Compositional Dynamics in Learning and Mechanics

David I. Spivak

TOPOS INSTITUTE

AFOSR Year-3 Review 2026 August 6

Award FA9550-23-1-0376 Program officers: Fred Leve, Tristan Nguyen, & Doug Riecken

Outline

Working language across three AFOSR programs

This is the third year of a grant that's funded by three AFOSR programs:

- Fred Leve: Dynamics and Control
- Doug Riecken: Computation and Learning, and
- Tristan Nguyen: Information Assurance.

Working language across three AFOSR programs

This is the third year of a grant that's funded by three AFOSR programs:

- Fred Leve: Dynamics and Control
- Doug Riecken: Computation and Learning, and
- Tristan Nguyen: Information Assurance.

Originating question: What is *working language*? How does language *work*?

- Language works in the sense of basic physics: it directs energy.
- If I say "pass the salt," 10^{25} atoms move through space.
- This involves compositional planning and high-precision control.
- Getting it set up in the first place requires (evolution and) learning.
- DNA is working language: ATCG symbols code for chemical reactions.
- Computer programming languages do a lot of work in the world.

Working language across three AFOSR programs

This is the third year of a grant that's funded by three AFOSR programs:

- Fred Leve: Dynamics and Control
- Doug Riecken: Computation and Learning, and
- Tristan Nguyen: Information Assurance.

Originating question: What is *working language*? How does language *work*?

- Language works in the sense of basic physics: it directs energy.
- If I say "pass the salt," 10^{25} atoms move through space.
- This involves compositional planning and high-precision control.
- Getting it set up in the first place requires (evolution and) learning.
- DNA is working language: ATCG symbols code for chemical reactions.
- Computer programming languages do a lot of work in the world.

Language and learning both run on "bare physics."

- Why is reality learnable, compressible, and governable?
- Conjecture: its mathematical basis is fundamental, pre-biological.
- Hamiltonian dynamics and learning share compositional structure.

Working language across three AFOSR programs

This is the third year of a grant that's funded by three AFOSR programs:

- Fred Leve: Dynamics and Control
- Doug Riecken: Computation and Learning, and
- Tristan Nguyen: Information Assurance.

Originating question: What is *working language*? How does language *work*?

- Language works in the sense of basic physics: it directs energy.
- If I say "pass the salt," 10^{25} atoms move through space.
- This involves compositional planning and high-precision control.
- Getting it set up in the first place requires (evolution and) learning.
- DNA is working language: ATCG symbols code for chemical reactions.
- Computer programming languages do a lot of work in the world.

Language and learning both run on "bare physics."

- Why is reality learnable, compressible, and governable?
- Conjecture: its mathematical basis is fundamental, pre-biological.
- Hamiltonian dynamics and learning share compositional structure.

This isn't a Cat. Th. audience, so I'll try to keep the talk self-contained.

Basic idea and plan

Hamiltonian physics and training a neural network share a lot of structure.

- They both use a map $\sharp\colon T^*V \to TV$, so that covectors let you hop.
- They both use potentials to drive dynamics.
- They are both local and compositional. That's today's story.

Basic idea and plan

Hamiltonian physics and training a neural network share a lot of structure.

- They both use a map $\sharp\colon T^*V \to TV$, so that covectors let you hop.
- They both use potentials to drive dynamics.
- They are both local and compositional. That's today's story.

Category theory rarely meets numerics; here, discretization is functorial.

Basic idea and plan

Hamiltonian physics and training a neural network share a lot of structure.

- They both use a map $\sharp\colon T^*V \to TV$, so that covectors let you hop.
- They both use potentials to drive dynamics.
- They are both local and compositional. That's today's story.

Category theory rarely meets numerics; here, discretization is functorial.

Plan for the remainder of the talk:

- Background: category theory and functorial semantics.
- Two compilers, Φ_{conf} and Φ_{phase}, functorially discretize the physics.
- Examples from ANN training, Hamiltonian mechanics, and control.
- Conclude with answers to Fred's questions + a summary.

Outline

Category theory recap

A *category* is a collection of objects and maps, closed under compos'n.

- The category **Set** has sets as objects and functions as maps.
- The category **Mfd** has manifolds as objects and smooth fns as maps.

Category theory recap

A *category* is a collection of objects and maps, closed under compos'n.

- The category **Set** has sets as objects and functions as maps.
- The category **Mfd** has manifolds as objects and smooth fns as maps.

Category theory is designed to surface the most reusable parts of math.

- There is a definition of product that makes sense in any category.
  - Some categories don't have products $A \times B$. But many do.
  - In **Set** and **Mfd**, this notion recovers the usual cartesian product.
  - Posets are cat'ies, and the same definition gives meets $A \wedge B$.

Category theory recap

A *category* is a collection of objects and maps, closed under compos'n.

- The category **Set** has sets as objects and functions as maps.
- The category **Mfd** has manifolds as objects and smooth fns as maps.

