Making sense of sensory input

Richard Evans[a,b,*], José Hernández-Orallo[c,d], Johannes Welbl[a,e], Pushmeet Kohli[a], Marek Sergot[b]

[a]*DeepMind, London*
[b]*Imperial College London*
[c]*Universitat Politècnica de València*
[d]*CFI, University of Cambridge*
[e]*University College London*

Abstract

This paper attempts to answer a central question in unsupervised learning: what does it mean to "make sense" of a sensory sequence? In our formalization, making sense involves constructing a symbolic causal theory that explains the sensory sequence and satisfies a set of unity conditions. This model was inspired by Kant's discussion of the "synthetic unity of apperception" in the *Critique of Pure Reason*. On our account, making sense of sensory input is a type of program synthesis, but it is *unsupervised* program synthesis.

Our second contribution is a computer implementation, the Apperception Engine, that was designed to satisfy the above requirements. Our system is able to produce interpretable human-readable causal theories from very small amounts of data, because of the strong inductive bias provided by the Kantian unity constraints. A causal theory produced by our system is able to predict future sensor readings, as well as retrodict earlier readings, and "impute" (fill in the blanks of) missing sensory readings, in any combination.

We tested the engine in a diverse variety of domains, including cellular automata, rhythms and simple nursery tunes, multi-modal binding problems, occlusion tasks, and sequence induction IQ tests. In each domain, we test our engine's ability to predict future sensor values, retrodict earlier sensor values,

*Corresponding author
Email address: `richardevans@google.com` (Richard Evans)

and impute missing sensory data. The Apperception Engine performs well in all these domains, significantly out-performing neural net baselines. We note in particular that in the sequence induction IQ tasks, our system achieved human-level performance. This is notable because our system is not a bespoke system designed specifically to solve IQ tasks, but a *general purpose* apperception system that was designed to make sense of *any* sensory sequence.

Keywords: apperception, unsupervised program synthesis, Kant

1. Introduction

Imagine a machine, equipped with various sensors, that receives a stream of sensory information. It must, somehow, *make sense* of this stream of sensory data. But what does it mean, exactly, to "make sense" of sensory data? We have an intuitive understanding of what is involved in making sense of the sensory stream – but can we specify precisely what is involved? Can this intuitive notion be formalized?

One approach is to treat the sensory sequence as the input to a supervised learning problem[1]: given a sequence $x_{1:t}$ of sensory data from time steps 1 to t, maximize the probability of the next datum x_{t+1}. This family of approaches seeks to maximize $p(x_{t+1} \mid x_{1:t})$.

We believe there is more to "making sense" than just predicting future sensory readings. Making sense of a sequence means interpreting that sequence as a representation of an external world composed of objects, persisting over time, with attributes that change over time, according to general laws. Predicting the future state of one's photoreceptors may be *part* of what is involved in making sense – but it is not on its own sufficient.

Deep convolutional neural networks are remarkably good at image classification when given large amounts of training data[2]. But it is important to

[1]See, for example, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. See also the Predictive Processing paradigm: [12, 13, 14, 15].

[2]For example, the AmoebaNet [16] achieves 99% accuracy on CIFAR-10, and the NASNet [17]

understand what they can and cannot do. Although they are extremely accurate at classifying an image as a cat or dog, they do not know what a cat or dog *is*. They do not know that cats and dogs are mammals, that no animal can be both a cat and a dog, or that these animals are suitable as household pets. More fundamentally, they do not even know that cats and dogs are enduring physical objects that inhabit 3D space. These nets are able to classify, but they not able to integrate that classification into a wider body of knowledge, to explore its inferential connections, and understand what it rules out. They are able, in other words, to perceive, but not to **apperceive**.

"Apperception" is understood in two distinct but related ways[3]. First, to apperceive something is to be self-consciously aware of it. Second, to apperceive something is to assimilate it into a larger collection of ideas, to unify the thought with others in a larger, coherent whole. In this paper, we focus on apperception in the second sense, as the process of assimilating sensory information into a coherent unified whole[4].

In this paper, we shall interpret "making sense" of a sensory sequence as apperceiving that sequence: constructing an interpretable (symbolic) causal theory that satisfies various unity conditions and that explains the sequence. The particular unity conditions we use are inspired by Kant's discussion of the "synthetic unity of apperception" in the *Critique of Pure Reason* [19]. However

achieves 96.2% top-five accuracy on ImageNet.

[3]Apperception (from the French 'apercevoir') is a term coined by Leibniz in the *New Essays Concerning Human Understanding* [18]. It is a central term in Kant's *Critique of Pure Reason* [19].

