Interaction Networks

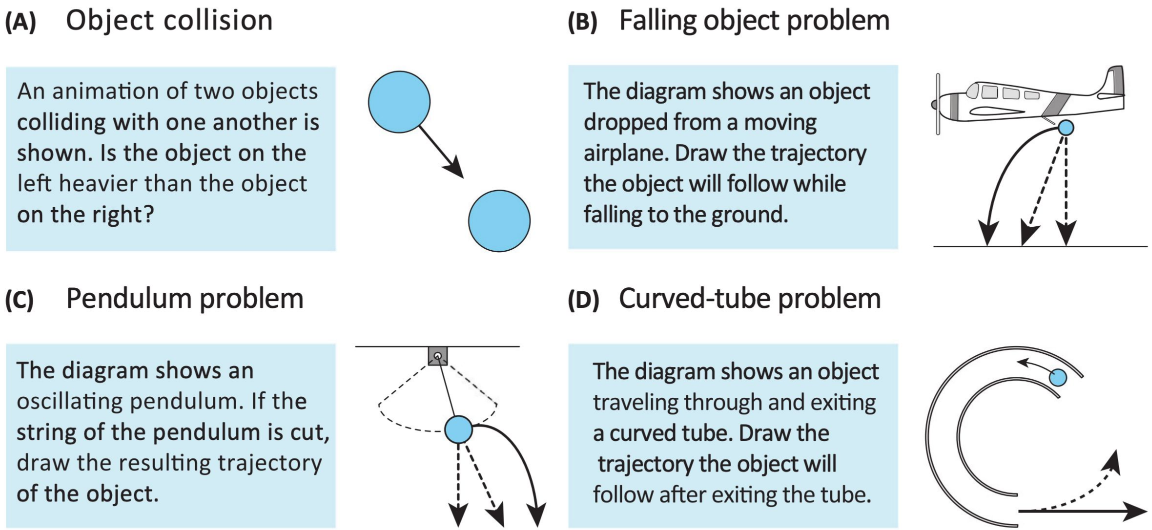
for Learning about Objects, Relations and Physics

Presenters: Shuwen Qiu, Jiayue Sun, Qing Li

2021-01-28

Motivation

- Representing and reasoning about objects, relations and physics is a “core” domain of human common sense knowledge



Introduction

- Scenes in daily life

Introduction

- Many everyday problems are challenging for models
  - predicting what will happen next in physical environments
  - inferring underlying properties of complex scenes
- People can nevertheless solve such problems by decomposing the scenario into distinct objects and relations, and reasoning about the consequences of their interactions and dynamics

Introduction

- Interaction Networks
  - perform an analogous form of reasoning about objects and relations in complex systems
  - combine three powerful approaches
    - structured models: exploit rich, explicit knowledge of relations among objects
    - simulation: an effective method for approximating dynamical systems
    - deep learning: couples generic architectures with efficient optimization algorithms to provide highly scalable learning and inference in challenging real-world settings
  - explicitly separate how they reason about relations from how they reason about objects
    - automatically generalize their learning across variable numbers of arbitrarily ordered objects and relations
    - recompose their knowledge of entities and interactions in novel and combinatorially many ways

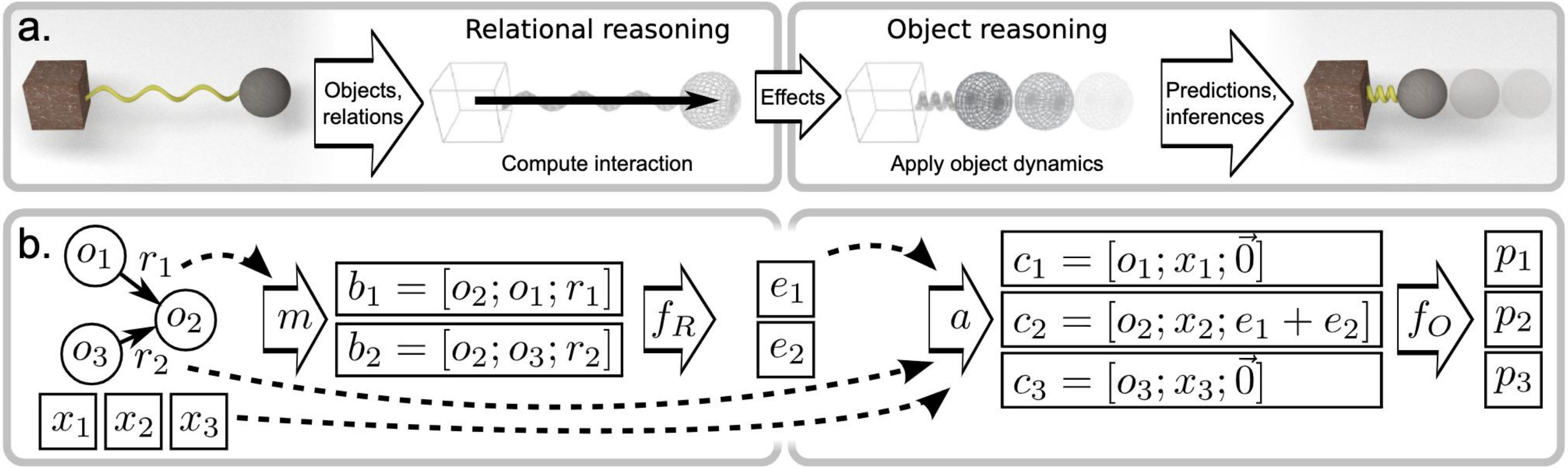
Introduction

- Evaluation focus
  - whether IN can predict future states
  - whether IN can predict abstract physical properties, such as energy
  - how they generalize to novel systems with different numbers and configurations of elements

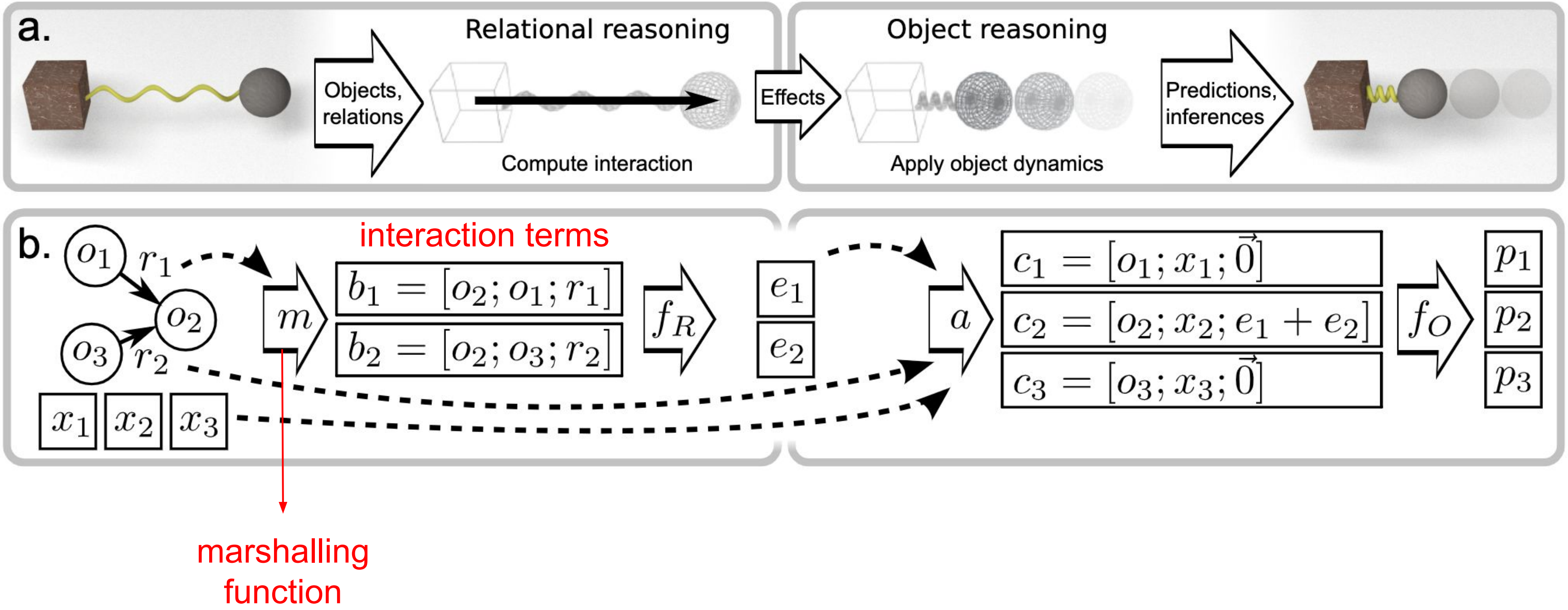
Related Work

- Physical simulation engine
  - generates sequences of states by repeatedly applying rules that approximate the effects of physical interactions and dynamics on objects over time
  - the interaction rules are relation-centric, operating on two or more objects that are interacting
  - the dynamics rules are object-centric, operating on individual objects and the aggregated effects of the interactions they participate in
- Previous AI work on physical reasoning
  - predict and control the state of articulated bodies
  - learn fluid dynamics
  - CNNs used to predict coarse-grained physical dynamics from images

