathbb{R}^n, y_0) \arrow[r, "B_2"] & (\mathbb{R}^p, z_0) \end{tikzcd}$$

Theorem #1: We have a category

This construction of C satisfies all of the category axioms.

We now have a category!

Definition #4: The category C

We define C such that

- Ob(C): objects are pointed Euclidean spaces, e.g, $(\mathbb{R}^n, x_0)$
- Mor(C): morphisms are algebraic bifunctions, i.e., polynomial B where

$$B(x,y) = \begin{cases} f(x) & g(x) \leq y_1, h(x) = y_2 \\ \infty & \text{otherwise} \end{cases}$$

- Composition uses local $\inf$-multiplication
- For $B_1 : (\mathbb{R}^m, x_0) \nrightarrow (\mathbb{R}^n, y_0)$ and $B_2 : (\mathbb{R}^n, y_0) \nrightarrow (\mathbb{R}^p, z_0)$, we have

$$\big(B_1 \circ_{y_0} B_2\big)(x,z) = \operatorname*{local\,min}_{y, y_0} \Big[B_1(x,y) + B_2(y,z)\Big]$$

$$\begin{tikzcd}
(\mathbb{R}^m, x_0) \arrow[r, "B_1"] \arrow[rr, bend right, "B_1 \circ B_2"'] & (\mathbb{R}^n, y_0) \arrow[r, "B_2"] & (\mathbb{R}^p, z_0)
\end{tikzcd}$$

Theorem #1: We have a category

This construction of C satisfies all of the category axioms.

- Fine, but where are the inputs?

We need to introduce external parameters

- We will use Para to introduce inputs[4]. To get there, we make a new category

[4] B. Fong, D. Spivak and R. Tuyéras, "Backprop as Functor: A compositional perspective on supervised learning," *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019

We need to introduce external parameters

- We will use Para to introduce inputs[4]. To get there, we make a new category

Definition #5: AlgBiFun

- Define $\mathsf{AlgBiFun} = \big(\mathsf{C}, \oplus, (\mathbb{R}^0, \bullet)\big)$ so that
  1. $(\mathbb{R}^n, x_0) \oplus (\mathbb{R}^m, y_0) = \left(\mathbb{R}^n \oplus \mathbb{R}^m, \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right)$
  2. $\big(B_1 \oplus B_2\big)\big((w, x), (y, z)\big) = B_1(w, y) + B_2(x, z)$

[4] B. Fong, D. Spivak and R. Tuyéras, "Backprop as Functor: A compositional perspective on supervised learning," *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019

We need to introduce external parameters

- We will use Para to introduce inputs[4]. To get there, we make a new category

Definition #5: AlgBiFun

- Define AlgBiFun $= \left(\mathsf{C}, \oplus, (\mathbb{R}^0, \bullet)\right)$ so that
  1. $(\mathbb{R}^n, x_0) \oplus (\mathbb{R}^m, y_0) = \left(\mathbb{R}^n \oplus \mathbb{R}^m, \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right)$
  2. $\left(B_1 \oplus B_2\right)\left((w,x),(y,z)\right) = B_1(w,y) + B_2(x,z)$

Theorem #2: We've made a new category

AlgBiFun is a strict symmetric monoidal category.

[4] B. Fong, D. Spivak and R. Tuyéras, "Backprop as Functor: A compositional perspective on supervised learning," *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019

We need to introduce external parameters

- We will use Para to introduce inputs[4]. To get there, we make a new category

Definition #5: AlgBiFun

- Define AlgBiFun $= \big(\mathsf{C}, \oplus, (\mathbb{R}^0, \bullet)\big)$ so that

1. $(\mathbb{R}^n, x_0) \oplus (\mathbb{R}^m, y_0) = \left(\mathbb{R}^n \oplus \mathbb{R}^m, \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right)$
2. $\Big(B_1 \oplus B_2\Big)\Big((w, x), (y, z)\Big) = B_1(w, y) + B_2(x, z)$

Theorem #2: We've made a new category

AlgBiFun is a strict symmetric monoidal category.