Category theory is designed to surface the most reusable parts of math.

- There is a definition of product that makes sense in any category.
  - Some categories don't have products $A \times B$. But many do.
  - In **Set** and **Mfd**, this notion recovers the usual cartesian product.
  - Posets are cat'ies, and the same definition gives meets $A \wedge B$.
- Functors are maps between categories that preserve composition.
  - There's a functor $\mathbf{Mfd} \to \mathbf{Set}$ taking the set of points.
  - There's a functor $\mathbf{Set}_{\text{cntbl}} \to \mathbf{Mfd}$, sending $N \mapsto \coprod_N \mathbb{R}^0$.

Category theory recap

A *category* is a collection of objects and maps, closed under compos'n.

- The category **Set** has sets as objects and functions as maps.
- The category **Mfd** has manifolds as objects and smooth fns as maps.

Category theory is designed to surface the most reusable parts of math.

- There is a definition of product that makes sense in any category.
  - Some categories don't have products $A \times B$. But many do.
  - In **Set** and **Mfd**, this notion recovers the usual cartesian product.
  - Posets are cat'ies, and the same definition gives meets $A \wedge B$.
- Functors are maps between categories that preserve composition.
  - There's a functor **Mfd** $\to$ **Set** taking the set of points.
  - There's a functor $\mathbf{Set}_{\text{cntbl}} \to \mathbf{Mfd}$, sending $N \mapsto \coprod_N \mathbb{R}^0$.
- Functors are classified by what structures they preserve.
  - Both of the above *preserve products*. But $N \mapsto \coprod_N \mathbb{R}$ doesn't,...
  - ...because $\left(\coprod_M \mathbb{R}\right) \times \left(\coprod_N \mathbb{R}\right) \cong \coprod_{M \times N} \mathbb{R}^2 \ncong \coprod_{M \times N} \mathbb{R}$.

Functorial semantics

Lawvere's idea: syntax is a cat'y $\mathcal{C}$; its models are functors $\mathcal{C} \to \mathbf{Set}$.

Functorial semantics

Lawvere's idea: syntax is a cat'y $\mathcal{C}$; its models are functors $\mathcal{C} \to \mathbf{Set}$.

- There are many algebraic theories: sets, groups, rings, vector spaces.
  - What they all have in common: a theory and its models.
  - A *theory* T provides oper'ns and prop'ties, e.g. $(*)$ and its assoc.
  - A *T-model* M is a set, closed under the op's, satisfying the prop's.

Functorial semantics

Lawvere's idea: syntax is a cat'y $\mathcal{C}$; its models are functors $\mathcal{C} \to \mathbf{Set}$.

- There are many algebraic theories: sets, groups, rings, vector spaces.
  - What they all have in common: a theory and its models.
  - A *theory* T provides oper'ns and prop'ties, e.g. $(*)$ and its assoc.
  - A *T-model* M is a set, closed under the op's, satisfying the prop's.
- Categorically, a model is a functor: $T \xrightarrow{M} \mathbf{Set}$.
  - Roughly: "a theory is a category T with finite products and...
  - ... a T-model is a product-preserving functor $M\colon T \to \mathbf{Set}$, and...
  - ... a map of T-models is a natural transformation $M \Rightarrow M'$. "
  - So there's a cat'y T_{grp} whose category of models is precisely **Grp**.

Functorial semantics

Lawvere's idea: syntax is a cat'y $\mathcal{C}$; its models are functors $\mathcal{C} \to \mathbf{Set}$.

- There are many algebraic theories: sets, groups, rings, vector spaces.
  - What they all have in common: a theory and its models.
  - A *theory* T provides oper'ns and prop'ties, e.g. $(*)$ and its assoc.
  - A *T-model* M is a set, closed under the op's, satisfying the prop's.
- Categorically, a model is a functor: $T \xrightarrow{M} \mathbf{Set}$.
  - Roughly: "a theory is a category T with finite products and...
  - ... a T-model is a product-preserving functor $M\colon T \to \mathbf{Set}$, and...
  - ... a map of T-models is a natural transformation $M \Rightarrow M'$. "
  - So there's a cat'y T_{grp} whose category of models is precisely **Grp**.
- Vocab: T is the *syntax*, **Set** is the *semantics*, M is the *model*.

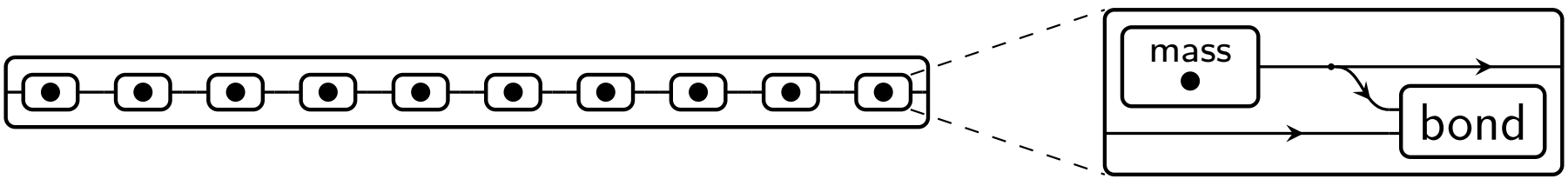
Very different-seeming examples: databases, discrete dynamical systems.

