Causal Inductive Synthesis Corpus

Zenna Tavares [1], Ria Das [1], Elizabeth Weeks [1], Kate Lin [1, 2], Joshua B. Tenenbaum [1], Armando Solar-Lezama [1]

[1]MIT, [2]Wellesley College

Motivation

Motivation Young children engage in forms of intuitive scientific discovery, building structured causal theories of their environment using many of the principles that underpin professional science [1, 2]. Despite progress in modeling many of these ideas, automatic discovery of models of realistic phenomena from observation and interaction remains largely out of reach.

Objective Facilitate progress towards automatic scientific discovery, first by introducing a representation of causal models that is expressive enough to succinctly capture the complexities of real world phenomena, and second by presenting a corpus of domains and accompanying benchmark challenge.

Contribution

Causal Inductive Synthesis Corpus A manually constructed collection of interactive domains, which abstract core causal concepts present in real-world mechanisms and environments.

Autumn The Autumn language is a Turing-complete language for specifying causal probabilistic models. It allows succinct expression for models that vary dynamically through time, respond to external input, have internal state and memory, exhibit probabilistic non-determinism, and have complex causal dependencies between variables.

The Autumn Language

Autumn is a language for probabilistic causal models. It is designed to express models that vary as a function of time or external input. Many constructs are standard. These include the use of `=` for variable assignment, and giving variables types by declaring them as `x:Int`.

```
fib = init 0
      next if time == 1
           then 1
           else (prev fib) + (prev prev fib)
```

Sequences Values in an Autumn program represent sequences that vary with time. A value v at time t may be (i) time invariant, i.e., $v_t = c$ for some constant c, (ii) stateless and time varying, i.e, $v_t = f(t)$ for some function f, or (iii) stateful / recurrent sequences defined in terms of previous values, i.e., $v_t = f(v_{t-1})$. Sequences are defined using the **`init next`** construct, which defines an initial value for a variable and an update rule to be performed on subsequent time steps. Previous values of a variable can be accessed using the **`prev`** primitive; for example, **`prev x`** gives the value of x at time $t-1$.

Events Beyond **`init next`**, temporal events may also be specified using the construct `on`, with the pattern `on event intervention`. An `event` is any sequence of type `Bool`, and an `intervention` is a modification to a value.

Canonical Autumn Example

```
-- define Ants and Food
object Ant {(Cell 0 0 gray)}
object Food {(Cell 0 0 red)}

-- ants initially randomly placed
ants : List Ant
ants = init map Ant (randPositions GRID 6)
       next update (prev ants) nextAnt

-- food gets removed if ant lands on it
foods : List Food
foods = init ()
        next update (prev foods)
                    obj -> if nextTo obj (closest obj Ant)
                           then removeObj obj
                           else obj

-- add random food to grid on click
on clicked
   foods = addObj (prev foods)
                  (map Food (randPositions GRID 4))

-- move every ant to the closest food
nextAnt : (Ant -> Ant)
nextAnt ant = move ant (unitVec ant (closest ant Food)))
```