This has two immediate outcomes:

1. We unlock "string diagrams". For
$f : (U, u_0) \nrightarrow (W, w_0) \oplus (X, x_0)$,
$g : (X, x_0) \nrightarrow (Y, y_0)$,
$h : (W, w_0) \oplus (Y, y_0) \nrightarrow (Z, z_0)$,
we can "draw" the composite
$h \circ_{(w_0, y_0)} (\mathrm{id}_W \oplus g) \circ_{(w_0, x_0)} f$

[4] B. Fong, D. Spivak and R. Tuyéras, "Backprop as Functor: A compositional perspective on supervised learning," *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019

We need to introduce external parameters

- We will use Para to introduce inputs[4]. To get there, we make a new category

Definition #5: AlgBiFun

- Define AlgBiFun $= \Big(\mathsf{C}, \oplus, (\mathbb{R}^0, \bullet)\Big)$ so that

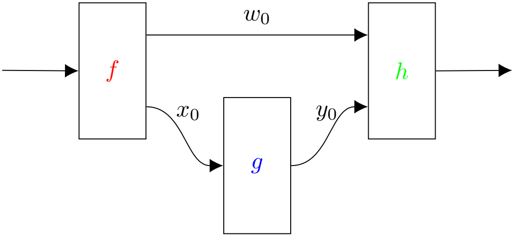
  1. $(\mathbb{R}^n, x_0) \oplus (\mathbb{R}^m, y_0) = \left(\mathbb{R}^n \oplus \mathbb{R}^m, \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right)$
  2. $\Big(B_1 \oplus B_2\Big)\Big((w,x),(y,z)\Big) = B_1(w,y) + B_2(x,z)$

Theorem #2: We've made a new category

AlgBiFun is a strict symmetric monoidal category.

This has two immediate outcomes:

1. We unlock "string diagrams". For $f : (U, u_0) \nrightarrow (W, w_0) \oplus (X, x_0)$, $g : (X, x_0) \nrightarrow (Y, y_0)$, $h : (W, w_0) \oplus (Y, y_0) \nrightarrow (Z, z_0)$, we can "draw" the composite $h \circ_{(w_0, y_0)} (\mathrm{id}_W \oplus g) \circ_{(w_0, x_0)} f$



[4] B. Fong, D. Spivak and R. Tuyéras, "Backprop as Functor: A compositional perspective on supervised learning," *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019

We need to introduce external parameters

- We will use Para to introduce inputs[4]. To get there, we make a new category

Definition #5: AlgBiFun

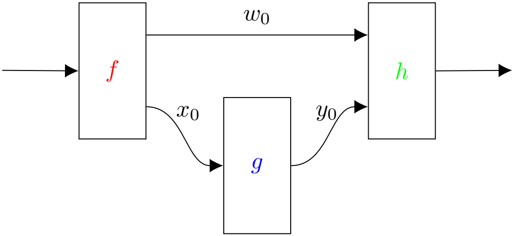
- Define $\mathsf{AlgBiFun} = \big(\mathsf{C}, \oplus, (\mathbb{R}^0, \bullet)\big)$ so that
  1. $(\mathbb{R}^n, x_0) \oplus (\mathbb{R}^m, y_0) = \left(\mathbb{R}^n \oplus \mathbb{R}^m, \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right)$
  2. $\big(B_1 \oplus B_2\big)\big((w,x),(y,z)\big) = B_1(w,y) + B_2(x,z)$

Theorem #2: We've made a new category

AlgBiFun is a strict symmetric monoidal category.

This has two immediate outcomes:

1. We unlock "string diagrams". For
   $f : (U, u_0) \nrightarrow (W, w_0) \oplus (X, x_0)$,
   $g : (X, x_0) \nrightarrow (Y, y_0)$,
   $h : (W, w_0) \oplus (Y, y_0) \nrightarrow (Z, z_0)$,
   we can "draw" the composite
   $h \circ_{(w_0,y_0)} (\mathrm{id}_W \oplus g) \circ_{(w_0,x_0)} f$
2. We can parameterize this category!