- Schemas are categories $\mathcal{S}$; the data is a functor $\mathcal{S} \to \mathbf{Set}$.
- A functor  $\to \mathbf{Set}$ is a set S equipped with an update map $S \to S$.

Operadic syntax for interaction

An operad $\mathcal{O}$ is a theory of how things arrange in higher-level things.

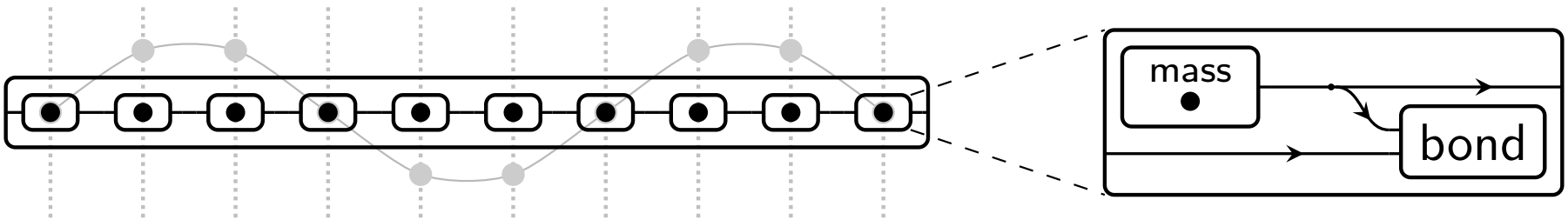
- $\mathcal{O}$ has a set $\mathrm{Ob}(\mathcal{O})$ of objects, which I like to call *interfaces*.
- And $\mathcal{O}$ has K-ary morphisms, which I like to call *arrangements*.
- Arrangements $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ are drawn as K things in a thing.
- Arrangements nest (composition), e.g. as shown:



Operadic syntax for interaction

An operad $\mathcal{O}$ is a theory of how things arrange in higher-level things.

- $\mathcal{O}$ has a set $\mathrm{Ob}(\mathcal{O})$ of objects, which I like to call *interfaces*.
- And $\mathcal{O}$ has K-ary morphisms, which I like to call *arrangements*.
- Arrangements $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ are drawn as K things in a thing.
- Arrangements nest (composition), e.g. as shown:



We'll be giving a functorial models for this theory.

- Our operad will be called $\mathbb{A}\mathbf{rr}$; its oper'ns are *dynamic arrangements*.
- We'll have models via two functors $\Phi_{\mathrm{phase}}, \Phi_{\mathrm{conf}}\colon \mathbb{A}\mathbf{rr} \to \mathbb{P}\mathbf{C}$.
- They compile diagrams like the above into discrete dynamical systems.
- The above diagram will compile into the discrete wave and heat eq'ns.

Outline

Adaptive arrangements

An adaptive arrangement is a morphism $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ in $\mathbb{A}\mathbf{rr}$

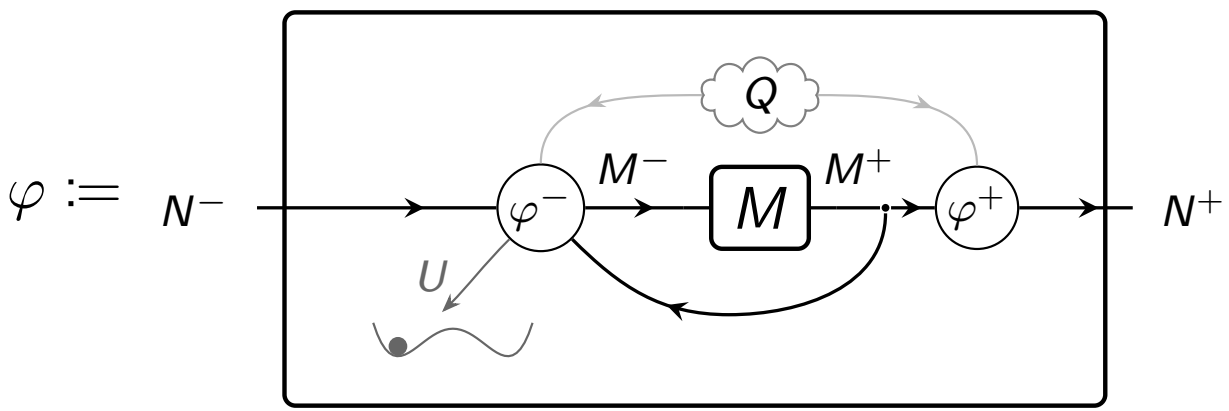
- Each m, called an *interface*, consists of two manifolds $m = \binom{M^-}{M^+}$.
- Roughly output M^+ is what m's exterior can include in potentials, ...
- ... and input M^- is what m's interior can include in potentials.