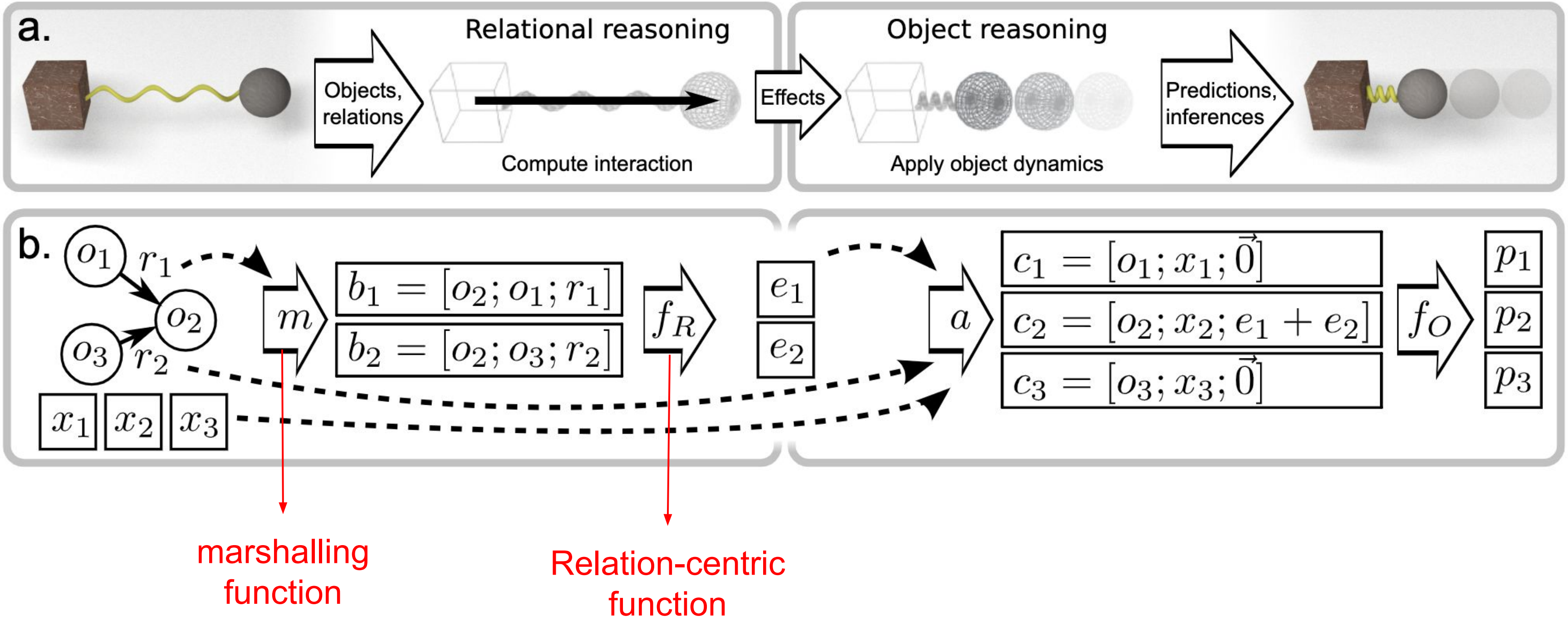
Schematic of Interaction Networks (IN)



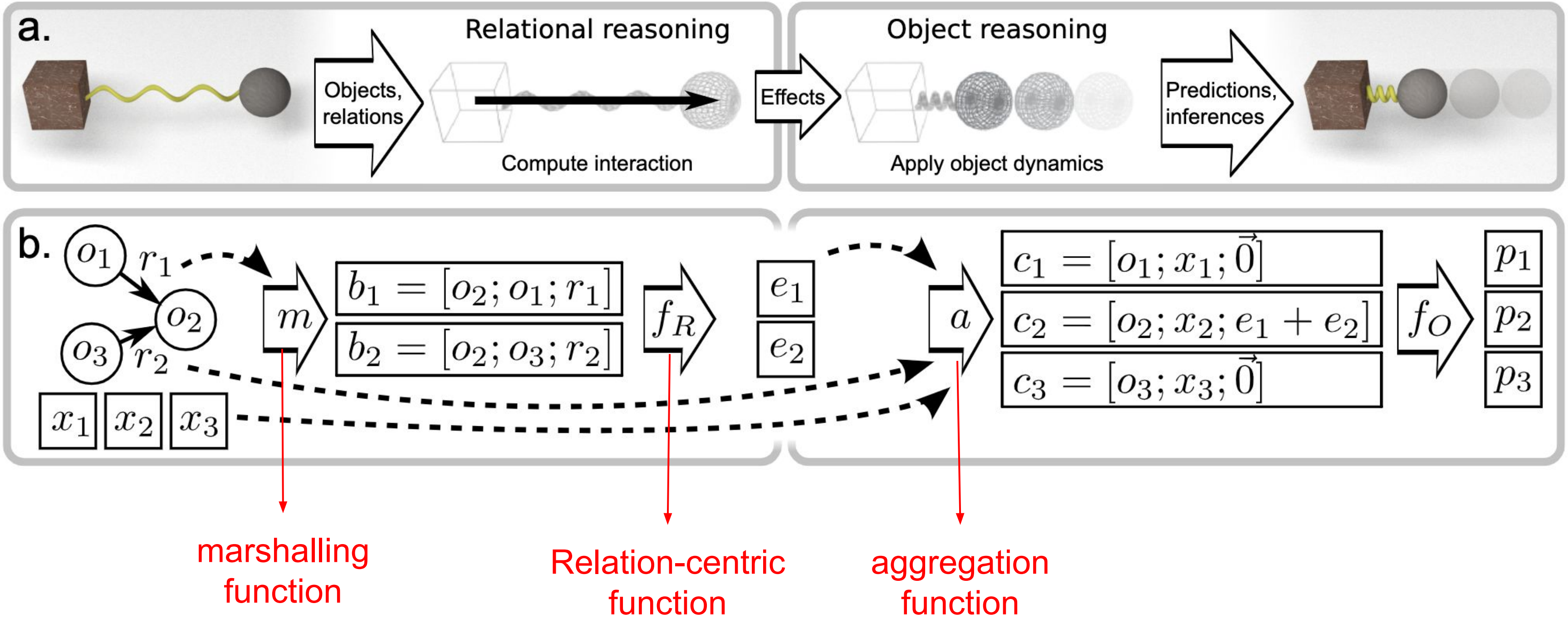
Schematic of Interaction Networks (IN)



