Interaction Networks

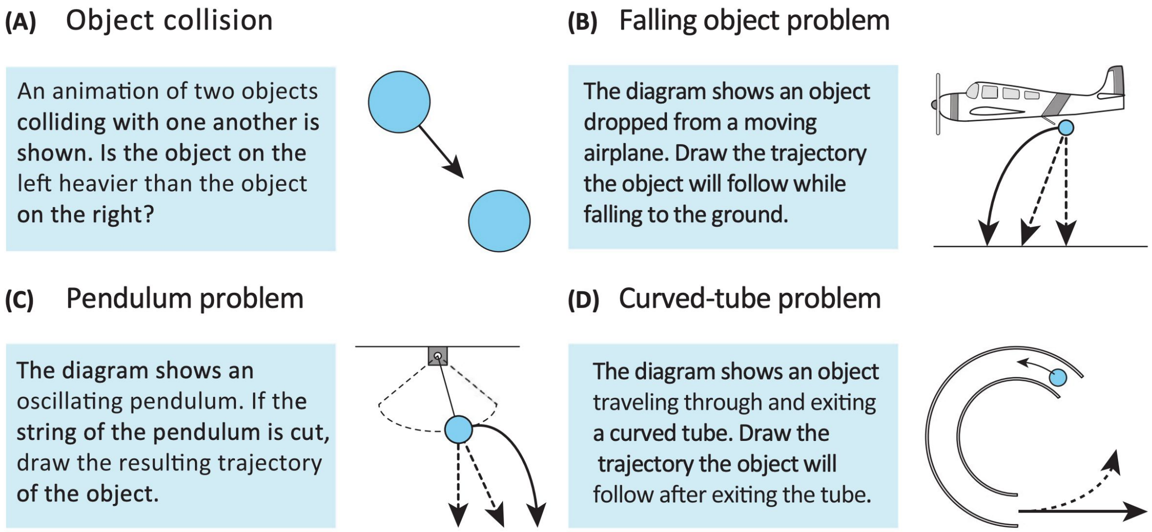
for Learning about Objects, Relations and Physics