Adaptive arrangements

An adaptive arrangement is a morphism $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ in $\mathbb{A}\mathbf{rr}$

- Each m, called an *interface*, consists of two manifolds $m = \binom{M^-}{M^+}$.
- Roughly output M^+ is what m's exterior can include in potentials, ...
- ... and input M^- is what m's interior can include in potentials.

The arrangement φ parameterizes information flow and carries a potential.



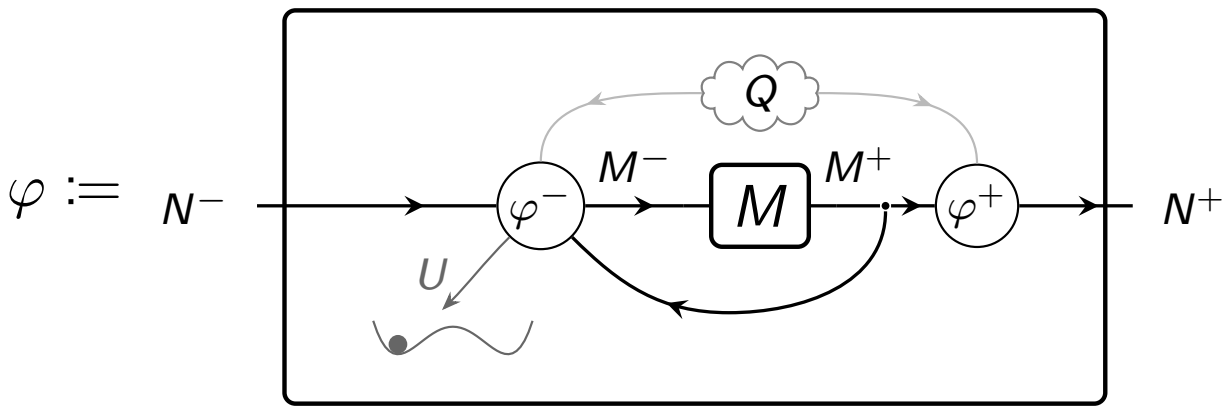
- $\varphi^+\colon Q \times M^+ \to N^+$ and $\varphi^-\colon Q \times M^+ \times N^- \to M^-$ "wiring"
- $U\colon Q \times M^+ \times N^- \to \mathbb{R}$ "potential"

Adaptive arrangements

An adaptive arrangement is a morphism $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ in $\mathbb{A}\mathbf{rr}$

- Each m, called an *interface*, consists of two manifolds $m = \binom{M^-}{M^+}$.
- Roughly output M^+ is what m's exterior can include in potentials, ...
- ... and input M^- is what m's interior can include in potentials.

The arrangement φ parameterizes information flow and carries a potential.



- $\varphi^+\colon Q \times M^+ \to N^+$ and $\varphi^-\colon Q \times M^+ \times N^- \to M^-$ "wiring"
- $U\colon Q \times M^+ \times N^- \to \mathbb{R}$ "potential"

This is all packaged using known CT formalisms. The operad is

$$\mathbb{A}\mathbf{rr} := \mathbb{P}\mathbf{ara}_{\mathbf{RVect}}(\mathbf{Lens}_{\mathbf{Mfd}}^{\mathbb{R}\times\square})$$

Semantics: systems that run on a computer

The semantics is a functor $\Phi\colon \mathbb{A}\mathbf{rr} \to \mathbb{P}\mathbf{C}$ into "polynomial coalgebras."

- A polynomial coalgebra is an input-output discrete dynamical system.
- Inputs and outputs will include points $x \in M$, covectors $\xi \in T^*_x M$,...
- ... and 1-forms $\omega\colon M \to T^*M$ on manifolds.

Semantics: systems that run on a computer

The semantics is a functor $\Phi\colon \mathbb{A}\mathbf{rr} \to \mathbb{P}\mathbf{C}$ into "polynomial coalgebras."

- A polynomial coalgebra is an input-output discrete dynamical system.
- Inputs and outputs will include points $x \in M$, covectors $\xi \in T^*_x M$,...
- ... and 1-forms $\omega\colon M \to T^*M$ on manifolds.

The arrangements $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ become dynamic coupling laws.

- Recall $\varphi = (Q, \sharp_Q, \varphi^+, \varphi^-, U)$ where $\varphi^+\colon Q \times M^+ \to N^+$,...
- ... $\varphi^-\colon Q \times M^+ \times N^- \to M^-$ and $U\colon Q \times M^+ \times N^- \to \mathbb{R}$
- φ determines how points, covectors, 1-forms are passed per timestep.
- The map $\sharp_Q$ determines how $dU|_q$ updates the parameter.