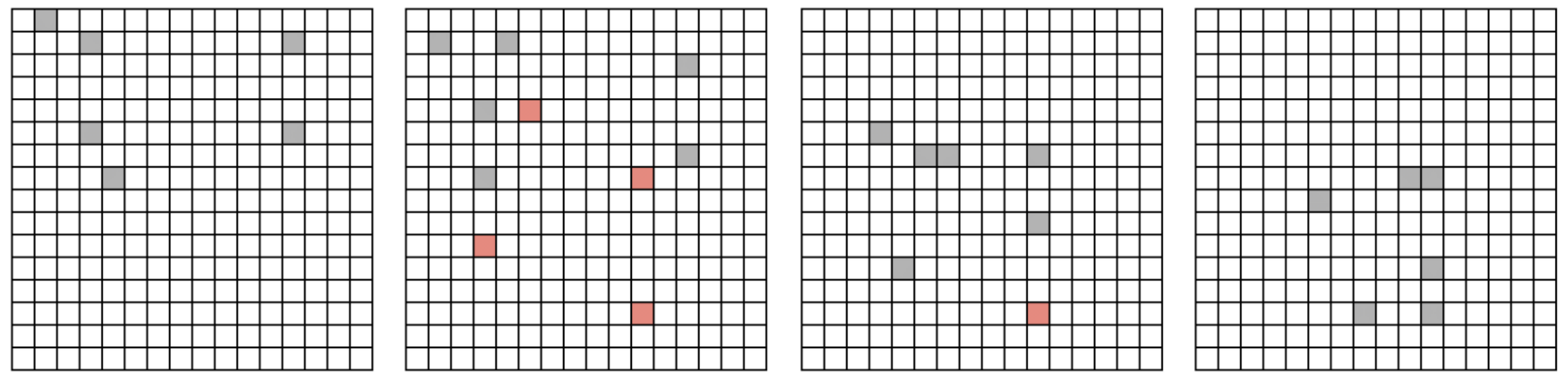

Figure 1: An Autumn program. This program simulates ants seeking food, starting at $t = 0$ on the leftmost grid. A number of ants (grey) are initially randomly positioned on the grid. On clicking, food (red) is placed at random positions on the board, and the ants move in the direction of the closest food item. The food disappears once an ant arrives at the same position. This process continues until all food items disappear.

The Causal Inductive Synthesis Corpus

We develop a corpus of interactive environments (CISC) using the Autumn language, a sampling of which are shown in Figure 2. We formalize CISC as well describe the two key synthesis challenges of passive and active discovery as follows.

Specification Let $\mathcal{L}$ denote the set of all Autumn models. CISC is a dataset $\mathcal{D} = (m_1, m_2, \ldots, m_N)$ of N Autumn models, i.e., $m_i \in \mathcal{L}$. For each model $m \in \mathcal{D}$, there is also a collection of M_m^{test} test trajectories $T_m^{\text{test}} = (\tau_1, \tau_2, \ldots, \tau_{M^m)}$ and M_e^{train} train trajectories $T_m^{\text{test}} = (\tau_1, \tau_2, \ldots, \tau_{M_m^{\text{train}}})$. A trajectory is a pair $(\mathbf{a}, \mathbf{o})$, where $\mathbf{a}$ and $\mathbf{o}$ are finite sequences (of identical length) of actions and observations respectively. The action space $\mathcal{A}$ allows for selecting a grid-cell, pressing an arrow, performing no action, or stopping a simulation. The observation space $\mathcal{O}$ is a colored grid of cells.