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

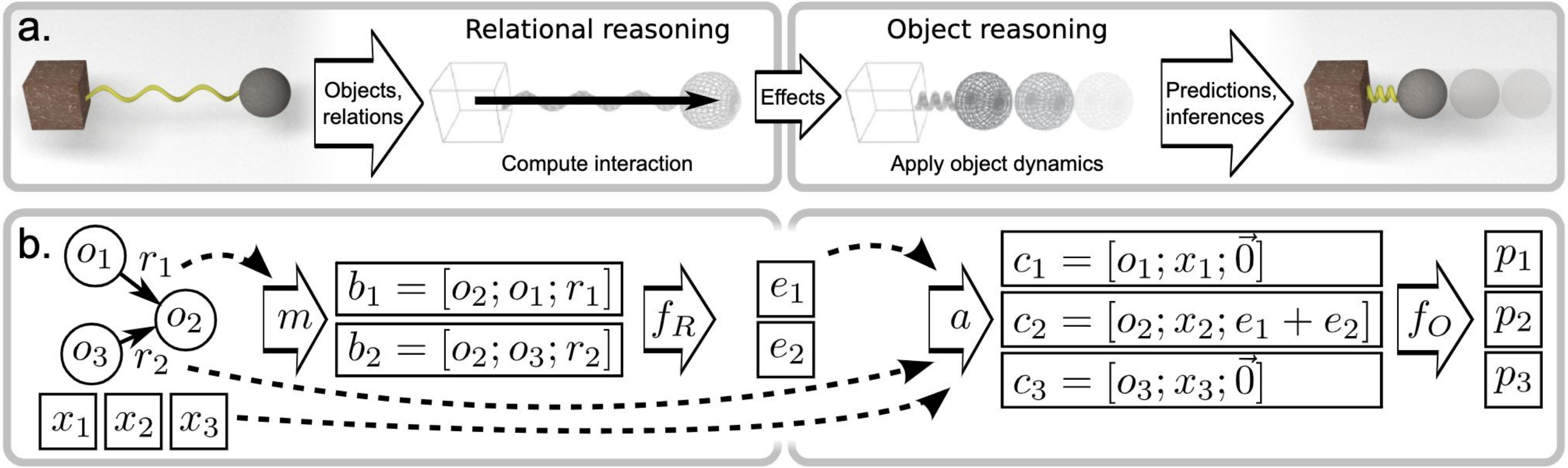
Introduction

- 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

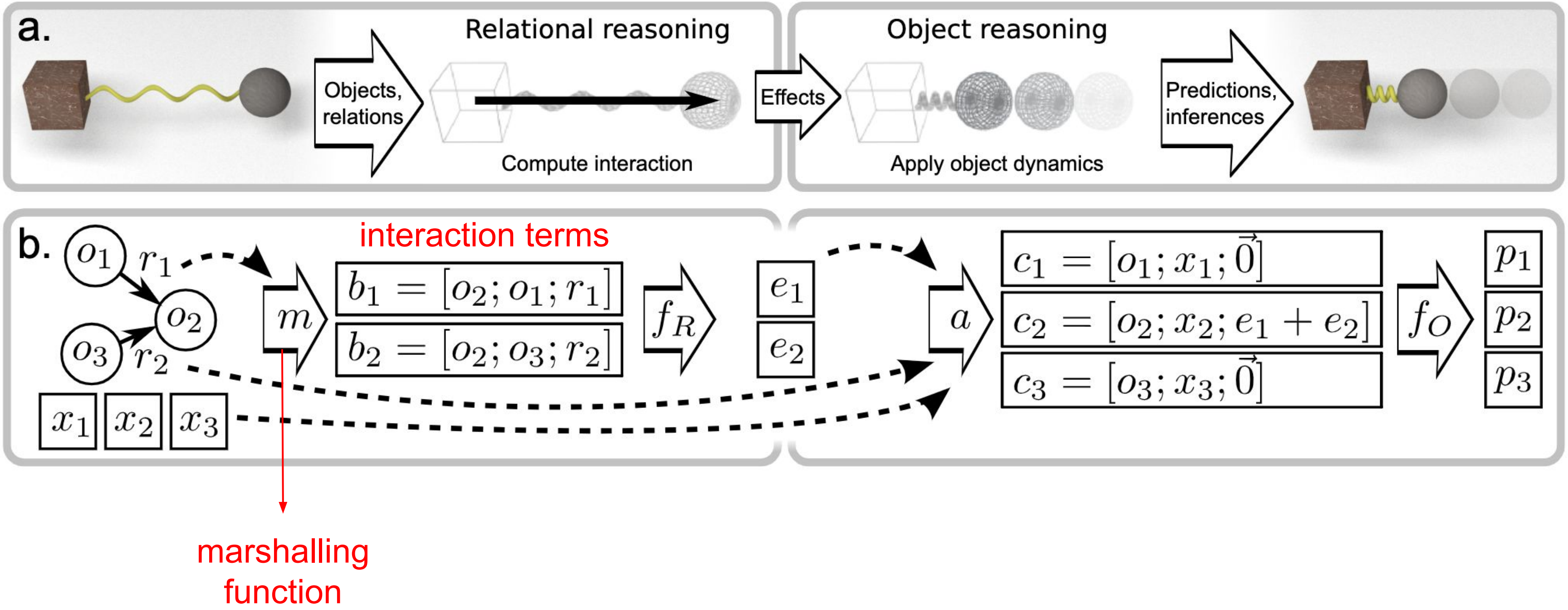
Related Work

- 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

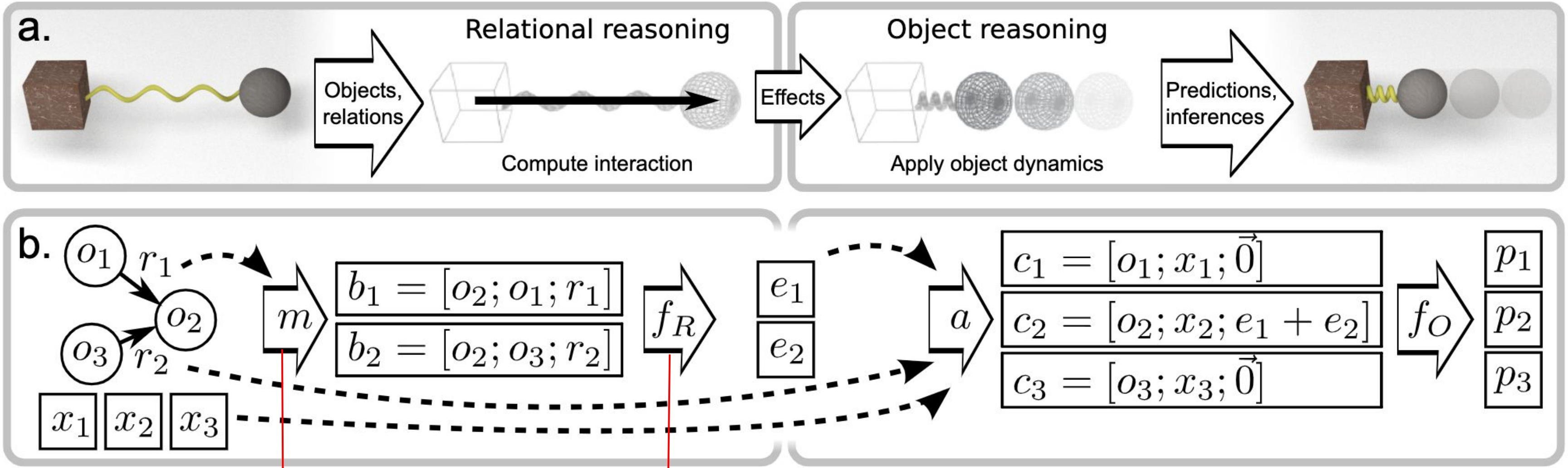
Schematic of Interaction Networks (IN)



Schematic of Interaction Networks (IN)



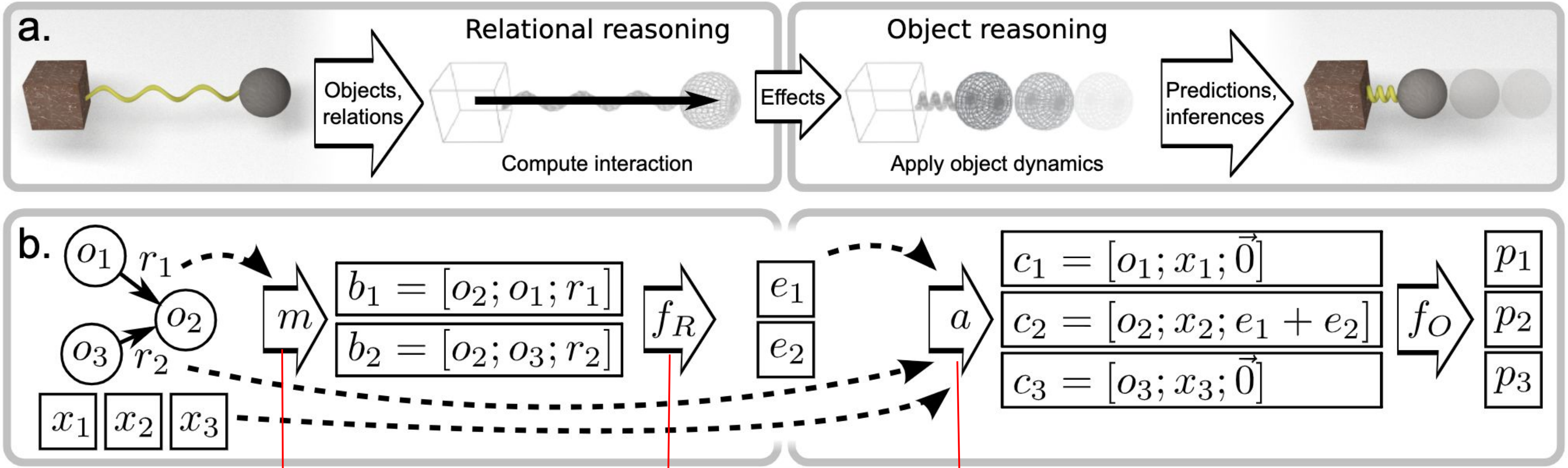
Schematic of Interaction Networks (IN)



marshalling function

Relation-centric function

Schematic of Interaction Networks (IN)

