

# Generalization by Construction

Learning reusable models.

Reasoning with them to solve new problems.

Yilun Du

Harvard University



**HARVARD**  
UNIVERSITY



**Kempner**  
INSTITUTE

# Building General-Purpose AI Agents

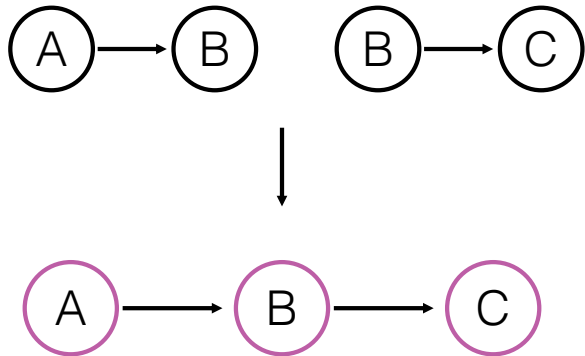


How do we build an agent  
that can **reliably generalize**  
to unseen tasks, environments,  
and other agents?

# What Changes at Test Time?

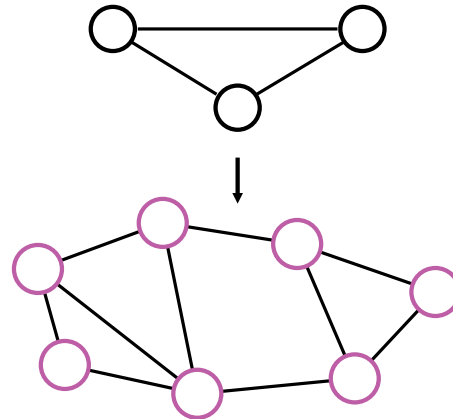
The full task can be unfamiliar, even when its relationships are known.

New combinations



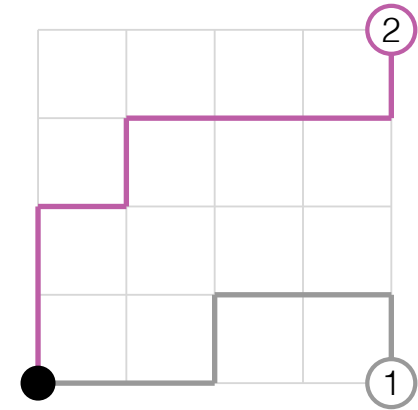
Known relations,  
a new joint specification

Larger problems



Known local rules,  
more coupled variables

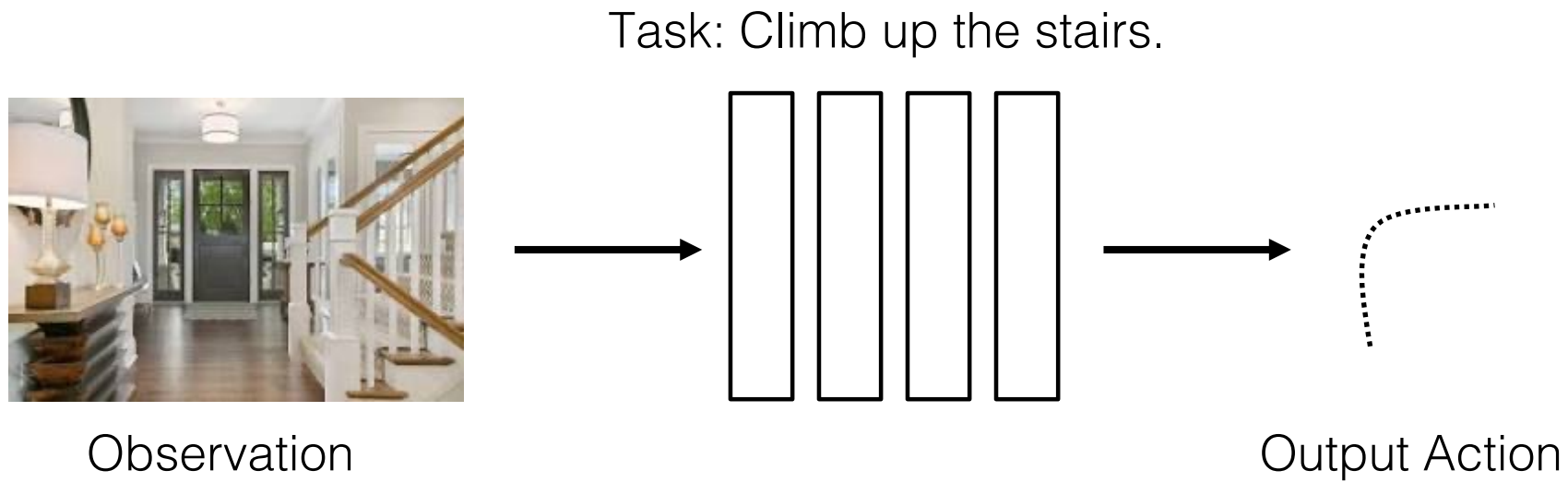
New goals or constraints



Known dynamics,  
a different objective

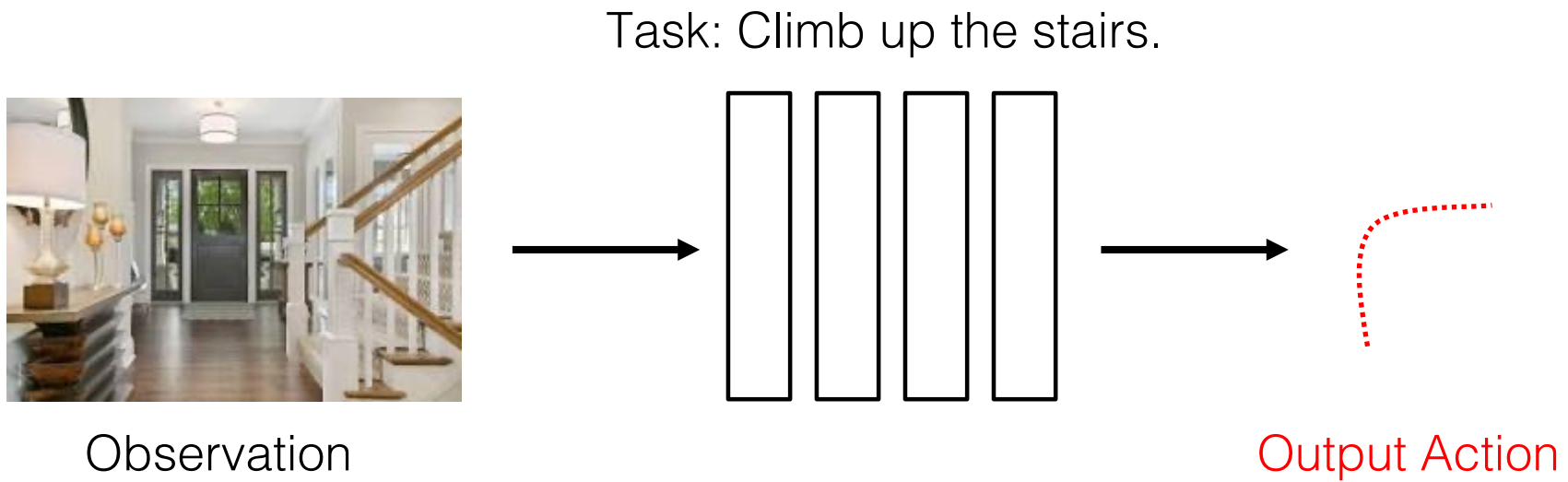
Generalization can come from recombining learned structure.

# Predicting an Action Directly



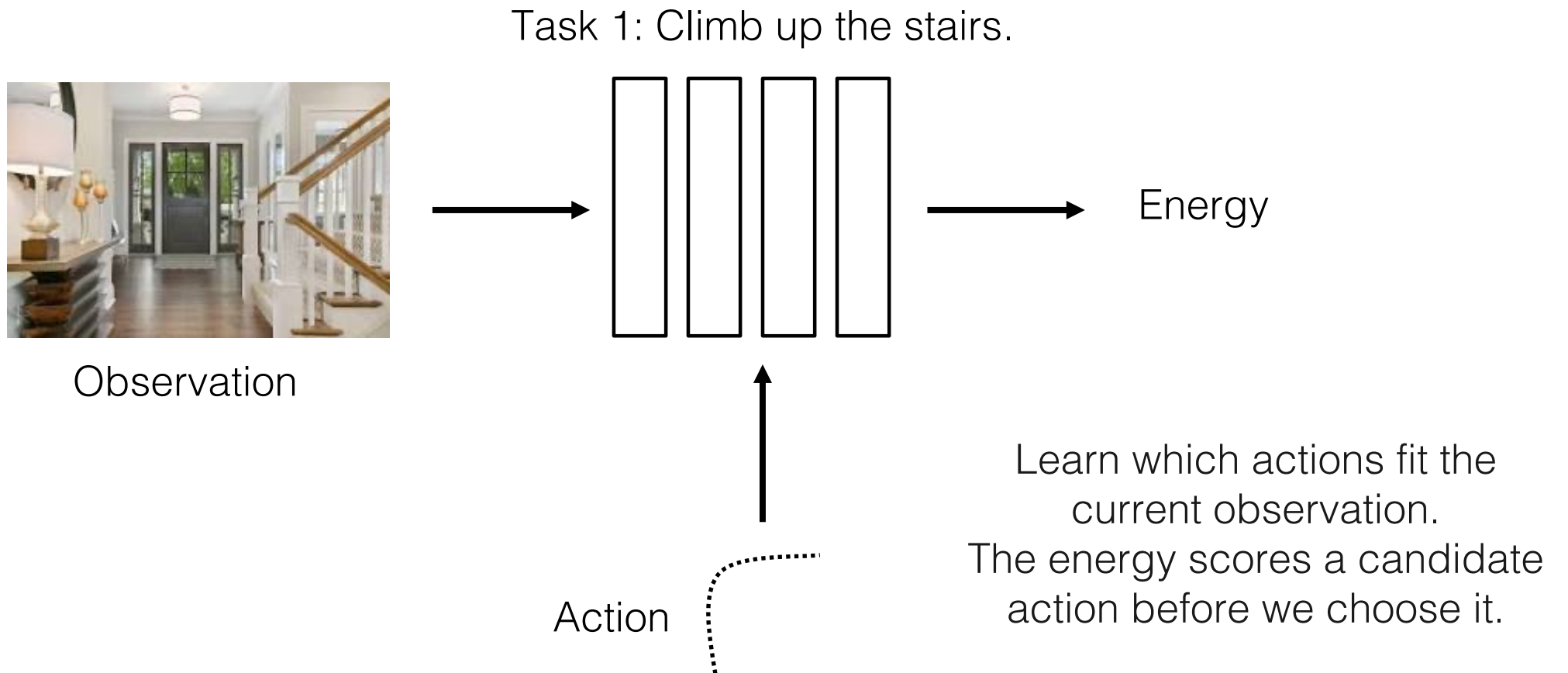
Learning can resolve the decision in advance.  
At execution time, the model directly proposes an action.

# When the Problem Changes

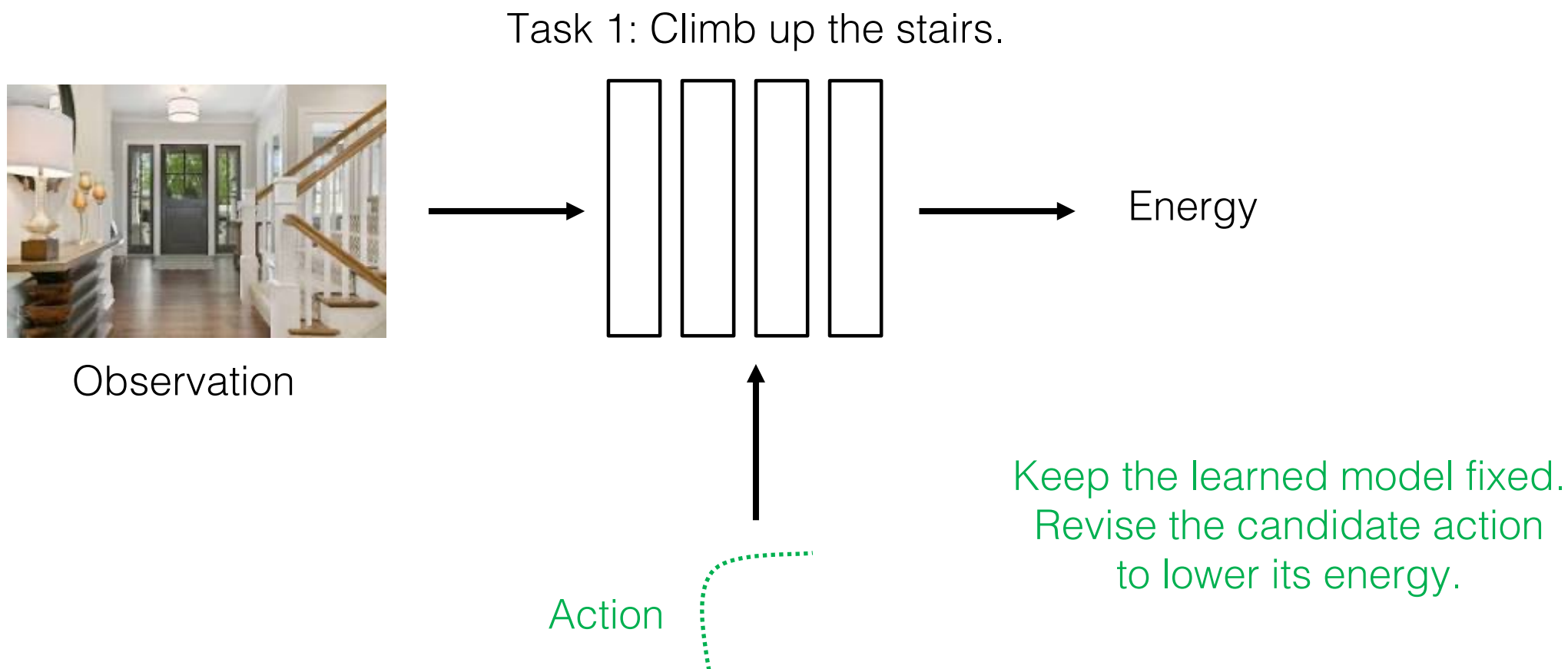


Changes in an environment can make a predicted action invalid

# Learning a Criterion over Actions



# Inferring the Action for the Current Problem

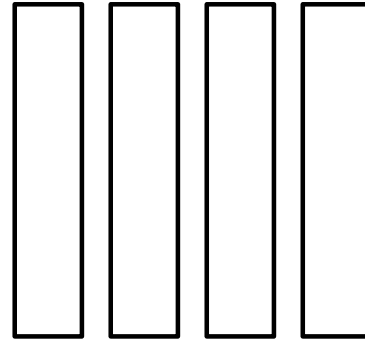


# Adding a New Constraint at Test Time



Observation

Task 1: Climb up the stairs.



Action

Prediction  
Time Costs

Energy 1

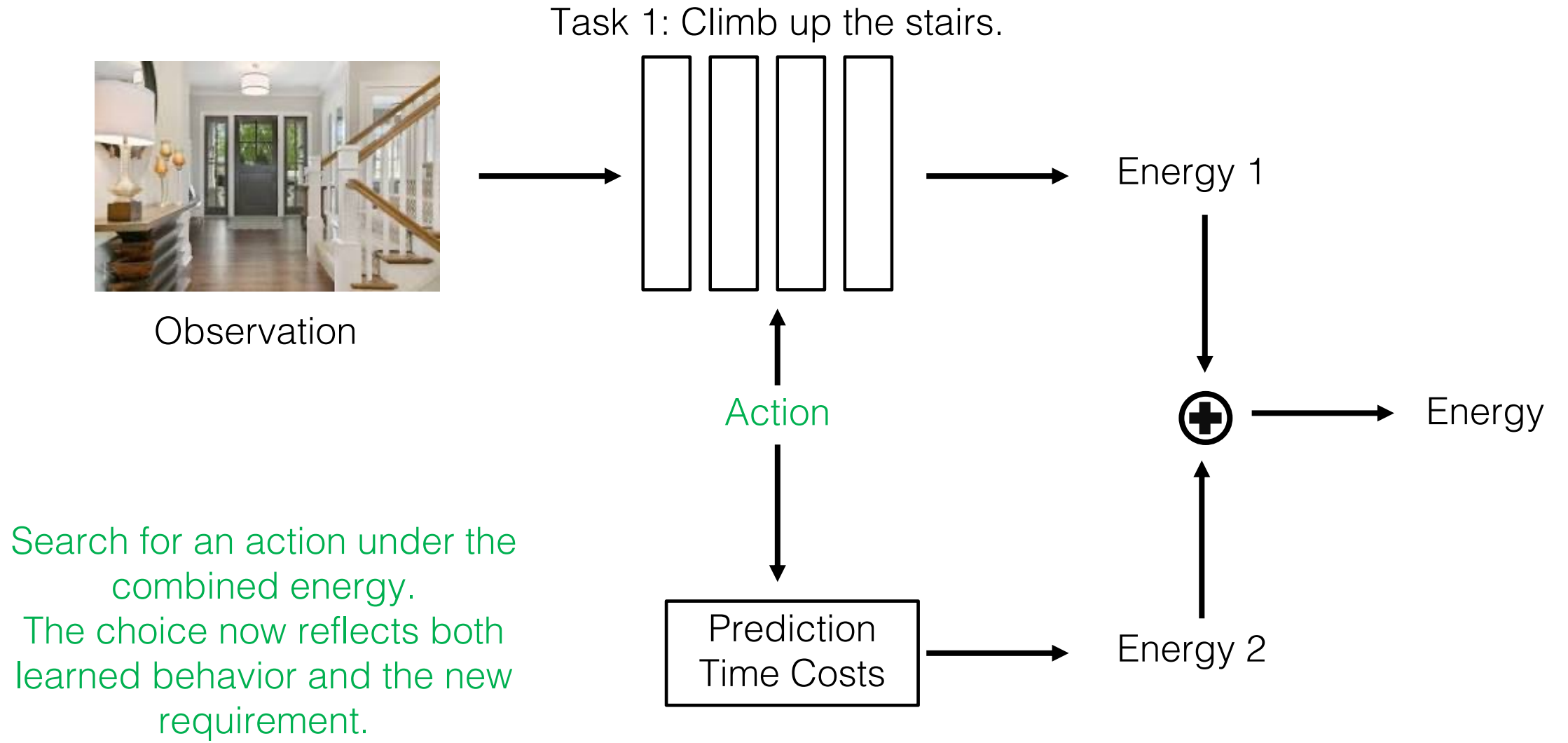


Energy

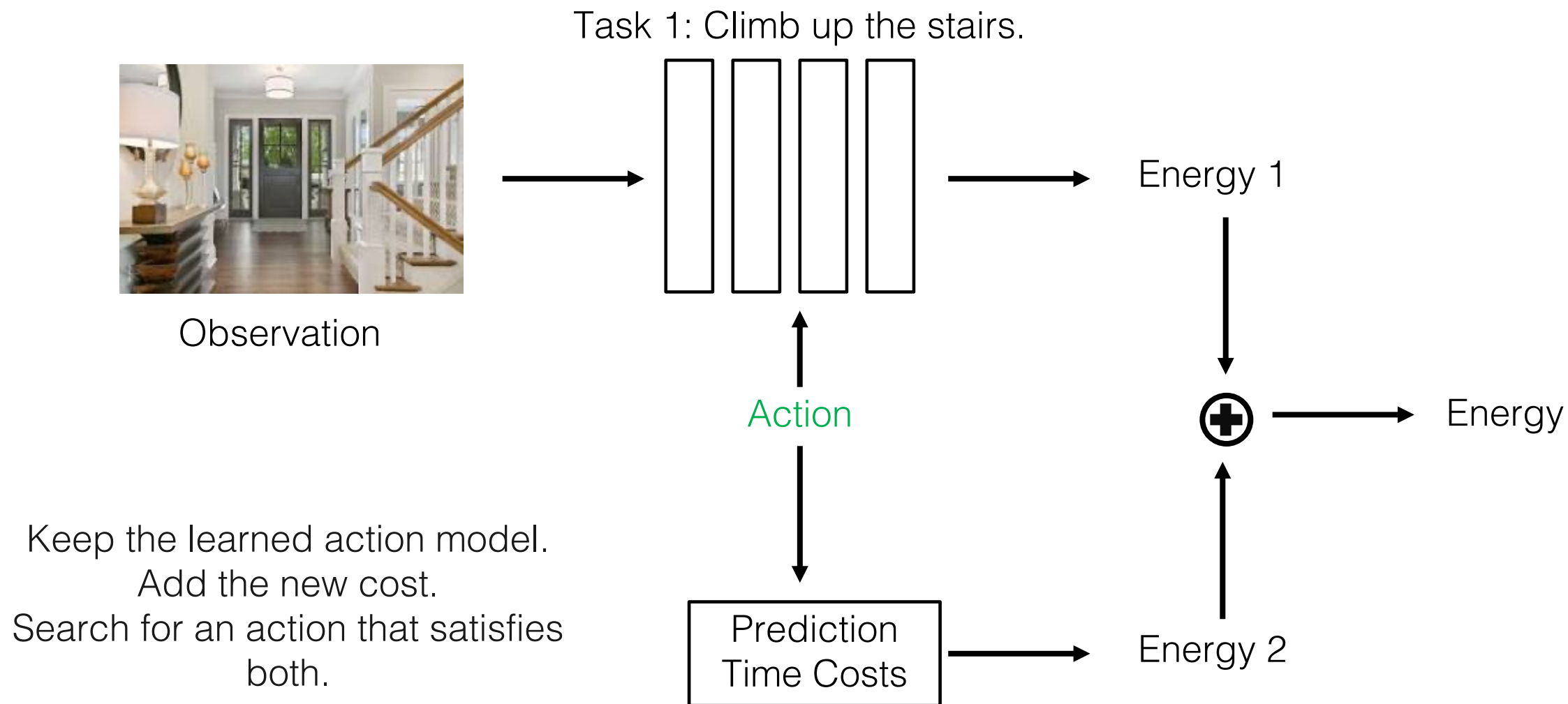
Energy 2

Keep the learned action energy  
fixed.  
Add a cost for the new  
requirement, such as battery  
use.

# Satisfying the Requirements Together

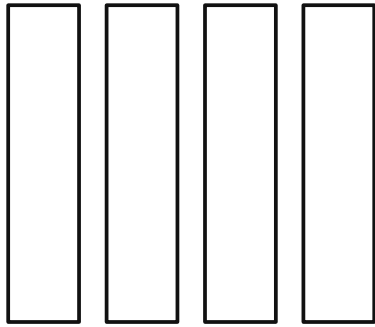


# New Constraints with the Same Learned Model



# What Should Be Learned, and What Should Be Inferred?

## Learning

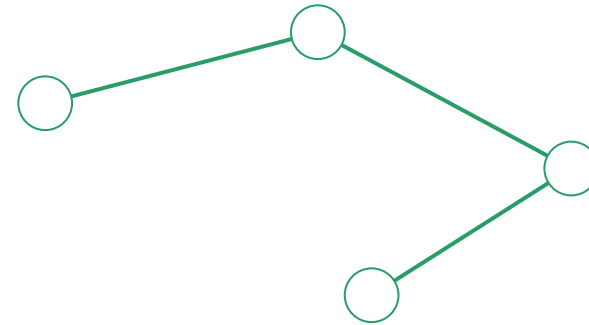


Learn relationships that recur across problems.

Concepts, constraints, dynamics, and capabilities.



## Inference

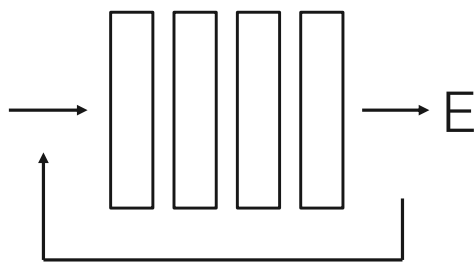


Construct a solution for the current problem.

Use the observed state, goal, and constraints.

# From Learned Models to New Behavior

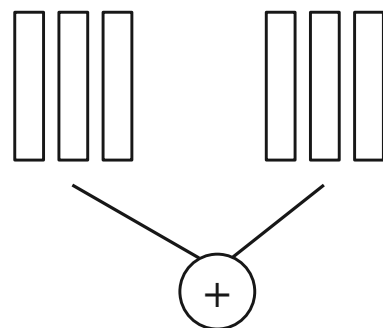
Generative  
inference



Learn a solution  
space.

Sample and refine  
a candidate.

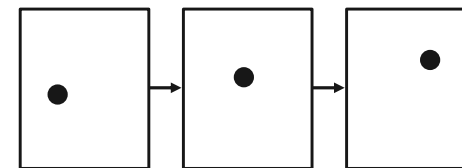
Compositional  
inference



Learn reusable  
relations.

Compose and satisfy  
them together.

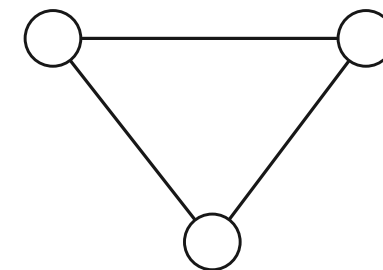
World models  
and planning



Learn possible  
futures.

Search for behavior  
under a new goal.

Multi-agent  
intelligence



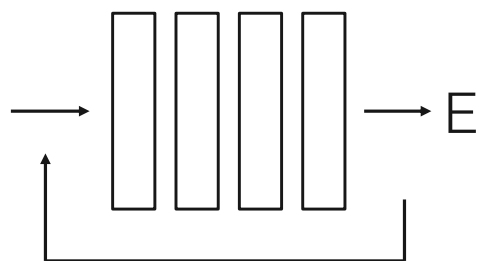
Reuse learned  
capabilities.

Communicate,  
evaluate,  
and choose who acts.

What is learned? What remains open to inference?

# From Learned Models to New Behavior

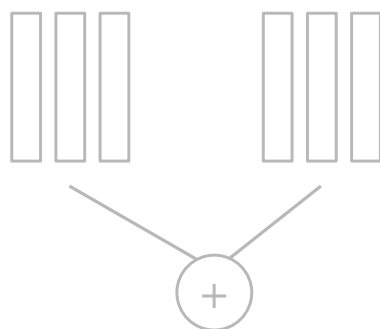
Generative  
inference



Learn a solution  
space.