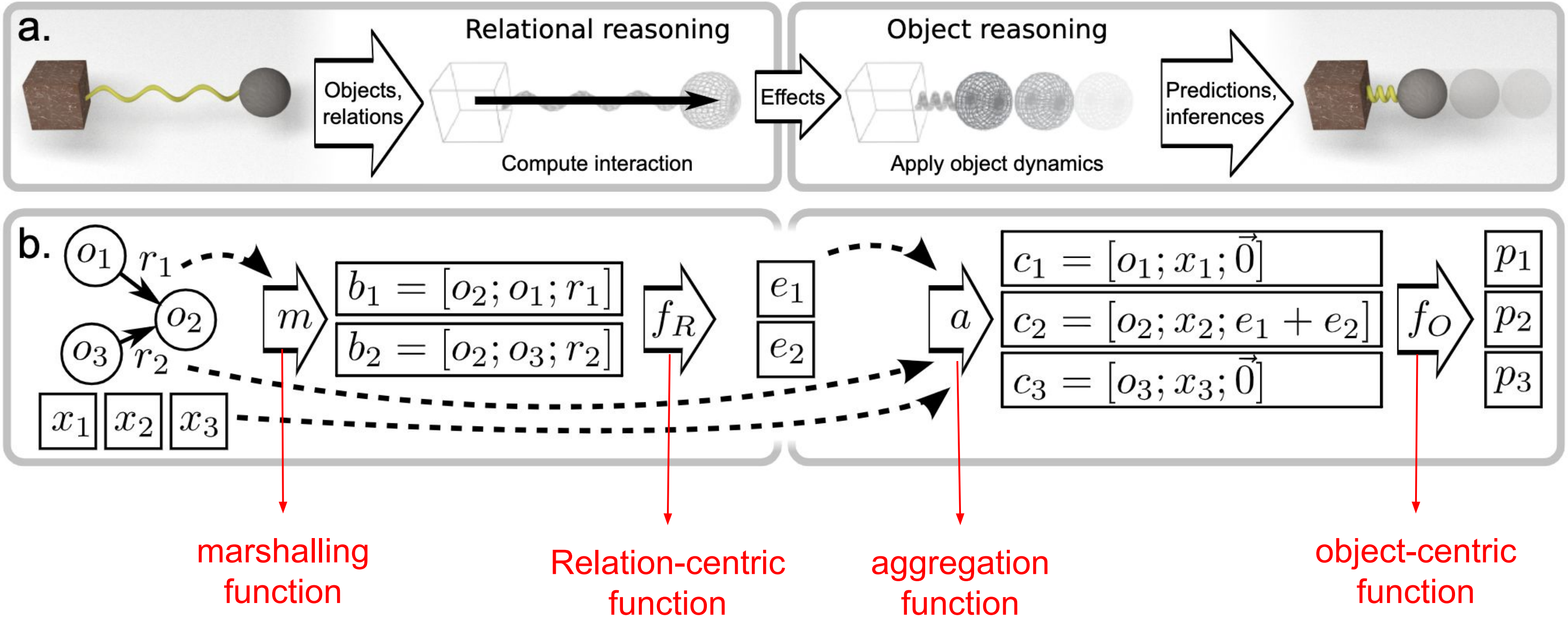
Schematic of Interaction Networks (IN)



Schematic of Interaction Networks (IN)



Schematic of Interaction Networks (IN)



A Formal Definition of IN

Represent a whole system as a graph $G = \langle O, R \rangle$

$$O = \{o_j\}_{j=1 \ldots N_O} \quad R = \{\langle i, j, r_k \rangle_k\}_{k=1 \ldots N_R} \quad X = \{x_j\}_{j=1 \ldots N_O}$$

$$\mathrm{IN}(G) = \phi_O(a(G,\ X,\ \phi_R(\ m(G)\ )\ ))$$

$$\begin{aligned} m(G) &= B = \{b_k\}_{k=1 \ldots N_R} \\ f_R(b_k) &= e_k \\ \phi_R(B) &= E = \{e_k\}_{k=1 \ldots N_R} \end{aligned}$$

$$\begin{aligned} a(G, X, E) &= C = \{c_j\}_{j=1 \ldots N_O} \\ f_O(c_j) &= p_j \\ \phi_O(C) &= P = \{p_j\}_{j=1 \ldots N_O} \end{aligned}$$

$$P \xrightarrow{\phi_A} q$$

N_R : number of relations

N_O : number of objects

b_k : $<o_i, o_j, r_k>$

(rearranges the objects and relations into interaction terms)

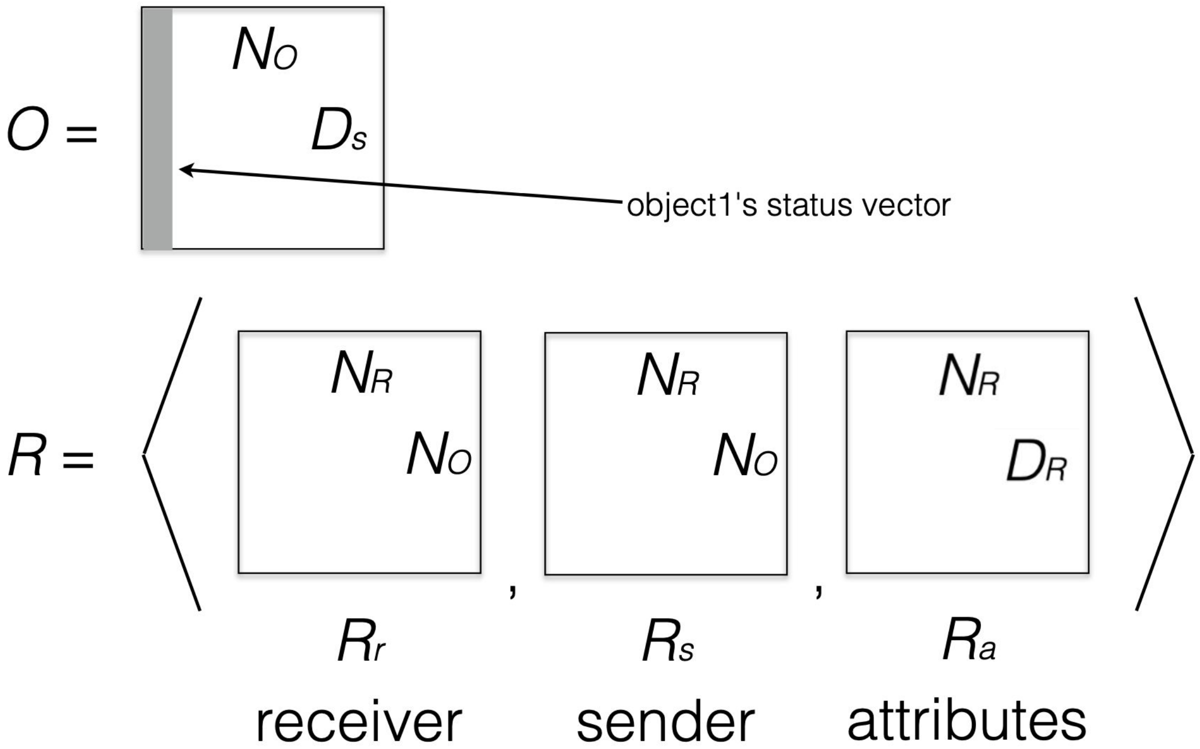
Relation

e: multiple for one object

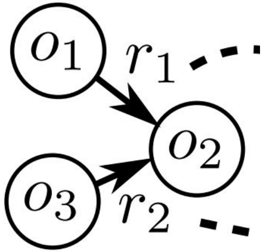
c: aggregated by *a*

A Learnable Implementation of IN

$$G = \langle O, R \rangle$$

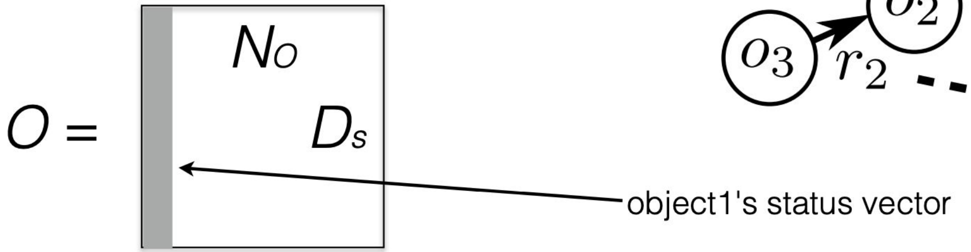


A Learnable Implementation of IN

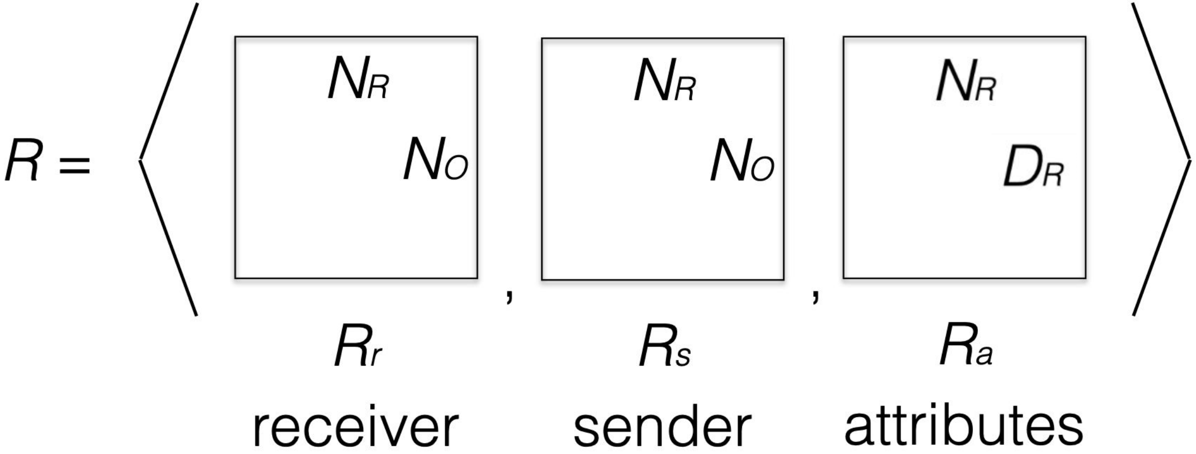


$$R_r = \begin{matrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{matrix}$$

$$R_s = \begin{matrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{matrix}$$