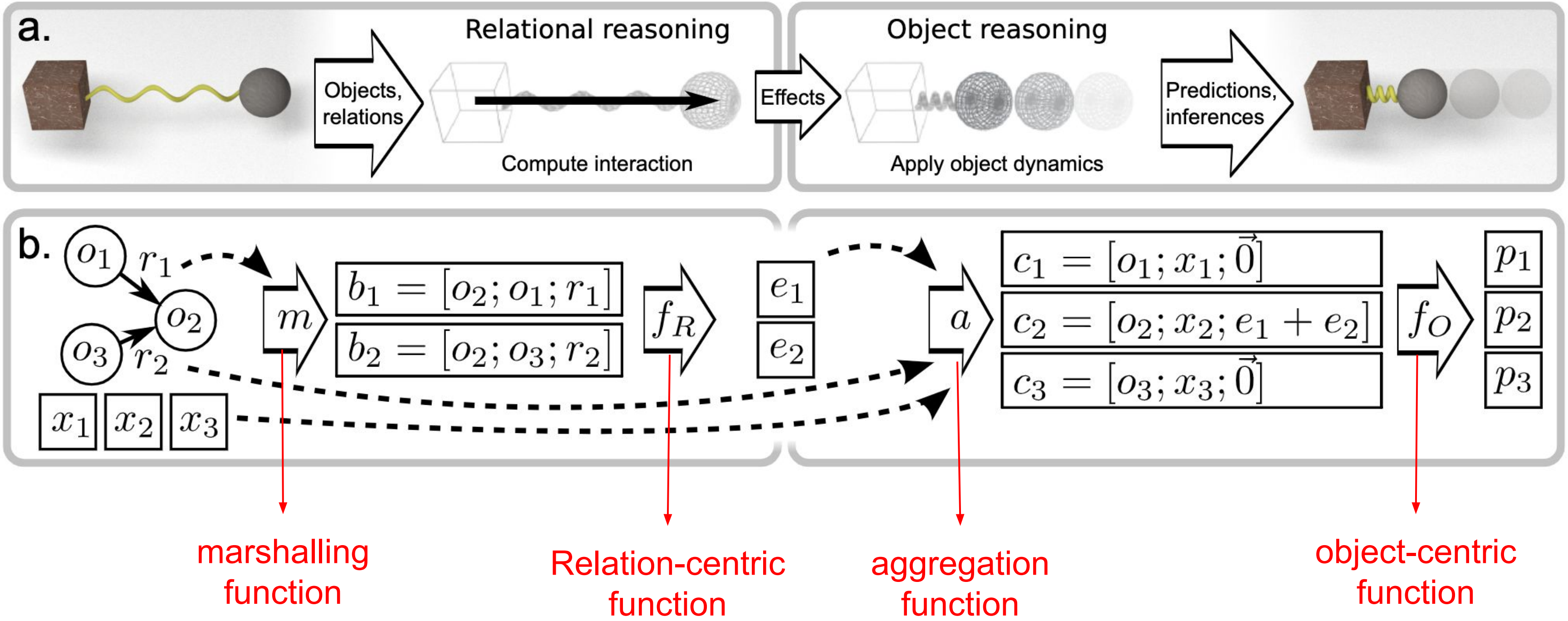


marshalling function

Relation-centric function

aggregation function

Schematic of Interaction Networks (IN)



A Formal Definition of IN

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

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

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

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

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

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

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)