[4] B. Fong, D. Spivak and R. Tuyéras, "Backprop as Functor: A compositional perspective on supervised learning," *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019

We apply the Para construction to model inputs

- So far: made C (category), then AlgBiFun (strict symmetric monoidal category)
- Now: Para(AlgBiFun)

We apply the Para construction to model inputs

- So far: made C (category), then AlgBiFun (strict symmetric monoidal category)
- Now: Para(AlgBiFun)

Definition #6: The category Para(AlgBiFun)

i. Objects are pointed Euclidean spaces, e.g., $(\mathbb{R}^n, x_0)$

We apply the Para construction to model inputs

- So far: made C (category), then AlgBiFun (strict symmetric monoidal category)
- Now: Para(AlgBiFun)

Definition #6: The category Para(AlgBiFun)

i. Objects are pointed Euclidean spaces, e.g., $(\mathbb{R}^n, x_0)$

ii. A morphism $(X, x_0) \to (Z, z_0)$ is a pair $\big((U, u_0), F\big)$, where

$$F : (U, u_0) \oplus (X, x_0) \to (Z, z_0)$$

We apply the Para construction to model inputs

- So far: made C (category), then AlgBiFun (strict symmetric monoidal category)
- Now: Para(AlgBiFun)

Definition #6: The category Para(AlgBiFun)

i. Objects are pointed Euclidean spaces, e.g., $(\mathbb{R}^n, x_0)$

ii. A morphism $(X, x_0) \to (Z, z_0)$ is a pair $\big((U, u_0), F\big)$, where

$$F : (U, u_0) \oplus (X, x_0) \to (Z, z_0)$$

iii. For bifunctions $F_1 : (X, x_0) \nrightarrow (Y, y_0)$ and $F_2 : (Y, y_0) \nrightarrow (Z, z_0)$, composition of the two morphisms $\big((U, u_0), F_1\big)$ and $\big((V, v_0), F_2\big)$ is

$$\big(V \oplus U, F_2 \circ_{y_0} (\mathrm{id}_V \oplus F_1)\big)$$

We apply the Para construction to model inputs

- So far: made C (category), then AlgBiFun (strict symmetric monoidal category)
- Now: Para(AlgBiFun)

Definition #6: The category Para(AlgBiFun)

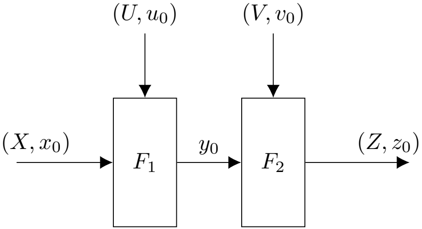
i. Objects are pointed Euclidean spaces, e.g., $(\mathbb{R}^n, x_0)$

ii. A morphism $(X, x_0) \to (Z, z_0)$ is a pair $\big((U, u_0), F\big)$, where

$$F : (U, u_0) \oplus (X, x_0) \to (Z, z_0)$$

iii. For bifunctions $F_1 : (X, x_0) \nrightarrow (Y, y_0)$ and $F_2 : (Y, y_0) \nrightarrow (Z, z_0)$, composition of the two morphisms $\big((U, u_0), F_1\big)$ and $\big((V, v_0), F_2\big)$ is

$$\big(V \oplus U, F_2 \circ_{y_0} (\mathrm{id}_V \oplus F_1)\big)$$



We apply the Para construction to model inputs

- So far: made C (category), then AlgBiFun (strict symmetric monoidal category)
- Now: Para(AlgBiFun)

Definition #6: The category Para(AlgBiFun)