Sample and refine  
a candidate.

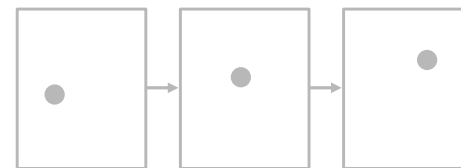
Compositional  
inference



Learn reusable  
relations.

Compose and satisfy  
them together.

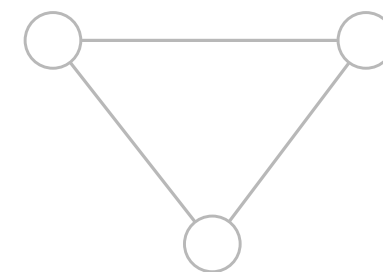
World models  
and planning



Learn possible  
futures.

Search for behavior  
under a new goal.

Multi-agent  
intelligence

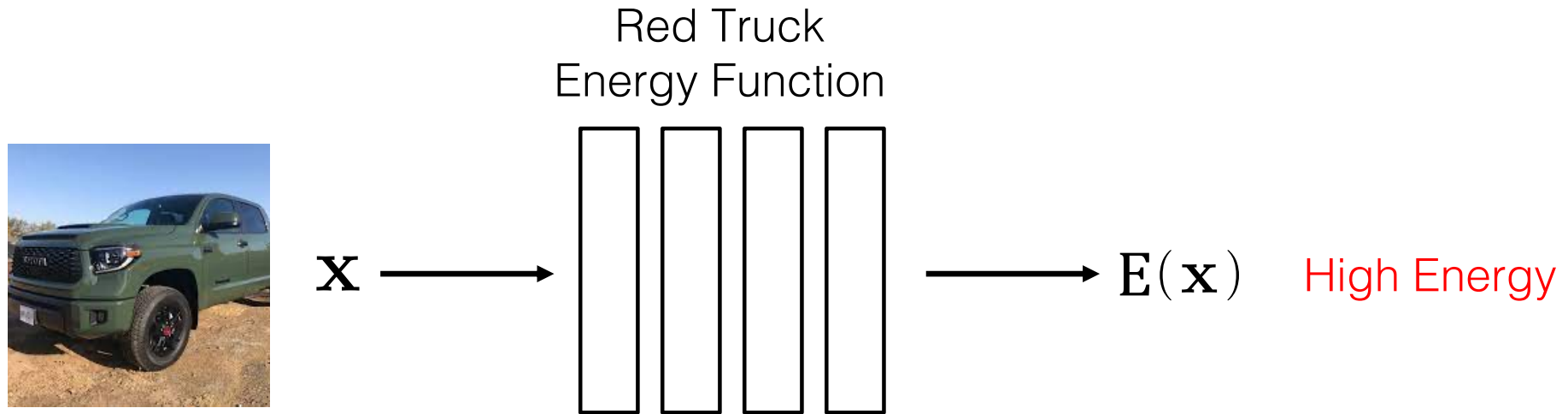


Reuse learned  
capabilities.

Communicate,  
evaluate,  
and choose who acts.

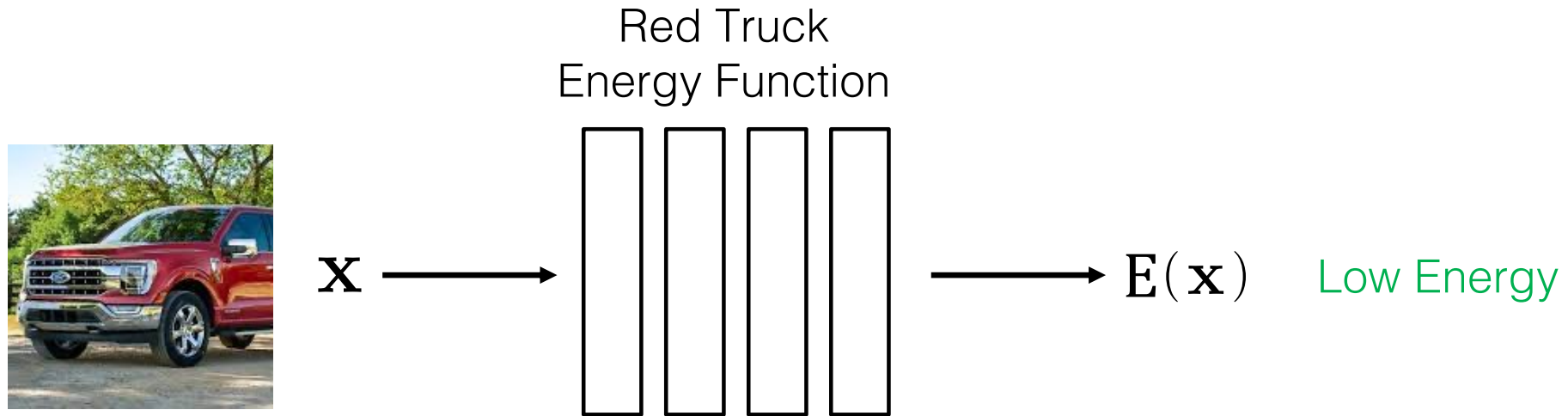
Can one learned model support different ways of finding a solution?

# Representing Compatibility with an Energy



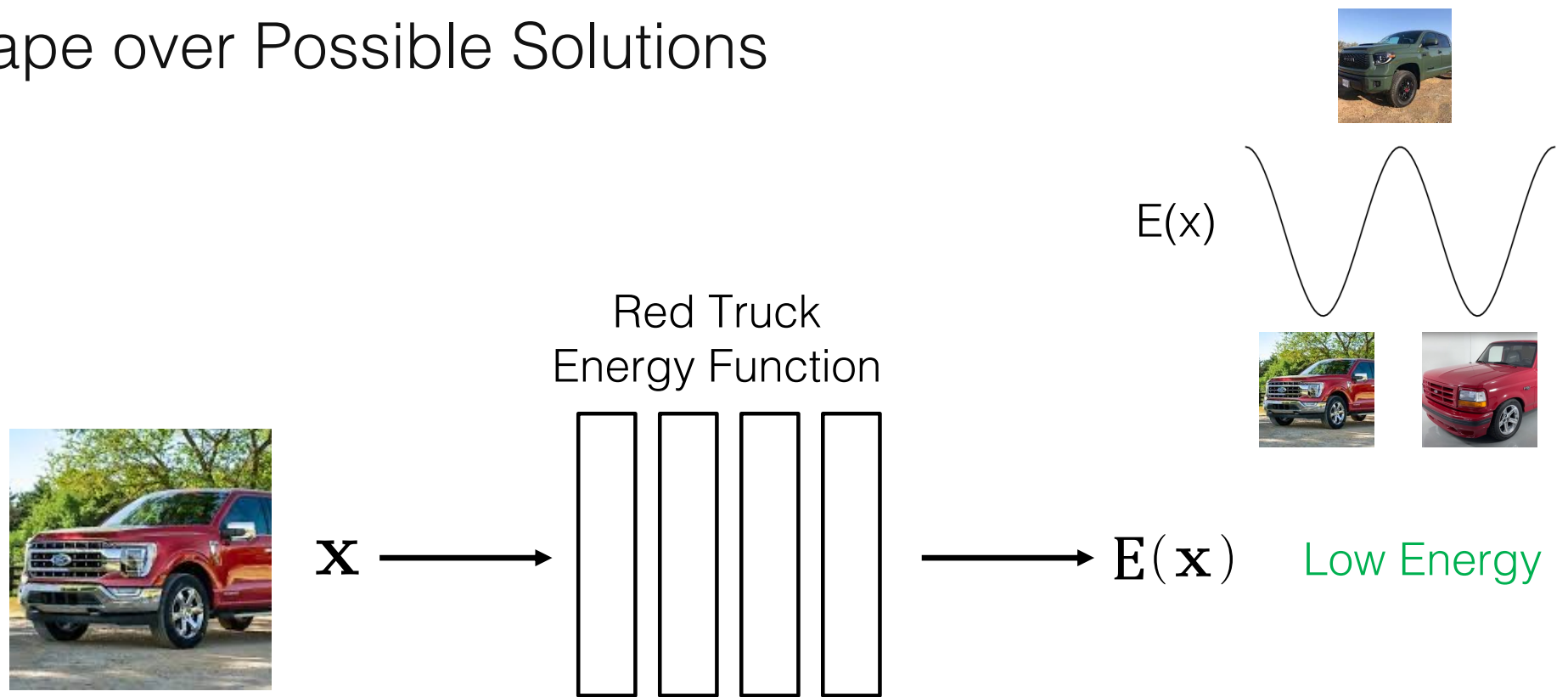
The model scores any proposed image.  
Inference uses these scores to construct an image.

# Low Energy for Compatible Candidates

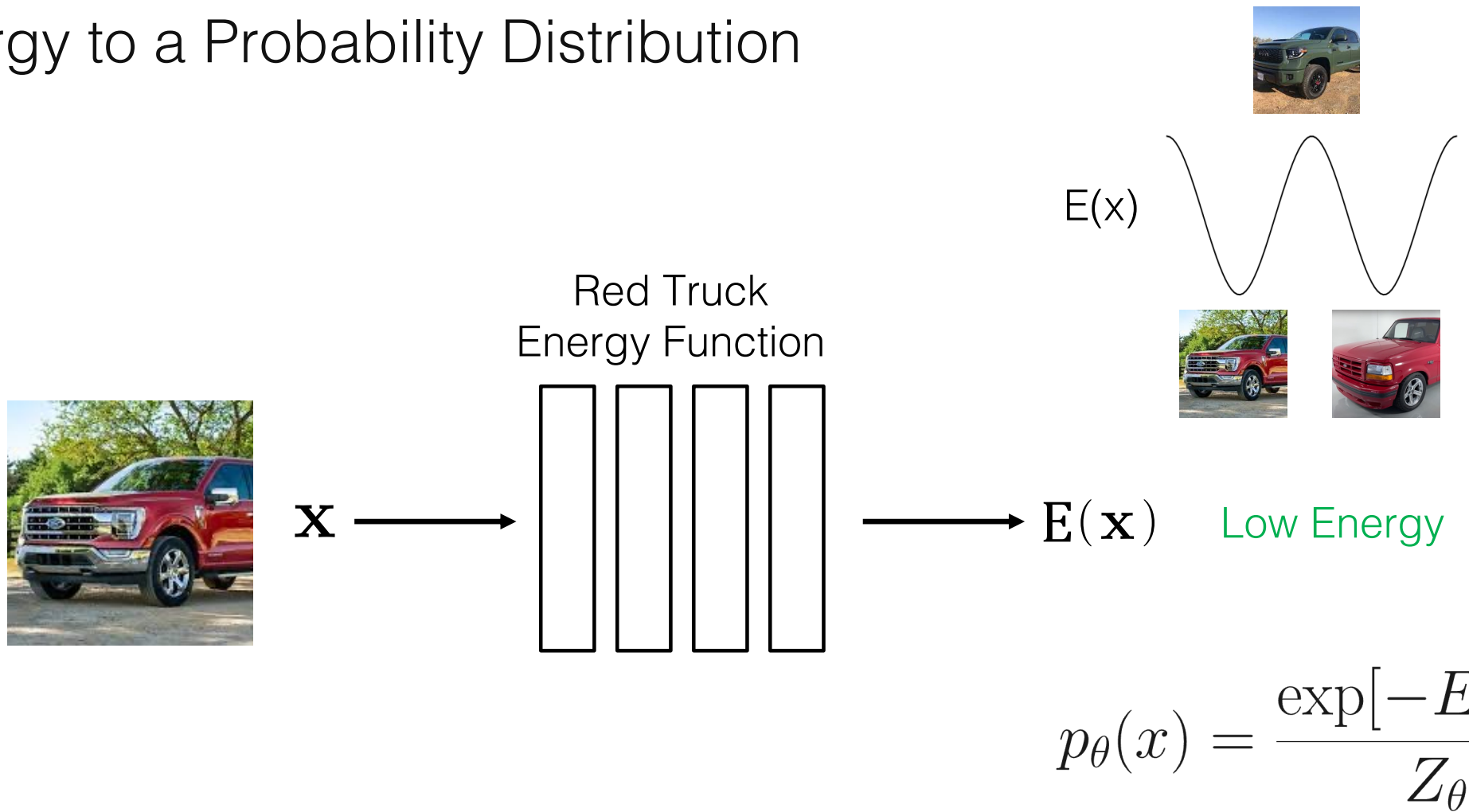


Different images can satisfy the same concept.  
The model scores compatibility, rather than one preferred image.

# A Landscape over Possible Solutions



# From Energy to a Probability Distribution



$Z$  normalizes the distribution.

Lower energy gives higher probability.  
Sampling searches this learned space of possibilities.

# Learning an Energy Function

Training compares observed data with samples from the current model.

$$\nabla_{\theta} \mathcal{L}_{\text{ML}} = \mathbb{E}_{x \sim p_{\text{data}}} [\nabla_{\theta} E_{\theta}(x)] - \mathbb{E}_{x \sim p_{\theta}} [\nabla_{\theta} E_{\theta}(x)]$$

Data samples

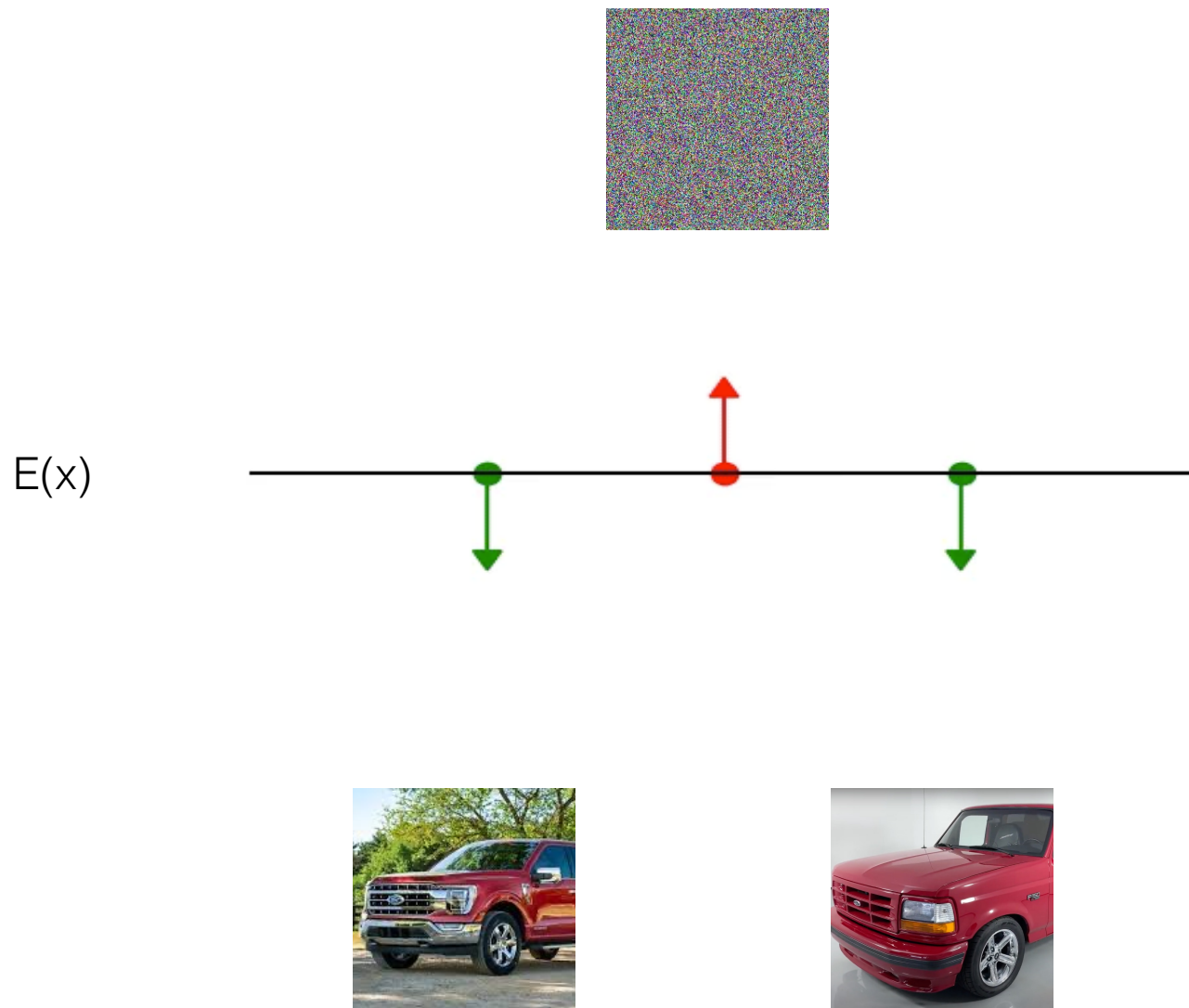
Lower energy on examples  
that actually occur.

Model samples

Raise energy where the model  
assigns too much probability.

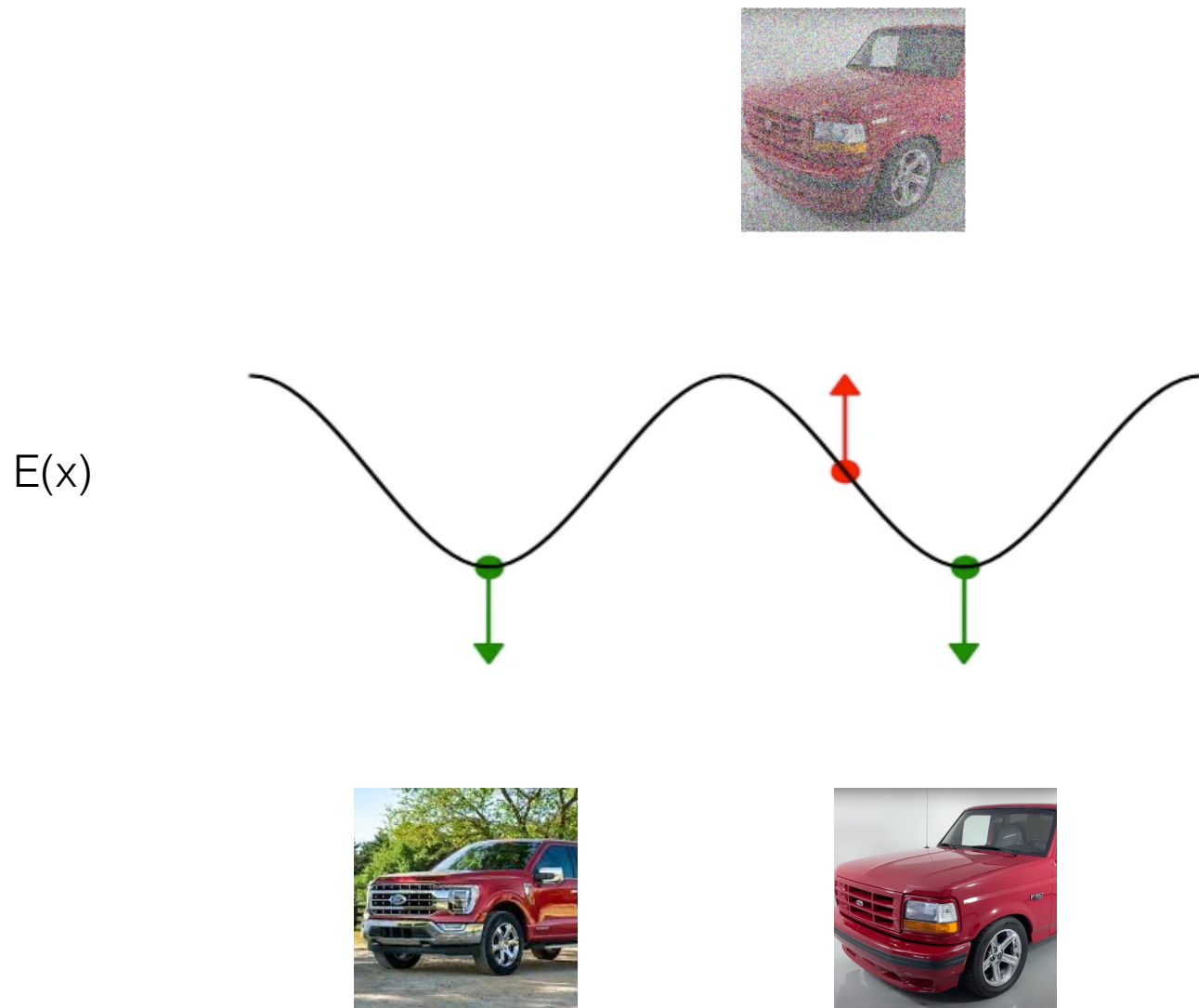
Learning shapes the landscape. Inference uses it.

# Learning an Energy Landscape



Data examples define  
where energy should be  
low.

# Learning an Energy Landscape

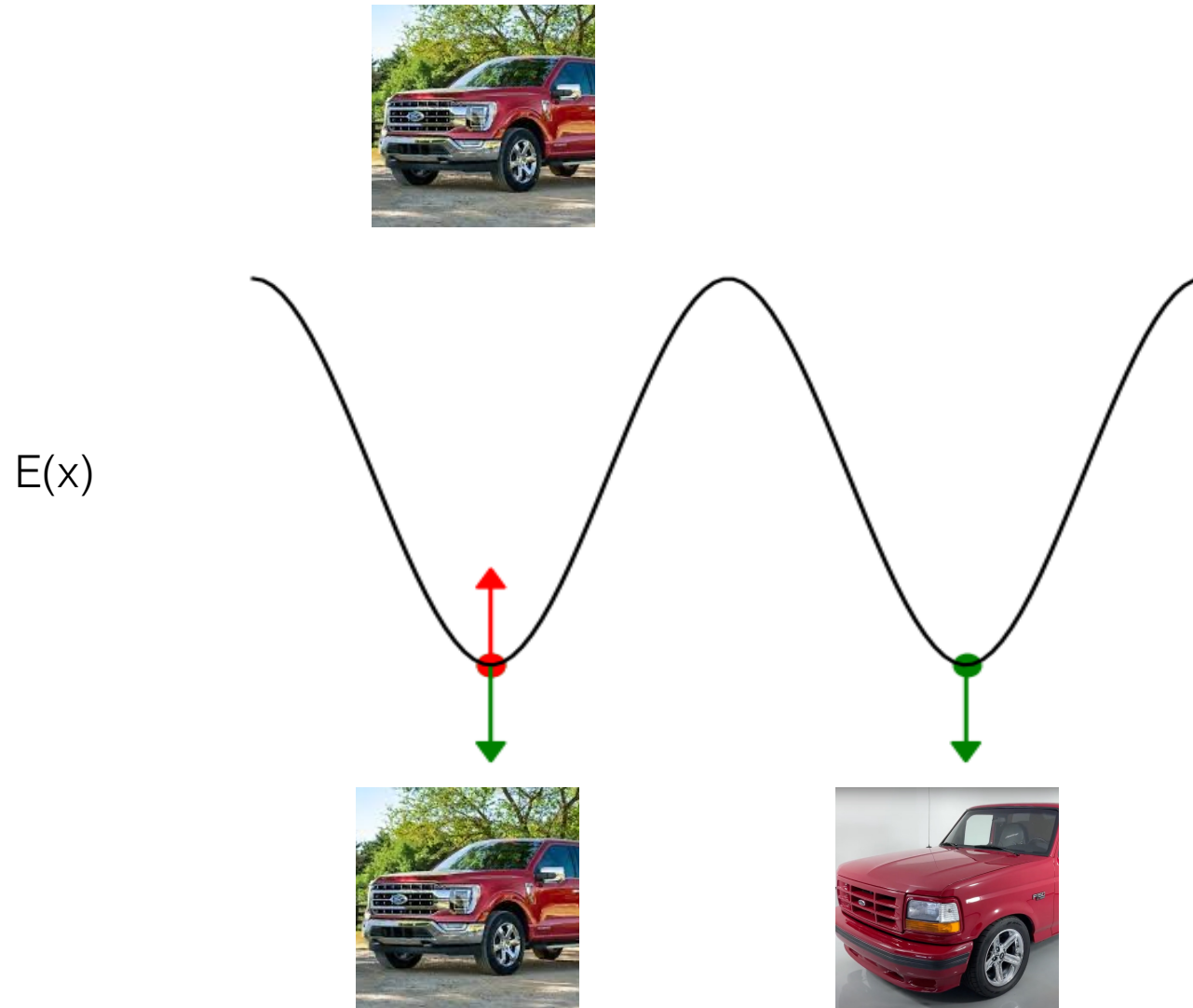


Model samples reveal where energy must be raised.

Explain first · click to start

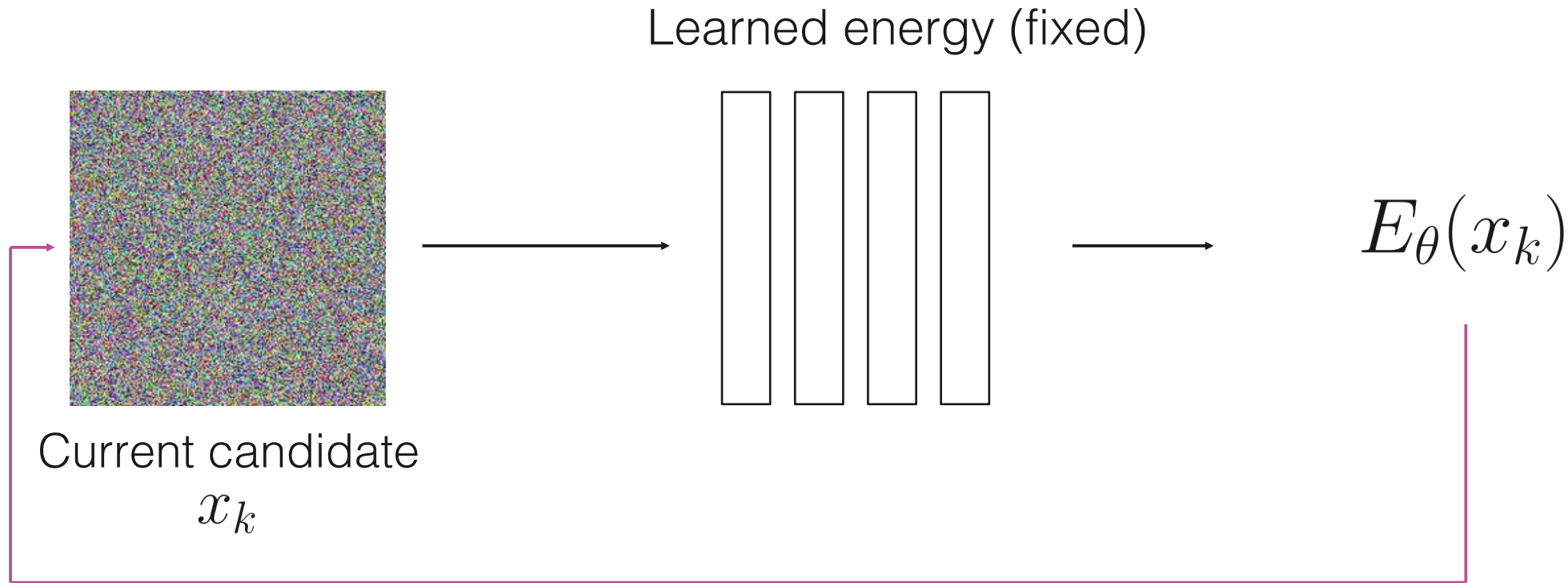
[Play / replay](#)

# Learning an Energy Landscape



After learning, sampling uses this landscape to generate new examples.