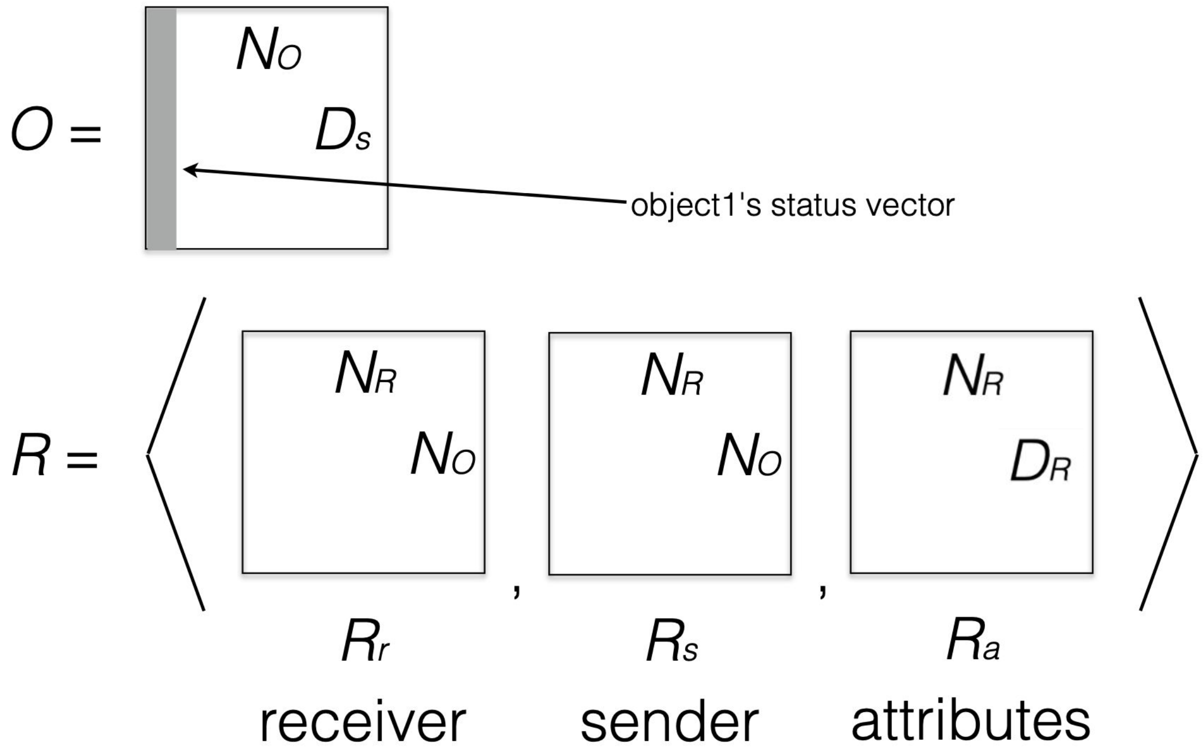
Relation

e: multiple for one object

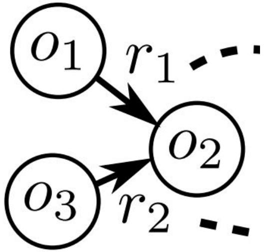
c: aggregated by *a*

A Learnable Implementation of IN

$G = \langle O, R \rangle$



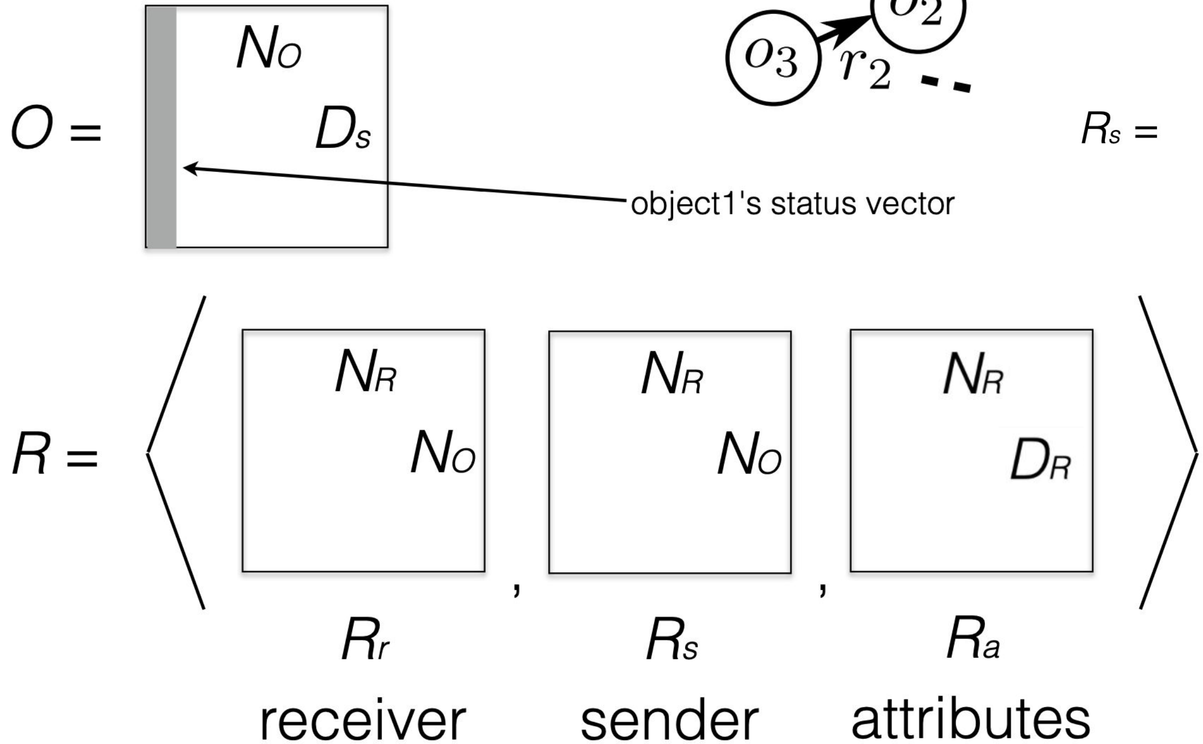
A Learnable Implementation of IN



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

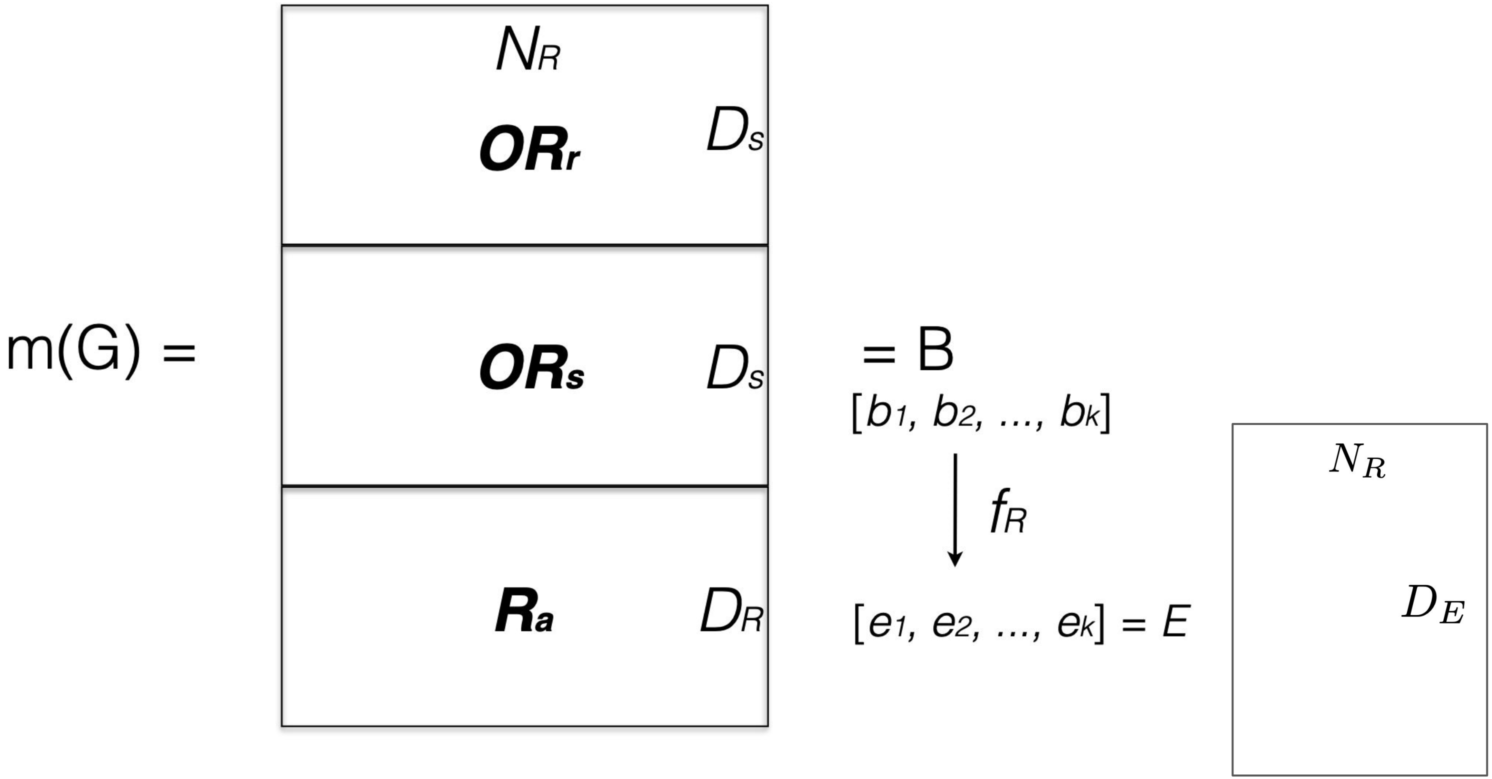
$$R_s = \begin{matrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{matrix}$$