Semantics: systems that run on a computer

The semantics is a functor $\Phi\colon \mathbb{A}\mathbf{rr} \to \mathbb{P}\mathbf{C}$ into "polynomial coalgebras."

- A polynomial coalgebra is an input-output discrete dynamical system.
- Inputs and outputs will include points $x \in M$, covectors $\xi \in T^*_x M$,...
- ... and 1-forms $\omega\colon M \to T^*M$ on manifolds.

The arrangements $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ become dynamic coupling laws.

- Recall $\varphi = (Q, \sharp_Q, \varphi^+, \varphi^-, U)$ where $\varphi^+\colon Q \times M^+ \to N^+$,...
- ... $\varphi^-\colon Q \times M^+ \times N^- \to M^-$ and $U\colon Q \times M^+ \times N^- \to \mathbb{R}$
- φ determines how points, covectors, 1-forms are passed per timestep.
- The map $\sharp_Q$ determines how $dU|_q$ updates the parameter.

Integrators hold state S and turn incoming covectors into state updates.

- Φ_{conf} uses $S := Q$ and $q^+ := q - \sharp_Q(\xi)$, gradient descent.
- Φ_{phase} uses $S := T^*Q$ and $(q, \xi)^+ := (q + \sharp(\xi), \xi - dU|_{q+\sharp(\xi)})$.
- Conj: We can also do multi-stage integrators, e.g. Runge-Kutta.

Semantics: systems that run on a computer

The semantics is a functor $\Phi\colon \mathbb{A}\mathbf{rr} \to \mathbb{P}\mathbf{C}$ into "polynomial coalgebras."

- A polynomial coalgebra is an input-output discrete dynamical system.
- Inputs and outputs will include points $x \in M$, covectors $\xi \in T^*_x M$,...
- ... and 1-forms $\omega\colon M \to T^*M$ on manifolds.

The arrangements $\varphi\colon m_1 \otimes \cdots \otimes m_K \to n$ become dynamic coupling laws.

- Recall $\varphi = (Q, \sharp_Q, \varphi^+, \varphi^-, U)$ where $\varphi^+\colon Q \times M^+ \to N^+$,...
- ... $\varphi^-\colon Q \times M^+ \times N^- \to M^-$ and $U\colon Q \times M^+ \times N^- \to \mathbb{R}$
- φ determines how points, covectors, 1-forms are passed per timestep.
- The map $\sharp_Q$ determines how $dU|_q$ updates the parameter.

Integrators hold state S and turn incoming covectors into state updates.

- Φ_{conf} uses $S := Q$ and $q^+ := q - \sharp_Q(\xi)$, gradient descent.
- Φ_{phase} uses $S := T^*Q$ and $(q, \xi)^+ := (q + \sharp(\xi), \xi - dU|_{q+\sharp(\xi)})$.
- Conj: We can also do multi-stage integrators, e.g. Runge-Kutta.

A monoidality condition in our def'n of integrator ensures compositionality:

- Coupling systems in the physics formalism, then discretizing...
- ...is precisely the same as discretizing each and then coupling.

Outline

Gradient descent & backpropagation

A smooth parameterized model $F\colon Q \times \mathbb{R}^m \to \mathbb{R}^n$ gives the arrangement

$$f := (Q, !, F, 0)\colon \begin{pmatrix} 1 \\ \mathbb{R}^m \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ \mathbb{R}^n \end{pmatrix}$$

For any label mfd Y, a smooth loss $U\colon \mathbb{R}^n \times Y \to \mathbb{R}$ is also an arrange'nt:

$$\mathrm{loss}_U := (0, !, !, U)\colon \begin{pmatrix} 1 \\ \mathbb{R}^n \end{pmatrix} \otimes \begin{pmatrix} 1 \\ Y \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Gradient descent & backpropagation

A smooth parameterized model $F\colon Q \times \mathbb{R}^m \to \mathbb{R}^n$ gives the arrangement

$$f := (Q, !, F, 0)\colon \begin{pmatrix} 1 \\ \mathbb{R}^m \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ \mathbb{R}^n \end{pmatrix}$$

For any label mfd Y, a smooth loss $U\colon \mathbb{R}^n \times Y \to \mathbb{R}$ is also an arrange'nt:

$$\mathrm{loss}_U := (0, !, !, U)\colon \begin{pmatrix} 1 \\ \mathbb{R}^n \end{pmatrix} \otimes \begin{pmatrix} 1 \\ Y \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Nesting them gives an arrang't $g_U := \mathrm{loss}_U \circ (f \otimes \mathrm{id}_Y)\colon \binom{1}{\mathbb{R}^m} \otimes \binom{1}{Y} \to \binom{1}{1}$

- Inputs $= (x, y)$; loss $L_{x,y}(q) := U(F(q, x), y)$
- $\Phi_{\mathrm{conf}}(g_U)$ gives $q^+ = q - \sharp_q(dL_{x,y}(q))$, gradient descent.
- NN layers nest in $\mathbb{A}\mathbf{rr}$; functoriality of Φ_{conf} gives backprop'n.