# Sampling by Iterative Refinement



Repeated gradient updates turn a learned criterion into a generator.

$$x_{k+1} = x_k - \eta \nabla_x E_\theta(x_k) + \sqrt{2\eta} \epsilon_k$$

# Diffusion Learns Refinement at Different Noise Levels

## Energy-based model

Learn a scalar energy.  
Differentiate it to refine a candidate.

$$x \longrightarrow E_{\theta}(x) \longrightarrow -\nabla_x E_{\theta}(x)$$

## Score-based diffusion

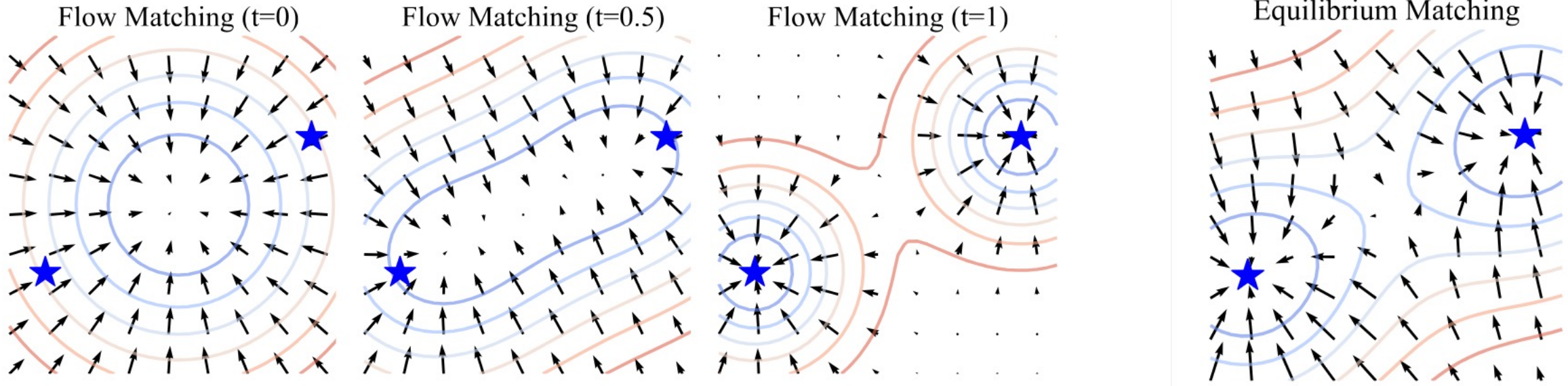
Learn a direction for each noise level.  
Refine as the noise level changes.

$$(x, t) \longrightarrow s_{\theta}(x, t)$$

$$s_{\theta}(x, t) \approx \nabla_x \log p_t(x) = -\nabla_x E_t(x)$$

Both expose directions for refinement, rather than only a final prediction.

# Equilibrium Matching Learns a Time-Invariant Field



Time-conditioned fields change with  $x$  and  $t$ .

Data points become equilibria of one learned field.

The same field supports repeated optimization updates.

$$x_{k+1} = x_k - \eta f_{\theta}(x_k)$$

One field:  $f_{\theta}(x)$

[EqM details](#)

# Optimization-Based Sampling Generates Images

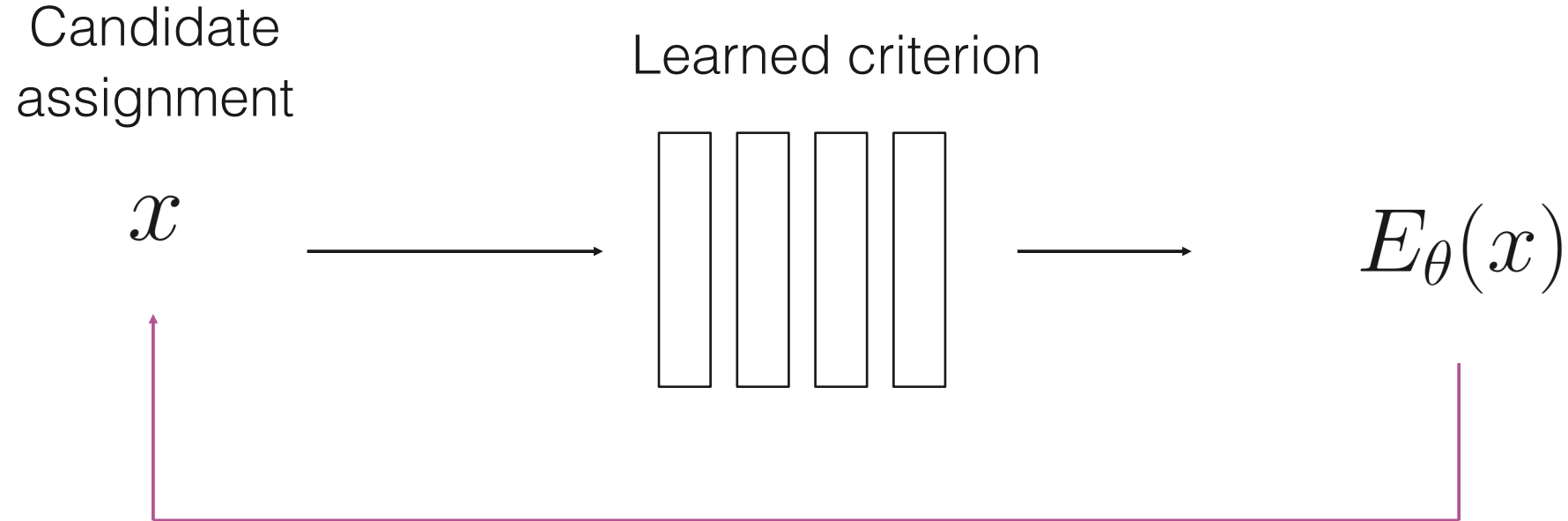


Class-conditional  
ImageNet

$256 \times 256$

The same learned field  
generates different  
images  
by iterative refinement.

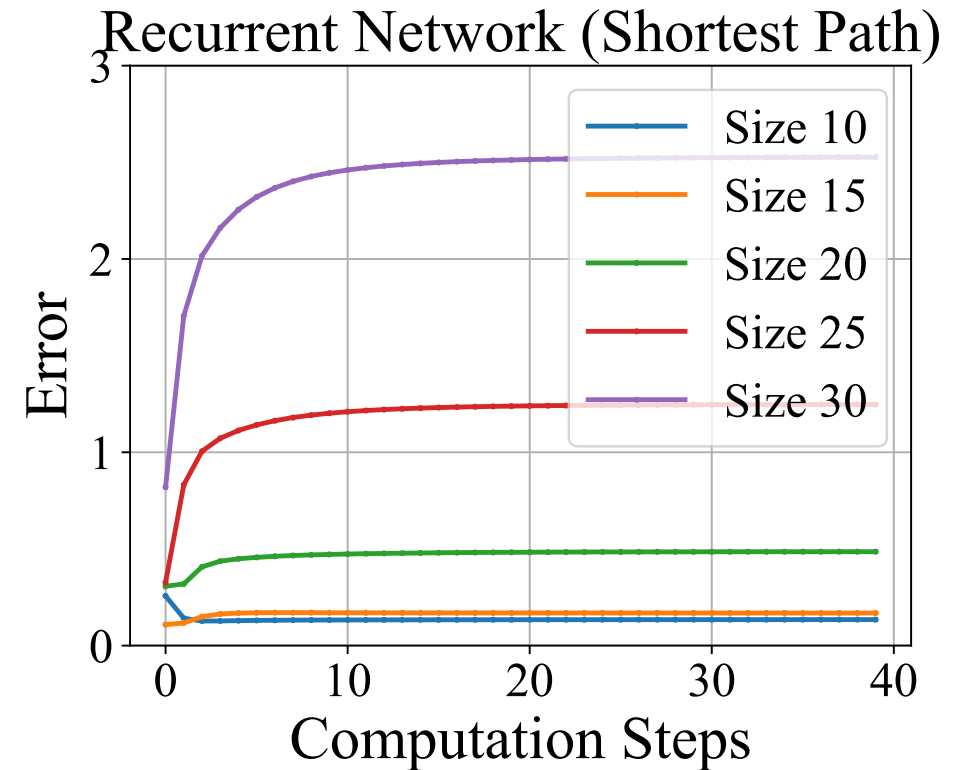
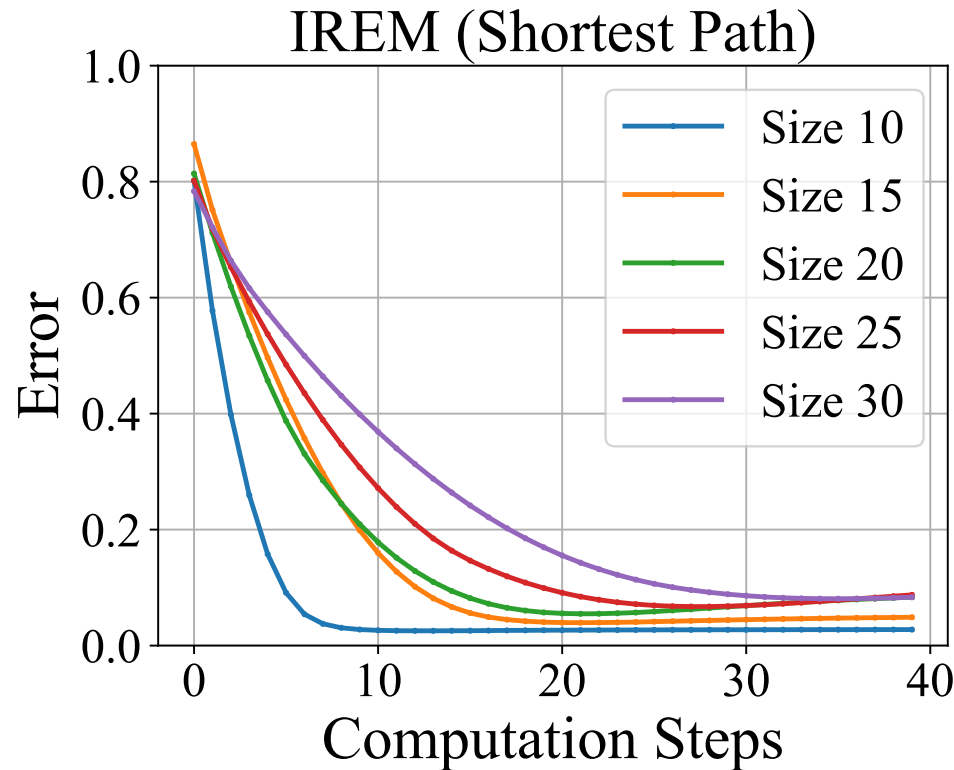
# Reasoning as Energy Minimization



$$x^{\star} \in \arg \min_x E_{\theta}(x)$$

Learn an energy over candidate assignments.  
At test time, revise the assignment to reduce that energy.

# Adapting Computation to the Instance



The same learned criterion can refine a solution for more steps.

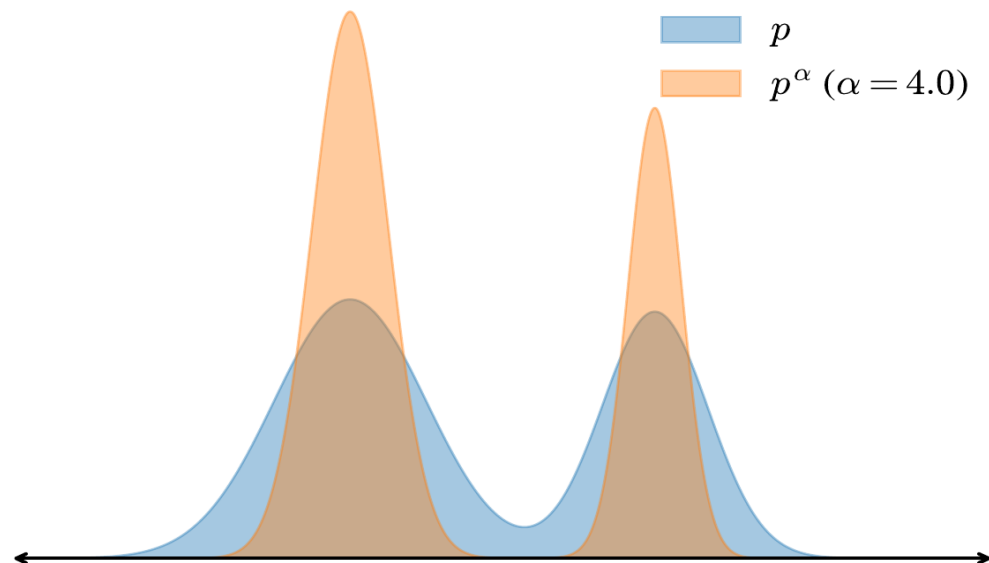
Larger problems can use more computation at test time.

# Using a Frozen Language Model To Reason

Change the distribution used for sampling

$$q_{\alpha}(x) = \frac{p(x)^{\alpha}}{Z_{\alpha}}$$

$$E(x) = -\alpha \log p(x)$$



$x$  is a complete reasoning trace.

$\alpha = 1$  keeps the base distribution.

$\alpha > 1$  emphasizes higher-likelihood traces.

No additional training or external verifier.

# Sequence-Level Power Is Not Tokenwise Temperature

Suppose A has one continuation: 0.45.  
B has two continuations: 0.30 and 0.25. Set  $\alpha = 2$ .

Sequence-level power

$$\sum_j p_j^\alpha$$

$$\text{A: } 0.45^2 = 0.2025$$

$$\text{B: } 0.30^2 + 0.25^2 = 0.1525$$

Tokenwise temperature

$$\left( \sum_j p_j \right)^\alpha$$

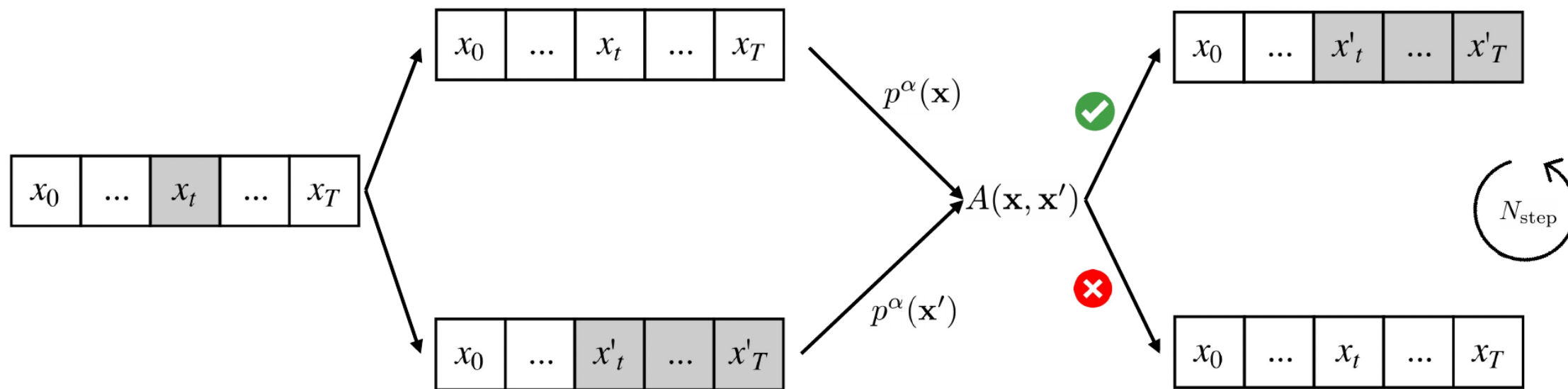
$$\text{A: } 0.45^2 = 0.2025$$

$$\text{B: } (0.30 + 0.25)^2 = 0.3025$$

Whole-sequence power favors A. Token temperature favors B.

Illustrative probabilities. Branch weights are shown before normalization.

# Revising a Reasoning Trace with MCMC

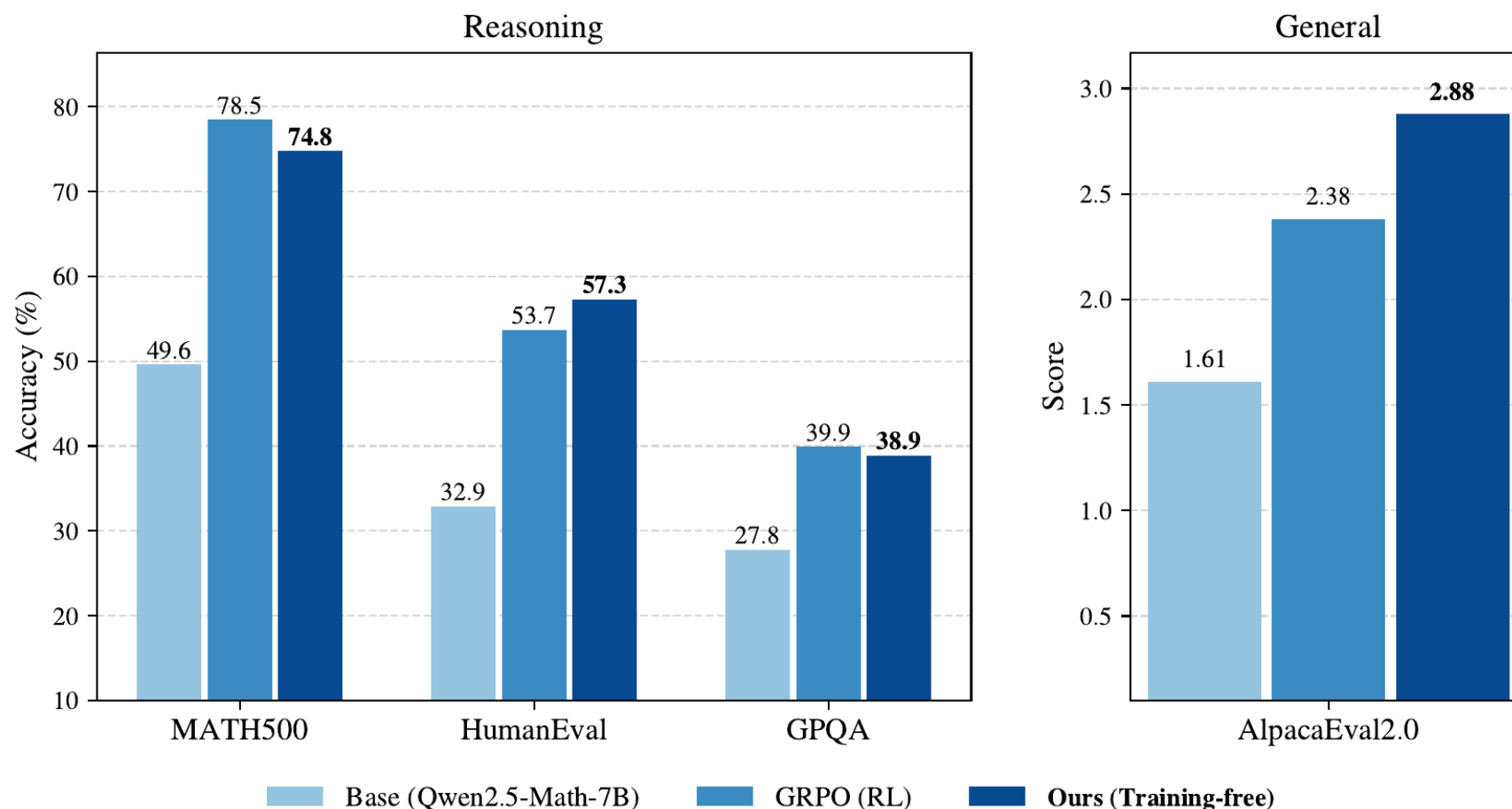


$$\emptyset \longrightarrow \pi_1 \longrightarrow \pi_2 \longrightarrow \dots \longrightarrow p^\alpha, \quad \pi_k(\text{prefix}) \propto p(\text{prefix})^\alpha$$

1 Extend by B tokens  $\rightarrow$  2 Choose a random cut  $\rightarrow$  3 Resample the suffix  $\rightarrow$  4 Accept by MH ratio

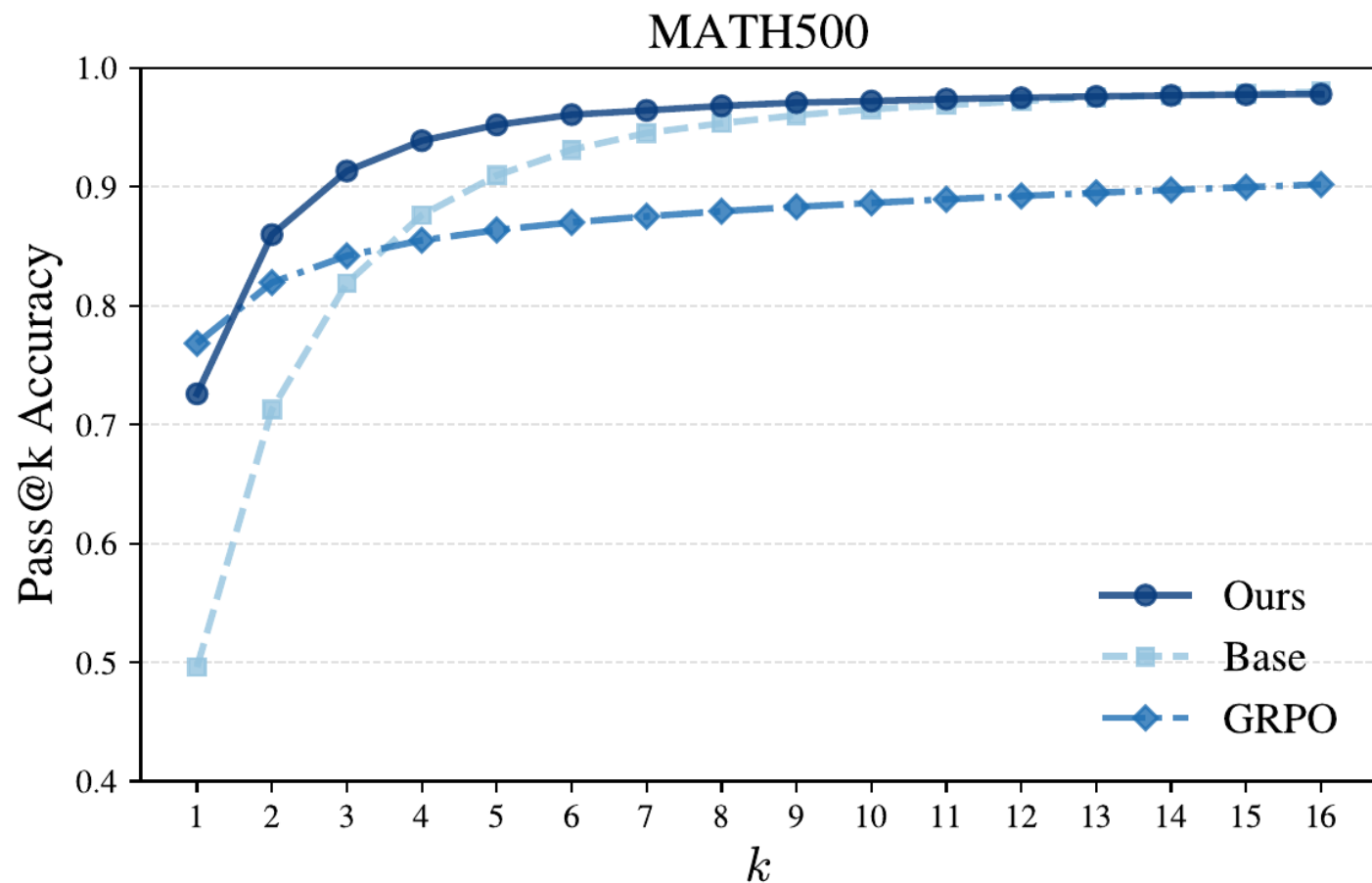
A suffix revision can replace an earlier reasoning decision.  
The fixed model scores the old and revised traces.

# Changing Inference Changes the Measured Capability



Revising complete reasoning traces improves accuracy  
while keeping the pretrained model fixed.

# Single-Sample Accuracy and Multi-Sample Coverage



Stronger first-answer accuracy; high- $k$  coverage close to the base model

# A Learned Model Is Not One Fixed Behavior

Learned structure

Inference

A space of plausible candidates       $\longrightarrow$       Sample, compare, and revise a candidate

What the examples establish

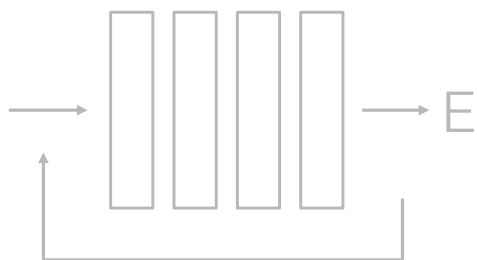
A learned distribution can support new reasoning procedures that improve the answers we obtain from the same model.

Next: can we also change the problem by composing models?

Inference determines how we use the knowledge already in the model.