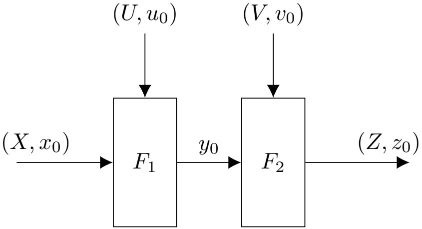
i. Objects are pointed Euclidean spaces, e.g., $(\mathbb{R}^n, x_0)$

ii. A morphism $(X, x_0) \to (Z, z_0)$ is a pair $\big((U, u_0), F\big)$, where

$$F : (U, u_0) \oplus (X, x_0) \to (Z, z_0)$$

iii. For bifunctions $F_1 : (X, x_0) \nrightarrow (Y, y_0)$ and $F_2 : (Y, y_0) \nrightarrow (Z, z_0)$, composition of the two morphisms $\big((U, u_0), F_1\big)$ and $\big((V, v_0), F_2\big)$ is

$$\big(V \oplus U, F_2 \circ_{y_0} (\mathrm{id}_V \oplus F_1)\big)$$



- Roughly, given $x(k)$, we have $x(k+1) = f\big(x(k), u\big)$
- Then $x(k+2) = f\big(x(k+1), v\big)$
- The figure essentially says

$$x(k+2) = f\Big(f\big(x(k), u\big), v\Big)$$

One-step MPC problems are morphisms

- Consider the one-step MPC problem

$$\begin{aligned} \underset{u(k)}{\text{minimize}} \quad & \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \end{aligned}$$

One-step MPC problems are morphisms

- Consider the one-step MPC problem

$$\begin{aligned} \underset{u(k)}{\text{minimize}} \quad & \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \end{aligned}$$

- Its associated *one-step bifunction* is $G : (U, u_0) \oplus (X, x_0) \nrightarrow (X, \xi_0)$, i.e.,

$$G\Big(\big((u, u_0), (x, x_0)\big), (x', \xi_0)\Big) = \underbrace{\ell(x, u)}_{\text{Cost}} + \underbrace{\delta\big(x' \mid f(x, u)\big)}_{\text{Dynamics}} + \underbrace{\delta\big(x, u \mid g(x, u) \leq 0\big)}_{\text{Constraints}}$$

One-step MPC problems are morphisms

- Consider the one-step MPC problem

$$\begin{aligned} \underset{u(k)}{\text{minimize}} \quad & \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \end{aligned}$$

- Its associated *one-step bifunction* is $G : (U, u_0) \oplus (X, x_0) \nrightarrow (X, \xi_0)$, i.e.,

$$G\Big(\big((u, u_0), (x, x_0)\big), (x', \xi_0)\Big) = \underbrace{\ell(x, u)}_{\text{Cost}} + \underbrace{\delta\big(x' \mid f(x, u)\big)}_{\text{Dynamics}} + \underbrace{\delta\big(x, u \mid g(x, u) \leq 0\big)}_{\text{Constraints}}$$

Theorem #3: One-step problems are morphisms

The pair $\big((U, u_0), G\big)$ is a morphism from (X, x_0) to (X, ξ_0) in Para(AlgBiFun).

One-step MPC problems are morphisms

- Consider the one-step MPC problem

$$\begin{aligned} \underset{u(k)}{\text{minimize}} \quad & \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \end{aligned}$$

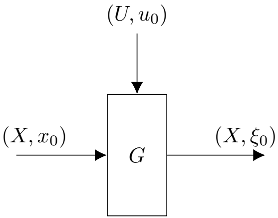
- Its associated *one-step bifunction* is $G : (U, u_0) \oplus (X, x_0) \nrightarrow (X, \xi_0)$, i.e.,

$$G\Big(\big((u, u_0), (x, x_0)\big), (x', \xi_0)\Big) = \underbrace{\ell(x,u)}_{\text{Cost}} + \underbrace{\delta\big(x' \mid f(x,u)\big)}_{\text{Dynamics}} + \underbrace{\delta\big(x, u \mid g(x,u) \leq 0\big)}_{\text{Constraints}}$$

Theorem #3: One-step problems are morphisms

The pair $\big((U, u_0), G\big)$ is a morphism from (X, x_0) to (X, ξ_0) in Para(AlgBiFun).