Gradient descent & backpropagation

A smooth parameterized model $F\colon Q \times \mathbb{R}^m \to \mathbb{R}^n$ gives the arrangement

$$f := (Q, !, F, 0)\colon \begin{pmatrix} 1 \\ \mathbb{R}^m \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ \mathbb{R}^n \end{pmatrix}$$

For any label mfd Y, a smooth loss $U\colon \mathbb{R}^n \times Y \to \mathbb{R}$ is also an arrange'nt:

$$\mathrm{loss}_U := (0, !, !, U)\colon \begin{pmatrix} 1 \\ \mathbb{R}^n \end{pmatrix} \otimes \begin{pmatrix} 1 \\ Y \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Nesting them gives an arrang't $g_U := \mathrm{loss}_U \circ (f \otimes \mathrm{id}_Y)\colon \binom{1}{\mathbb{R}^m} \otimes \binom{1}{Y} \to \binom{1}{1}$

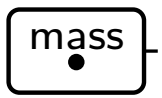
- Inputs $= (x, y)$; loss $L_{x,y}(q) := U(F(q, x), y)$
- $\Phi_{\mathrm{conf}}(g_U)$ gives $q^+ = q - \sharp_q(dL_{x,y}(q))$, gradient descent.
- NN layers nest in $\mathbb{A}\mathbf{rr}$; functoriality of Φ_{conf} gives backprop'n.

One-step system id: using $m = n$ and $y_t := x_{t+1}$ fits q online by $L_{x_t, x_{t+1}}(q)$.

Wave & heat equations (same arrangement)

Let $G = (V, m, E, \kappa)$ be a weighted graph, $m\colon V \to \mathbb{R}_{>0}$, $\kappa\colon E \to \mathbb{R}_{>0}$.

- Each vertex v gives a potential-free mass arrangement

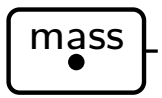


$$\mathrm{mass}_v := (\mathbb{R}, !, \mathrm{id}, 0)\colon \begin{pmatrix} 1 \\ 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 \\ \mathbb{R} \end{pmatrix}, \qquad \sharp_v(\xi) = \xi / m_v$$

Wave & heat equations (same arrangement)

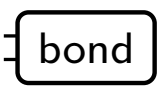
Let $G = (V, m, E, \kappa)$ be a weighted graph, $m\colon V \to \mathbb{R}_{>0}$, $\kappa\colon E \to \mathbb{R}_{>0}$.

- Each vertex v gives a potential-free mass arrangement



$$\mathrm{mass}_v := (\mathbb{R}, !, \mathrm{id}, 0)\colon \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}1\\\mathbb{R}\end{pmatrix}, \qquad \sharp_v(\xi) = \xi/m_v$$

- Each edge $e = (u, v)$ gives a stateless harmonic-bond arrangement

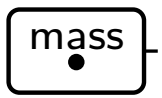


$$\mathrm{bond}_e := (0, !, !, U_e)\colon \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}\mathbb{R}^2\\1\end{pmatrix}, \qquad U_e(x, y) = \frac{\kappa_e}{2}(x-y)^2.$$

Wave & heat equations (same arrangement)

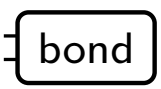
Let $G = (V, m, E, \kappa)$ be a weighted graph, $m\colon V \to \mathbb{R}_{>0}$, $\kappa\colon E \to \mathbb{R}_{>0}$.

- Each vertex v gives a potential-free mass arrangement



$$\mathrm{mass}_v := (\mathbb{R}, !, \mathrm{id}, 0)\colon \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}1\\\mathbb{R}\end{pmatrix}, \qquad \sharp_v(\xi) = \xi/m_v$$

- Each edge $e = (u, v)$ gives a stateless harmonic-bond arrangement



$$\mathrm{bond}_e := (0, !, !, U_e)\colon \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}\mathbb{R}^2\\1\end{pmatrix}, \qquad U_e(x, y) = \frac{\kappa_e}{2}(x-y)^2.$$

Composing them according to G gives a closed arrange't, $\mathrm{wire}_G\colon \binom{1}{1} \to \binom{1}{1}$

- The following is all calculated from the machinery, not imposed.
- wire_G's parameter is $Q_G = \mathbb{R}^V$, with $\sharp_q(\xi) = M^{-1}\xi$. $M = \mathrm{diag}(m)$.
- wire_G's potential is $U_G(q) = \frac{1}{2}q^\top L_G q$, so $dU_G(q) = L_G q$.