# From Learned Models to New Behavior

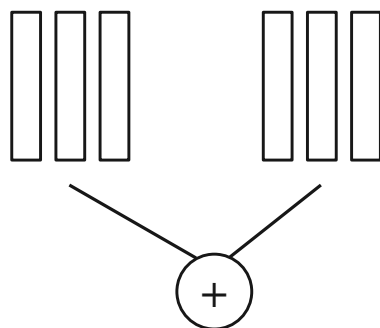
Generative  
inference



Learn a solution  
space.

Sample and refine  
a candidate.

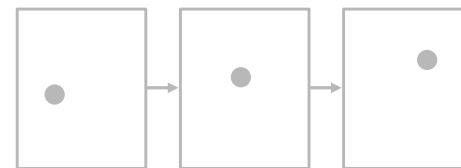
Compositional  
inference



Learn reusable  
relations.

Compose and satisfy  
them together.

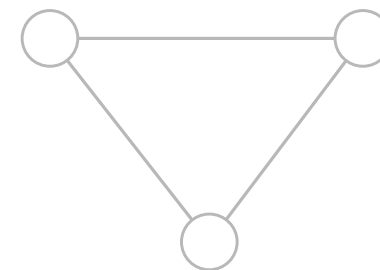
World models  
and planning



Learn possible  
futures.

Search for behavior  
under a new goal.

Multi-agent  
intelligence



Reuse learned  
capabilities.

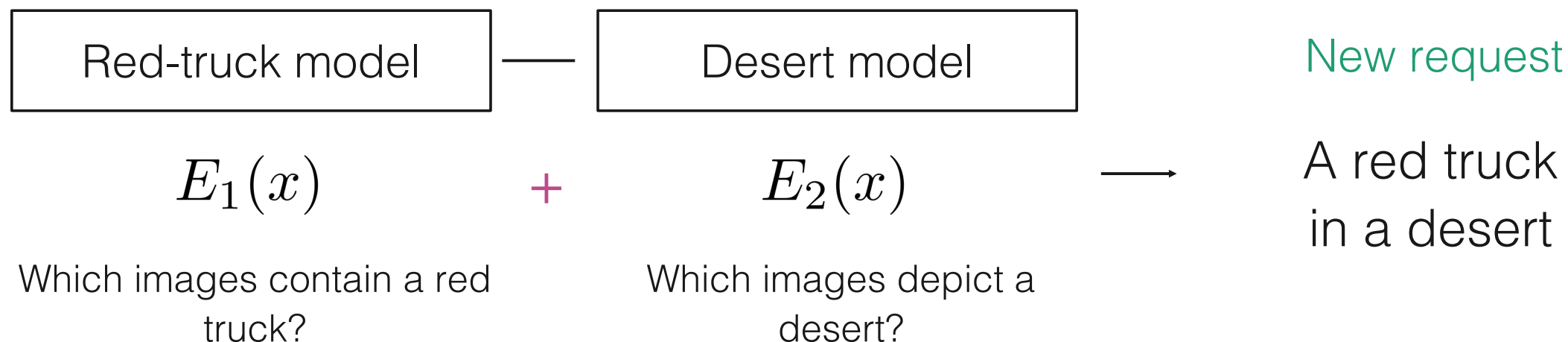
Communicate,  
evaluate,  
and choose who acts.

Can familiar learned relations define an unfamiliar complete problem?

# Composition Changes the Problem We Ask the Model to Solve

So far: use one learned model differently.

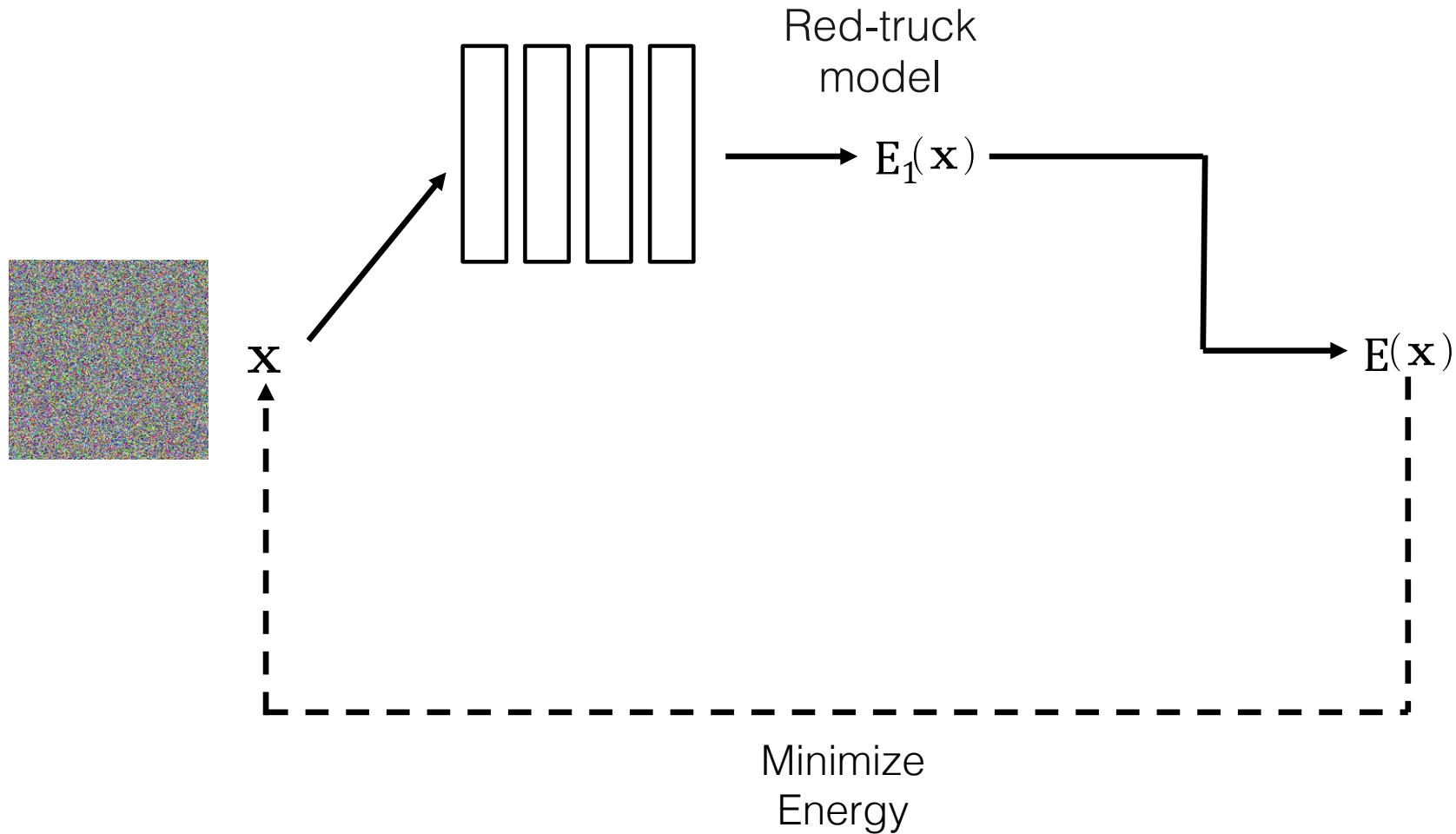
Now: assemble the requirements for a different problem.



The same candidate must satisfy both learned requirements.

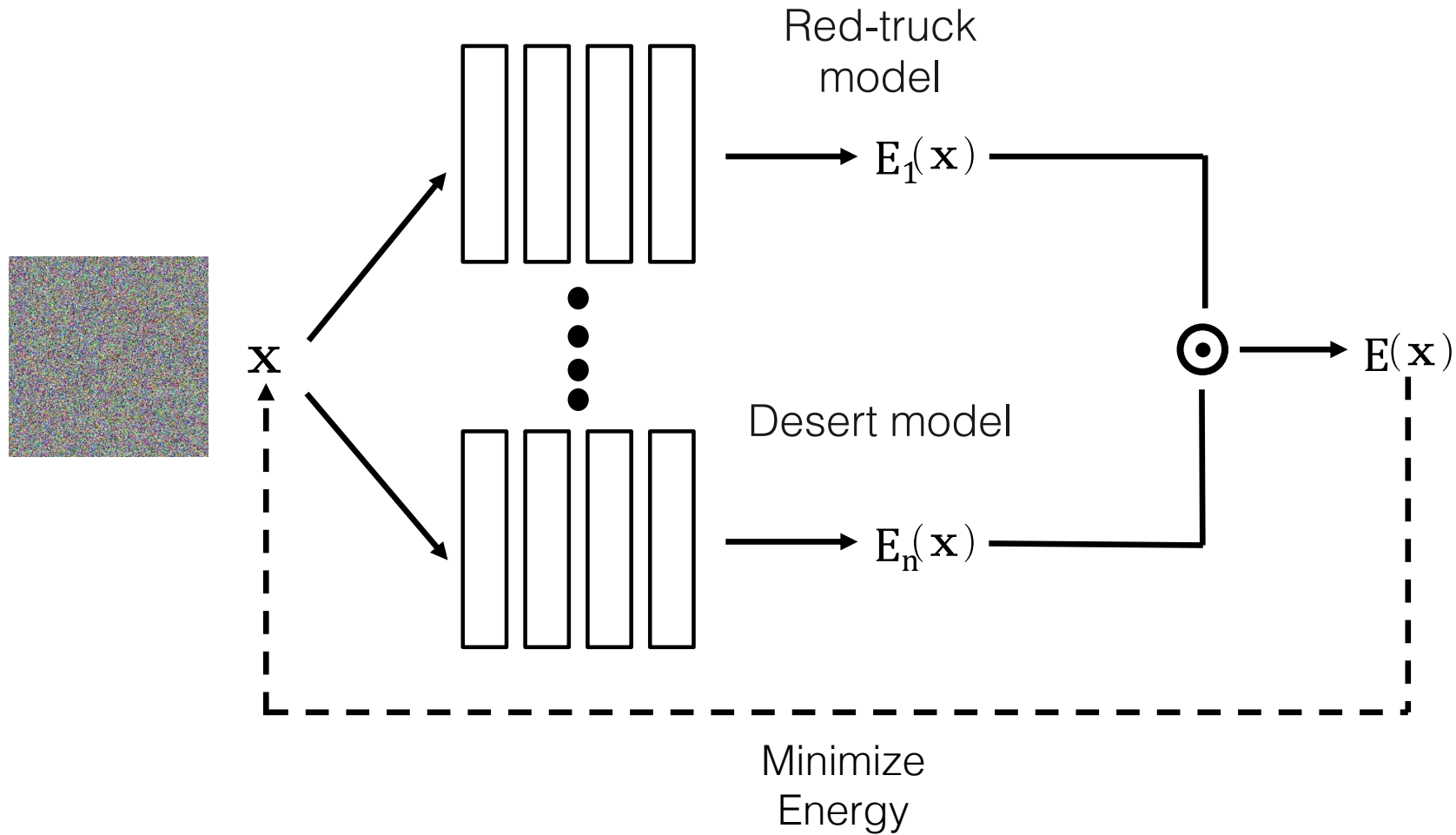
Keep the components fixed. Change how they define the task.

# A Learned Component Can Represent One Requirement



A component can represent one concept or requirement. It can later be reused as part of a larger specification.

# Composing Requirements at Inference Time

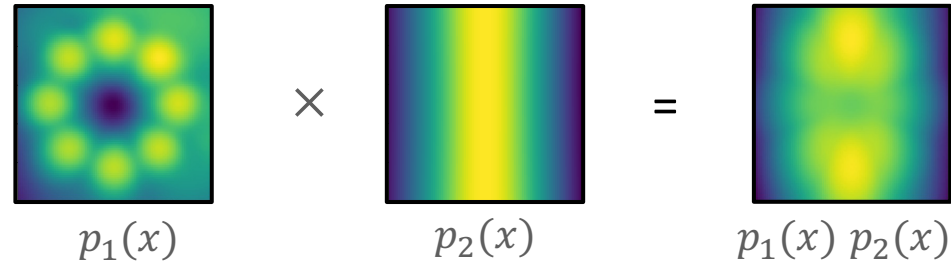


Both models score the same candidate image.

Minimizing their summed energy searches for an image that fits both concepts.

# Products Combine Requirements

Products:

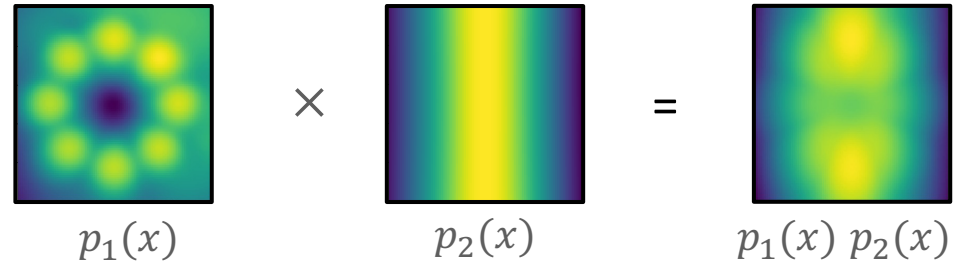


Both models evaluate the same variable  $x$ .

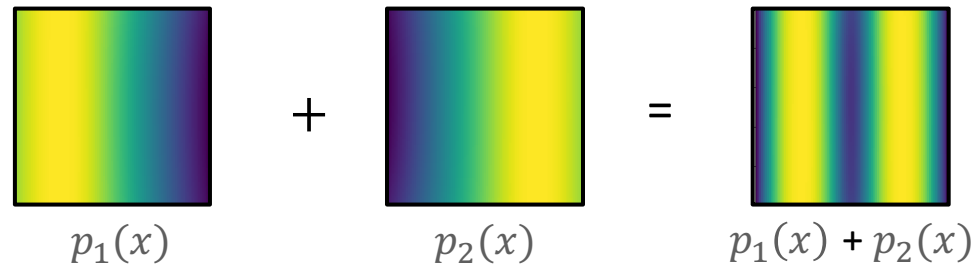
A product favors candidates supported by both models.

# Mixtures Represent Alternatives

Products:



Mixtures:



Products combine requirements.

Mixtures represent alternative possibilities.

# Generating from a New Composed Specification



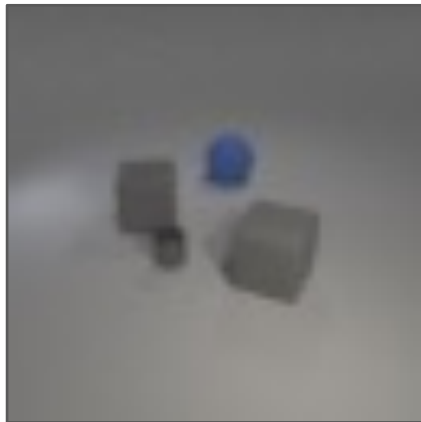
The same component models generate different combinations  
when we change the composed specification.

# Learning Components from Different Data

Dataset 1



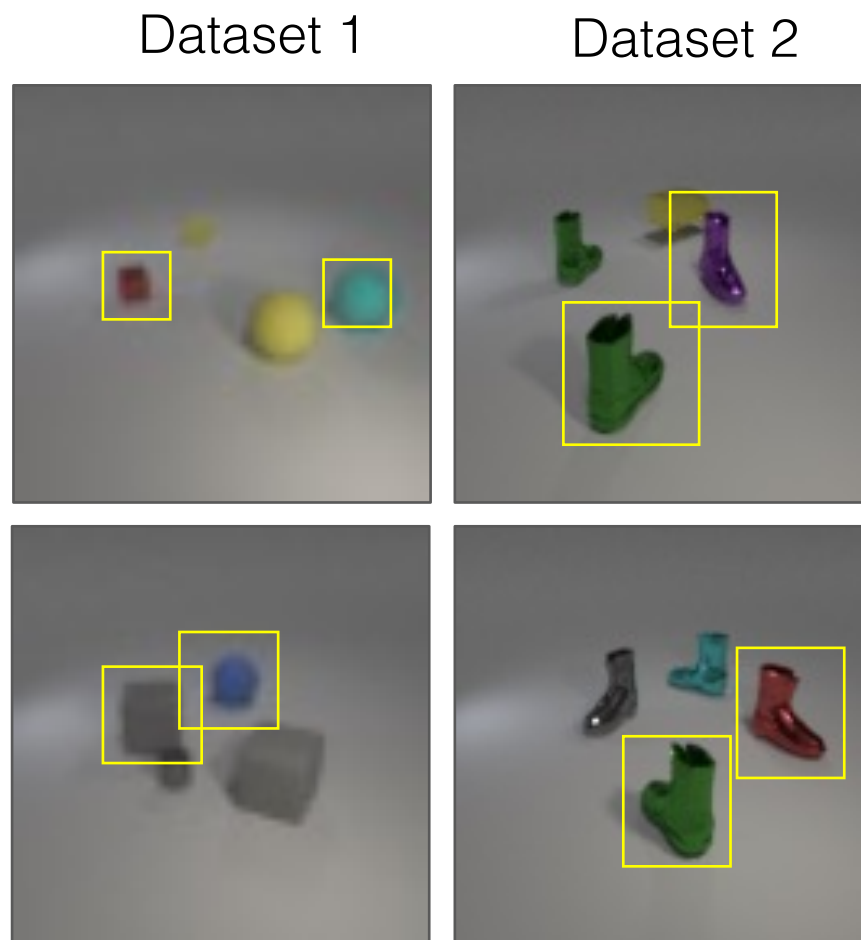
Dataset 2



Images

Can knowledge learned in  
separate datasets  
be used in one new scene?

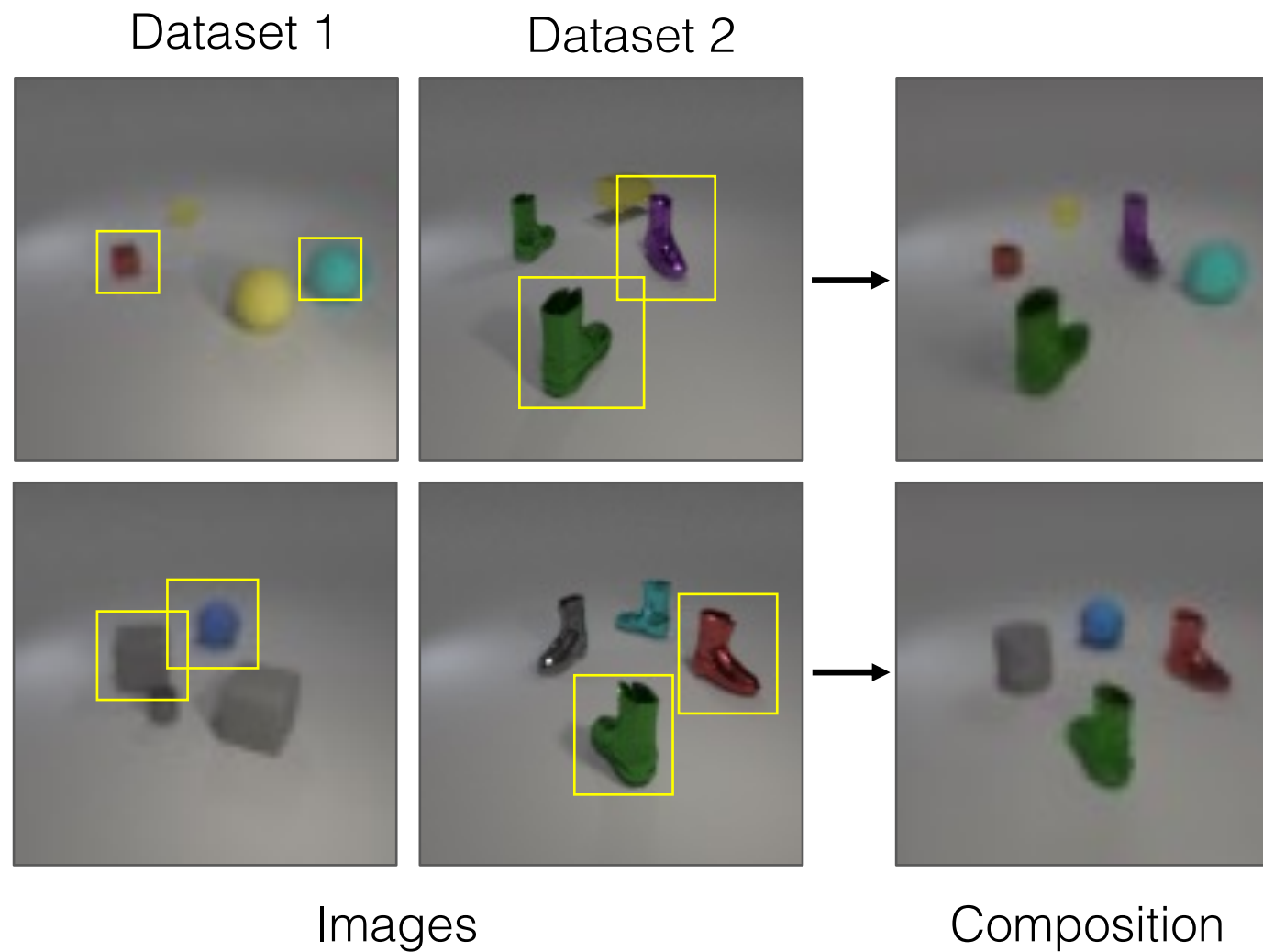
# Identifying the Reusable Components



The highlighted objects identify components that can be reused across scenes.

Images

# Recombining Components into a New Scene

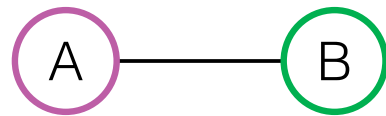


Sample from the  
composed  
distribution:

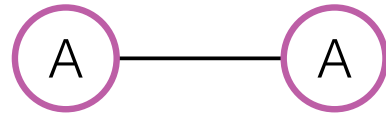
$$p_{\theta}(x|z_1) p_{\theta}(x|z_2) \\ p_{\phi}(x|z_3) p_{\phi}(x|z_4)$$

# Composing Relations into a Reasoning Problem

Learn a local relation



Low energy

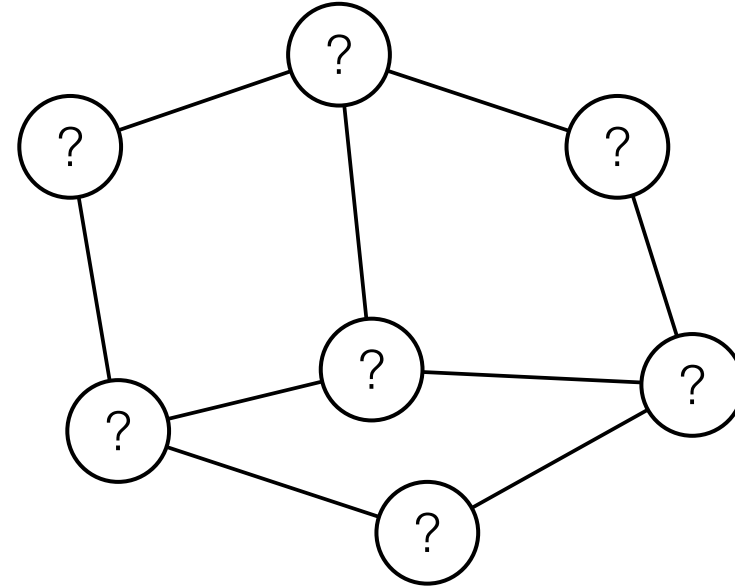


High energy

Same criterion  
on every edge



Reuse it on a larger graph



$$E_G(\mathbf{y}) = \sum_{(i,j) \in \mathcal{E}(G)} E_\theta(y_i, y_j)$$

Reuse the learned relation on every edge.

Optimize the shared node assignments under the summed energy.

# Learning a Simple Relation on a Small Subproblem

Train



Learn the energy of a small relation.

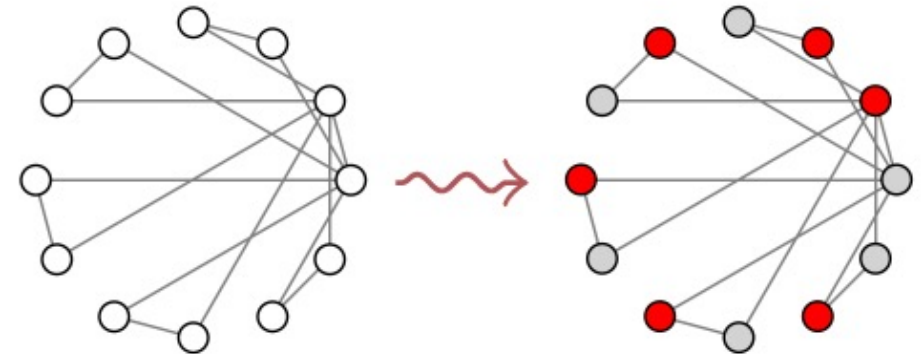
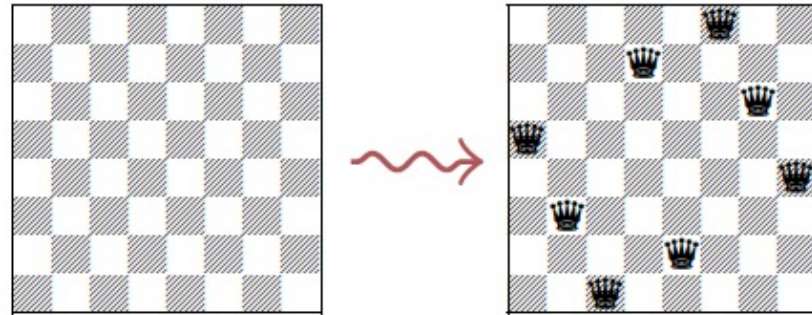
Reuse that energy whenever the relation appears in a larger problem.

# Reusing the Relation in a Larger Problem

Train



Evaluation



The complete problem changes.