- Pictorially, a 1-step MPC problem is



N-step MPC problems are N-fold bifunction compositions

- Now consider the N-step MPC problem

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{k=t}^{t+N-1} \ell\big(x(k), u(k)\big) \\ \text{subject to} & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \quad k = t, t+1, \ldots, t+N-1 \end{array}$$

- Let $\big((U, u_0^i), G\big)$ be the morphism corresponding to time i

N-step MPC problems are N-fold bifunction compositions

- Now consider the N-step MPC problem

$$\begin{aligned} \text{minimize} \quad & \sum_{k=t}^{t+N-1} \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \quad k = t, t+1, \dots, t+N-1 \end{aligned}$$

- Let $\big((U, u_0^i), G\big)$ be the morphism corresponding to time i

Theorem #4: N-step problems are compositions of N morphisms

The N-step MPC problem can be represented as an N-fold composition of morphisms:

$$\big((U, u_0^t), G\big) \circ_{x_0} \big((U, u_0^{t+1}), G\big) \circ_{\xi_0} \cdots \circ_{\zeta_0} \big((U, u_0^{t+N-2}), G\big) \circ_{\psi_0} \big((U, u_0^{t+N-1}), G\big)$$

N-step MPC problems are N-fold bifunction compositions

- Now consider the N-step MPC problem

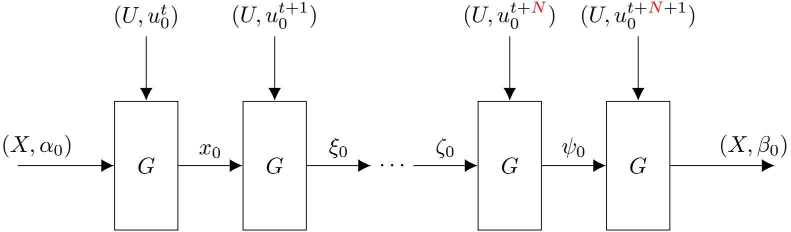
$$\begin{aligned} \text{minimize} \quad & \sum_{k=t}^{t+N-1} \ell\big(x(k), u(k)\big) \\ \text{subject to} \quad & x(k+1) = f\big(x(k), u(k)\big) \\ & g\big(x(k), u(k)\big) \leq 0 \quad k = t, t+1, \ldots, t+N-1 \end{aligned}$$

- Let $\big((U, u_0^i), G\big)$ be the morphism corresponding to time i

Theorem #4: N-step problems are compositions of N morphisms

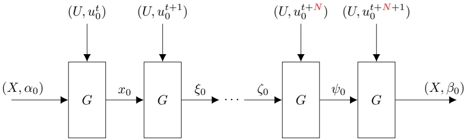
The N-step MPC problem can be represented as an N-fold composition of morphisms:

$$\big((U, u_0^t), G\big) \circ_{x_0} \big((U, u_0^{t+1}), G\big) \circ_{\xi_0} \cdots \circ_{\zeta_0} \big((U, u_0^{t+N-2}), G\big) \circ_{\psi_0} \big((U, u_0^{t+N-1}), G\big)$$



What's next?

- We can automatically generate correct-by-construction software
- How easy can we make the implementation of MPC/multi-stage optimization?



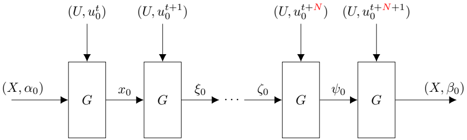

```
# Create the one-step bifunction.
one_step = one_step_bifunction(dim_x,dim_u,cost,
    constraints, dynamics)

N = 10 # prediction horizon
MPC_bifunc = compose(repeat([one_step], N))

# Create variables to store control inputs
# and final state achieved.
us = [Variable(1) for i in 1:N-1]
xN = Variable(2)

# Convert to a Convex.jl problem and solve.
MPC_prob = make_problem(MPC_bifunc, us, x0, xN)
solve!(MPC_prob, SCS.Optimizer)
```

[5] T. Hanks, B. She, M. Hale, E. Patterson, M. Klawonn, J. Fairbanks, "Modeling Model Predictive Control: A Category Theoretic Framework for Multistage Control Problems", *2024 American Control Conference*, 2024.