[4]For similar uses, see Dewey: "Apperception is the relating activity which combines the various sensuous elements presented to the mind at one time into a whole, and which unites these wholes, recurring at successive times, into a continuous mental life, thereby making psychical life intelligent" [20]. See also James: "Every impression that comes in from without, be it a sentence which we hear, an object of vision, or an effluvium which assails our nose, no sooner enters our consciousness than it is drafted off in some determinate direction or other, making connection with the other materials already there, and finally producing what we call our reaction. The particular connections it strikes into are determined by our past experiences and the 'associations' of the present sort of impression with them... This way of taking in the object is the process of apperception." [21]

we do not focus on Kant exegesis in this paper, but rather focus on the formal and experimental consequences of the definitions.

This paper makes two main contributions. The first is a formalization of what it means to "make sense" of (*apperceive*) the stream of sensory data. According to our definition, making sense of a sensory sequence involves positing a symbolic causal theory – a set of objects, a set of concepts, a set of initial conditions, a set of rules, and a set of constraints – that together satisfy two conditions. First, the theory must explain the sensory readings it is given. Second, the theory must satisfy a particular type of "unity". Our Kant-inspired definition of unity involves four conditions. (i) *Spatial unity*: all objects must be unified in space via a chain of binary relations. (ii) *Conceptual unity*: all concepts must be unified via constraints. (iii) *Static unity*: all propositions that are true at the same time must jointly satisfy the set of constraints. (iv) *Temporal unity*: all the states must be unified into a sequence by causal rules.

Our second contribution is a description of a particular computer system, the Apperception Engine (AE), that was designed to satisfy the conditions described above[5]. We introduce a causal language, Datalog$^{\ni}$, that was designed for reasoning about infinite temporal sequences. Given a sensory sequence, our system synthesizes a Datalog$^{\ni}$ program that, when executed, generates a trace that both explains the sensory sequence and also satisfies the four conditions of unity. This can be seen as a form of *unsupervised program synthesis* [22]. In traditional supervised program synthesis, we are given input/output pairs, and search for a program that, when executed on the inputs, produces the desired outputs. Here, in unsupervised program synthesis, we are given a sensory sequence, and search for a causal theory that, when executed, generates a trajectory that both respects the sensory sequence and also satisfies the conditions of unity.

The Apperception Engine has a number of appealing features. (1) Because the causal theories it generates are symbolic, they are human-readable and

[5]Our code and datasets are publicly available. See the supplementary material for details.

hence verifiable. We can understand precisely how the system is apperceiving its sensory data[6]. (2) Because of the strong inductive bias (both in terms of the design of the causal language, $\mathrm{Datalog}^{\ni}$, but also in terms of the Kantian unity conditions that must be satisfied), the system is data-efficient, able to make sense of the shortest and scantiest of sensory sequences[7]. (3) Our system generates a causal model that is able to accurately predict future sensory input. But *that is not all it can do*; it is also able to *retrodict* previous values and *impute* missing sensory values in the middle of the sensory stream. In fact, our system is able to predict, retrodict, and impute simultaneously[8]. (4) The Apperception Engine has been tested in a diverse variety of domains, with encouraging results. The five domains we use are elementary cellular automata, rhythms and nursery tunes, "Seek Whence" sequence induction IQ tasks [24], multi-modal binding tasks, and occlusion problems. These tasks were chosen because they require apperception rather than mere classificatory perception, and because they are simple for humans but not for modern machine learning systems e.g. neural networks[9]. The Apperception Engine performs well in all these domains, significantly out-performing neural net baselines. These results are significant because neural systems typically struggle to solve the binding problem (where information from different modalities must somehow be combined into different aspects of one unified object) and fail to solve occlusion tasks (in which objects are sometimes visible and sometimes obscured from view).

We note in particular that in the IQ tasks, our system achieved human-level performance. This is notable because the Apperception Engine was not

[6]Human readability is a much touted feature of Inductive Logic Programming (ILP) systems, but when the learned programs become large and include a number of invented auxiliary predicates, the resulting programs become less readable (see [23]). But even a large and complex machine-generated logic program will be easier to understand than a large tensor of floating point numbers.

[7]Our sensory sequences are less than 300 bits. See Table 2.

[8]See Example 6 for a case where the Apperception Engine jointly imputes and predicts.

[9]Figure 10 shows how neural baselines struggle to solve these tasks.

designed to solve these IQ tasks; it is not a bespoke hand-engineered solution to this particular domain. Rather, it is a *general purpose* apperception system that attempts to make sense of *any* sensory sequence. This is, we believe, a highly suggestive result [25].

In ablation tests, we tested what happened when each of the four Kantian unity conditions was turned off. Since the system's performance deteriorates noticeably when each unity condition is ablated, this indicates that the unity conditions are indeed doing vital work in our engine's attempts to make sense of the incoming barrage of sensory data.

The sensory sequences that we tested on are very simple and very short (see Table 2). These tasks are, from the perspective of many in the machine learning community, "toy". Nevertheless we believe that, as well as trying to achieve impressive but inscrutable predictive results on huge and complex datasets, it is also important to find *interpretable* causal theories for simpler datasets. Which domains count as "toy", in other words, depends on what we are going to do with them – and the goal of interpretable integrative apperception is significantly more demanding than the goal of merely predicting future sensory states.