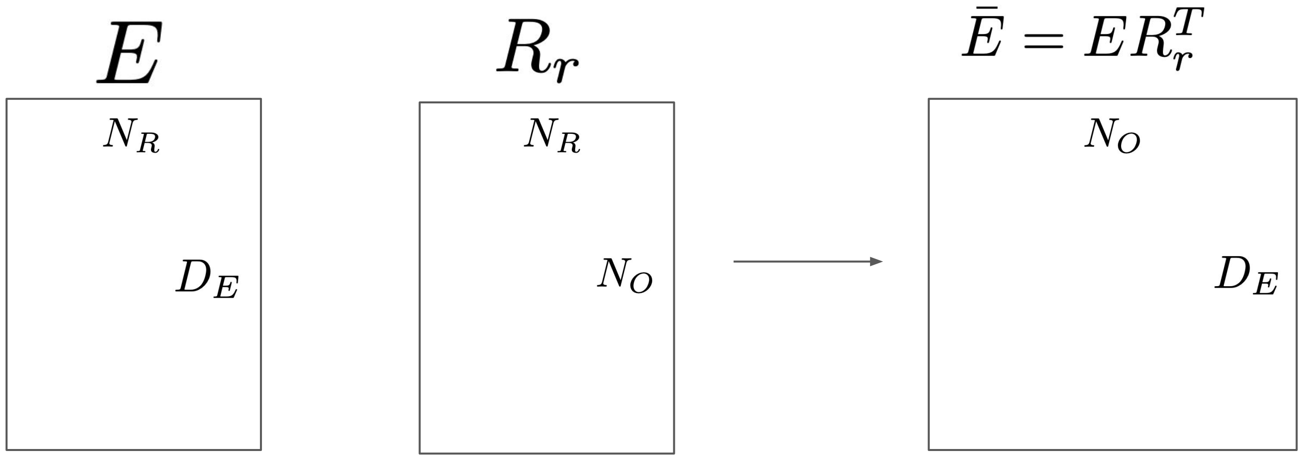
$$G = \langle O, R \rangle$$

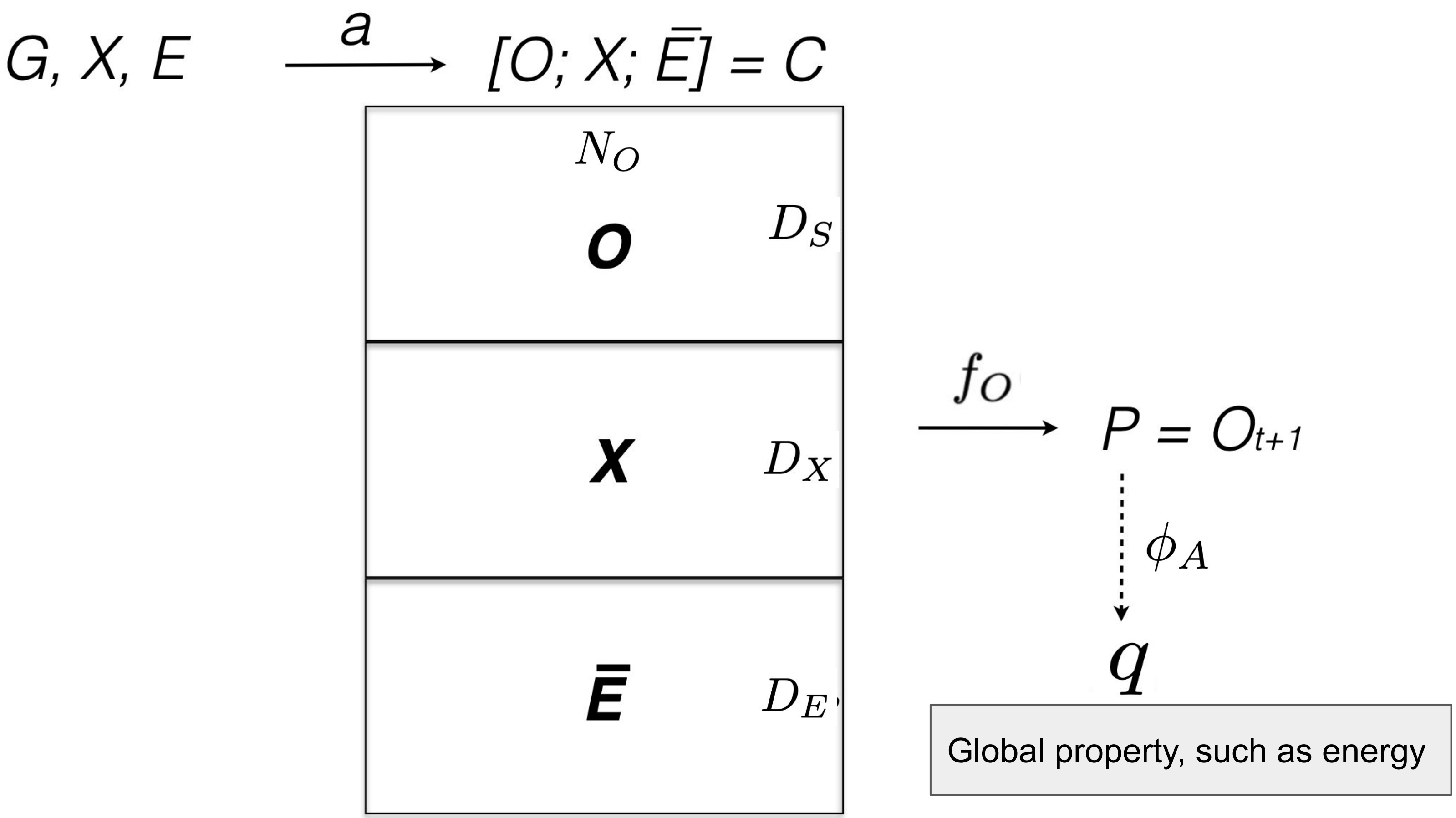


$$m(G) = \begin{array}{|c|}\hline \quad N_R \quad \\ \boldsymbol{OR_r} \quad D_s \\ \hline \boldsymbol{OR_s} \quad D_s \\ \hline \boldsymbol{R_a} \quad D_R \\ \hline \end{array} = B$$

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


m(G) =
N_R
OR_r
D_s
OR_s
D_s
R_a
D_R
= B
$[b_1, b_2, ..., b_k]$
f_R
$[e_1, e_2, ..., e_k] = E$
N_R
D_E


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


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

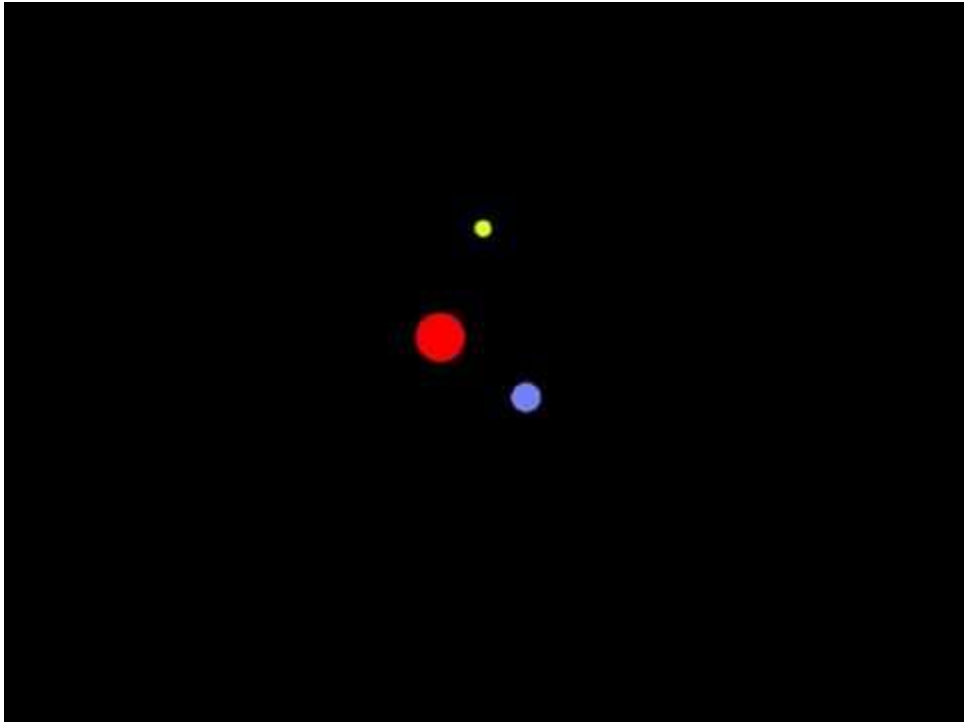
Experiments - Physical Reasoning Tasks

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

- 3 Physical domains:
  - N-body systems
  - Balls bouncing in a box
  - Strings composed of springs that collide with rigid objects