$G = \langle O, R \rangle$



Slides adapted from Ken Kuroki's presentation.

$$m(G) = \begin{array}{|c|}\hline N_R \\ \boldsymbol{OR_r} \qquad D_s \\ \hline \\ \boldsymbol{OR_s} \qquad D_s \\ \\ \hline \\ \boldsymbol{R_a} \qquad D_R \\ \\ \hline\end{array} = B$$

$$[b_1, b_2, \ldots, b_k]$$

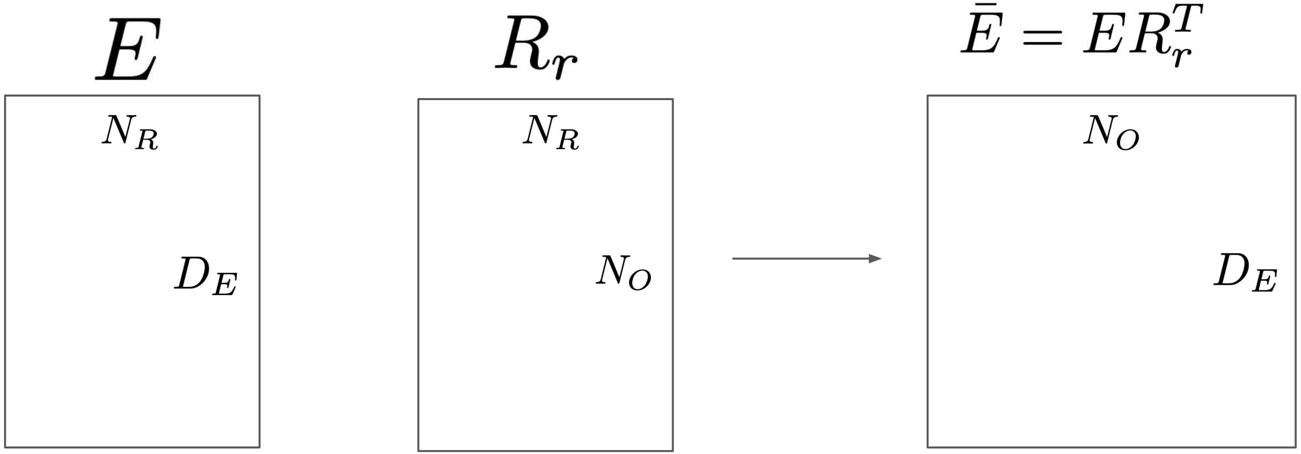
$$\big\downarrow f_R$$

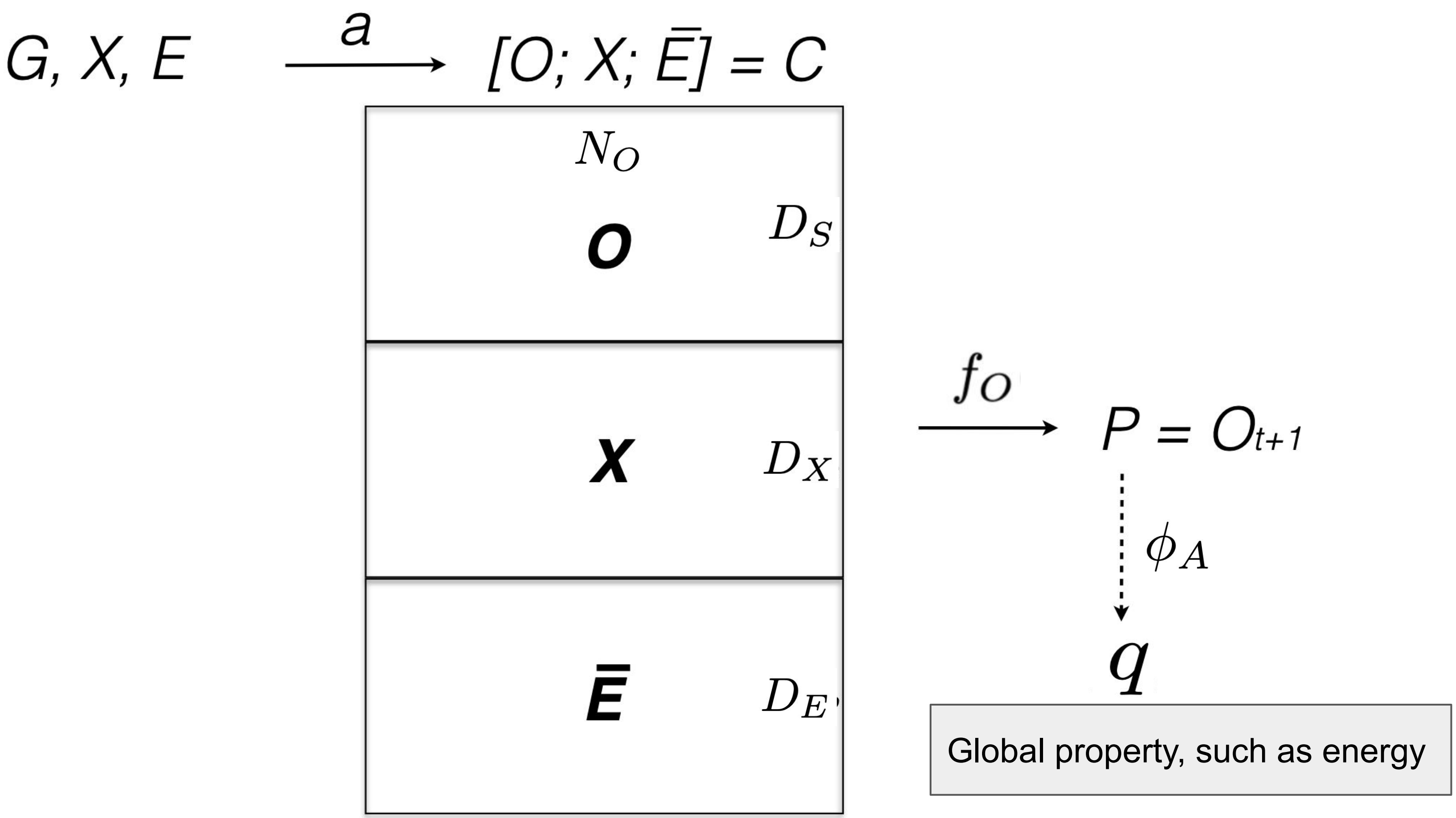
$$[e_1, e_2, \ldots, e_k] = E$$

$$m(G) = \begin{array}{cc} N_R & \\ \left[\begin{array}{c} \boldsymbol{OR_r} \\ \hline \boldsymbol{OR_s} \\ \hline \boldsymbol{R_a} \end{array}\right] & \begin{array}{c} D_s \\ D_s \\ D_R \end{array} \end{array} = B$$