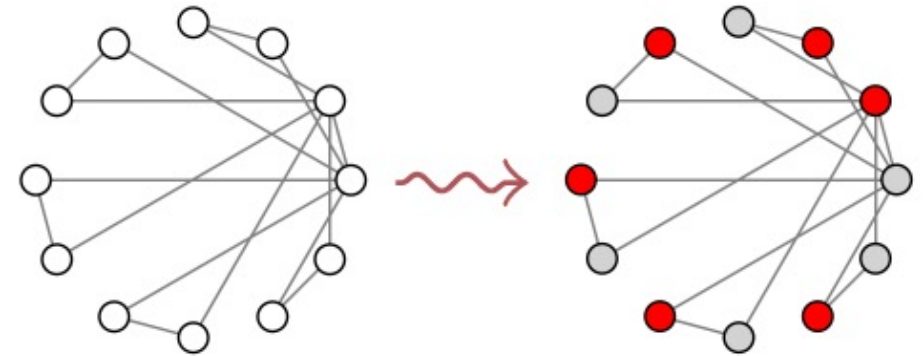
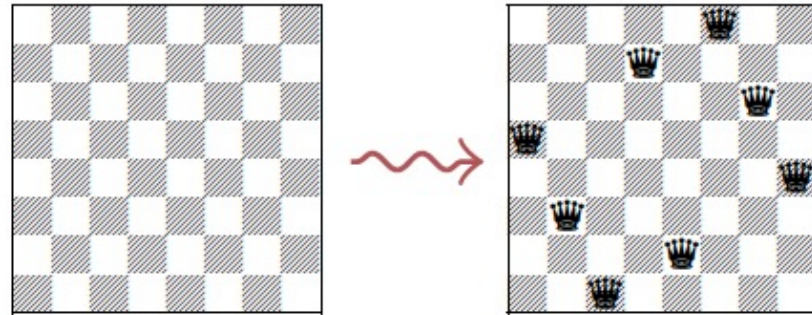
Inference resolves the interactions between the familiar subproblems.

# Zero-Shot Composition on N-Queens

Train

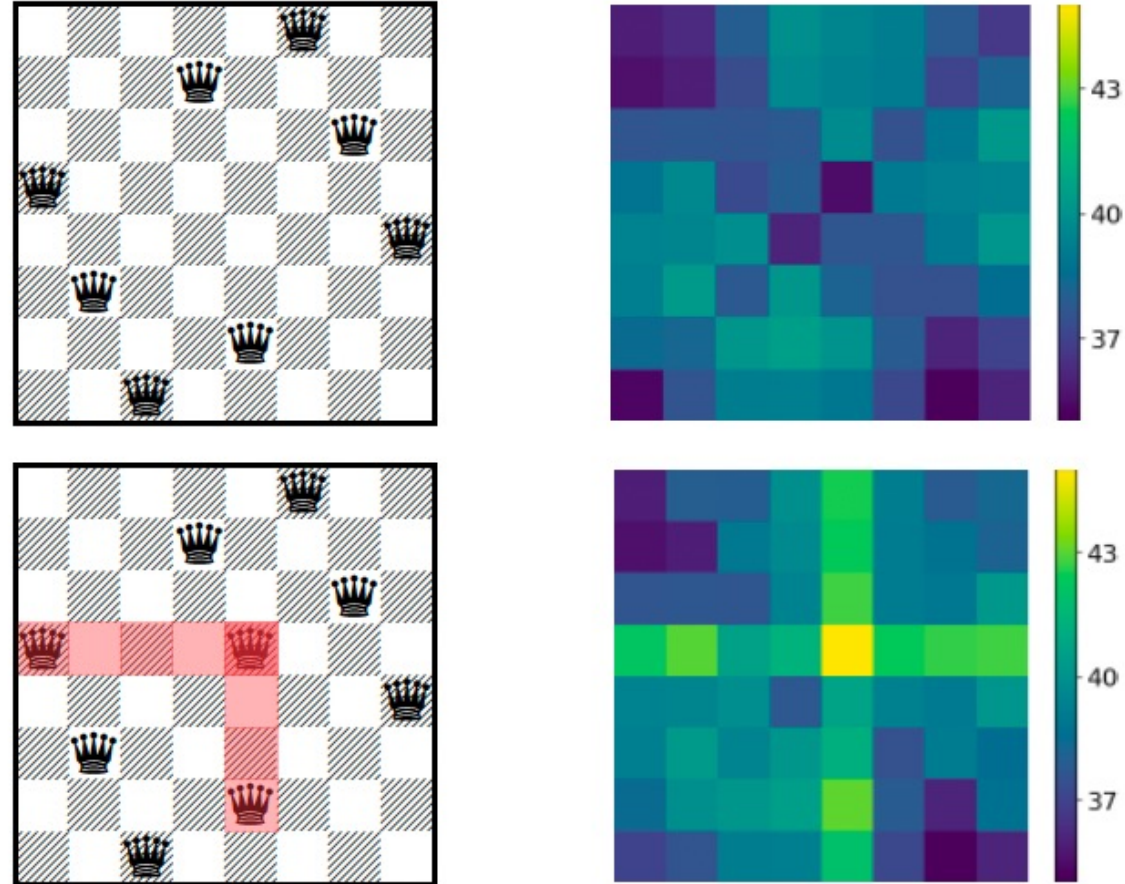


Evaluation



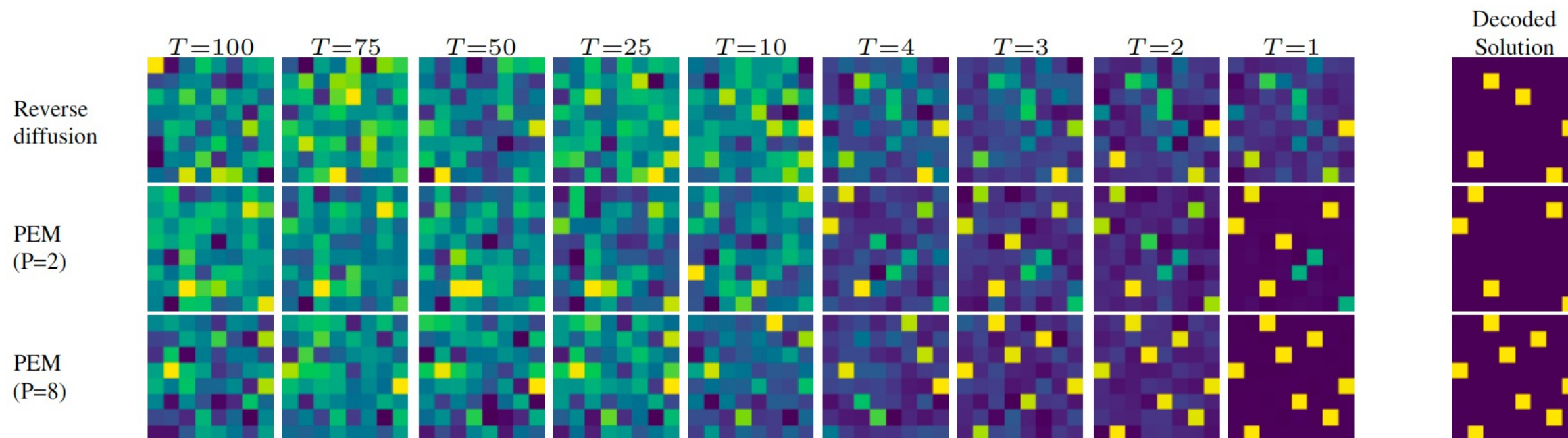
The test problem couples many familiar local constraints.  
Inference must find one assignment that satisfies them together.

# Inspecting the Learned Energy over Positions



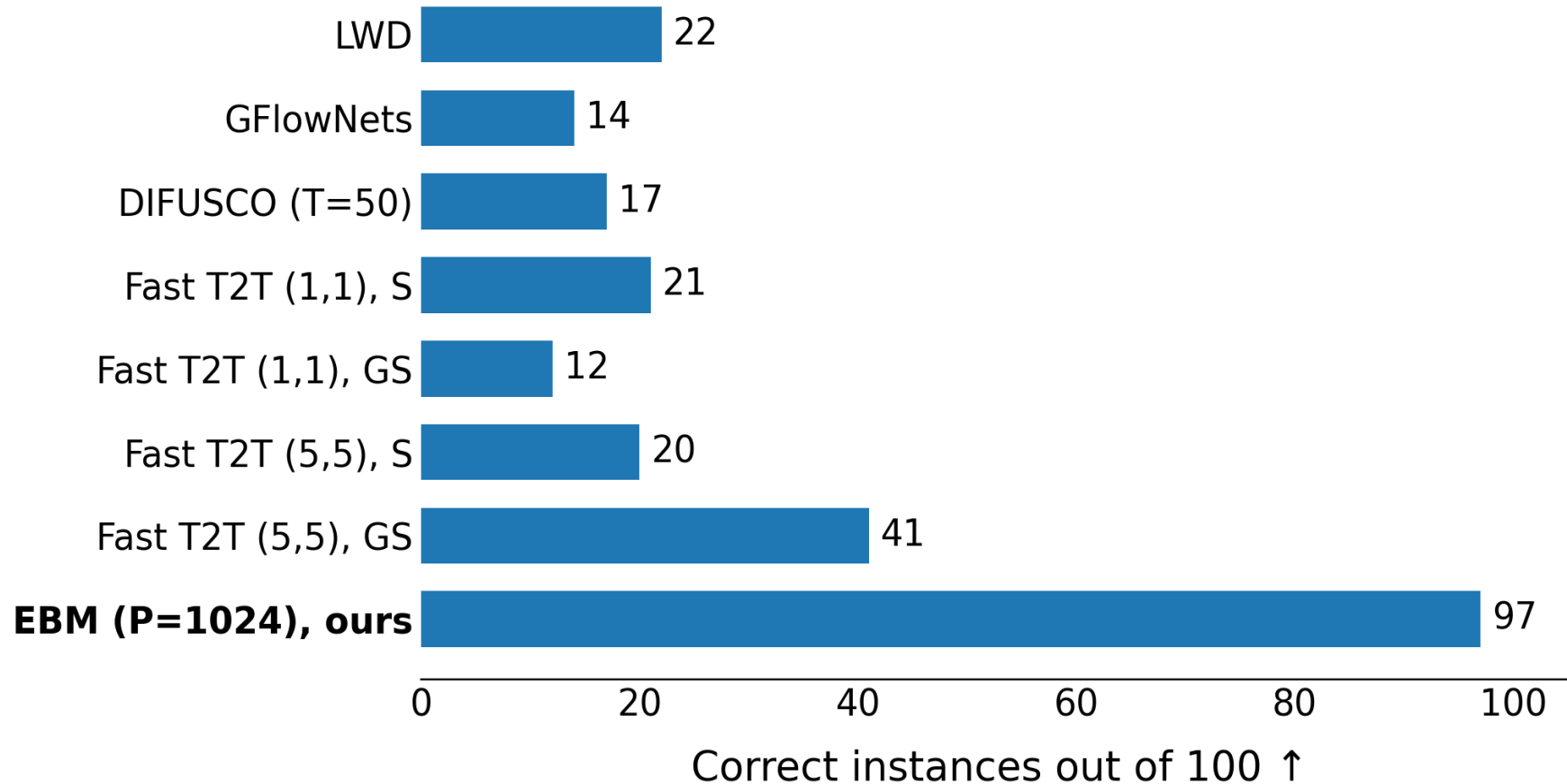
The energy map scores possible positions under the current partial assignment.

# Refining the Joint Assignment



Each variable receives updates from every constraint that uses it.  
Joint refinement coordinates the local requirements.

# Measuring Complete N-Queens Solutions



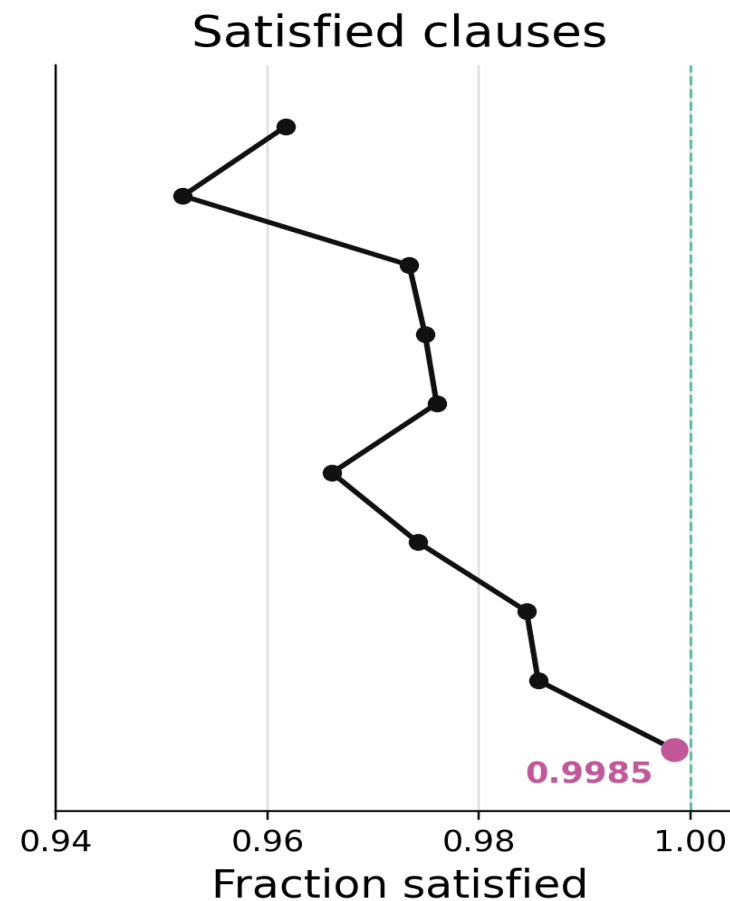
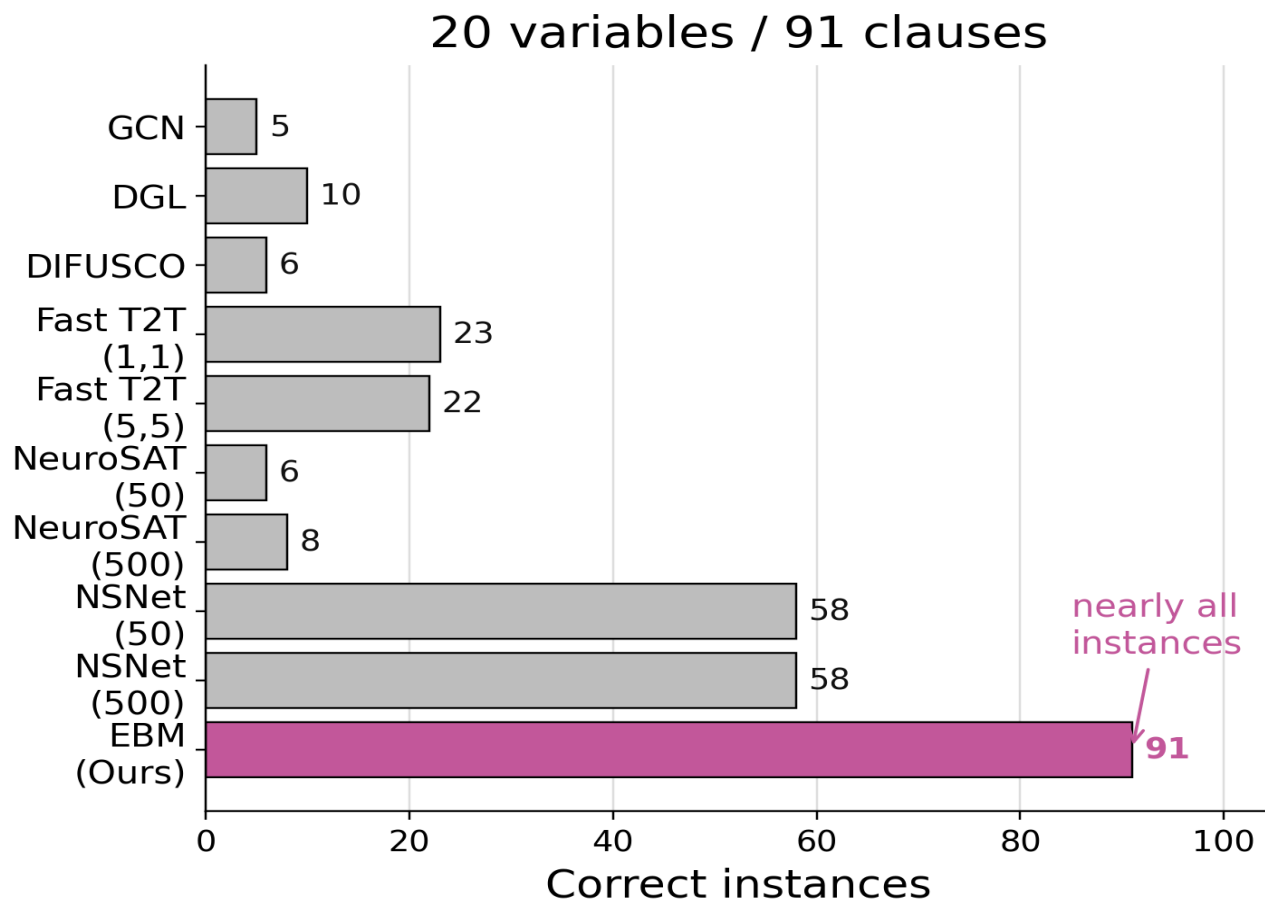
A complete solution must satisfy all constraints together.

97 / 100

correct with the composed EBM

Full table [↗](#)

# Composed Reasoning on 3-SAT



On the training-sized SAT problems, the composed EBM solves many more complete instances while also satisfying nearly

# Reusable Relations, Coupled Solutions

Learned structure

Inference

Local concepts and constraints



Assemble them and solve for  
shared variables

What the examples establish

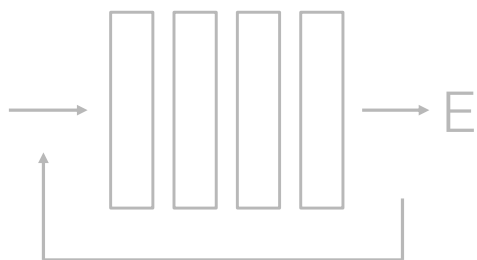
A fixed set of learned relations can define new scenes  
and larger reasoning problems at test time.

Next: can a model of the world help choose what to do?

The new behavior comes from assembling and solving a new joint objective.

# From Learned Models to New Behavior

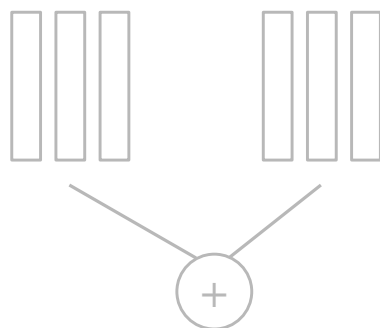
Generative  
inference



Learn a solution  
space.

Sample and refine  
a candidate.

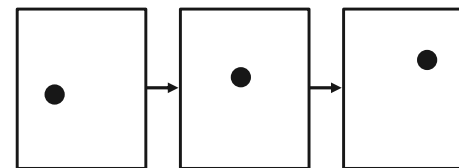
Compositional  
inference



Learn reusable  
relations.

Compose and satisfy  
them together.

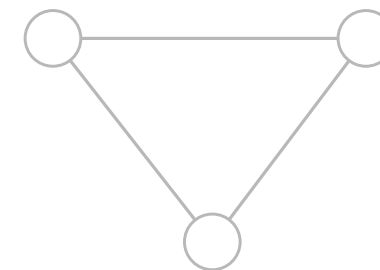
World models  
and planning



Learn possible  
futures.

Search for behavior  
under a new goal.

Multi-agent  
intelligence

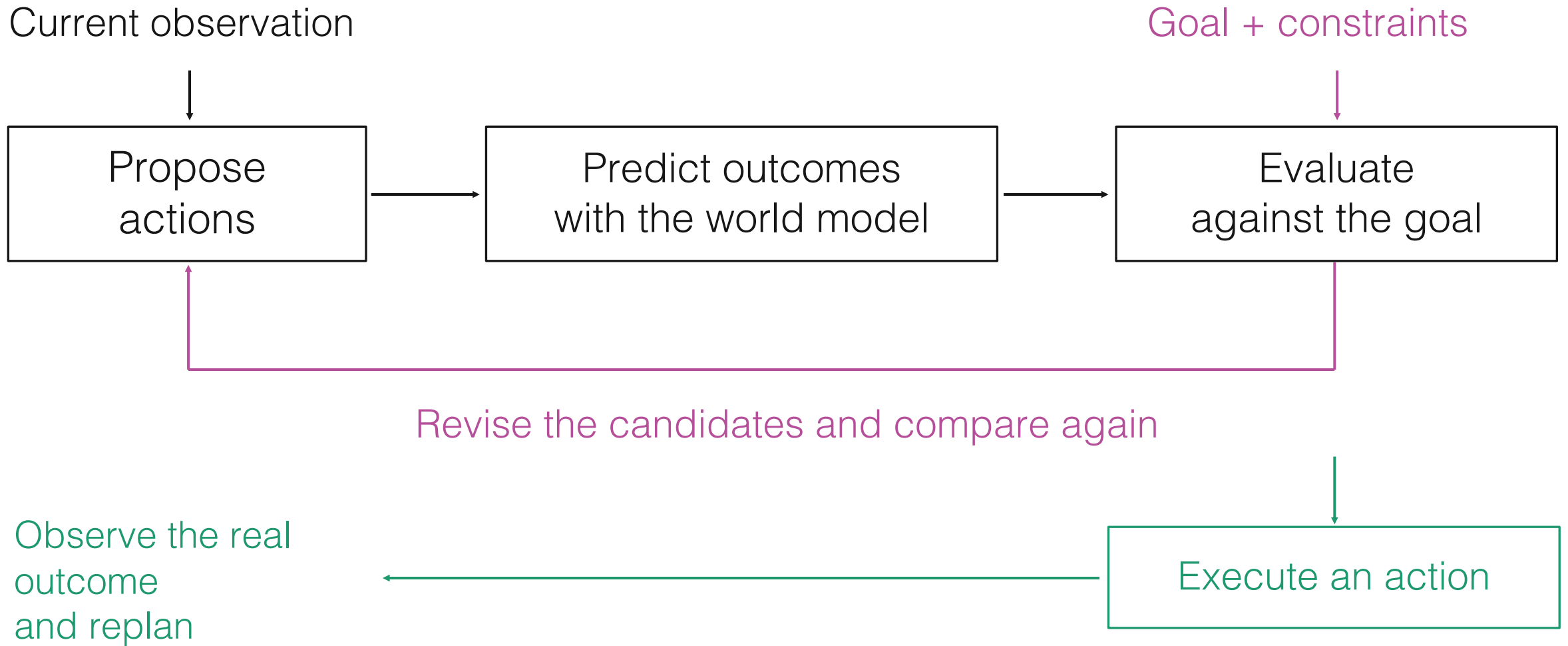


Reuse learned  
capabilities.

Communicate,  
evaluate,  
and choose who acts.

Can a model of possible futures support a different goal?

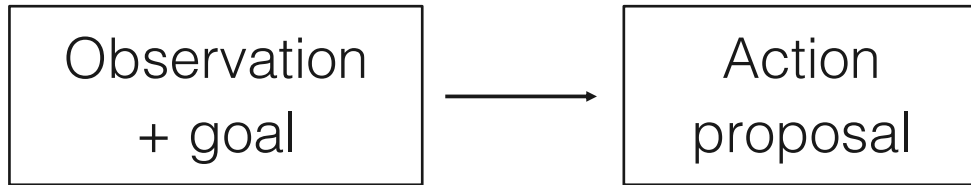
# A World Model Turns Acting into Inference



Predict what each action would do. Choose using the goal and constraints for this task.

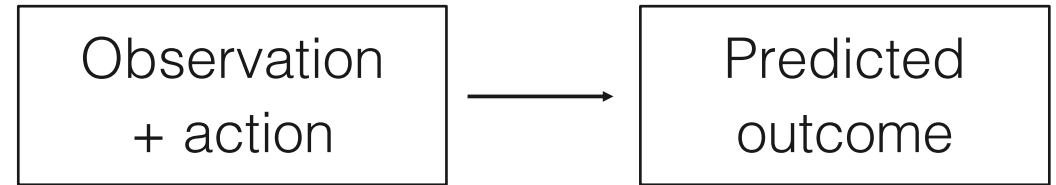
# Policies and Models Play Different Roles

A policy proposes actions



Predict or sample an action  
using the learned controller.

A world model predicts  
consequences

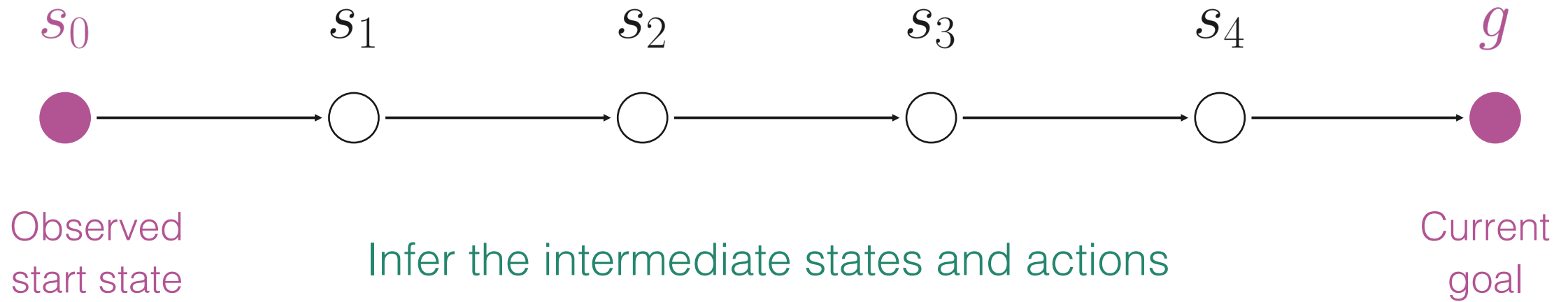


A planner compares predicted  
outcomes  
under the current objective.

The same world model can support different decisions as the goal or cost changes.