1.1. Related work

A human being who has built a mental model of the world can use that model for counterfactual reasoning, anticipation, and planning [26, 27, 28, 29, 30, 31, 32]. Similarly, computer agents endowed with mental models are able to achieve impressive performance in a variety of domains. For instance, Lukasz Kaiser et al. [33] show that a model-based RL agent trained on 100K interactions compares with a state-of-the-art model-free agent trained on tens or hundreds of millions of interactions. David Silver et al. [34] have shown that a model-based Monte Carlo tree search planner with policy distillation can achieve superhuman level performance in a number of board games. The tree search relies, crucially, on an accurate model of the game dynamics.

When we have an accurate model of the environment, we can leverage that model to anticipate and plan. But in many domains, we do not have an accurate model. If we want to apply model-based methods in these domains, we must *learn* a model from the stream of observations. In the rest of this section, we shall describe various different approaches to representing and learning models, and show where our particular approach fits into the landscape of model learning systems.

Before we start to build a model to explain a sensory sequence, one fundamental question is: what form should the model take? We shall distinguish three dimensions of variation of models (adapted from [35]): first, whether they simply model the observed phenomena, or whether they also model latent structure; second, whether the model is explicit and symbolic or implicit; and third, what type of prior knowledge is built into the model structure.

We shall use the hidden Markov model (HMM)[10] [36, 37] as a general framework for describing sequential processes. Here, the observation at time t is x_t, and the latent state is z_t. In a HMM, the observation x_t at time t depends only on the latent (unobserved) state z_t. The state z_t in turn depends only on the previous latent state z_{t-1}.

The first dimension of variation amongst models is whether they actually use latent state information z_t to explain the observation x_t. Some approaches [38, 39, 40, 41, 42, 43] assume we are *given* the underlying state information $z_{1:t}$. In these approaches, there is no distinction between the observed phenomena and the latent state: $x_i = z_i$. With this simplifying assumption, the only thing a model needs to learn is the transition function. Other approaches [5, 7, 44] focus only on the observed phenomena $x_{1:t}$ and ignore latent information $z_{1:t}$ altogether. These approaches predict observation x_{t+1} given observation x_t

[10]Many systems predict state dynamics for partially observable Markov decision processes (POMDPs), rather than HMMs. In a POMDP, the state transition function depends on the previous state z_t and the action a_t performed by an agent. See Jessica Hamrick's paper for excellent overview [35] of model-based methods in deep learning that is framed in terms of POMDPs. In this paper, we consider HMMs. Adding actions to our model is not particularly difficult, but is left for further work.

without positing any hidden latent structure. Some approaches take latent information seriously [1, 10, 45, 11, 46]. These jointly learn a perception function (that produces a latent z_t from an observed x_t), a transition function (producing a next latent state z_{t+1} from latent state z_t) and a rendering function (producing a predicted observation x_{t+1} from the latent state z_{t+1}). Our approach also builds a latent representation of the state. As well as positing latent properties (unobserved properties that explain observed phenomena), we also posit latent *objects* (unobserved objects whose relations to observed objects explain observed phenomena).

The second dimension of variation concerns whether the learned model is explicit, symbolic and human-readable, or implicit and inscrutable. In some approaches [1, 10, 45, 11], the latent states are represented by vectors and the dynamics of the model by weight tensors. In these cases, it is hard to understand what the system has learned. In other approaches [47, 48, 49, 50], the latent state is represented symbolically, but the state transition function is represented by the weight tensor of a neural network and is inscrutable. We may have some understanding of what state the machine thinks it is in, but we do not understand why it thinks there is a transition from this state to that. In some approaches [51, 52, 53, 54, 55, 56], both the latent state and the state transition function are represented symbolically. Here, the latent state is a set of ground atoms and the state transition function is represented by a set of universally quantified rules. Our approach falls into this third category. Here, the model is fully interpretable: we can interpret the state the machine thinks it is in, and we can understand the reason why it believes it will transition to the next state.

A third dimension of variation between models is the amount and type of prior knowledge that they include. Some model learning systems have very little prior knowledge. In some of the neural systems (e.g. [7]), the only prior knowledge is the spatial invariance assumption implicit in the convolutional network's structure. Other models incorporate prior knowledge about the way objects and states should be represented. For example, some models assume objects can be composed in hierarchical structures [48]. Other systems additionally incorporate prior knowledge about the type of rules that are used to define the state transition function. For example, some [54, 55, 56] use prior knowledge of the event calculus [57]. Our approach falls into this third category. We impose a language bias in the form of rules used to define the state transition function and also impose additional requirements on candidate sets of rules: they must satisfy the four Kant-inspired unity conditions introduced above (and elaborated in Section 3.2 below).