$$[b_1, b_2, \ldots, b_k] \xrightarrow{f_R} [e_1, e_2, \ldots, e_k] = E$$

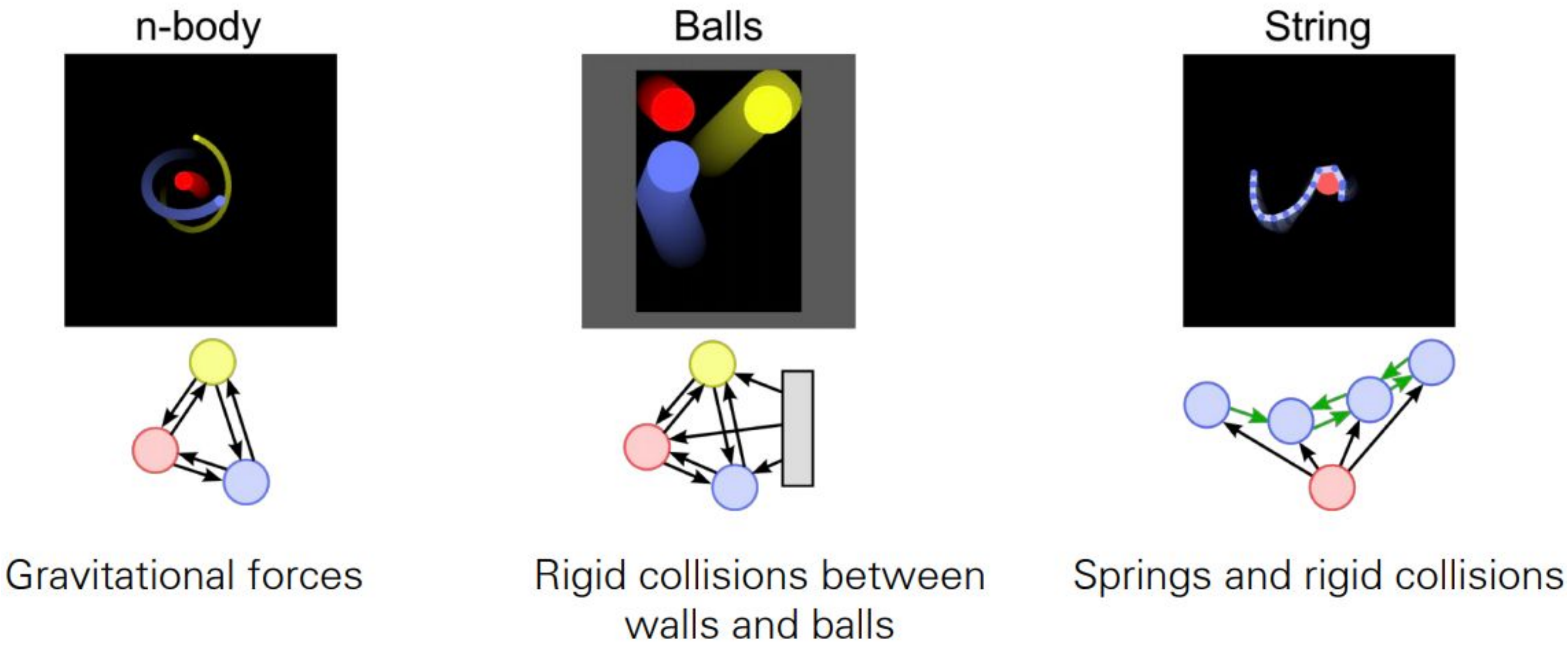
$$\begin{array}{cc} N_R & \\ \left[\begin{array}{c} \quad \end{array}\right] & D_E \end{array}$$


E
R_r
$\bar{E} = ER_r^T$
N_R
D_E
N_R
N_O
N_O
D_E


G, X, E
a
[O; X; Ē] = C
N_O
O
D_S
X
D_X
Ē
D_E
f_O
P = O_{t+1}
ϕ_A
q
Global property, such as energy

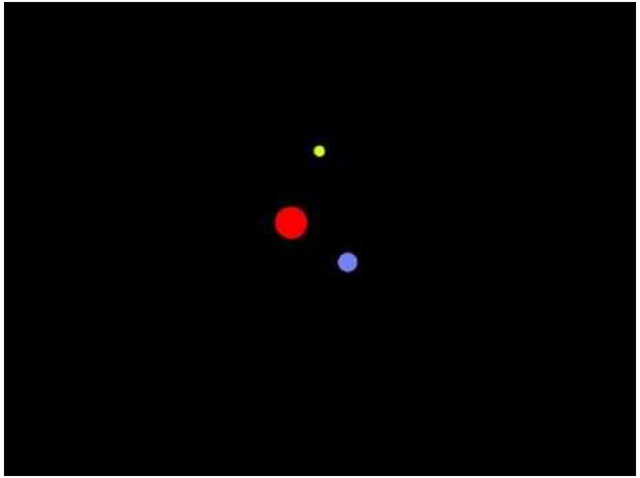
Experiments - Physical Reasoning Tasks

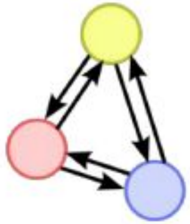
- 2 Reasoning Tasks:
  - Predict future states of a system (velocity)
  - Estimate abstract properties of a system (e.g. potential energy)
- 3 Physical domains:



N-body Domain

- N bodies with gravitational forces in between (distance and mass dependent)
- Varied object attributes (inputs):
  - Mass
- Train on 6-body scenes,
  test on 3, 6, 12-body scenes.
- Each body initialized with velocities
  - Half systems with random velocities
  - Half systems with velocities that would cause stable orbits



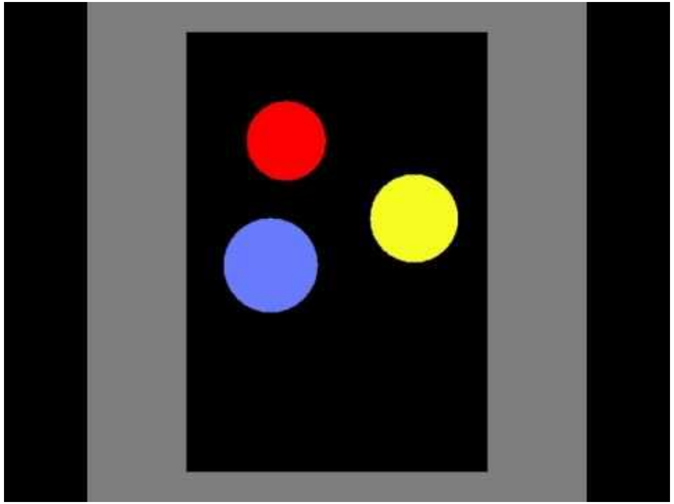


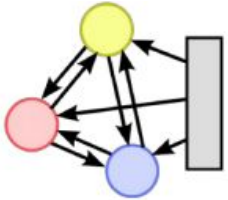
n(n-1)
relations

Gravitational forces

Balls bouncing in a box

- Moving balls collide with each other and static walls.
- Collision or straight-line motion
    - For each ball, only 1% steps with collision.
- Varied Attributes (inputs):
    - Object (include walls):
        - Shape: rectangle for wall, circle for balls
        - Scale: size of a shape
        - Mass
    - Relation: Coefficient of restitution
- Train on 6 balls, test on 3, 6, 9 balls



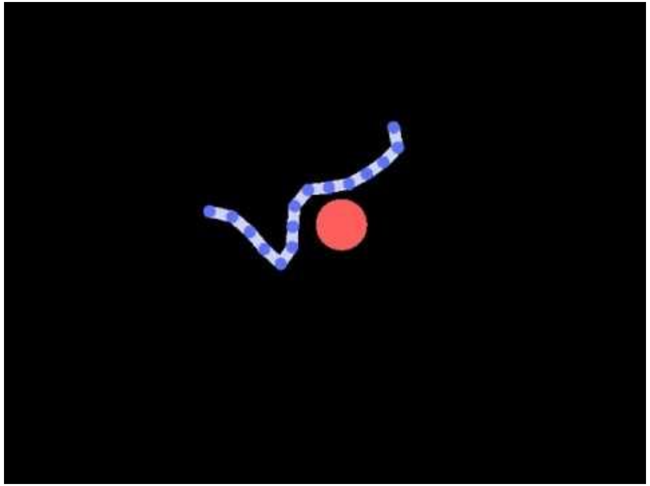


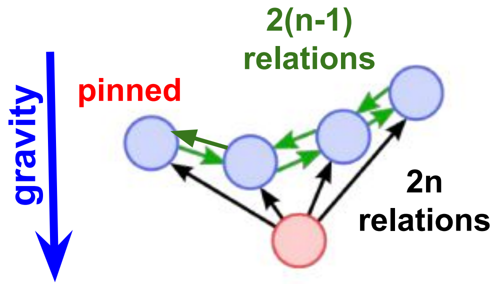
n(n-1) relations

Rigid collisions between walls and balls

N-mass String Domain

- A string and a circle:
  - **String:** n masses connected by springs
  - **Ends:** 0, 1 or both ends fixed
  - **Circle:** static circle below the string
- Relation Input:
  - **2(n-1) spring relations** with their immediate neighbors. (spring constant)
  - **2n relations** with the **rigid circle** (coefficient of restitution)
- External input:
  - Gravitational acceleration (varied across simulations)
- Train on 1-pin and 15 points, test on 0,1,2-pin and 5,15, 30 points.