N-body Domain

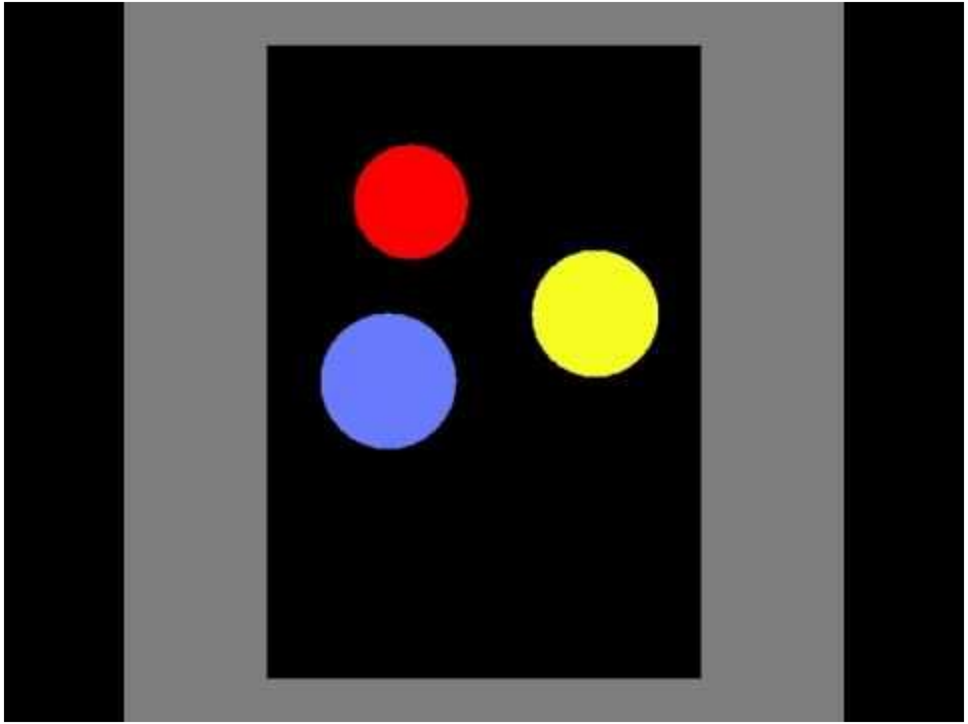
- N bodies with gravitational forces in between (distance and mass dependent)
- n(n-1) relations
- Varied object attributes:
  - Mass
- Each body initialized with velocities
  - Half random velocities
  - Half velocities that cause stable orbits



Balls bouncing in a box

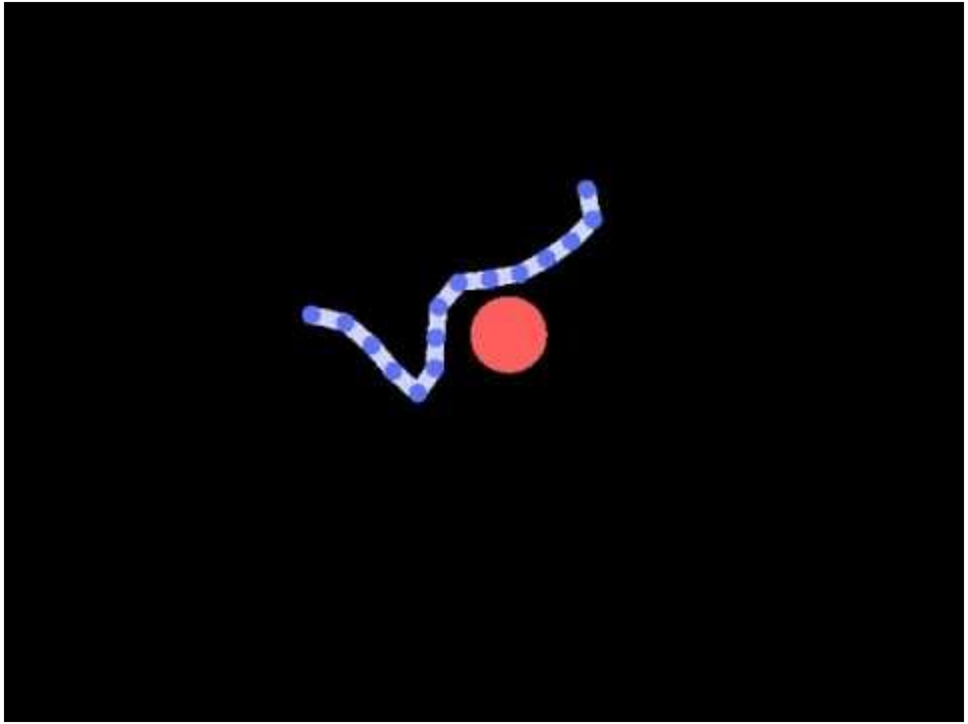
- Moving balls collide with each and static walls.
- n(n-1) relations
- Collision or straight-line motion
  - For each ball, only 1% steps with collision.
- Varied Attributes:
  - Object: Shape, scale and mass
  - Relation: Coefficient of restitution

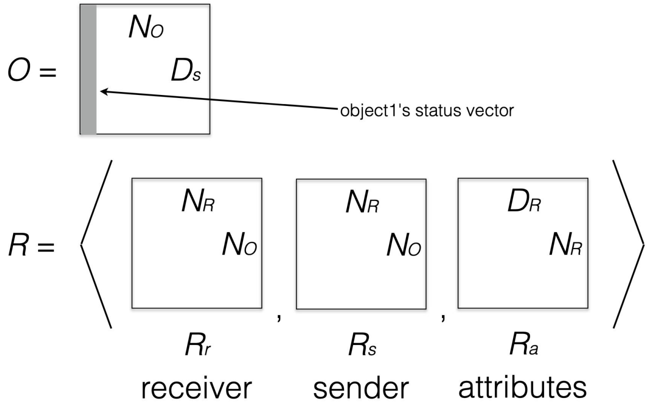
$$e = \frac{\text{velocity of separation (after collision)}}{\text{velocity of approach (before collision)}} = \frac{(v_2 - v_1)}{(u_1 - u_2)} \qquad (4.68)$$



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
- Relations:
  - 2(n-1) spring relations with their immediate neighbors. (spring constant)
  - 2n relations with the rigid object
- External input:
  - Gravitational acceleration (varied across simulations)





Prediction Tasks

- 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 represented quantities such as the coefficient of restitution, and 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
  - The output velocity on time step t became the input velocity on t+1
  - the position at t+1 was updated by the predicted velocity at 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
- Performance reported in the results was measured on held-out test data.
- Adding noise:
    - Adding noise to 20% of the data's input positions and velocities during the initial phase of training.The noise was reduced to 0% from epochs 50 to 250.
    - Learn to project physically impossible states back to nearby, possible states.
    - Prediction MSE is not influenced, generated rollout videos from models trained with noise are slightly more visually realistic.
    - Static objects were less subject to drift

Model Architecture

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

L2 Regularization

- Apply to effects E and model parameters
- Regularize E:
    - Improved generalization to different number of objects
    - Reduced drift over many rollout steps
- Regularize parameters:
    - 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

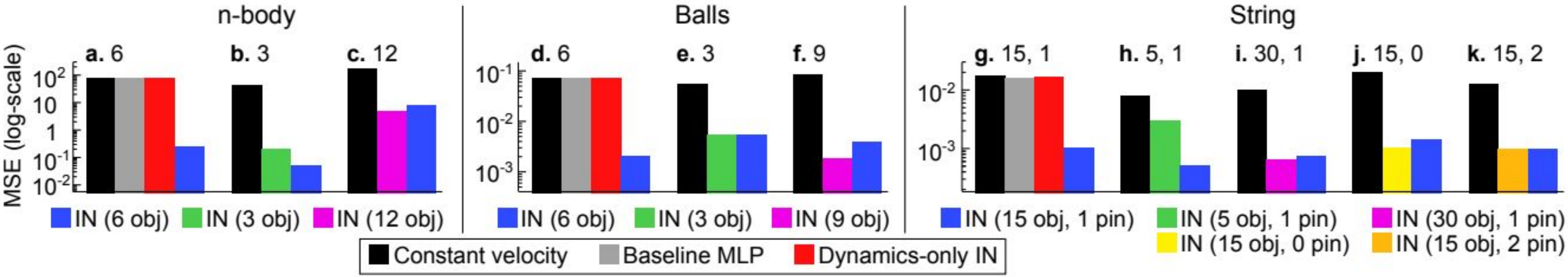
- Next-step velocity: high accuracy
  - Orders of lower test error than other baseline models
- Since IN can exploit dynamic interactions among objects for its predictions.
- Dynamics-only IN had no mechanism for processing interaction, and perform similarly to the constant velocity model.
- Baseline MLP in theory can learn object interactions but
  - would be required to learn how to select interactions based on relation indices.
  - it shares learning across relations and objects.