1.1.1. Our approach

To summarize, in order to position our approach within the landscape of other approaches, we have distinguished three dimensions of variation. Our approach differs from neural approaches in that the posited theory is explicit and human readable. Not only is the representation of state explicit (represented as a set of ground atoms) but the transition dynamics of the system are also explicit (represented as universally quantified rules in a domain specific language designed for describing causal structures). Our approach differs from other inductive program synthesis methods in that it posits significant latent structure in addition to the induced rules to explain the observed phenomena: in our approach, explaining a sensory sequence does not just mean constructing a set of rules that explain the transitions; it also involves positing a type signature containing a set of latent relations and a set of latent *objects*. Our approach also differs from other inductive program synthesis methods in the type of prior knowledge that is used: as well as providing a strong language bias by using a particular representation language (a typed extension of datalog with causal rules and constraints), we also inject a substantial inductive bias: the Kant-inspired unity conditions, the key constraints on our system, represent domain-*independent* prior knowledge. Our approach also differs from other inductive program synthesis methods in being entirely unsupervised. In contrast, OSLA and OLED [54, 55] are supervised, and SPLICE [56] is semi-supervised.

Ellis et al. [22] also learn programs in an unsupervised setting. We discuss related work in detail in the supplementary material, but focus here on one key difference: while we interpret time sequences, they focus on a single individual moment of time. Inoue, Ribeiro, and Sakama [52] describe a system (**LFIT**) for learning logic programs from sequences of sets of ground atoms. But they learn a program that generates a sequence that must match the input sequence exactly, while we impose the looser requirements that the generated sequence must "cover" the input sequence (see Definition 2). This extra lassitude enables us to construct interpretations that include both latent properties and also latent objects (see Example 11).

1.2. Paper outline

Section 2 introduces basic notation. Section 3 presents the main definition of what it means for a theory to count as a unified interpretation of a sensory sequence. Section 3.4 provides some technical results, guaranteeing that there is always some unified interpretation for every acceptable sensory sequence. Section 4 describes a computer system that is able to generate unified interpretations of sensory sequences. Section 5 describes our experiments in five different types of task: elementary cellular automata, rhythms and nursery tunes, "Seek Whence" sequence induction IQ tasks, multi-modal binding tasks, and occlusion problems. In Section 6, we compare our Apperception Engine with various baselines and ablated systems.

2. Background

In this paper, we use basic concepts and standard notation from logic programming [58, 59, 60]. A function-free atom is an expression of the form $p(t_1, ..., t_n)$, where p is a predicate of arity $n \geq 0$ and each t_i is either a variable or a constant. We shall use $a, b, c, ...$ for constants, $X, Y, Z, ...$ for variables, and $p, q, r, ...$ for predicate symbols.

A **Datalog clause** is a definite clause of the form $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_0$ where each α_i is an atom and $n \geq 0$. It is traditional to write clauses from right to left:

$\alpha_0 \leftarrow \alpha_1, ..., \alpha_n$. In this paper, we will define a Datalog interpreter implemented in another logic programming language, ASP (answer-set programming). In order to keep the two languages distinct, we write Datalog rules from left to right and ASP clauses from right to left. A **Datalog program** is a set of Datalog clauses.

A key result of logic programming is that every Datalog program has a unique subset-minimal least Herbrand model that can be directly computed by repeatedly generating the consequences of the ground instances of the clauses [61].

We turn now from Datalog to normal logic programs under the answer set semantics [62]. A **literal** is an atom α or a negated atom *not* α. A **normal logic program** is a set of clauses of the form:

$$\alpha_0 \leftarrow \alpha_1, ..., \alpha_n$$

where α_0 is an atom, $\alpha_1, ..., \alpha_n$ is a conjunction of literals, and $n \geq 0$. Normal logic clauses extend Datalog clauses by allowing functions in terms and by allowing negation by failure in the body of the rule.

Answer Set Programming (ASP) is a logic programming language based on normal logic programs under the answer set semantics. Given a normal logic program, an ASP solver finds the set of answer sets for that program. Modern ASP solvers can also be used to solve optimization problems by the introduction of weak constraints [63]. A **weak constraint** is a rule that defines the cost of a certain tuple of atoms. Given a program with weak constraints, an ASP solver can find a preferred answer set with the lowest cost.

3. A computational framework for apperception

Imagine that our perceptual machine is equipped with certain sensors. Assume that the sensor readings have already been *discretized*, so a sensory reading featuring sensor a can be represented by a ground atom $p(a)$ for some unary predicate p, or by an atom $r(a, b)$ for some binary relation r and unique value b.

In this paper, for performance reasons, we restrict our attention to unary and binary predicates[11].