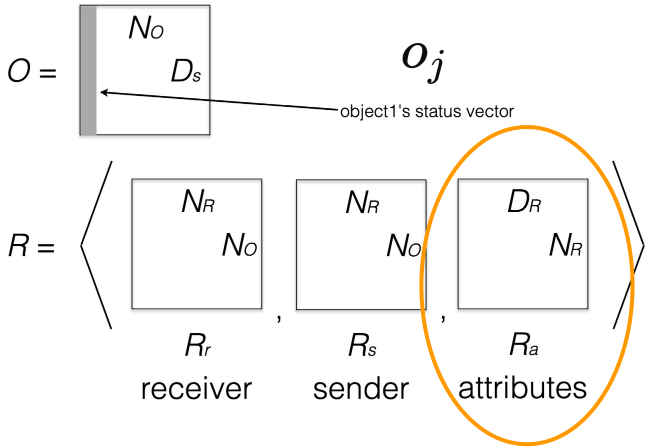




Springs and rigid collisions

Prediction Tasks

$$G = \langle O, R \rangle$$



- State G = Objects O and physical relations R
- Each object state, o_j is composed of
  - dynamic state component (e.g. position and velocity)
  - Static attribute component (e.g. mass, size, shape)
- The relation attributes R_a :
  e.g. coefficient of restitution, spring constant
- Prediction Target:
  - **Velocities** of the objects at the **subsequent** times step
  - **Potential energy** of the system at the **current** time step
- Multi-step rollouts
  - output **velocity** at t became the input velocity at t+1 $\mathbf{v}_t \rightarrow \mathbf{o}_{v,\, t+1}$
  - the **position** at t+1 was updated by the predicted velocity at t $\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_t \cdot t$

Data

- Data generated by simulating 200 scenes over 1000 time steps
  - Training: **1 million** one-step input/target pairs
  - Validation: **200k** one-step input/target pairs
  - Test: **200k** one-step input/target pairs
- Adding noise:
  - Adding **Gaussian noise** to 20% of the data's input positions and velocities initially. In epochs 50 - 250, the noise was gradually reduced to 0%.
  - Learn to project physically impossible states back to nearby, possible states.
  - Prediction error (MSE) is not influenced, generated rollout videos from models trained with noise are **slightly more visually realistic**.
  - Static objects (e.g. walls) less subject to drift.

Model Architecture

MLP Models	#hidden layers	hidden layer size	output size (D)
f_R	4	150	50 (E)
f_O	1	100	2 (velocity)
ϕ_A Input dim = 10	1	25	1

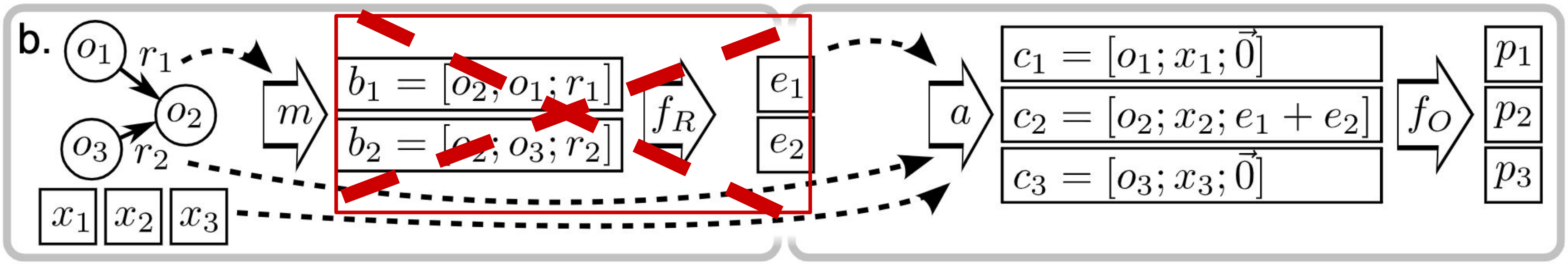
- MLP: multiple hidden layers of linear transformations + biases, followed by ReLU.
- Model architecture was selected by a grid search over layer sizes and depths.
- All training objectives and test measures used MSE.

L2 Regularization

- Apply to effects E and model parameters
- Regularize effects E: $\phi_R(B) \quad = E$
  - Improved generalization to different number of objects
  - Reduced drift over many rollout steps
- Regularize parameters:
  - Generally improved performance
  - Reduced overfitting
- Penalty factors were selected by a grid search

Model Comparison

- Constant velocity: outputs the input velocity
- Baseline MLP:
  - Architecture: two 300-length hidden layers
  - Took input as a flattened vector of all the input data (O, R and X)
- Dynamics-only IN:
  - Variant of IN with the relation model ϕ_R removed



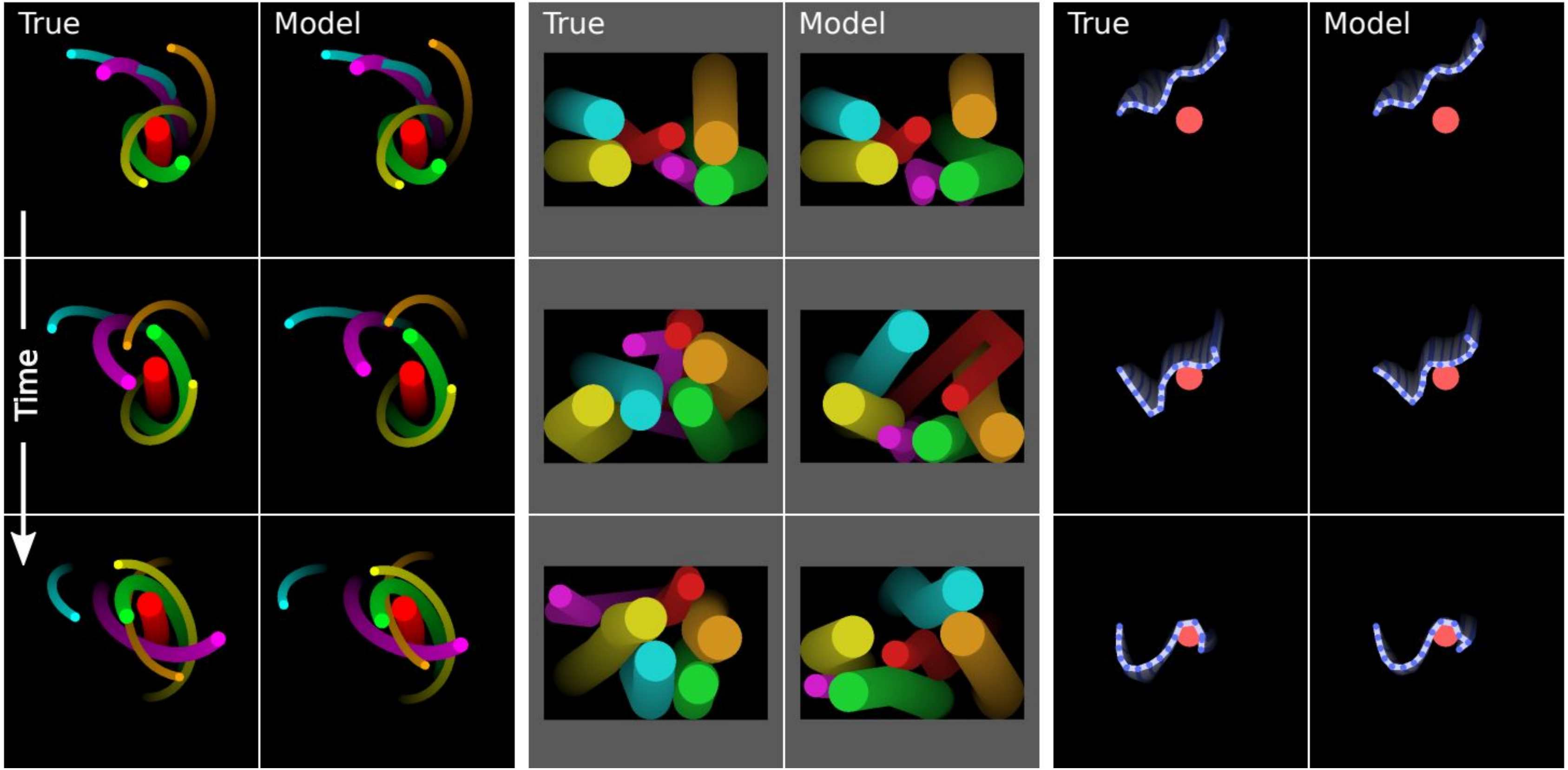
Results: Velocity Prediction

- Predict **next-step dynamics**: high accuracy
  - Orders of lower test error than other models
- Since **IN** can exploit dynamic interactions among objects for its predictions.
- **Dynamics-only IN** had no mechanism for processing interaction.
  It performs similarly to the constant velocity model.
- Baseline **MLP** in theory can learn object interactions but
  - hard to learn how to selectly process interactions based on relation indices.