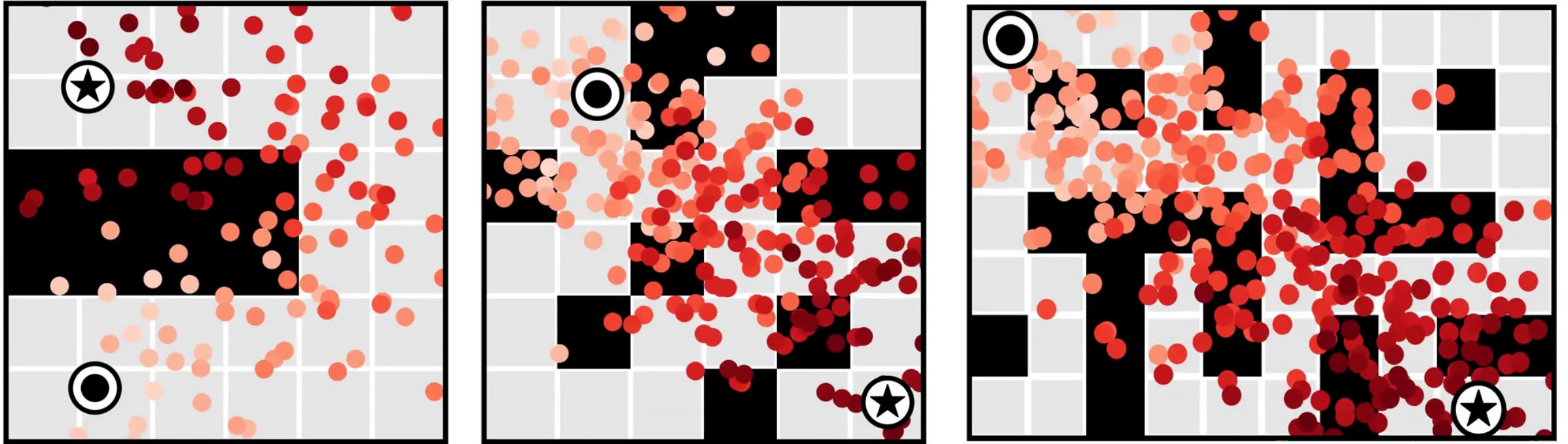
# Changing the Goal without Relearning the Model

Keep the learned trajectory model fixed.



Change the start and goal; infer a new trajectory with the same learned prior.

# Generating a Goal-Directed Trajectory



The start and goal define this task.

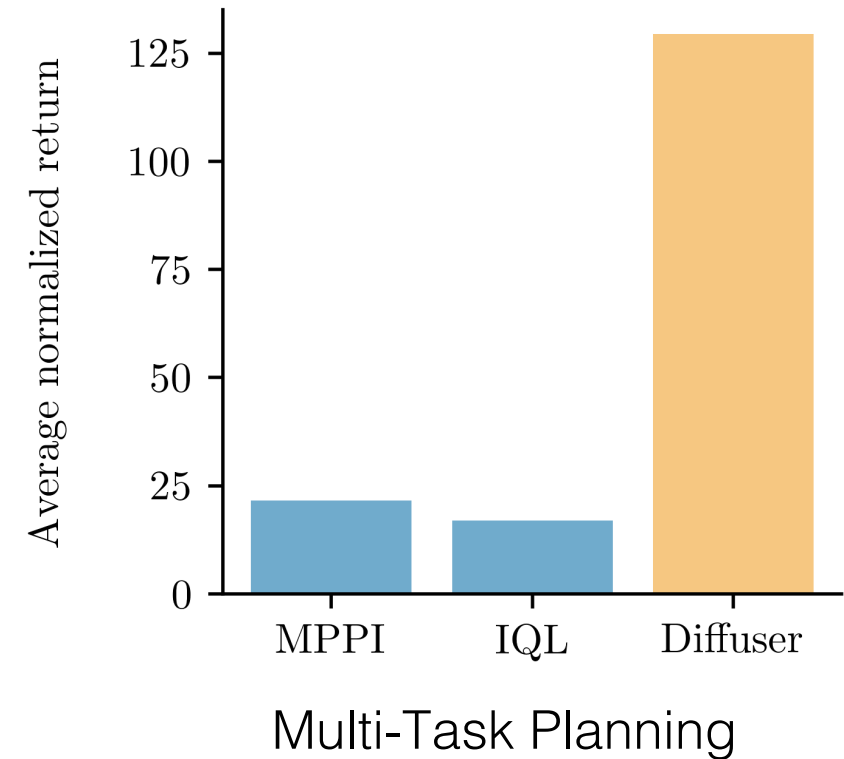
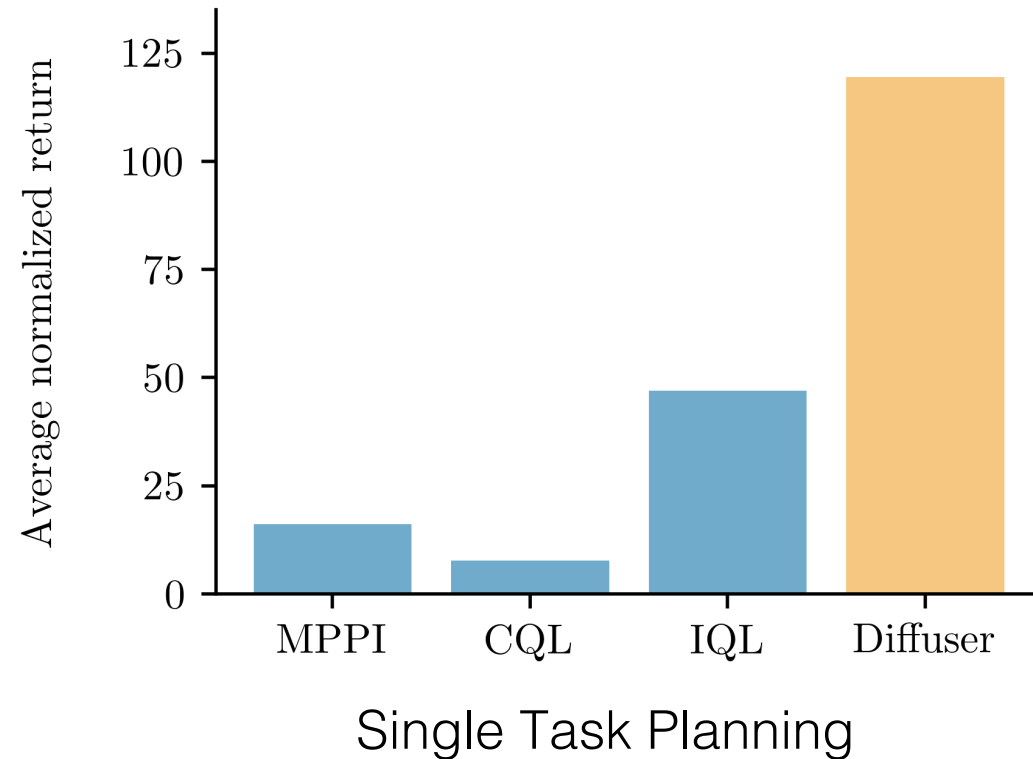
The trajectory model supplies the structure used to connect them.

[Explain first](#) · [click to start](#)

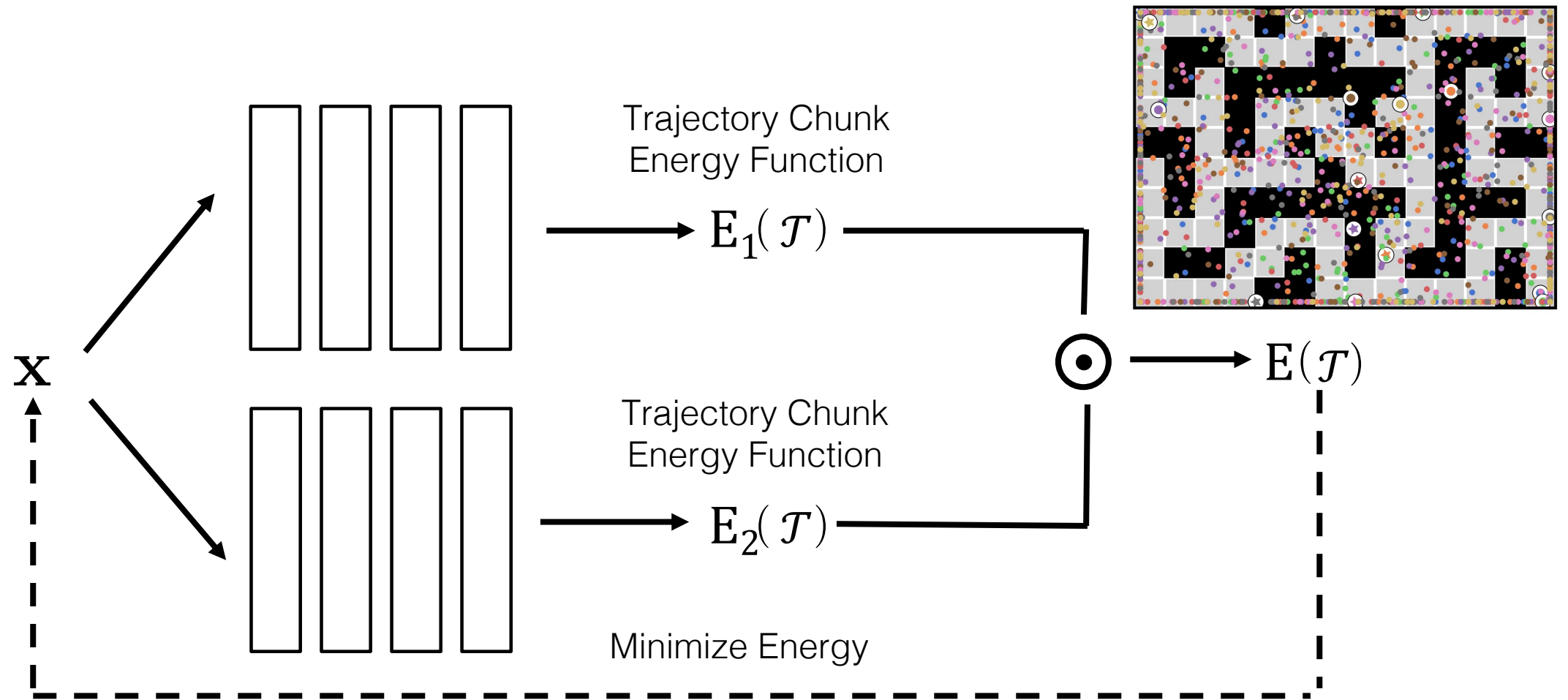
[Play / replay](#)

# Reusing One Model across Goal-Conditioned Tasks

Baselines require per task retraining  
while our trained model can be applied  
across tasks!



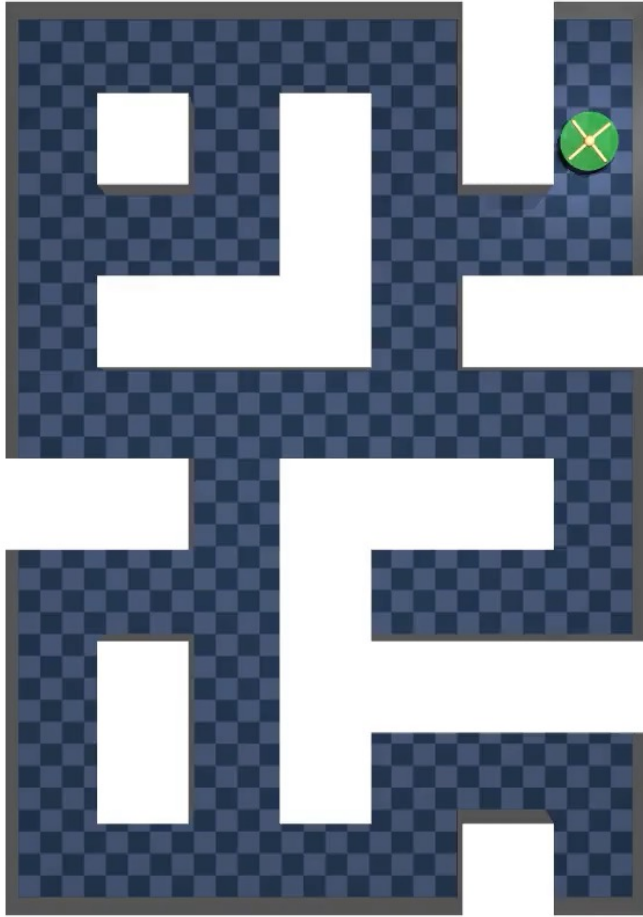
# Composing Short Trajectories into a Longer Plan



Explain first · click to start

Play / replay

# What the Trajectory Model Was Trained On



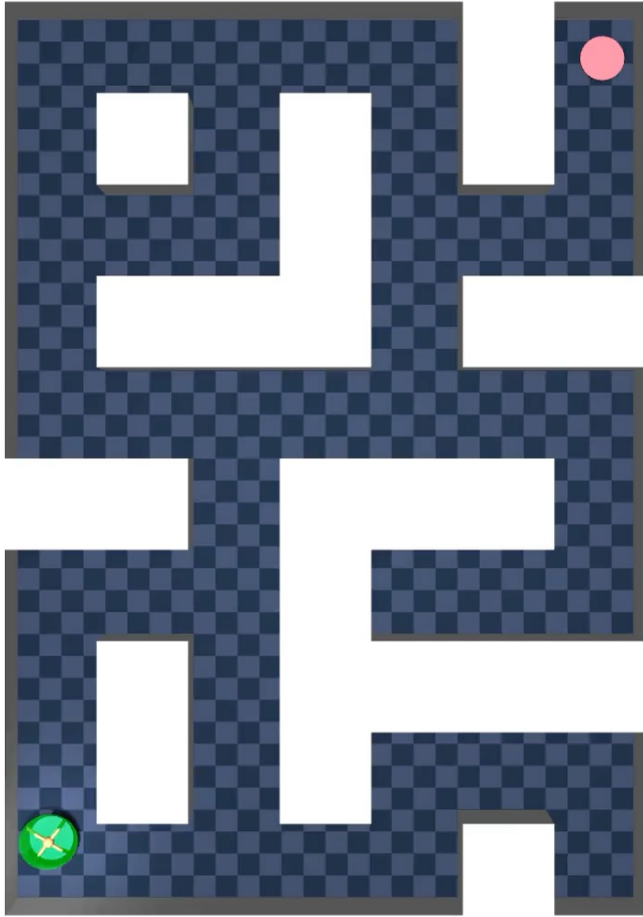
Offline trajectory data

The model learns local trajectory structure from offline data.  
Inference constructs the complete route for the current task.

Explain first · click to start

[Play / replay](#)

# A New Complete Route from Learned Trajectory Chunks



Generated plan and execution

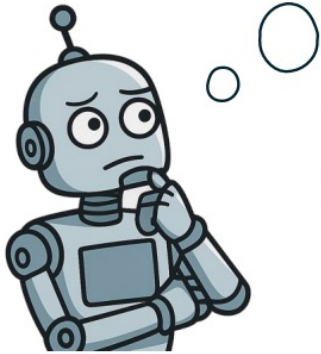
Optimize the chunks jointly so their overlapping states agree.  
This constructs a complete route from reusable local motion.

[Explain first · click to start](#)

[Play / replay](#)

# A Generated Video Can Specify a Robot Plan

Please pull a piece of tape



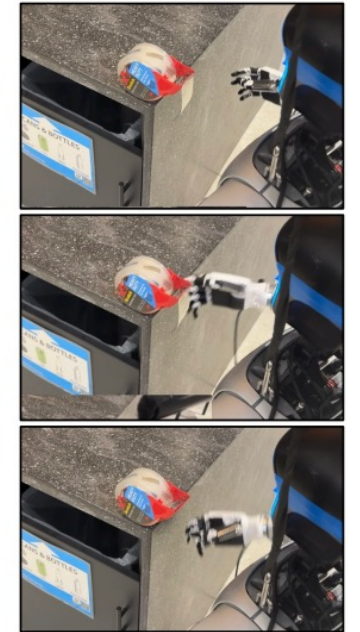
Generated videos from Large Video Planner



↓ Retarget to robot hand



Robot execution



Generate the motion needed for the current instruction.  
Retarget the visual plan to the robot for execution.

# The Same Scene, a Different Goal



Generated video plan  
Goal: put the can on the  
green cube.

Put Coke Can on Green Cube

Explain first · click to start

[Play / replay](#)

# The Same Scene, a Different Goal



Generated video plan  
Goal: put the green cube  
on the can.

Put Green Cube on Coke Can

Explain first · click to start

[Play / replay](#)

# Model Predictive Control: Propose Candidate Actions

## 1. Propose actions

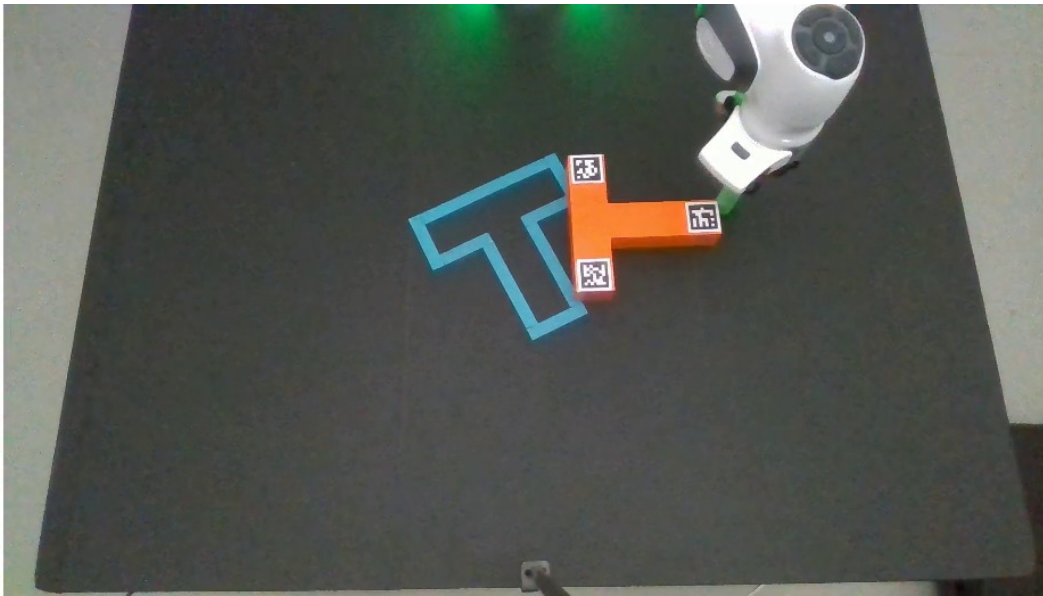
## 2. Predict outcomes

## 3. Select and execute

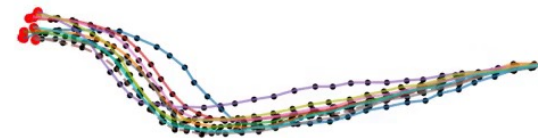
$$\pi(I_t) = a_{t:t+T}^{(k^*)}, \quad k^* \in \arg \max_{k=1, \dots, K} \mathcal{R}\left(\mathcal{W}(I_t, a_{t:t+T}^{(k)})\right)$$

$\{a_{t:t+T}^{(1)}, \dots, a_{t:t+T}^{(K)}\}$  are candidate action chunks.

Current observation



Candidate action chunks



The learned policy supplies several candidate action chunks.

# Model Predictive Control: Predict Each Outcome

1. Propose actions

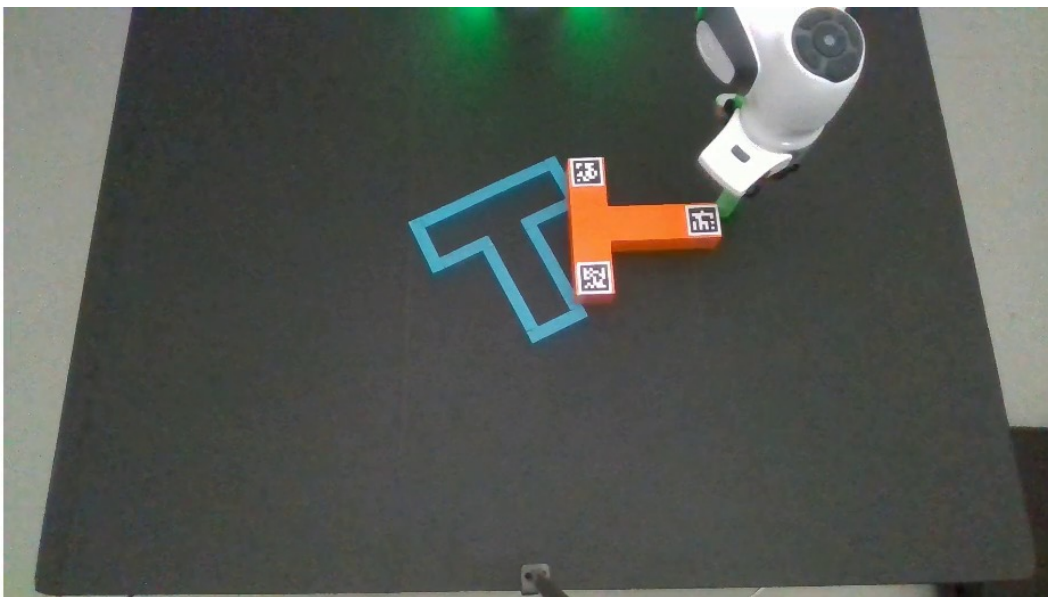
2. Predict outcomes

3. Select and execute

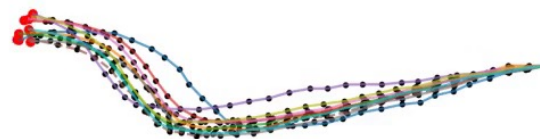
$$\pi(I_t) = a_{t:t+T}^{(k^*)}, \quad k^* \in \arg \max_{k=1, \dots, K} \mathcal{R}\left(\mathcal{W}(I_t, a_{t:t+T}^{(k)})\right)$$

$\mathcal{W}(I_t, a_{t:t+T}^{(k)})$  predicts the outcome of candidate  $k$ .

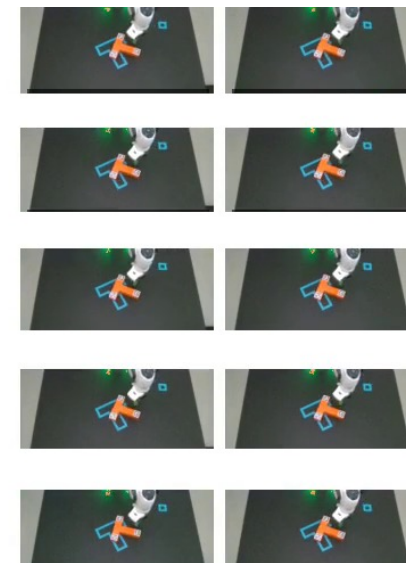
Current observation



Candidate action chunks



Predicted outcomes



For each proposed action chunk, predict the resulting scene.

# Model Predictive Control: Select the Best Predicted Outcome

1. Propose actions

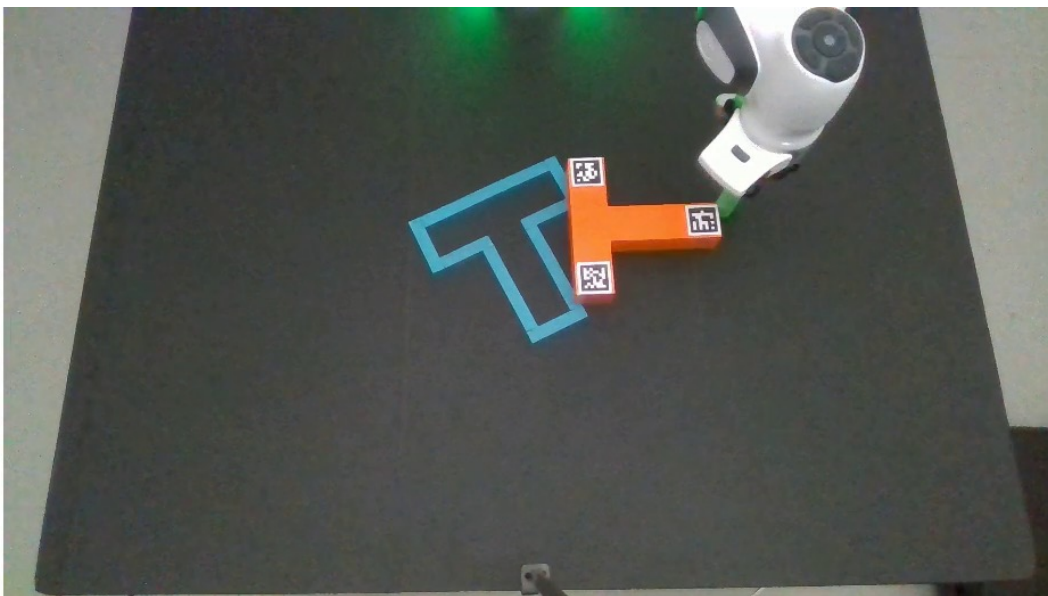
2. Predict outcomes

3. Select and execute

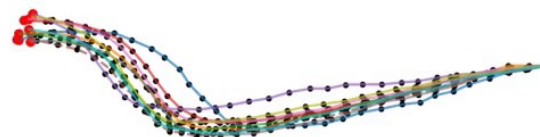
$$\pi(I_t) = a_{t:t+T}^{(k^*)}, \quad k^* \in \arg \max_{k=1, \dots, K} \mathcal{R}(\mathcal{W}(I_t, a_{t:t+T}^{(k)}))$$

$k^*$  indexes the candidate with highest predicted reward.

Current observation



Candidate action chunks



Predicted outcomes + scores



Score each predicted scene for this goal and execute the highest-scoring chunk.

# Read One Decision Numerically

One frozen frame from the demonstration



Top-right candidate: predicted reward  $-0.614$ .