Definition 1. Let $\mathcal{G}$ be the set of all ground atoms. A **sensory sequence** $S \in \left(2^{\mathcal{G}}\right)^*$ is a sequence of sets of ground atoms. Given a sequence $S = (S_1, S_2, ...)$, a **state** $S_t \subseteq \mathcal{G}$ is a set of atoms, representing a partial description of the world at a discrete time-step t. An atom $p(a) \in S_t$ represents that sensor a has property p at time t. An atom $r(a, b) \in S_t$ represents that sensor a is related via relation r to value b at time t.

Example 1. Note that there is no expectation that a sensory sequence contains readings for all sensors at all time-steps. Some of the readings may be missing. Consider, for example, the following sequence $S_{1:6}$:

$$\begin{array}{ll} S_1 = \{green(b)\} & S_2 = \{off(a), green(b)\} \\ S_3 = \{on(a)\} & S_4 = \{off(a), green(b)\} \\ S_5 = \{on(a), green(b)\} & S_6 = \{\} \end{array}$$

Here, sensor a is missing a reading at time 1, sensor b is missing a reading at time 3, and both sensors are missing readings at time 6. We shall use this sequence as a running example. $\triangleleft$

We shall make extensive use of the following partial order on sensory sequences:

Definition 2. Given two finite sequences $S = (S_1, ..., S_t)$ and $S' = (S'_1, ..., S'_{t'})$, define $\sqsubseteq$ as:

$$S \sqsubseteq S' \text{ iff } t \leq t' \text{ and } \forall 1 \leq i \leq t,\ S_i \subseteq S'_i$$

[11]This restriction can be made without loss of generality, since every k-ary relationship can be expressed as $k + 1$ binary relationships [64].

So far, we assume both sequences are finite. Now we extend this partial order so that S' may be infinitely long: $S \sqsubseteq S'$ if S' has a finite subsequence S'' such that $S \sqsubseteq S''$. If $S \sqsubseteq S'$, we say that S is **covered by** S', or that S' covers S.

Example 2. Compare sequence $S_{1:6}$ from Example 1 with the following sequence $S' = (S'_1, S'_2, S'_3, S'_4)$ where:

$$S'_1 = \{\} \qquad S'_2 = \{green(b)\}$$
$$S'_3 = \{on(a)\} \qquad S'_4 = \{off(a)\}$$

Here, $S' \sqsubseteq S$, but $S \not\sqsubseteq S'$. ◁

Theorem 1. $\sqsubseteq$ is a partial order. The proof is straightforward. Proofs of all theorems are in the supplementary material.

In this paper, we "make sense" of sensory sequences by constructing theories that explain these sequences. A central component of a theory is a *frame*, a type signature specifying the set of concepts that can be used in the theory:

Definition 3. Given a set $\mathcal{T}$ of types, a set $\mathcal{O}$ of constants representing individual objects, and a set $\mathcal{P}$ of predicates representing properties and relations, let $\mathcal{G}$ be the set of all ground atoms formed from $\mathcal{T}$, $\mathcal{O}$, and $\mathcal{P}$. Given a set $\mathcal{V}$ of variables, let $\mathcal{U}$ be the set of all unground atoms formed from $\mathcal{T}$, $\mathcal{V}$, and $\mathcal{P}$.

A **frame** is a tuple (T, O, P, V) where $T \subseteq \mathcal{T}$ is a finite set of types, $O \subseteq \mathcal{O}$ is a finite set of constants representing objects, $P \subseteq \mathcal{P}$ is a finite set of predicates representing properties and relations, and $V \subseteq \mathcal{V}$ is a finite set of variables. We write $\kappa_O : O \rightarrow T$ for the type of an object, $\kappa_P : P \rightarrow T^*$ for the types of the predicate's arguments, and $\kappa_V : V \rightarrow T$ for the type of a variable.

Now some frames are suitable for some sensory sequences, while other frames are unsuitable, because they do not contain the right constants and predicates. The following definition formalizes this:

Definition 4. Let $G_S = \bigcup_{t \geq 1} S_t$ be the set of all ground atoms that appear in sensory sequence $S = (S_1, ...)$. Let G_ϕ be the set of all ground atoms that are well-typed according to frame ϕ. If $\phi = (T, O, P, V)$ then $G_\phi = \{p(a_1, ..., a_n) \mid p \in P, \kappa_P(p) = (t_1, ..., t_n), a_i \in O, \kappa_O(a_i) = t_i \text{ for all } i = 1..n\}$. A frame ϕ is **suitable** for a sensory sequence S if all the atoms in S are well-typed according to frame ϕ, i.e. $G_S \subseteq G_\phi$.

Let U_ϕ be the set of all **unground** atoms that are well-typed according to frame ϕ. If $\phi = (T, O, P, V)$ then $U_\phi = \{p(v_1, ..., v_n) \mid p \in P, \kappa_P(p) = (t_1, ..., t_n), v_i \in V, \kappa_V(v_i) = t_i \text{ for all } i = 1..n\}$. Note that, according to this definition, an atom is unground if *all* its terms are variables. "Unground" here means more than simply not ground. For example, $p(a, X)$ is neither ground nor unground.