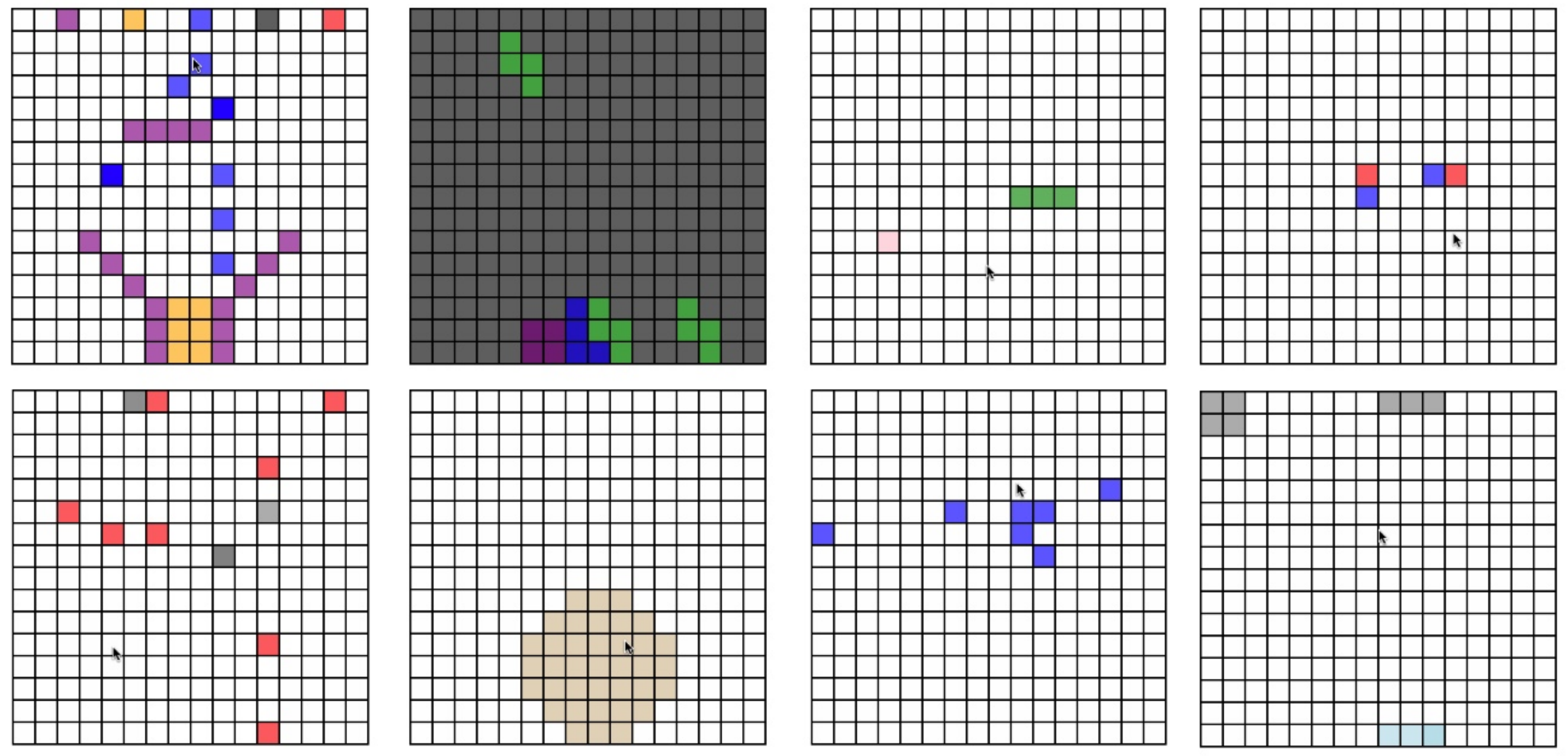

Figure 2: Example domains from the Causal Inductive Synthesis Corpus. From top-left clockwise: a simulation of water interacting with a sink, a Tetris clone, a snake clone, interacting magnets, food-seeking ants, obfuscated objects, a particle simulation, a simple weather simulation.

Passive Discovery The passive inductive synthesis problem is to produce a synthesizer s that maps a set of trajectories T_m^{train} produced from a ground truth Autumn model m onto a hypothesis Autumn model $\hat{m} = s(T_m^{\text{train}})$.

The score of a hypothesis $\hat{m}$ is a measure of the degree to which it matches m on the test trajectories. Letting sim denote a stochastic simulation function such that $\text{sim}(m, \mathbf{a})$ is a random variable over observations, the score of m is marginal likelihood averaged over T_m^{test}:

$$\text{score}_m(\hat{m}) = \frac{1}{M_m^{\text{test}}} \sum_{(\mathbf{a}_i^m, \mathbf{o}_i^m) \in T_m^{\text{test}}} p(\text{sim}(\hat{m}, \mathbf{a}_i^m) = \mathbf{o}_i^m).$$

The score of a synthesizer s is then the average score over $\mathcal{D}$:

$$\text{score}(s) = \frac{1}{N} \sum_{m \in \mathcal{D}} \text{score}_{m_i}(s(T_m^{\text{train}})).$$

Active Discovery In contrast to the passive case, in active discovery the observational data is not given and must be produced by an active agent. The active inductive synthesis problem is to produce a pair (π, σ) where $\pi : \mathcal{O} \times \Phi \to \mathcal{A} \times \Phi$ is a policy with internal memory Φ, and σ is stateful synthesizer. The agent interacts with a model, producing observational data until a `stop` action is performed. At this point a hypothesis model $\hat{m} = \sigma(T, \phi)$ is produced as function of the internal state $\phi \in \Phi$ of the agent and the trajectory $T = (\mathbf{a}, \mathbf{o})$ it has observed.

References

[1] Laura Schulz. The origins of inquiry: Inductive inference and exploration in early childhood. *Trends in cognitive sciences*, 16(7):382–389, 2012.

[2] Alison Gopnik. Scientific thinking in young children: Theoretical advances, empirical research, and policy implications. *Science*, 337(6102):1623–1627, 2012.