The highest score selects this action chunk.

$$k^* = \arg \max_{k=1, \dots, 10} \mathcal{R}_k = 2$$

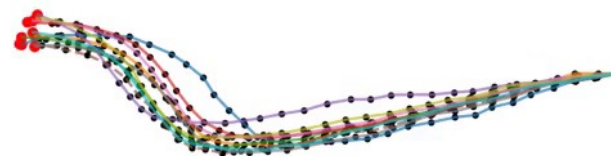
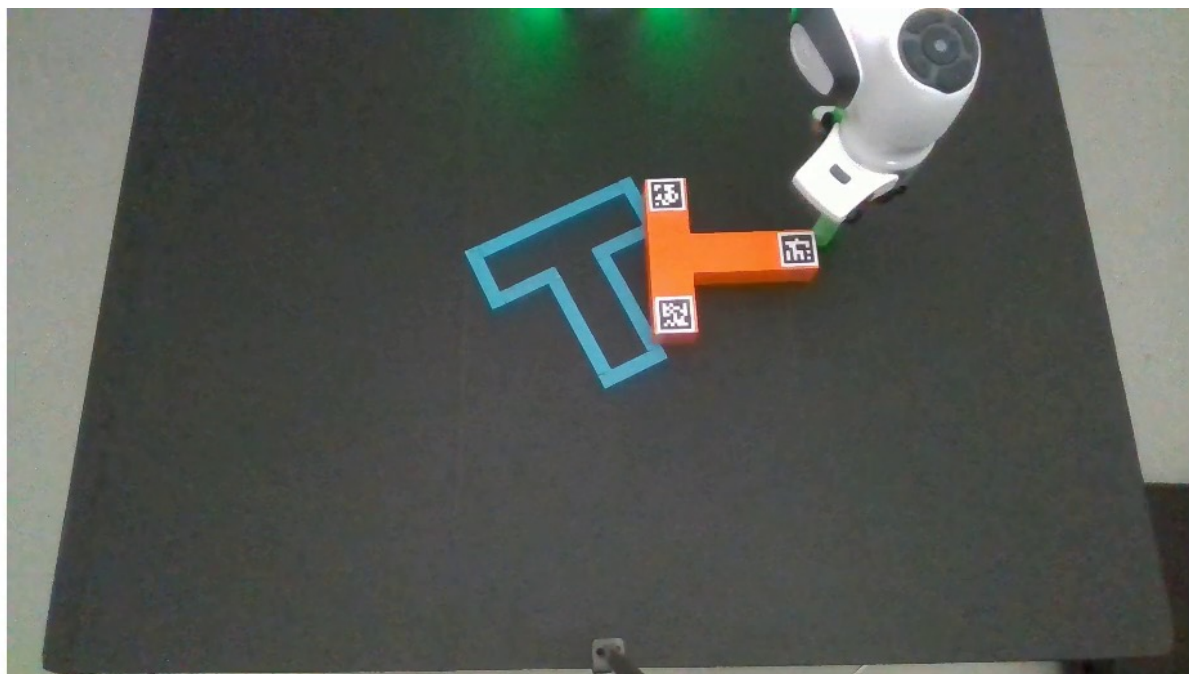
$$\mathcal{R}_2 = -0.614 > -0.625 > -0.637 > \dots$$

$$\pi(I_t) = a_{t:t+T}^{(2)}$$

Planning converts multiple policy proposals into one selected action.

# Watch the Propose–Predict–Select Loop

First identify the action candidates, predicted outcomes, and selected score. Then play.



$$\pi(I_t) = a_{t:t+T}^{(k^*)}, \quad k^* \in \arg \max_{k=1, \dots, K} \mathcal{R}(\mathcal{W}(I_t, a_{t:t+T}^{(k)}))$$

Prediction guides action. New observations restart the loop.

Explain first · click to start

Play / replay

# Predictive Control in Physical Manipulation



The policy proposes candidate actions.  
The world model predicts their effects.  
A goal score selects the next action.

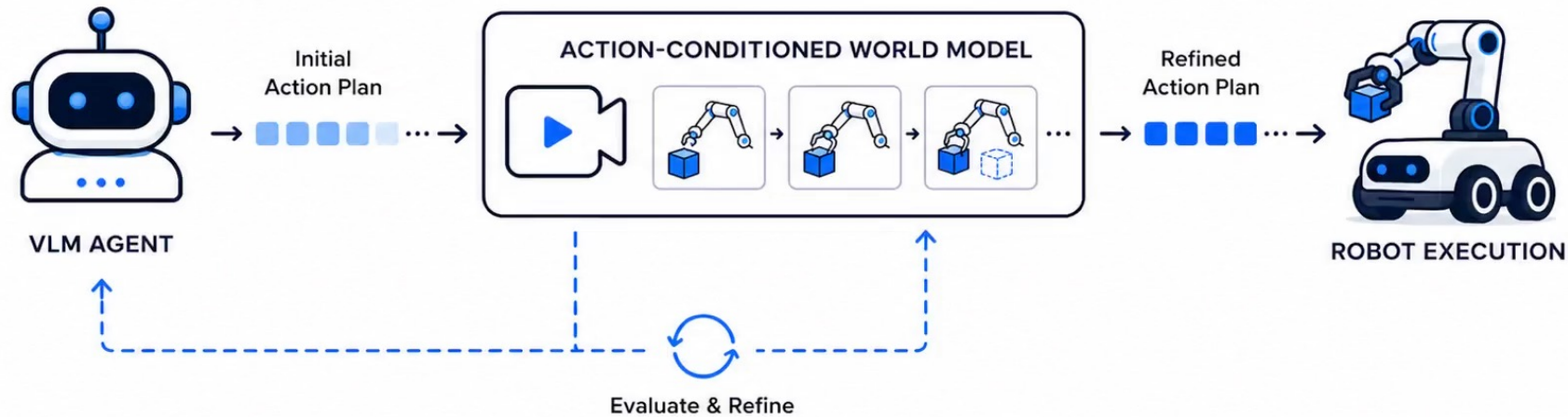
Compare predictions, act, then replan from the next observation.

Explain first · click to start

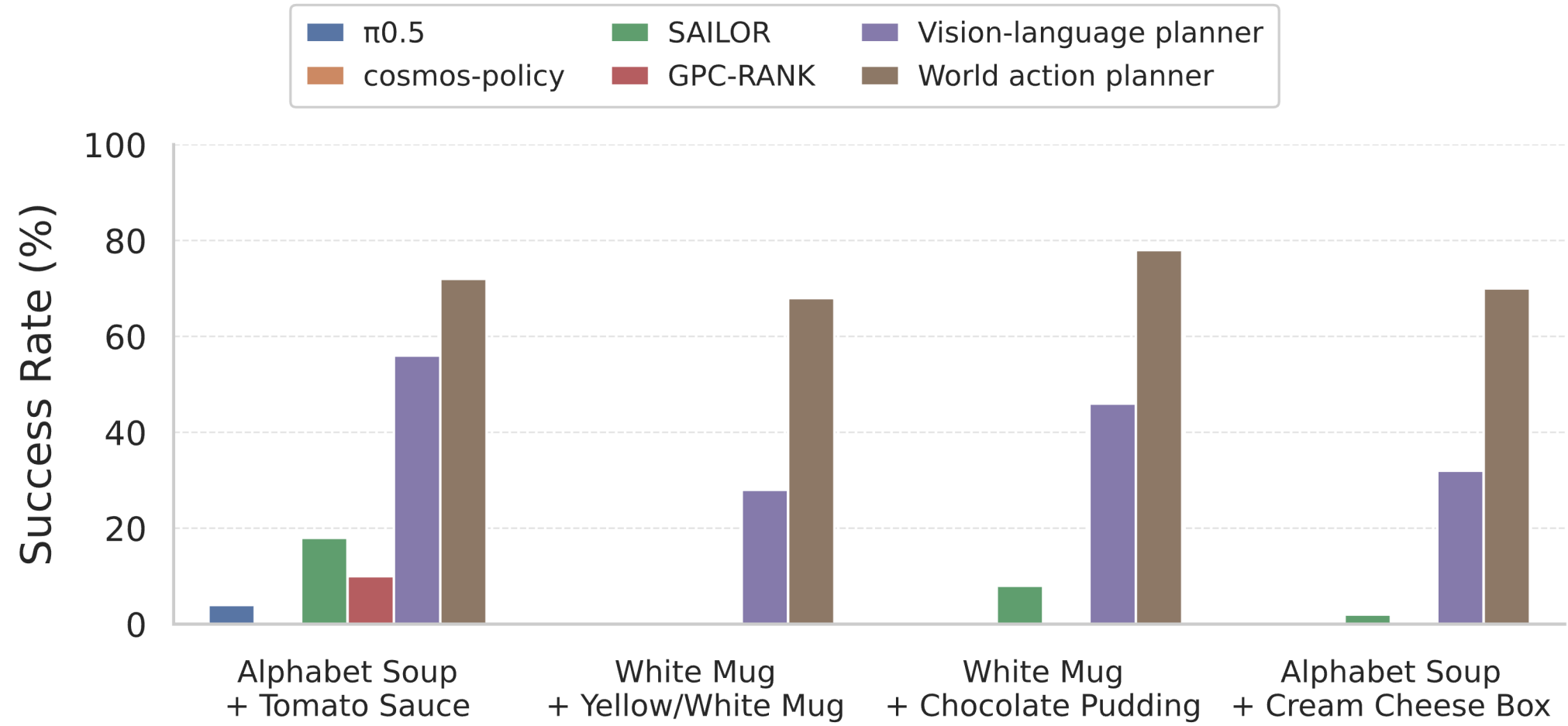
[Play / replay](#)

# World Action Planner

## Generalizable Robot Decision-Making

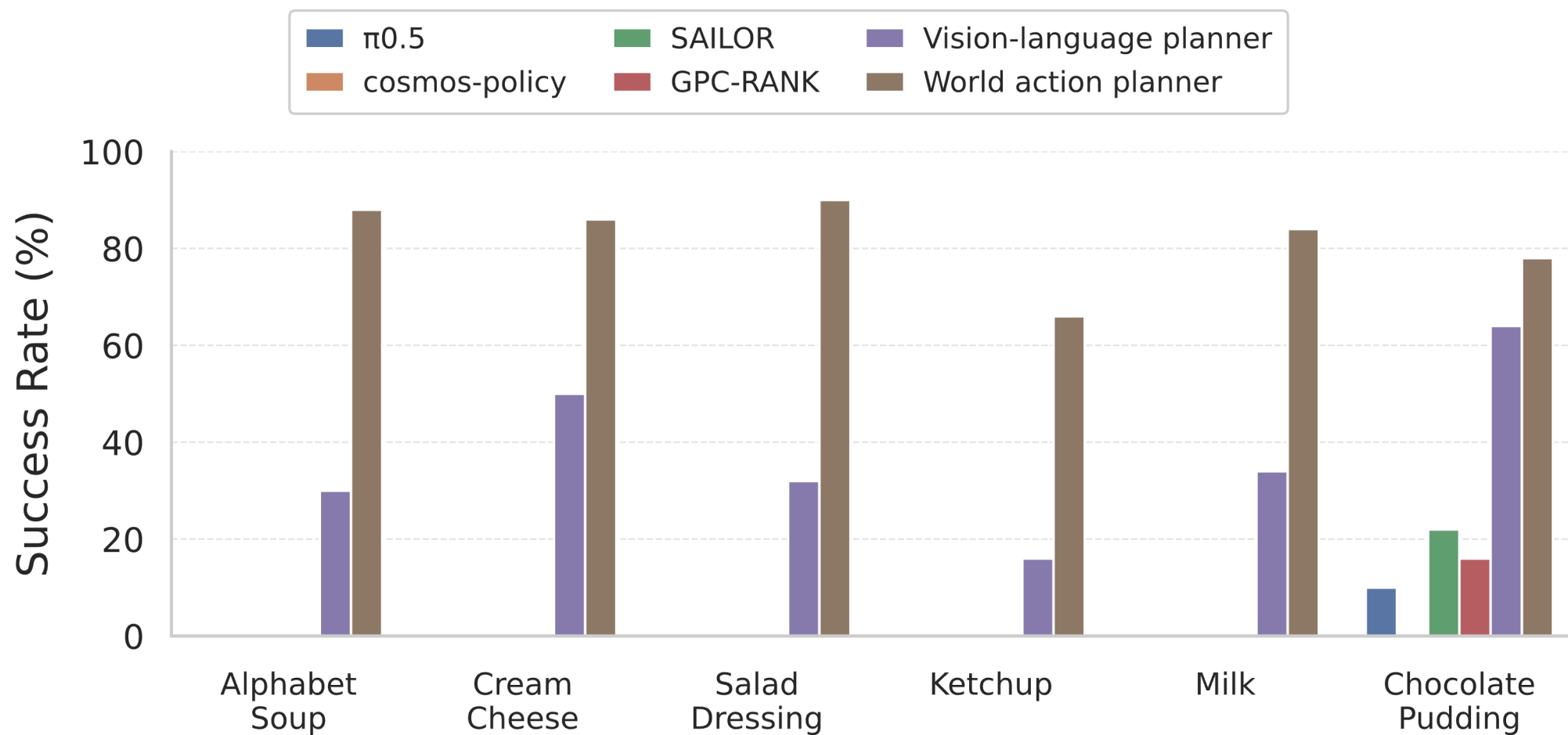


# World Action Planner: New Task Combinations



Combine familiar subtasks into a new objective.

# World Action Planner: New Scene Layouts



Keep the task, but change the arrangement of the scene.

# Reusable Models Support New Plans

Learned structure

Inference

Trajectory structure and action  
consequences



Construct and compare plans  
for this task

What the examples establish

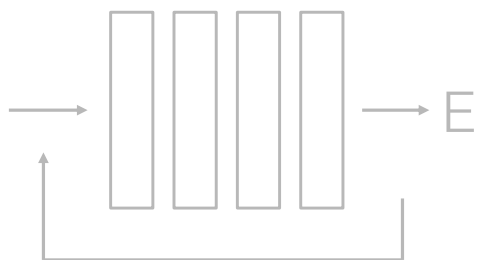
Reuse learned models under a new goal, along a new route, or to recover from  
a stalled action.

Next: can inference also coordinate who proposes and evaluates?

The models supply reusable knowledge. Inference decides how to act in the current situation.

# From Learned Models to New Behavior

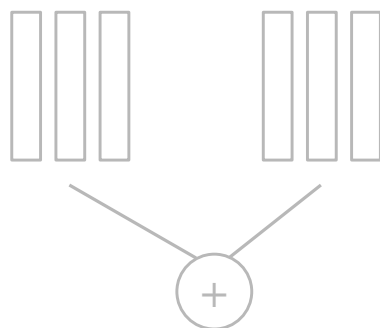
Generative  
inference



Learn a solution  
space.

Sample and refine  
a candidate.

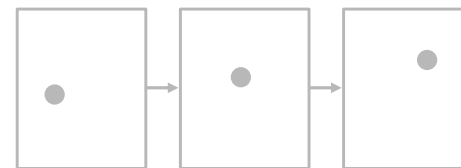
Compositional  
inference



Learn reusable  
relations.

Compose and satisfy  
them together.

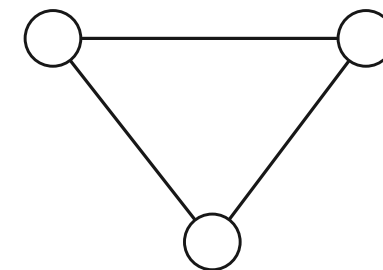
World models  
and planning



Learn possible  
futures.

Search for behavior  
under a new goal.

Multi-agent  
intelligence



Reuse learned  
capabilities.

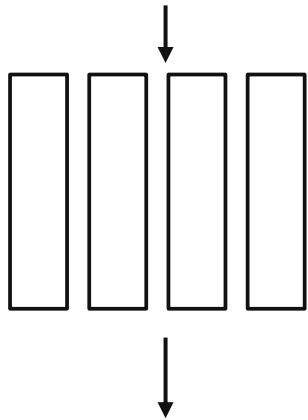
Communicate,  
evaluate,  
and choose who acts.

Can inference also decide how learned capabilities work together?

# Organizing Learned Capabilities through Interaction

## One model

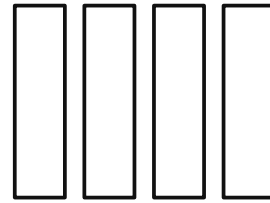
Observation and task



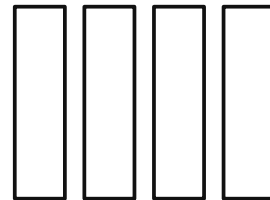
Response or action

## A coordinated set of capabilities

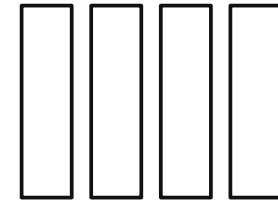
Language model



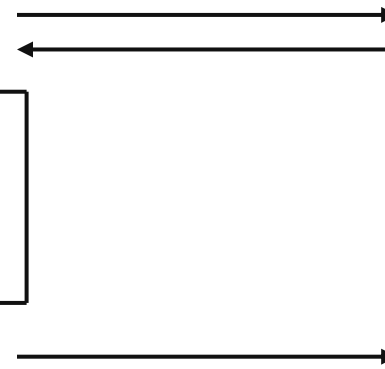
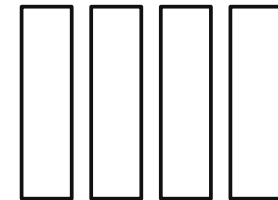
Video model



Language model



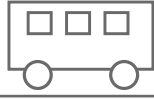
Action model



Agents can use different models or different reasoning paths from one model.

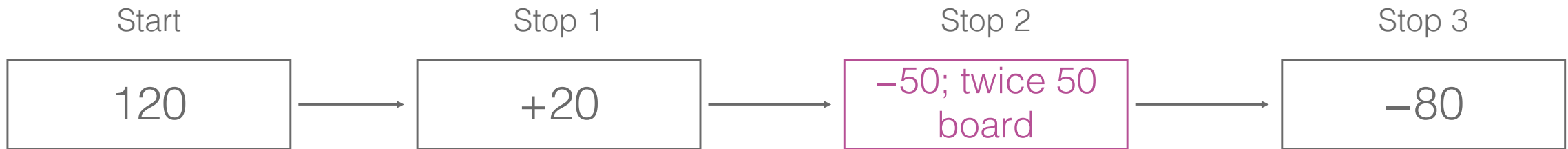
Communication brings their contributions into the same task.

# Debate Starts with Independent Reasoning



A train starts with 120 people. At each stop, passengers enter or leave.

**How many people remain after the third stop?**



Agent 1 — abbreviated initial response

After stop 2: 165  
After stop 3: 85

**85**

Agent 2 — abbreviated initial response

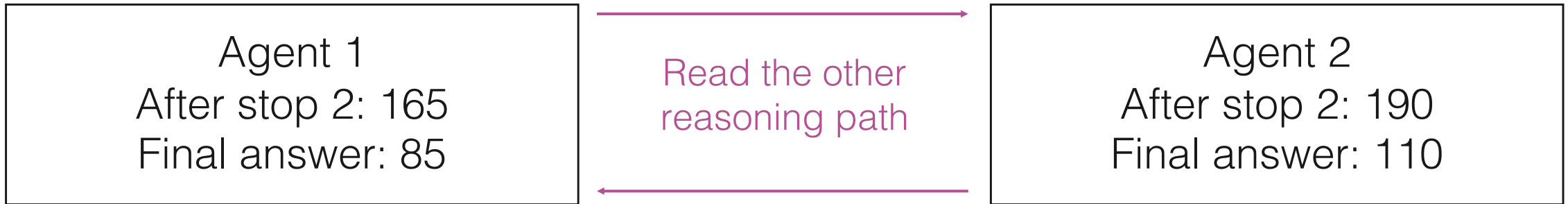
After stop 2: 190  
After stop 3: 110

**110**

Independent reasoning exposes a specific intermediate disagreement.

# Exchange Arguments, Not Just Answers

The disputed step is what happens at the second stop.



Check the same claim against the original task

50 leave and 100 board: a net gain of 50.

$120 + 20 - 50 + 100 = 190$  after the second stop

Use the exchanged reasoning to locate and correct the disputed step.

# Revise the Reasoning and Recompute

The correction changes an intermediate step, then the final answer.

$$120 + 20 - 50 + 100 - 80 = 110$$

Agent 1  
 $85 \rightarrow 110$

Agent 2  
 $110 \rightarrow 110$

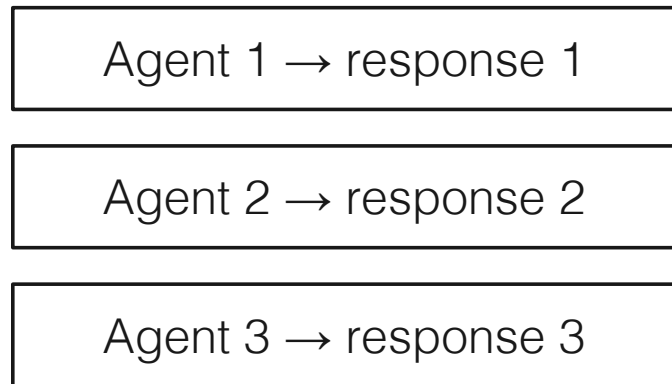
A response can contribute a useful intermediate step.

Another agent can use that step to repair its own solution.

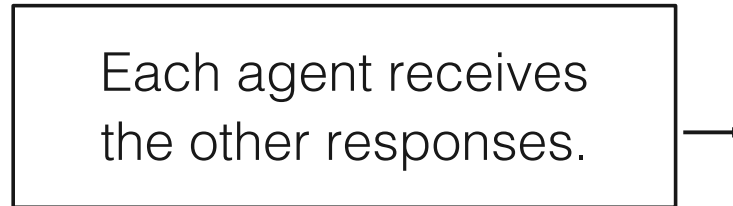
# Debate: Propose, Exchange, Then Revise

Each agent keeps its model fixed and updates its own response.

1. Propose independently



2. Exchange responses



3. Inspect and revise

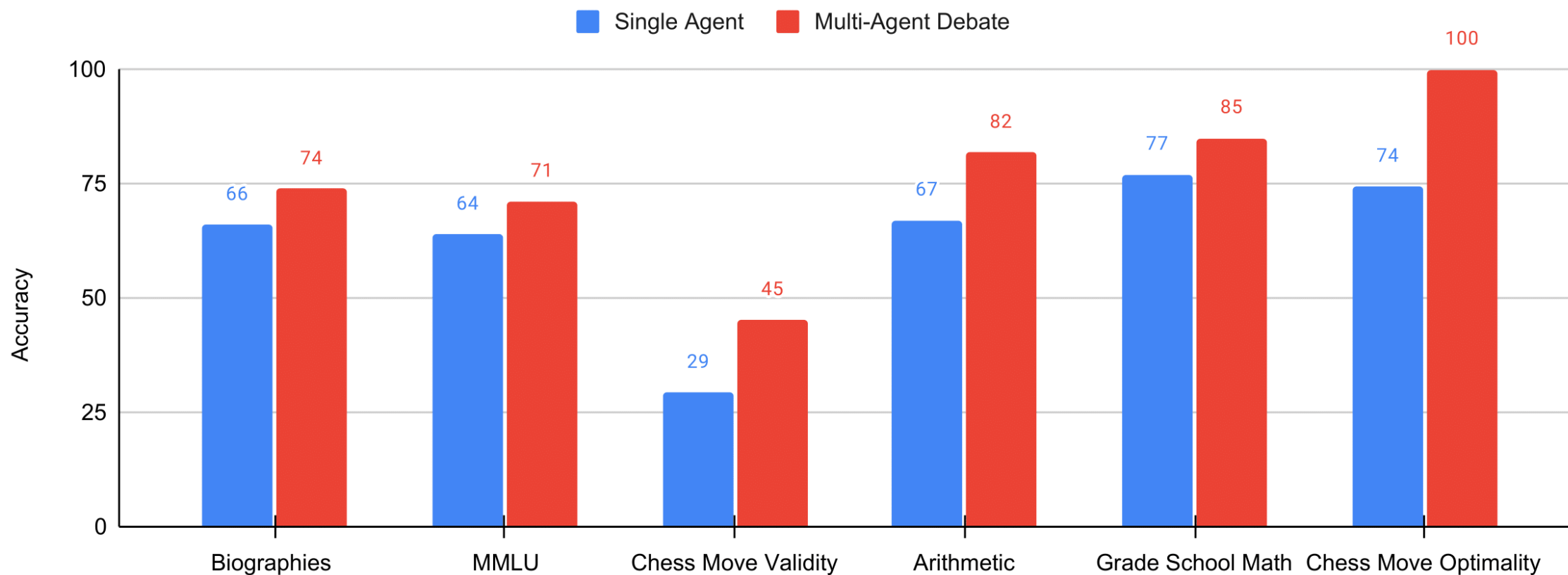
Inspect the disputed step.  
Check the reasoning.  
Produce a revised answer.

Arrows point from the central box in step 2 to these three lines of text.

Each round exposes new reasoning for the other agents to use.

The group refines its proposals before choosing a final answer.

# Evaluating Debate across Tasks

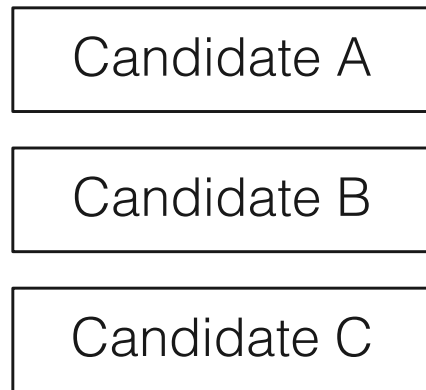


Agents read one another's reasoning and revise their responses.  
The reported results show gains across several kinds of tasks.

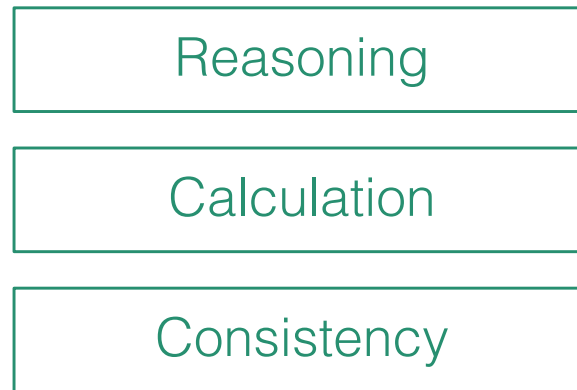
# Verification Selects among Proposed Answers

Debate revises answers. Verification selects among them.