Example 3. One suitable frame for the sequence of Example 1 is (T, O, P, V) where:

$$\begin{aligned} T &= \{t_1, t_2\} \\ O &= \{a{:}t_1, b{:}t_2\} \\ P &= \{on(t_1), \mathit{off}(t_1), green(t_2), red(t_2), next(t_1, t_2)\} \\ V &= \{X{:}t_1, Y{:}t_2\} \end{aligned}$$

Here, and throughout, we write $a{:}t_1$ to mean that object a is of type t_1, $next(t_1, t_2)$ to mean that binary predicate *next* takes two arguments of types t_1 and t_2, and $X{:}t_1$ to mean that variable X is of type t_1.

There are, of course, an infinite number of other suitable frames. Here, $U_\phi = \{on(X), \mathit{off}(X), green(Y), red(Y), next(X, Y)\}$. ◁

We shall make sense of sequences by positing theories that explain those sequences. We represent theories using concepts from logic programming, in general, and Datalog in particular [65]. Our theories contain two types of rules: static rules and causal rules.

Definition 5. Given a frame ϕ, a **static rule** is a definite clause of the form $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_0$, where $n \geq 0$ and each $\alpha_i \in U_\phi$. Informally, if conditions $\alpha_1, ...\alpha_n$ hold at the current time-step, then α_0 also holds at that time-step. So, for example, $on(X) \wedge next(X, Y) \rightarrow green(Y)$ states that if object X is *on* and X is related to Y via *next*, then Y is simultaneously *green*.

A **causal rule** is a clause of the form $\alpha_1 \wedge ... \wedge \alpha_n \ni\!\!\!\!- \alpha_0$, where $n \geq 0$ and each $\alpha_i \in U_\phi$. A causal rule expresses how facts change over time. $\alpha_1 \wedge ... \wedge \alpha_n \ni\!\!\!\!- \alpha_0$ states that if conditions $\alpha_1, ...\alpha_n$ hold at the current time-step, then α_0 will be true at the *next* time-step. So, for example, $on(X) \ni\!\!\!\!- off(X)$ states that if object X is currently *on*, then X will become *off* at the next-time-step.

Let R_ϕ be the set of all well-formed static and causal rules according to the frame ϕ:

$$R_\phi = \left\{ A \circ \alpha \mid \circ \in \{\rightarrow, \ni\!\!\!\!-\}, A \cup \{\alpha\} \subseteq U_\phi, var(\alpha) \subseteq \bigcup_{\beta \in A} var(\beta) \right\}$$

As well as static and dynamic rules, a theory also contains *constraints* that eliminate certain possibilities:

Definition 6. We shall define three types of constraint. Given a frame $\phi = (T, O, P, V)$, a **unary xor constraint** is an expression of the form:

$$\forall X{:}t, p_1(X) \oplus ... \oplus p_n(X)$$

where $t \in T$, $p_i \in P$, $\kappa_P(p_i) = (t)$, $X \in V$, $\kappa_V(X) = t$. Here, $\oplus$ means exclusive-or, so for example $\forall X{:}t, p_1(X) \oplus p_2(X)$ means that for every object X of type t, X either satisfies p_1 or p_2, but not both.

A **binary xor constraint** is an expression of the form:

$$\forall X{:}t_1, \forall Y{:}t_2, r_1(X, Y) \oplus ... \oplus r_n(X, Y)$$

where $t_1, t_2 \in T$, $r_i \in P$, $\kappa_P(r_i) = (t_1, t_2)$, $X, Y \in V$, $\kappa_V(X) = t_1$, and $\kappa_V(Y) = t_2$.

A **uniqueness constraint** is an expression of the form:

$$\forall X{:}t_1, \exists! Y{:}t_2, r(X, Y)$$

where $t_1, t_2 \in T$, $r \in P$, $\kappa_P(r) = (t_1, t_2)$, $X, Y \in V$, and $\kappa_v(X) = t_1, \kappa_v(Y) = t_2$. Here, $\forall X{:}t_1, \exists! Y{:}t_2, r(X, Y)$ means that for all objects X of type t_1 there exists a *unique* object Y of type t_2 such that $r(X, Y)$.

Let C_ϕ be the set of all such well-formed constraints according to the frame ϕ.