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**

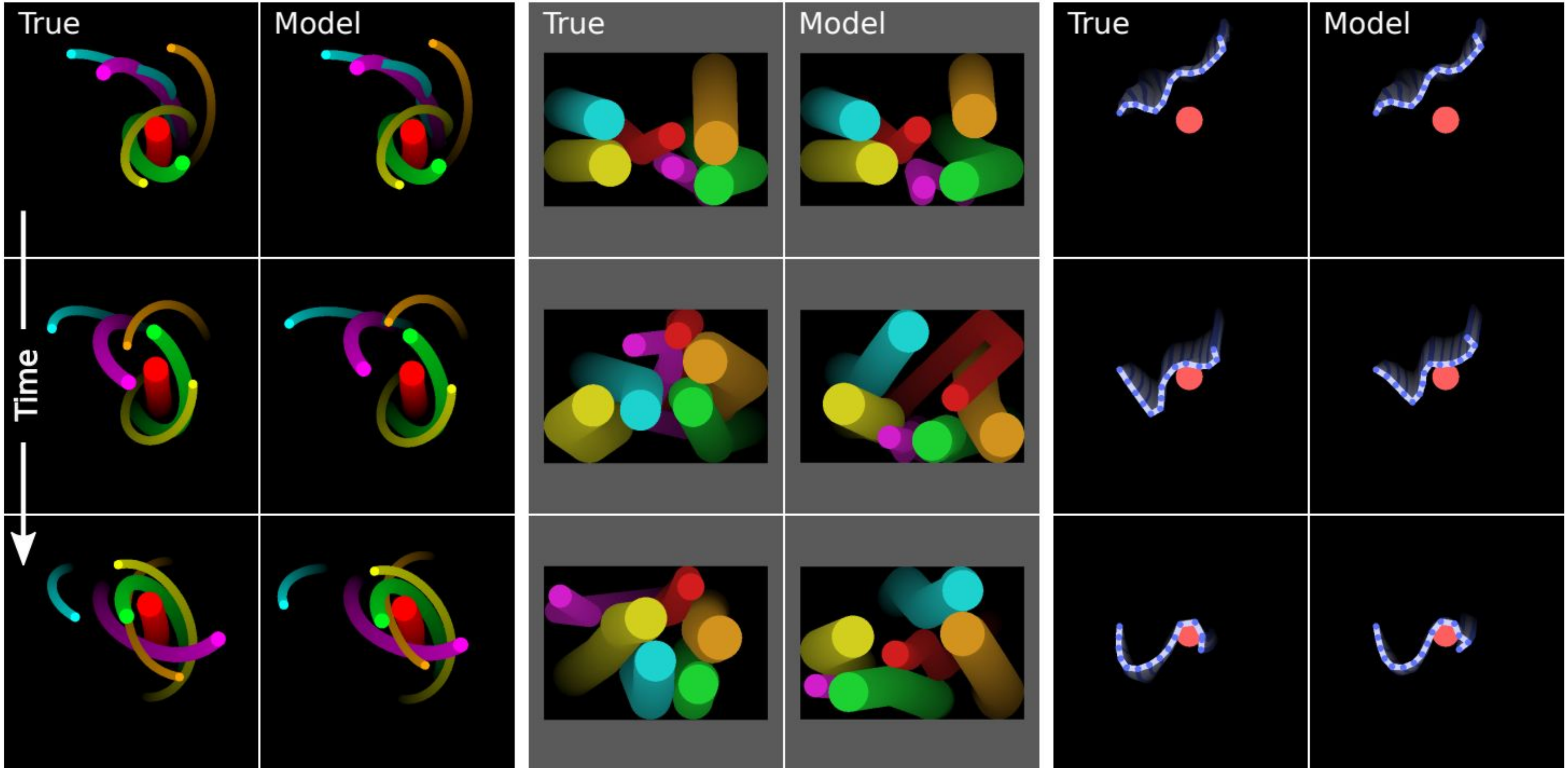
Generalizability

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



- Model trained on larger systems perform better than model trained on the smaller system.

Trajectory Simulation



Estimating abstract

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 are always 0.

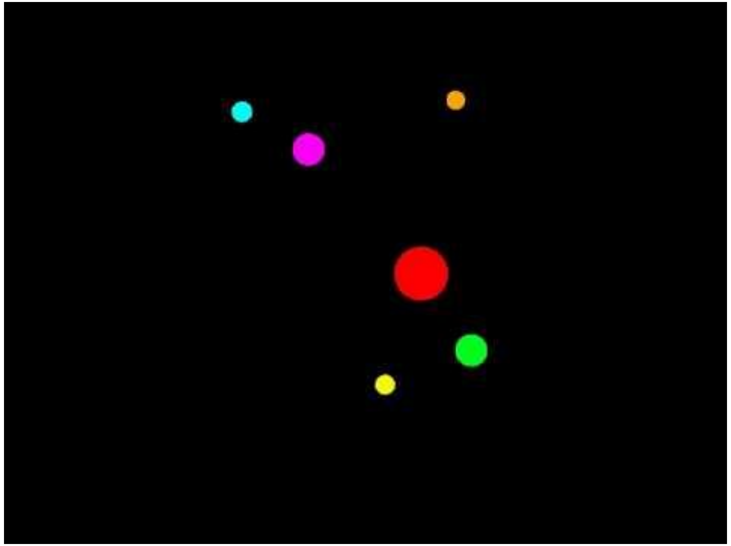
	N-body MSE	String MSE
IN	1.4	1.1
MLP	19	425