[6] W. Warke, J. Ramos, P. Ganesh, K. Brink, and M.T. Hale, "Pose Graph Optimization over Planar Unit Dual Quaternions: Improved Accuracy with Provably Convergent Riemannian Optimization", Accepted to IROS 2024. Preprint: `https://arxiv.org/abs/2404.00010v2`

What's next?

- We can automatically generate correct-by-construction software
- How easy can we make the implementation of MPC/multi-stage optimization?




```
# Create the one-step bifunction.
one_step = one_step_bifunction(dim_x,dim_u,cost,
    constraints, dynamics)

N = 10 # prediction horizon
MPC_bifunc = compose(repeat([one_step], N))

# Create variables to store control inputs
# and final state achieved.
us = [Variable(1) for i in 1:N-1]
xN = Variable(2)

# Convert to a Convex.jl problem and solve.
MPC_prob = make_problem(MPC_bifunc, us, x0, xN)
solve!(MPC_prob, SCS.Optimizer)
```

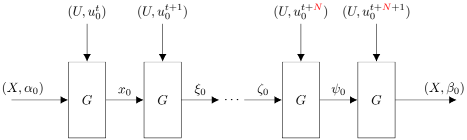
- This talk generalized work from ACC '24 on compositional models for *convex* MPC problems[5]
- How can we generalize to non-convex (and non-MPC) decision problems on other spaces, e.g., manifolds[6]?

[5] T. Hanks, B. She, M. Hale, E. Patterson, M. Klawonn, J. Fairbanks, "Modeling Model Predictive Control: A Category Theoretic Framework for Multistage Control Problems", *2024 American Control Conference*, 2024.

[6] W. Warke, J. Ramos, P. Ganesh, K. Brink, and M.T. Hale, "Pose Graph Optimization over Planar Unit Dual Quaternions: Improved Accuracy with Provably Convergent Riemannian Optimization", Accepted to IROS 2024. Preprint: `https://arxiv.org/abs/2404.00010v2`

What's next?

- We can automatically generate correct-by-construction software
- How easy can we make the implementation of MPC/multi-stage optimization?




```
# Create the one-step bifunction.
one_step = one_step_bifunction(dim_x,dim_u,cost,
    constraints, dynamics)

N = 10 # prediction horizon
MPC_bifunc = compose(repeat([one_step], N))

# Create variables to store control inputs
# and final state achieved.
us = [Variable(1) for i in 1:N-1]
xN = Variable(2)

# Convert to a Convex.jl problem and solve.
MPC_prob = make_problem(MPC_bifunc, us, x0, xN)
solve!(MPC_prob, SCS.Optimizer)
```

- This talk generalized work from ACC '24 on compositional models for *convex* MPC problems[5]
- How can we generalize to non-convex (and non-MPC) decision problems on other spaces, e.g., manifolds[6]?

ACC 2024

Next steps: Can we prove stability of MPC in purely categorical terms?
(Talking to Aaron Ames and Joe Moeller about this)

[5] T. Hanks, B. She, M. Hale, E. Patterson, M. Klawonn, J. Fairbanks, "Modeling Model Predictive Control: A Category Theoretic Framework for Multistage Control Problems", *2024 American Control Conference*, 2024.

[6] W. Warke, J. Ramos, P. Ganesh, K. Brink, and M.T. Hale, "Pose Graph Optimization over Planar Unit Dual Quaternions: Improved Accuracy with Provably Convergent Riemannian Optimization", Accepted to IROS 2024. Preprint: `https://arxiv.org/abs/2404.00010v2`

Thank you

Matthew Hale

Georgia Tech
matthale@gatech.edu

http://corelab.ece.gatech.edu

YIP Grant no. FA9550-23-1-0120