Example 4. In Example 3, R_ϕ contains the following rules among others:

$$\begin{aligned} on(X) &\rightarrow on(X) \\ on(X) \land \mathit{off}(X) &\rightarrow on(X) \\ on(X) &\ni\!\!\!\rightarrow \mathit{off}(X) \\ on(X) \land next(X, Y) &\ni\!\!\!\rightarrow red(Y) \end{aligned}$$

C_ϕ contains the following constraints among others:

$$\begin{aligned} \forall X{:}t_1, on(X) \oplus \mathit{off}(X) \\ \forall Y{:}t_2, green(Y) \oplus red(Y) \\ \forall X{:}t_1, \exists! Y{:}t_2, next(X, Y) \end{aligned}$$

◁

Definition 7. Datalog$^{\ni\!\rightarrow}$ is a variant of Datalog that contains causal rules as well as static rules, as defined in Definition 5, and that also contains constraints, as defined in Definition 6.

Datalog$^{\ni\!\rightarrow}$ extends Datalog by adding causal rules and constraints. But it is not, strictly speaking, an extension of Datalog, since Datalog$^{\ni\!\rightarrow}$ does not allow ground rules. Dedalus [66] is a related extension to Datalog for representing information that changes over time. The key element that distinguishes Datalog$^{\ni}$ from Dedalus is that our logic uses constraints to eliminate possibilities.

A theory is a collection of initial conditions, rules, and constraints that together purport to "make sense" of the given sensory sequence. The initial conditions specify what is true at some (arbitrarily chosen) initial time, the rules determine how facts change over time, and the constraints eliminate certain possibilities.

Definition 8. A **theory** θ is a tuple (ϕ, I, R, C) where:

- ϕ is a frame
- $I \subseteq G_\phi$ is a set of initial conditions
- $R \subseteq R_\phi$ is a set of rules (static rules and causal rules)
- $C \subseteq C_\phi$ is a set of constraints (unary and binary constraints)

Note that constants are not allowed in rules or in constraints[12]. The only terms that are allowed in rules and constraints are variables. The only place where ground atoms appear is in the initial conditions I.

3.1. Semantics

We now explain how theories are interpreted. We define what it is for a constraint to be satisfied in a particular state (a set of ground atoms), and we define how the rules generate an infinite sequence of states.

[12]This bias occurs in various ILP systems [52, 67].

Definition 9. Let A_t represent a set of ground atoms, α an unground atom, and σ a substitution. Then $A_t \models_\sigma \alpha$ denotes that A_t *satisfies* α with respect to substitution σ.

$A_t \models_\sigma p(X_1, ..., X_n)$ iff $p(X_1, ..., X_n)\sigma \in A_t$

$A_t \models_\sigma \alpha_1 \oplus ... \oplus \alpha_n$ iff $|\{i \mid A_t \models_\sigma \alpha_i\}| = 1$

$A_t \models_\sigma \forall X\ \exists! Y\ p(X, Y)$ iff there is exactly one variant substitution σ_2 (i.e. a substitution that differs from σ in at most the assignment to Y) such that $A_t \models_{\sigma_2} p(X, Y)$

3.1.1. Traces

In this section, we define the trace of a theory as an infinite sequence of sets of ground atoms. In order to do this, we need first to define a notion of 'incompossibility'.

Definition 10. Two ground atoms α and α' are **incompossible** with respect to a set C of constraints if either

- There is a constraint $\beta_1 \oplus ... \oplus \beta_n$ in C, and a ground substitution σ such that $\beta_i\sigma = \alpha$ and $\beta_j\sigma = \alpha'$ for some $i \neq j$
- There is a constraint $\forall X\ \exists! Y\ p(X, Y)$ in C, and a ground substitution σ such that there exist ground substitutions σ_1 and σ_2 extending σ, but differing in their assignment to Y, such that $p(X, Y)\sigma_1 = \alpha$ and $p(X, Y)\sigma_2 = \alpha'$

We shall also say that a ground atom α is **compossible** (with respect to C) with a *set* A of ground atoms if there is no α' in A such that α and α' are incompossible (with respect to C).

Let $\mathcal{G}$ be the set of all possible ground atoms, let Φ be the set of all possible frames, and let Θ be the set of all possible theories.

Definition 11. Every theory $\theta = (\phi, I, R, C)$ generates an infinite sequence of sets of ground atoms, called the **trace** of that theory. The τ function is used to denote the trace of a theory:

$$\tau : \Theta \rightarrow \left(2^{\mathcal{G}}\right)^*$$

Here, $\tau(\theta) = (A_1, A_2, ...)$, where each A_t is the smallest set of atoms satisfying the following conditions:

- $I \subseteq A_1$
- If there is a static rule $\beta_1 \wedge ... \wedge \beta_m \rightarrow \alpha'$ in R and a ground substitution σ such that $A_t \models_\sigma \beta_i$ for each antecedent β_i and $\alpha = \alpha'\sigma$, then $\alpha \in A_t$
- If there is a causal rule $\beta_1 \wedge ... \wedge \beta_m \ni\!\!\!\rightarrow \alpha'$ in R and a ground substitution σ such that $A_t \models_\sigma \beta_i$ for each antecedent β_i and $\alpha = \alpha'\sigma$, then $\alpha \in A_{t+1}$
- *Frame axiom*: if α is in A_{t-1} and there is no atom in A_t that is incompossible with α w.r.t constraints C, then $\alpha \in A_t$

Note that the state transition function is deterministic: A_{t+1} is uniquely determined by A_t.