Table 2: Prediction experiment MSEs

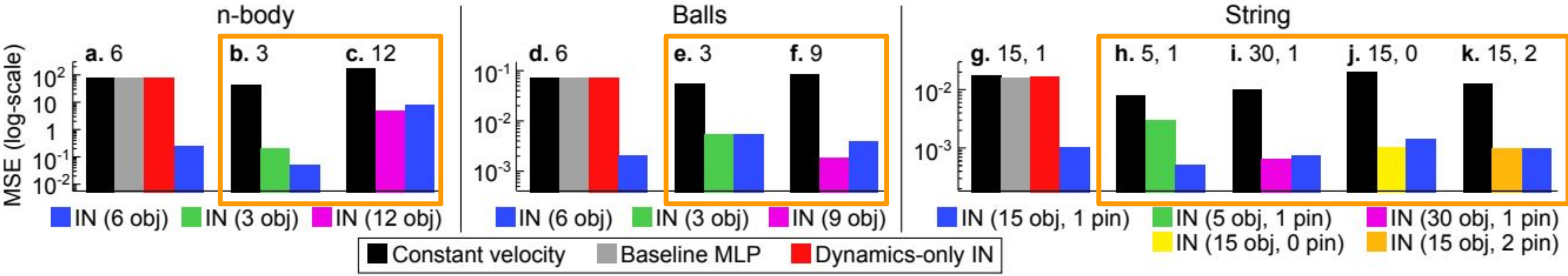
Domain	Constant velocity	Baseline	Dynamics-only IN	IN
n-body	82	79	76	**0.25**
Balls	0.074	0.072	0.074	**0.0020**
String	0.018	0.016	0.017	**0.0011**

Trajectory Simulation (1000-step rollouts)



Generalizability

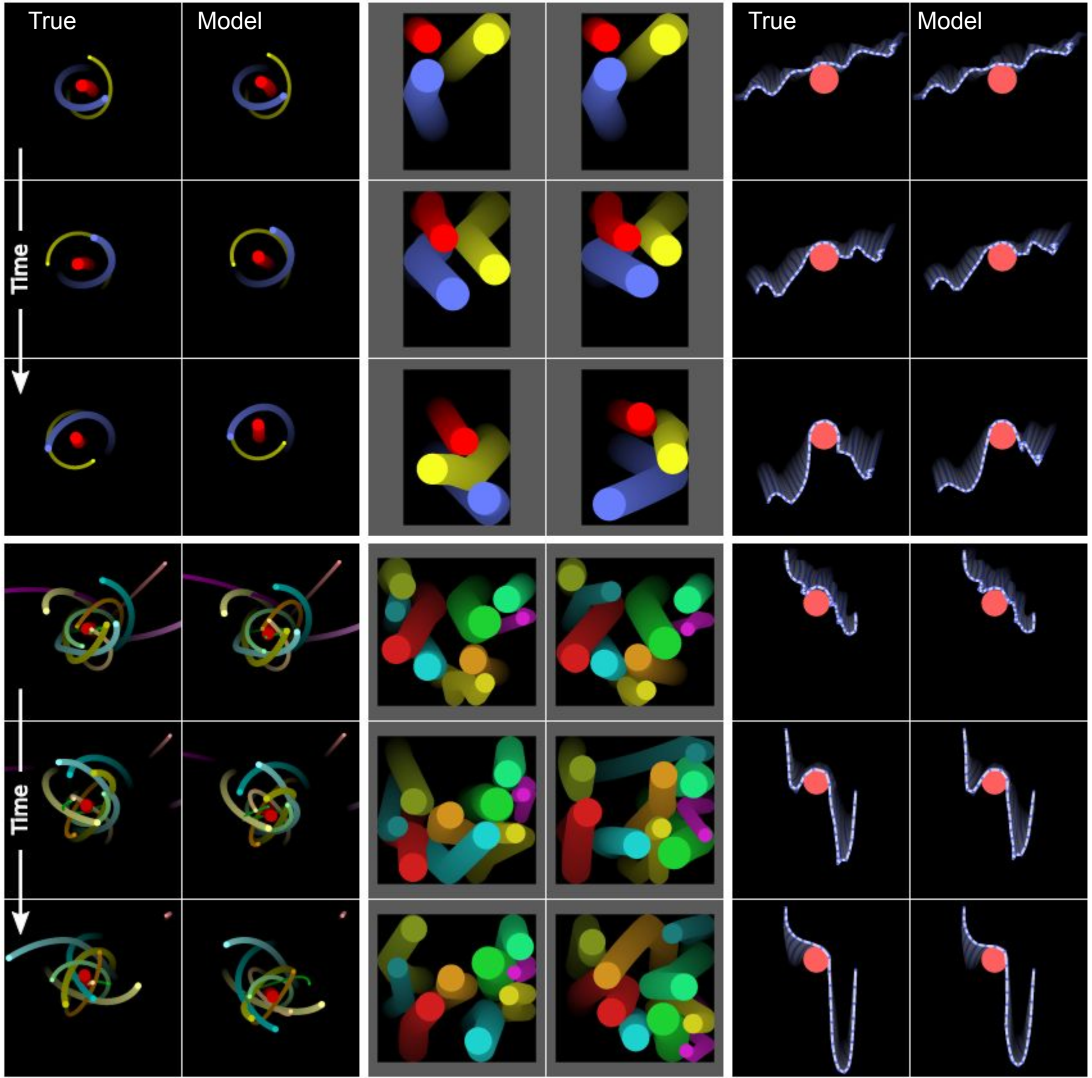
- IN **generalized well** to systems with fewer and greater number of objects.
- MSE evaluated on different systems:



- When tested on smaller n-body and spring systems, model trained on systems with more objects perform better than model trained on the smaller system.

systems with **fewer** objects

systems with **more** objects



IN generalizes well

Estimating abstract property

IN is also much more accurate in potential energy prediction than baseline MLP:

- It presumably learns the **gravitational and spring potential** energy functions
- Applies them to the relations in their respective domains
- And combines the results

Potential energy of bouncing balls is always 0.

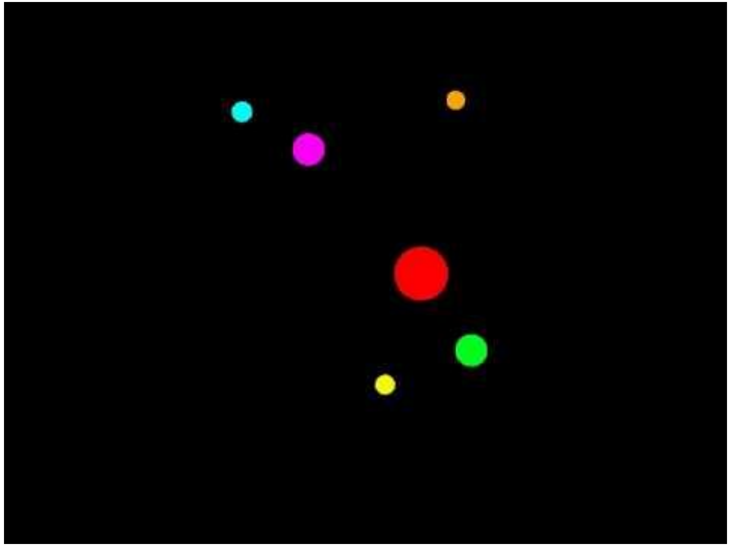
	N-body MSE	String MSE
IN	1.4	1.1
MLP	19	425