Contributions

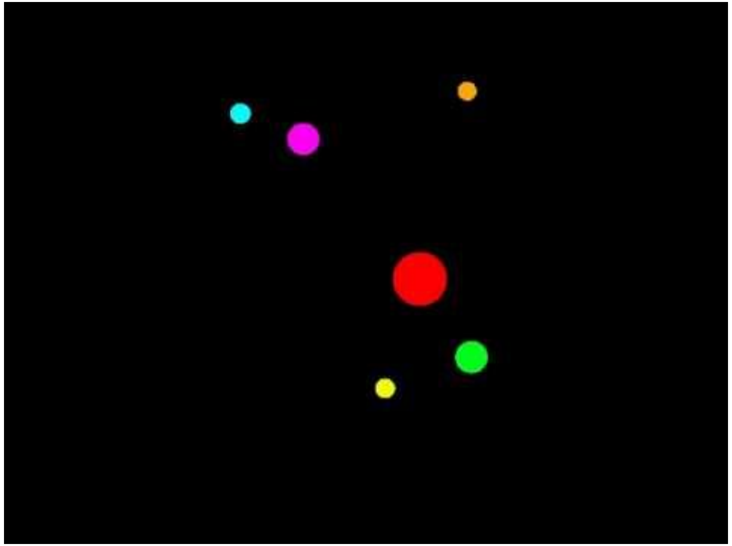
- IN shows strong ability to learn accurate physical simulations and can automatically generalize their training to novel contexts.
- It can roll out thousands of realistic future state predictions, even when trained only on single-step predictions.
- IN present the first general-purpose learnable physics 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



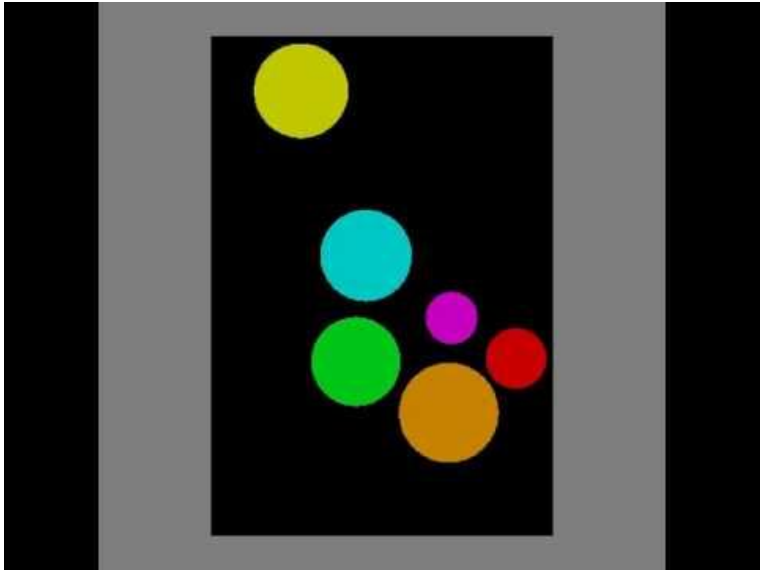
Ground Truth



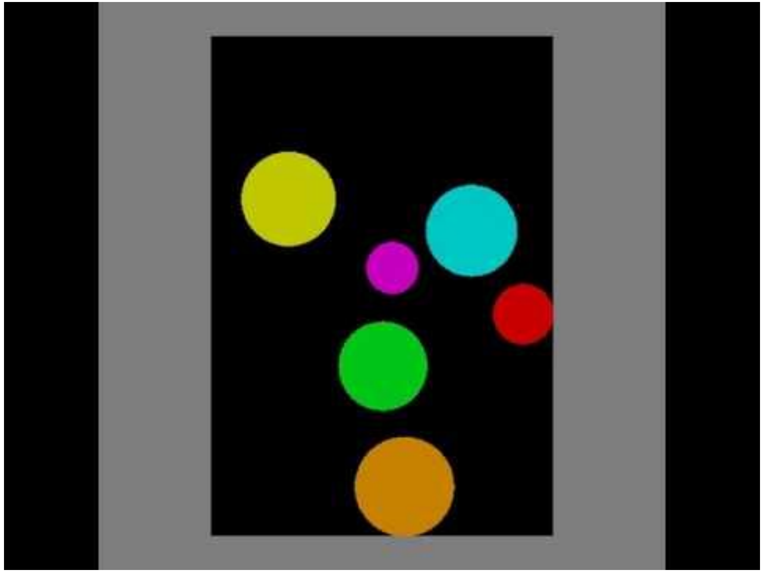
Model Prediction

Demo

- 6 Balls



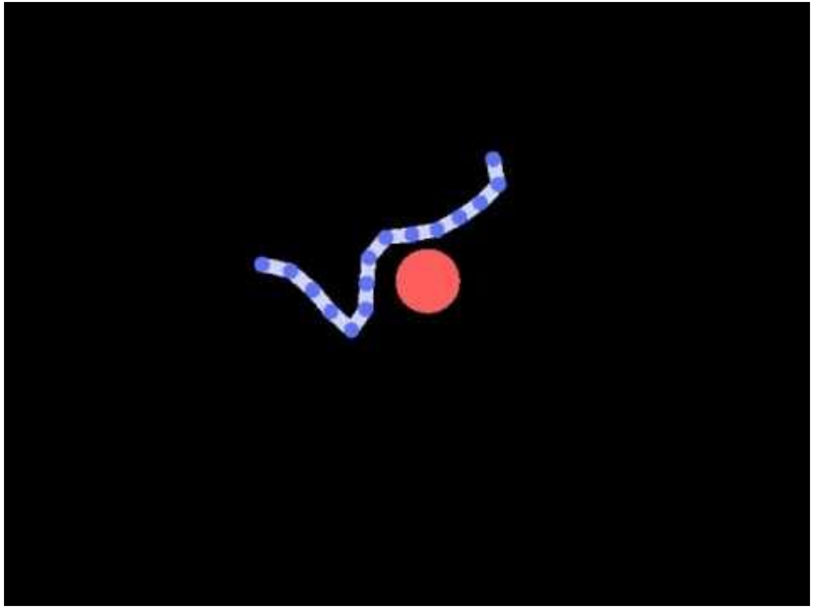
Ground Truth



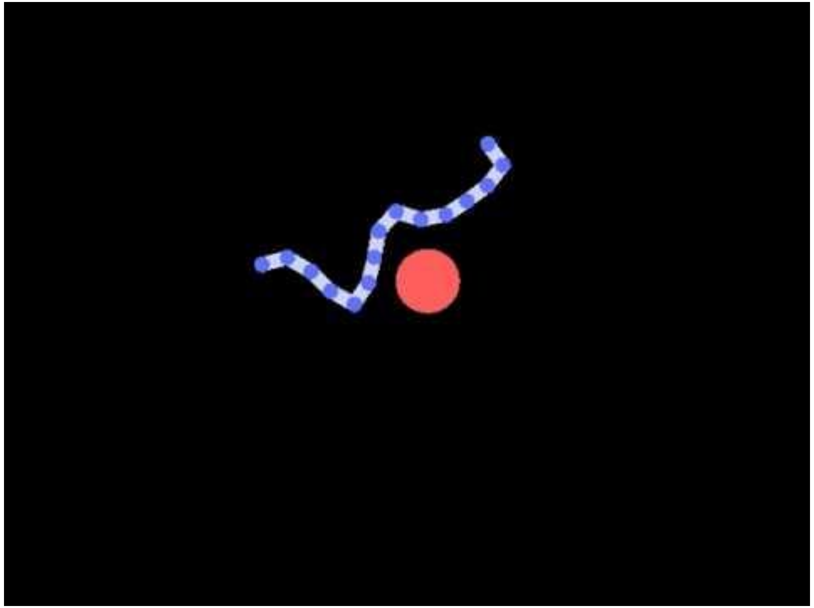
Model Prediction

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?