Candidate answers



For each candidate



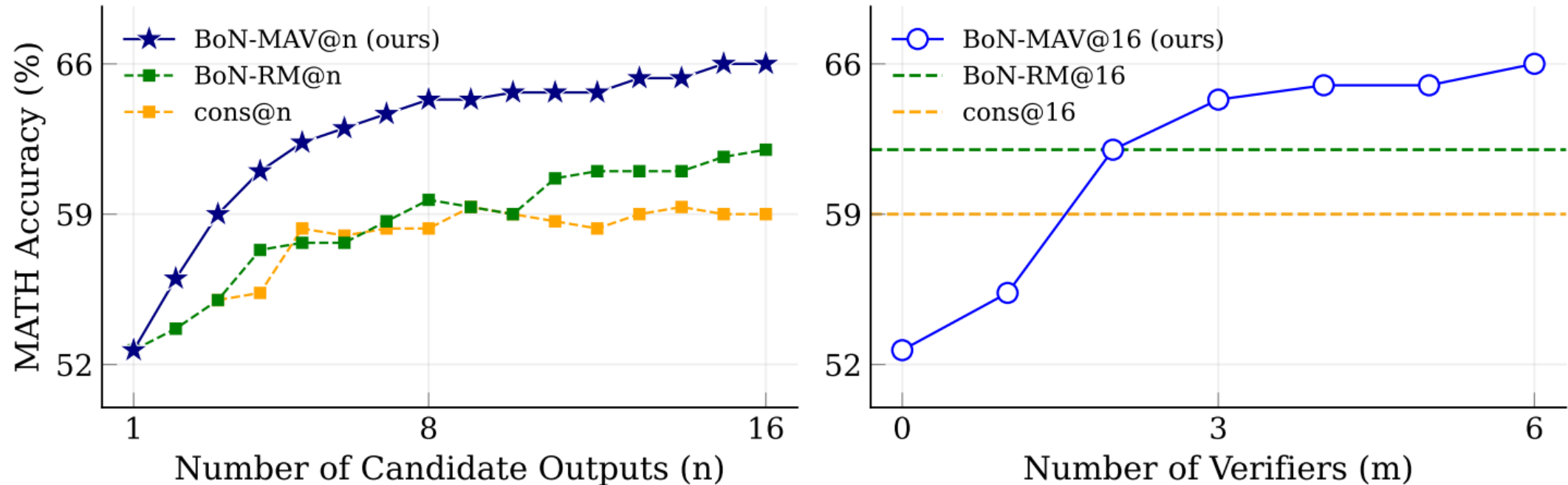
Aggregate judgments



The same answer is checked from multiple perspectives.

Combine checks of reasoning, calculation, and consistency to select an answer.

# Scaling Candidates and Verifiers Separately

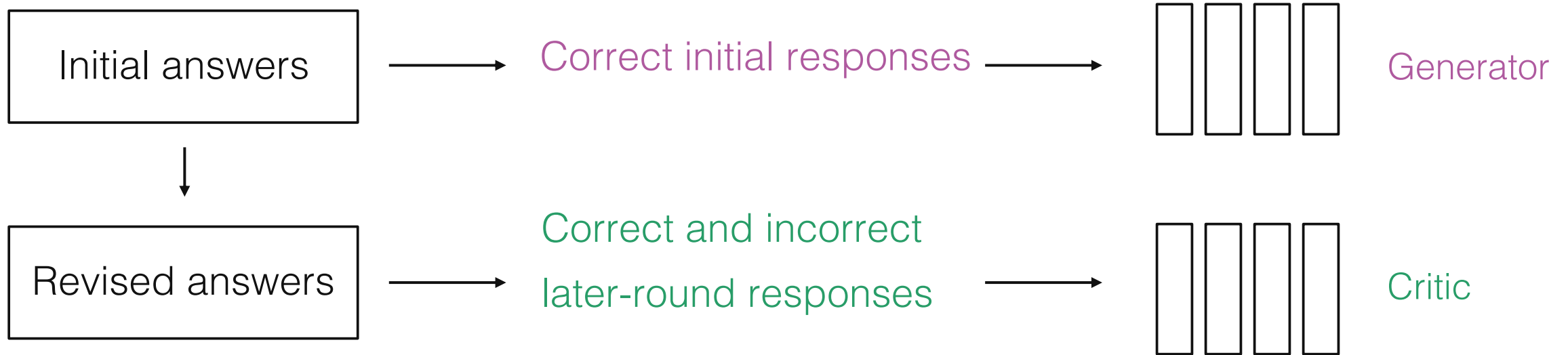


More candidates expand the set of proposed solutions.

More verifiers add checks to guide the selection.

# Learning Better Generators and Critics from Interaction

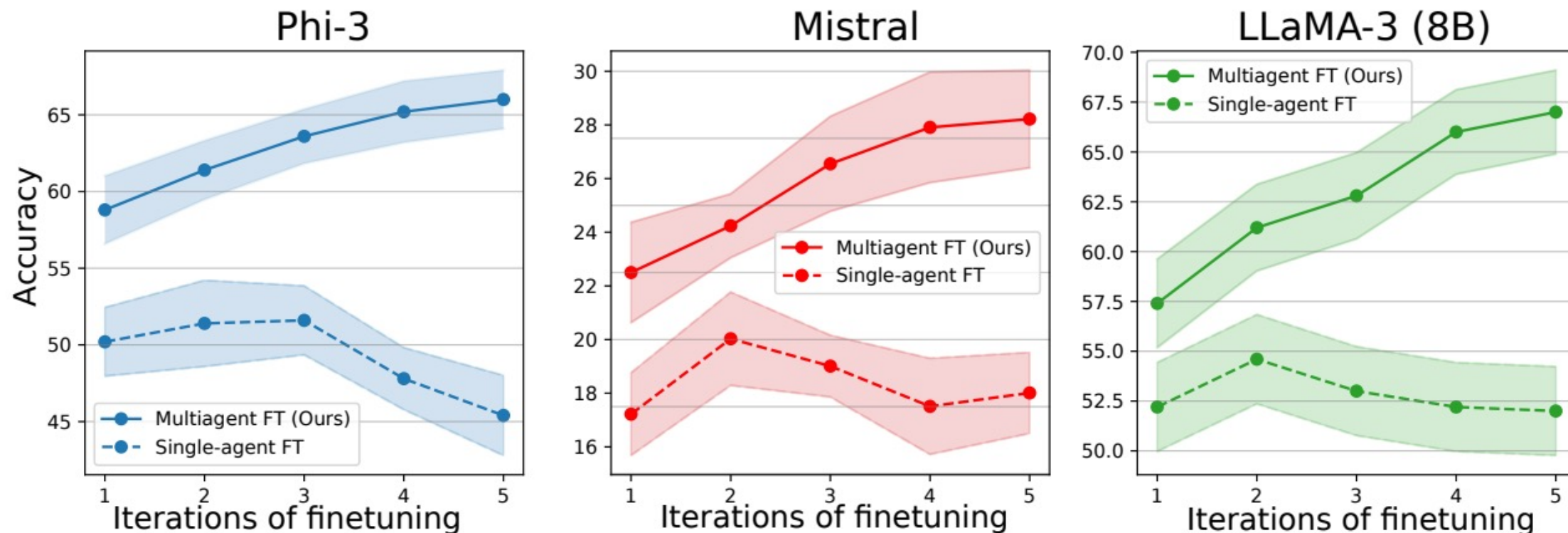
Debate traces



The roles learn from different parts of the interaction.

Generators improve proposals. Critics improve revisions in later rounds.

# Self-Improvement over Rounds of Finetuning

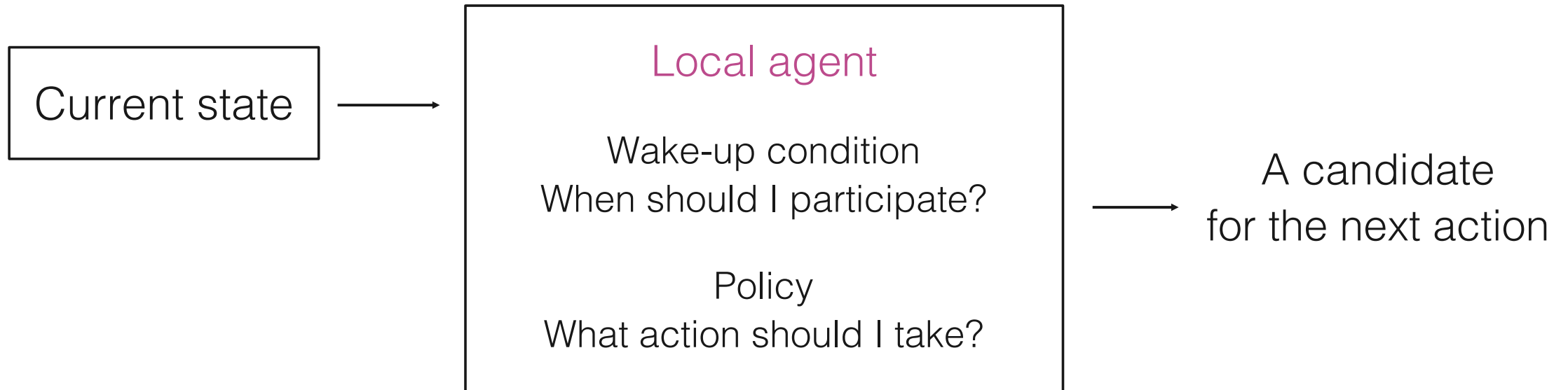


Successful proposals train the generators.

Revisions train the critics to improve later rounds.

# An Agent Economy Decides Who Acts Next

Choose the next actor from the current state, allowing the workflow to change.

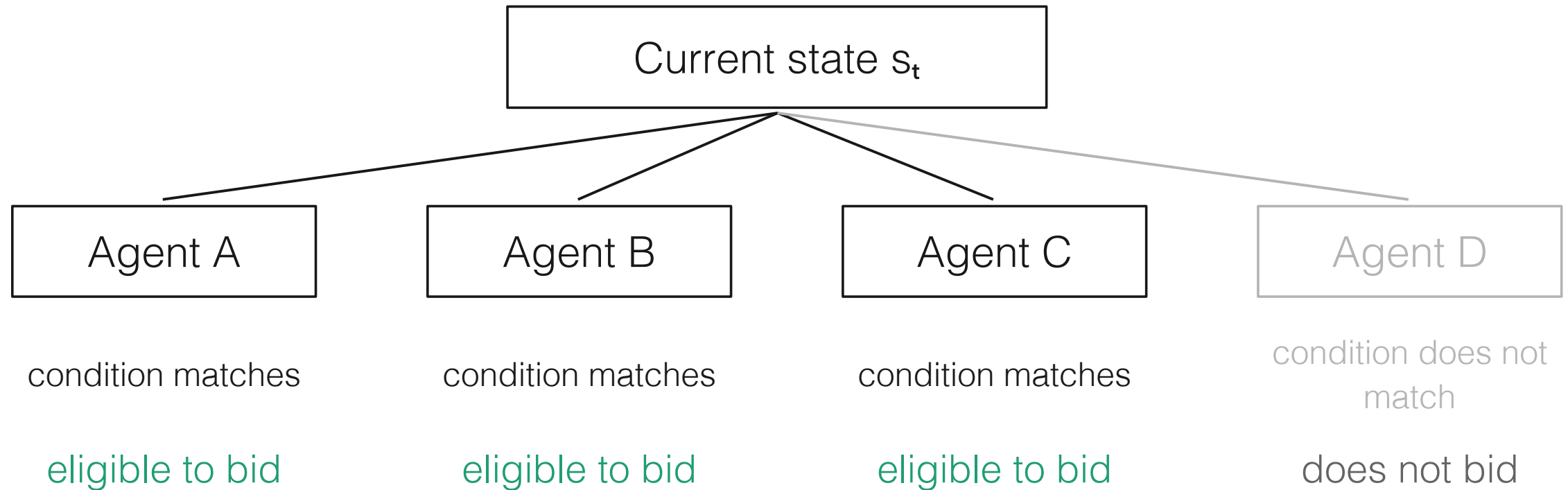


The economic rules coordinate local capabilities.

The next step can be chosen after seeing the current state.

# First Determine Which Agents Are Eligible

An agent can bid when its wake-up condition matches the current state.



Only eligible agents enter this auction.

## Then Choose the Highest Eligible Bid

Illustrative auction: A, B, and C are eligible.

Agent A

$$b_A = 1.0$$

Agent B

$$b_B = 1.2$$

Agent C

$$b_C = 1.5$$

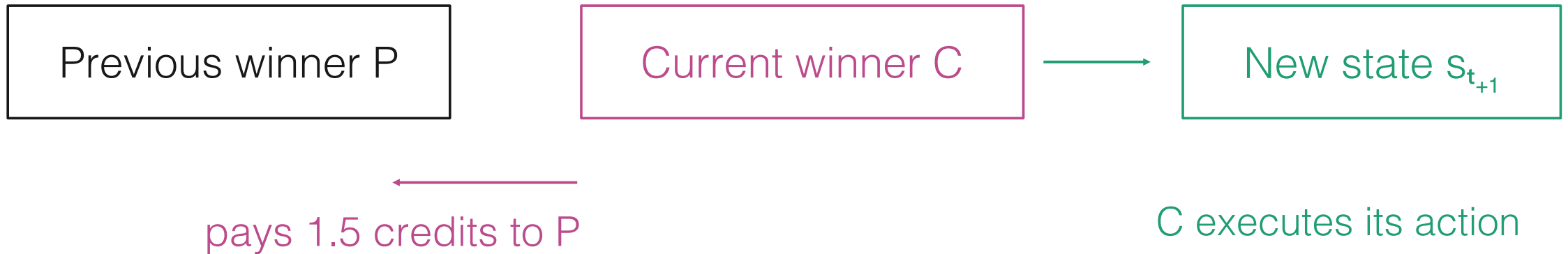
wins the right to act

$$i_t = \arg \max_{i \in \mathcal{A}(s_t)} b_i(s_t) = C$$

The current state determines who can bid; the auction determines who acts next.

# The Winner Pays Its Predecessor, Then Acts

Transactions and actions travel in different directions.

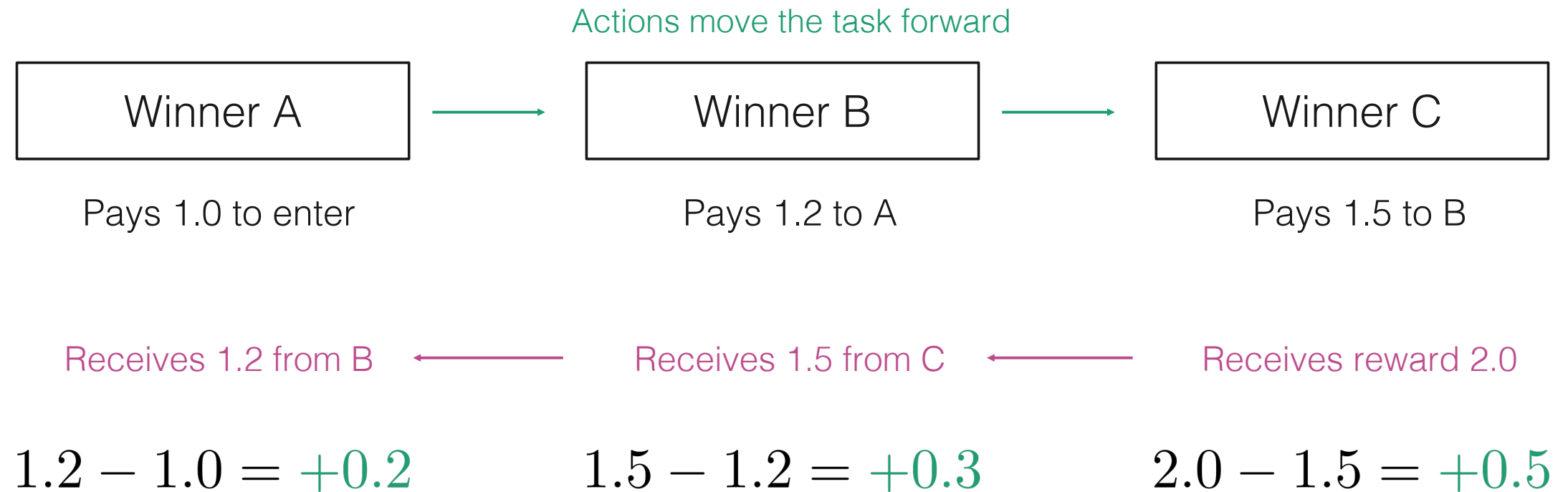


The next winner will pay C.  
At termination, the environment pays the final actor.

The payment rewards the predecessor for leaving a useful state.

# Profit Links Each Action to What Follows

An illustrative three-step episode, in credits—not an experimental result.



Profit = received payment or terminal reward – own bid

An early step earns credit when another agent can build on it.

# Across Tasks, Wealth Changes the Available Agents

The learning module changes the population—not just the next action.

## Exploitation

---

Wealthy agents remain.  
Propose mutations of their  
behavior.

## Exploration

---

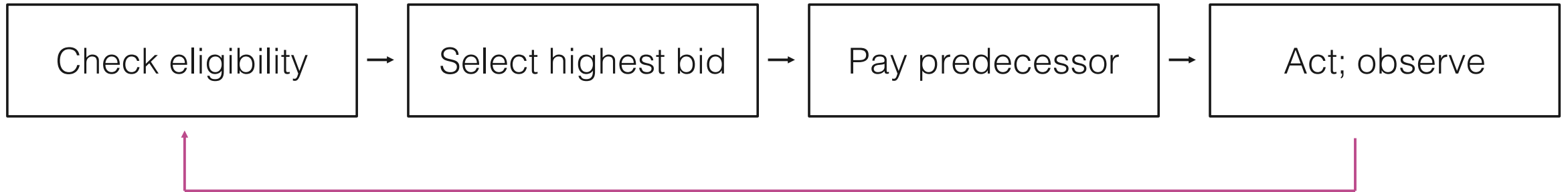
Propose new agents.  
Amend unsuccessful agents and  
reintroduce them.

Bankrupt agents are removed; amended variants can rejoin.

Repeated tasks change which local capabilities are available to the next auction.

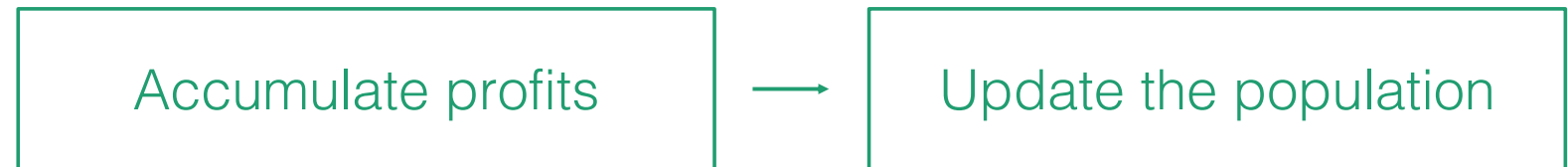
# Separate Inference Within a Task from Learning Across Tasks

Within one task



Repeat from the new state

Across tasks



The auction chooses the next actor. Experience changes who can act in the future.

# Partial Specialists Can Form a Stronger System

## Finance-Agent-Bench

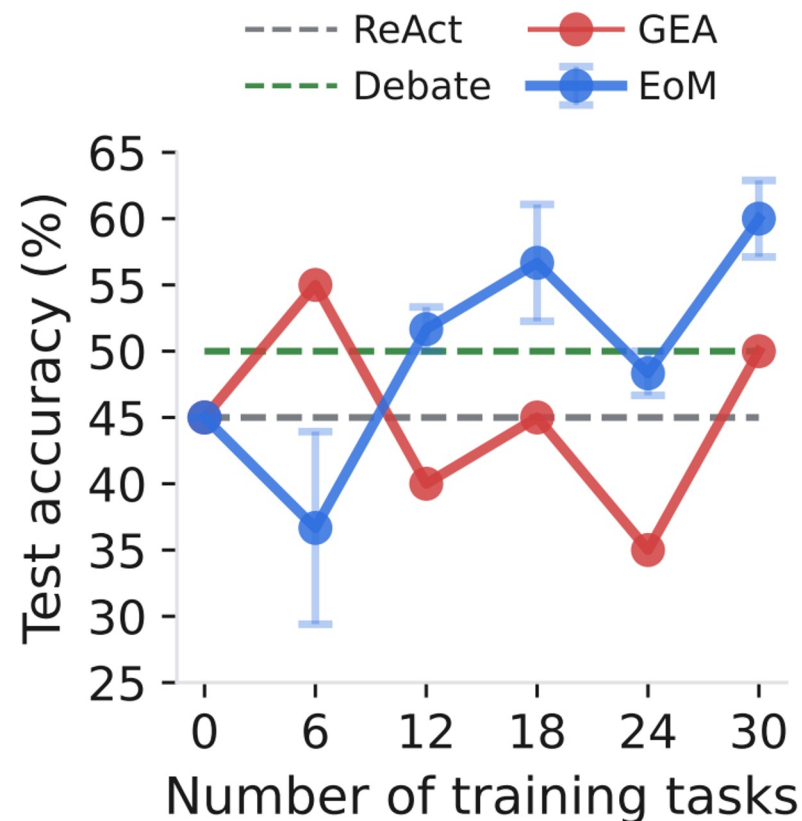
Each agent has one of the four tools needed to complete the task.

45% → 60%

Accuracy after 30 training tasks

At the reported endpoint:

Debate 50% · ReAct 45% · GEA 50%



# What Changes in the Organization?

An example from mathematical reasoning (MATH500)

| Training             | Example trace                                                           | Test accuracy |
|----------------------|-------------------------------------------------------------------------|---------------|
| 5 tasks<br>31 steps  | Planner → Executor → Verifier → ... → Executor                          | 14%           |
| 45 tasks<br>16 steps | Planner → Executor → Planner → ... → Executor                           | 25%           |
| 55 tasks<br>8 steps  | Planner → Verifier → Planner → ... → Executor                           | 32%           |
| 100 tasks<br>7 steps | Planner → Executor → Planner → Verifier → Planner → Executor → Verifier | 43%           |

The system learns which reasoning steps to use and how to combine them.

Trace lengths are examples; percentages are the reported test accuracy.

# Inference Can Also Organize the Reasoning Process

## Learned structure

Local capabilities for  
proposing, checking, and acting



## Inference

Compose their contributions  
into a solution

What can be constructed

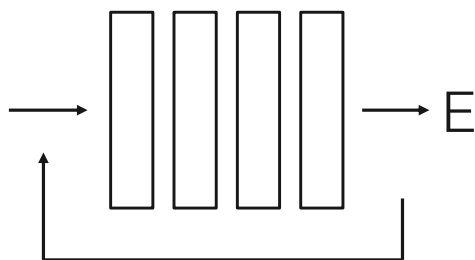
Debate revises a reasoning path.  
Verification selects from alternative answers.

Agent economies assemble and improve a workflow.

The constructed object can be an answer, a plan, or a sequence of specialists.

# From Learned Models to New Behavior

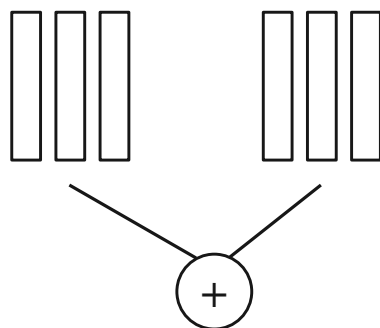
Generative  
inference



Learn a solution  
space.

Sample and refine  
a candidate.

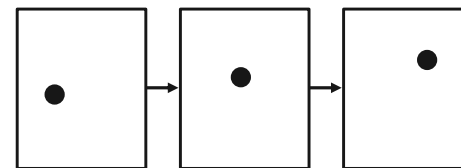
Compositional  
inference



Learn reusable  
relations.

Compose and satisfy  
them together.

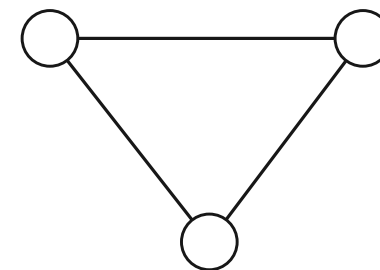
World models  
and planning



Learn possible  
futures.

Search for behavior  
under a new goal.

Multi-agent  
intelligence



Reuse learned  
capabilities.

Communicate,  
evaluate,  
and choose who acts.

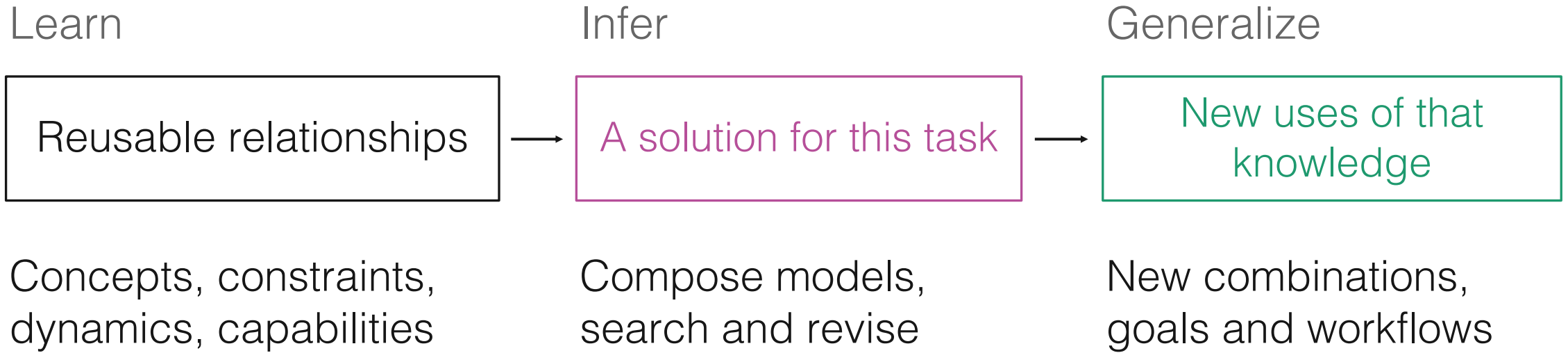
What is learned? What remains open to inference?

# What Can Be Constructed after Training?

|                    |                                                                                  |
|--------------------|----------------------------------------------------------------------------------|
| New specifications | Combine familiar concepts, goals, and constraints into a new task specification. |
| Larger problems    | Reuse local relations across more objects or variables.                          |
| New workflows      | Coordinate specialized capabilities around the current task.                     |

The complete solution can be new even when its learned components are familiar.

# Generalization by Construction



Learning does not need to directly generate the solution.  
It should provide structure from which new solutions can be found.

# Learn Reusable Structure. Infer New Solutions.

What should models represent so that inference  
can solve problems not anticipated during training?

Yilun Du

Harvard University



**HARVARD**  
UNIVERSITY

[yilundu.github.io](https://yilundu.github.io)



**Kempner**  
INSTITUTE