Contributions

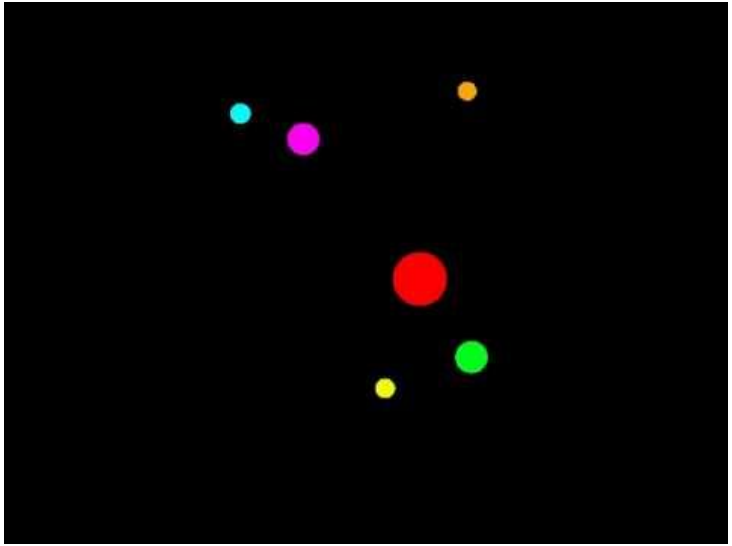
- IN shows strong ability to learn accurate physical simulations and can automatically generalize their training to novel contexts.
- It can roll out thousands of realistic future state predictions, even when trained only on single-step predictions.
- IN present the first general-purpose learnable physics engine that can scale up to real-world problems.
- IN provides a powerful general framework for reasoning about object and relations in complex real-world domains.

Demo

- 6-body



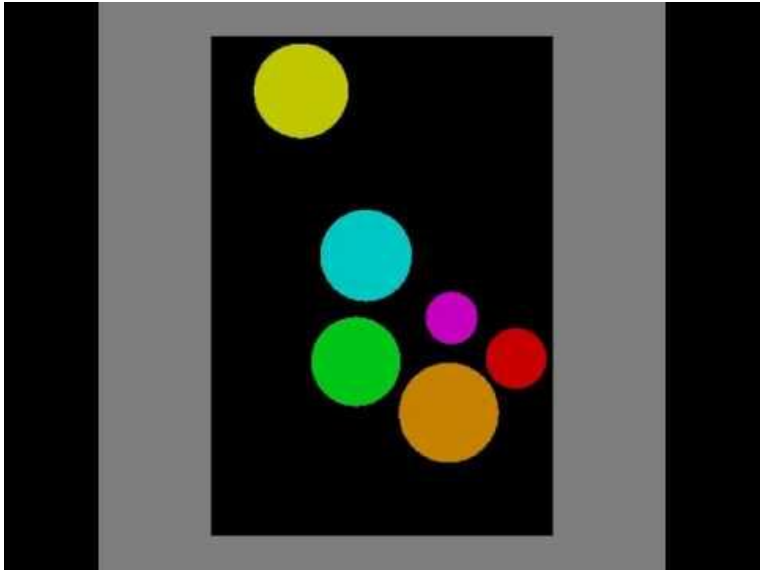
Ground Truth



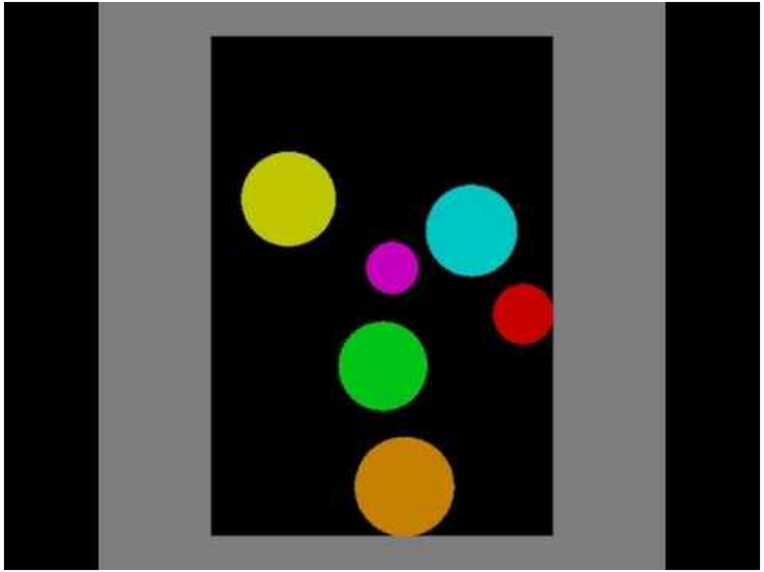
Model Prediction

Demo

- 6 Balls



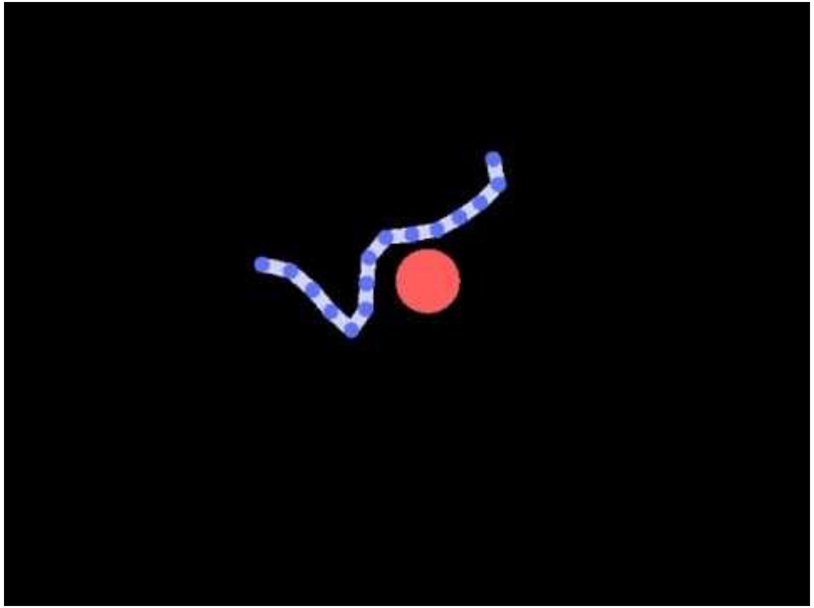
Ground Truth



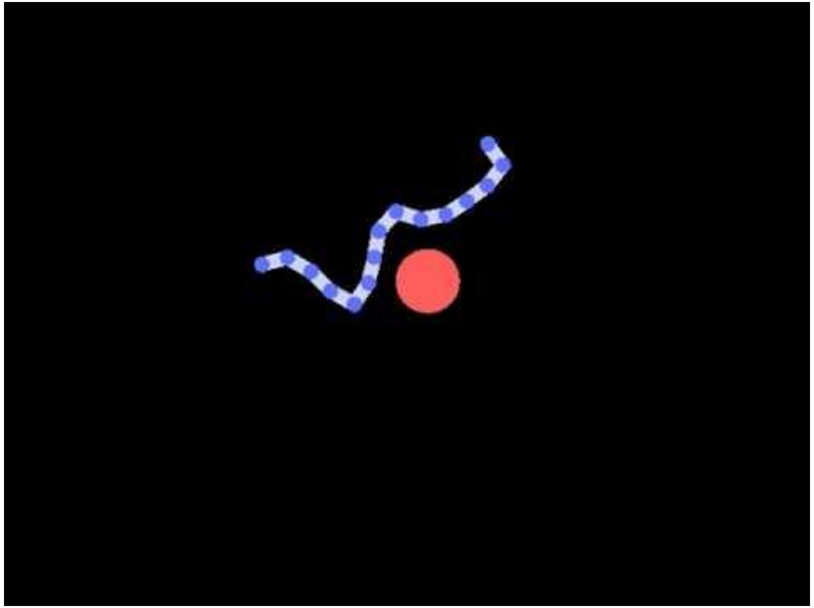
Model Prediction

Demo

- Spring 15 point masses, 1 pinned



Ground Truth



Model Prediction

Pros and Cons

- Pros
  - first general-purpose physical engine
  - generalize their training to novel systems with different numbers and configurations of objects and relations
  - could also learn to infer abstract properties of physical systems, such as potential energy
- Cons
  - if to handle very large systems with many interactions, then we need to reduce computation through methods like culling interaction computations with negligible effects
  - Only support binary relation
    - How to extend it to n-th order relations by combining n senders in each bk.
    - The interactions could even have variable order, where each bk includes all sender objects that interact with a receiver, but would require a f_R than can handle variable-length inputs.
  - Take the graph as input
    - Objects and relations are known
    - prepend a perceptual front-end that can infer the graph from raw observations

Questions?