Wave & heat equations (same arrangement)

Let $G = (V, m, E, \kappa)$ be a weighted graph, $m\colon V \to \mathbb{R}_{>0}$, $\kappa\colon E \to \mathbb{R}_{>0}$.

- Each vertex v gives a potential-free mass arrangement

mass $\quad \mathrm{mass}_v := (\mathbb{R}, !, \mathrm{id}, 0)\colon \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}1\\\mathbb{R}\end{pmatrix}, \qquad \sharp_v(\xi) = \xi/m_v$

- Each edge $e = (u, v)$ gives a stateless harmonic-bond arrangement

bond $\quad \mathrm{bond}_e := (0, !, !, U_e)\colon \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}\mathbb{R}^2\\1\end{pmatrix}, \qquad U_e(x, y) = \frac{\kappa_e}{2}(x-y)^2.$

Composing them according to G gives a closed arrange't, $\mathrm{wire}_G\colon \binom{1}{1} \to \binom{1}{1}$

- The following is all calculated from the machinery, not imposed.
- wire_G's parameter is $Q_G = \mathbb{R}^V$, with $\sharp_q(\xi) = M^{-1}\xi$. $M = \mathrm{diag}(m)$.
- wire_G's potential is $U_G(q) = \frac{1}{2}q^\top L_G q$, so $dU_G(q) = L_G q$.

One can investigate the resulting discrete dynamical systems and find:

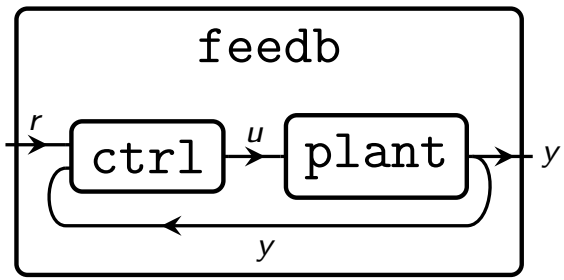
$$\Phi_{\mathrm{conf}}(\mathrm{wire}_G) \quad \textit{satisfies} \quad M\dot{q} = -L_G q \quad \text{discrete heat equation,}$$
$$\Phi_{\mathrm{phase}}(\mathrm{wire}_G) \quad \textit{satisfies} \quad M\ddot{q} = -L_G q \quad \text{discrete wave equation.}$$

Same graph, potential, and reaction; only the integrator changes.

Energy-based feedback

Each port passes values forward and covectors—generalized forces—backward.

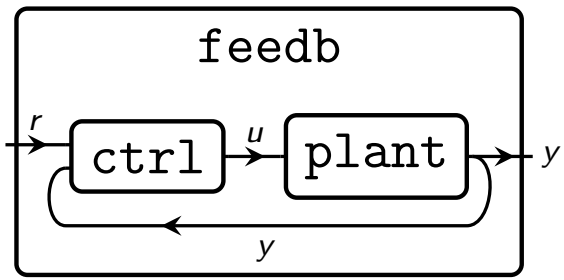
- `plant`: $y \coloneqq y_p(x_p), \quad U_p(x_p, u) \in \mathbb{R}.$
- `ctrl`: $u \coloneqq u_c(x_c), \quad U_c(x_c, r, y) \in \mathbb{R}.$
- Covectors drive state updates through $\sharp$.



Energy-based feedback

Each port passes values forward and covectors—generalized forces—backward.

- `plant`: $y \coloneqq y_p(x_p), \quad U_p(x_p, u) \in \mathbb{R}.$
- `ctrl`: $u \coloneqq u_c(x_c), \quad U_c(x_c, r, y) \in \mathbb{R}.$
- Covectors drive state updates through $\sharp$.



At reference input r, the static wiring `feedb` produces the composite potential:

$$U(x_p, x_c, r) \coloneqq U_p\big(x_p, u_c(x_c)\big) + U_c\big(x_c, r, y_p(x_p)\big).$$

The configuration integrator gives the following state update at $\xi_y = 0$:

$$(x_p^+, x_c^+) = (x_p, x_c) - (\sharp_p \oplus \sharp_c)\, dU(x_p, x_c, r).$$

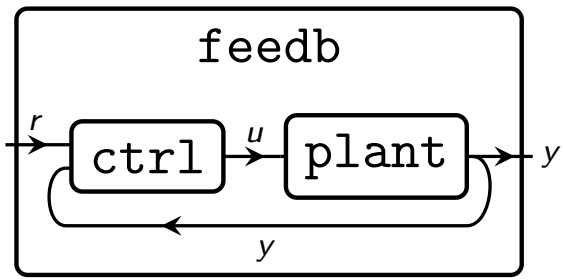
The backward-traveling covectors are:

$$d_y U_c + \xi_y \rightsquigarrow \texttt{plant}, \qquad d_u U_p \rightsquigarrow \texttt{ctrl}, \qquad d_r U_c \rightsquigarrow \texttt{exterior}.$$

Energy-based feedback

Each port passes values forward and covectors—generalized forces—backward.

- `plant`: $y \coloneqq y_p(x_p), \quad U_p(x_p, u) \in \mathbb{R}.$
- `ctrl`: $u \coloneqq u_c(x_c), \quad U_c(x_c, r, y) \in \mathbb{R}.$
- Covectors drive state updates through $\sharp$.



At reference input r, the static wiring `feedb` produces the composite potential:

$$U(x_p, x_c, r) \coloneqq U_p\big(x_p, u_c(x_c)\big) + U_c\big(x_c, r, y_p(x_p)\big).$$

The configuration integrator gives the following state update at $\xi_y = 0$:

$$(x_p^+, x_c^+) = (x_p, x_c) - (\sharp_p \oplus \sharp_c)\, dU(x_p, x_c, r).$$

The backward-traveling covectors are:

$$d_y U_c + \xi_y \rightsquigarrow \texttt{plant}, \qquad d_u U_p \rightsquigarrow \texttt{ctrl}, \qquad d_r U_c \rightsquigarrow \texttt{exterior}.$$

Monoidal integrators make discretization commute with interconnection:

$$\Phi\big(\texttt{feedb} \circ (\texttt{ctrl} \otimes \texttt{plant})\big) = \Phi(\texttt{feedb}) \circ \big(\Phi(\texttt{ctrl}) \otimes \Phi(\texttt{plant})\big).$$

Outline

Connection to the portfolio, and impact

Fit with the AFOSR portfolio

- Serves three funding programs at once: dynamics, learning, assurance.
- Learning and physical dynamics fit into a single high-assurance syntax.
- Comp'l theory of *open* systems: plants, controllers, and learners interconnect.

Connection to the portfolio, and impact

Fit with the AFOSR portfolio

- Serves three funding programs at once: dynamics, learning, assurance.
- Learning and physical dynamics fit into a single high-assurance syntax.
- Comp'l theory of *open* systems: plants, controllers, and learners interconnect.

What it enables for the DAF, if successful

- Discretized models compose exactly: no error introduced as subsystems join.
- The math fully specifies simulation algorithms; LLMs automat'ly write code.
- Swap a component; the assembled dynamics follows, with no re-derivation.

Connection to the portfolio, and impact

Fit with the AFOSR portfolio

- Serves three funding programs at once: dynamics, learning, assurance.
- Learning and physical dynamics fit into a single high-assurance syntax.
- Comp'l theory of *open* systems: plants, controllers, and learners interconnect.

What it enables for the DAF, if successful

- Discretized models compose exactly: no error introduced as subsystems join.
- The math fully specifies simulation algorithms; LLMs automat'ly write code.
- Swap a component; the assembled dynamics follows, with no re-derivation.

Qualitative & quantitative impact

- Book with Nelson Niu (Cambridge UP); papers on comonads, dynamics.
- Code released publicly; Topos colloquium and research staff both growing.
- Grant-supported attractor-lattice work seeded a collaboration with B. Kalies.
- Citation rate up 40% since 2023 grant start.

Connection to the portfolio, and impact

Fit with the AFOSR portfolio

- Serves three funding programs at once: dynamics, learning, assurance.
- Learning and physical dynamics fit into a single high-assurance syntax.
- Comp'l theory of *open* systems: plants, controllers, and learners interconnect.

What it enables for the DAF, if successful

- Discretized models compose exactly: no error introduced as subsystems join.
- The math fully specifies simulation algorithms; LLMs automat'ly write code.
- Swap a component; the assembled dynamics follows, with no re-derivation.

Qualitative & quantitative impact

- Book with Nelson Niu (Cambridge UP); papers on comonads, dynamics.
- Code released publicly; Topos colloquium and research staff both growing.
- Grant-supported attractor-lattice work seeded a collaboration with B. Kalies.
- Citation rate up 40% since 2023 grant start.

Personal impact of the funding

- Funds 100% of my time; enabled this move into physics and dynamics.
- Grant supports postdocs and students that move on to good academic jobs.

Summary

Learning and Hamiltonian dynamics share a compositional syntax.

- Arrangements specify interfaces, wiring, parameters, and potentials.
- Observations flow forward; error covectors flow backward.
- Each integrator gives a functorial compiler into executable machines.

Summary

Learning and Hamiltonian dynamics share a compositional syntax.

- Arrangements specify interfaces, wiring, parameters, and potentials.
- Observations flow forward; error covectors flow backward.
- Each integrator gives a functorial compiler into executable machines.

Examples include gradient descent, heat flow, waves, energy-based systems.

- The semantics applies far beyond harmonic systems.
- Physical machines can be interconnected with learning machines.
- The same machinery estimates physical parameters online.

Future work: linearization, continuum limits, generalized attention.

Thanks! Comments and questions welcome...