Example 5. Let $\theta = (\phi, I, R, C)$ where ϕ is the frame from Example 3 and:

$$\begin{aligned}
I &= \{on(a), green(b), next(a,b)\} \\
R &= \left\{ \begin{array}{l} on(X) \ni\!\!\!\rightarrow off(X) \\ off(X) \ni\!\!\!\rightarrow on(X) \end{array} \right\} \\
C &= \left\{ \begin{array}{l} \forall X{:}t_1, on(X) \oplus off(X) \\ \forall Y{:}t_2, green(Y) \oplus red(Y) \\ \forall X{:}t_1, \exists! Y{:}t_2, next(X,Y) \end{array} \right\}
\end{aligned}$$

Then $on(a)$ and $off(a)$ are incompossible w.r.t. C and so are $next(a,a)$ and $next(a,b)$. The trace of θ oscillates between two states: $\tau(\theta) = (A_1, A_2, A_1, A_2, A_1, A_2, ...)$

where:

$$\begin{aligned} A_1 &= \{on(a), green(b), next(a, b)\} \\ A_2 &= \{off(a), green(b), next(a, b)\} \end{aligned}$$

$\triangleleft$

The following theorem will be useful. It states that once the trace of a theory repeats itself, it will continue to repeat thereafter[13]:

Theorem 2. If $\tau(\theta) = (A_1, A_2, ...)$ and $A_i = A_j$, then $\forall k \geq 0$:

$$A_{i+k} = A_{j+k}$$

The proof is straightforward since the state transition is Markov and deterministic.

One important consequence of Theorem 2 is:

Theorem 3. Given a theory θ and a ground atom α, it is decidable whether α appears somewhere in $\tau(\theta)$. The proof is straightforward: since the set of ground atoms is finite, the set of possible states is also finite, so any infinite trace must eventually contain a repeated state, at which point Theorem 2 applies.

3.2. Apperception

We have now assembled the materials we need to define the key concepts of this paper. Throughout, we shall use the expressions "apperceiving sensory sequence S" and "making sense of S" interchangeably.

Definition 12. A theory θ counts as an **interpretation** of S if the trace of θ covers S, i.e. $S \sqsubseteq \tau(\theta)$

[13]Inoue et al make a similar argument in [52].

In providing a theory θ that counts as an interpretation of a sensory sequence S, we make S intelligible by placing it within a bigger picture: while S is a scanty and incomplete description of a fragment of the time-series, $\tau(\theta)$ is a complete and determinate description of the whole time-series.

In order for θ to make sense of S, it is *necessary* that $\tau(\theta)$ covers S. But this condition is not, on its own, *sufficient*. The extra condition that is needed for θ to count as "making sense" of S is for θ to be *unified*.

In this paper, we formalize what it means for a theory to be "unified" using Kant's[14] key notion of the "synthetic unity of apperception"[15]. A trace $\tau(\theta)$ is (i) a sequence of (ii) sets of ground atoms composed of (iii) predicates and (iv) objects[16]. For the theory θ to be unified is for unity to be achieved at each of these four levels:

1. objects are united in space[17]
2. predicates are united via constraints[18]
3. ground atoms are united into sets (states) by jointly respecting constraints and static rules[19]
4. states (sets of ground atoms) are united into a sequence by causal rules[20]

[14]In this paper, we do not focus on Kant exegesis, but do provide some key references. All references to the *Critique of Pure Reason* use the standard [A/B] reference system, where A refers to the First Edition (1781), and B refers to the Second Edition (1787). Our interpretation of Kant is indebted to Longuenesse's *Kant and the Capacity to Judge* and to Waxman's *Kant and the Anatomy of the Intelligent Mind*.

[15]"The principle of the synthetic unity of apperception is the supreme principle of all use of the understanding" [B136]; it is "the highest point to which one must affix all use of the understanding, even the whole of logic and, after it, transcendental philosophy" [B134].

[16]Strictly speaking, we mean two *constants* denoting two objects. Throughout, we follow the common practice of treating the Herbrand interpretation as a distinguished interpretation, blurring the difference between a constant and the object denoted by the constant.

[17]See [B203]. See also the *Third Analogy* [A211-215/B256-262].

[18]See [A103-11]. See also: "What the form of disjunctive judgment may do is contribute to the acts of forming categorical and hypothetical judgments the perspective of their possible *systematic unity*", [68, p.105]

[19]See the *schema of community* [A144/B183-4].

[20]See the *schema of causality* [A144/B183].

3.2.1. Spatial unity

Definition 13. A theory θ satisfies **spatial unity** if for each state A_t in $\tau(\theta) = (A_1, A_2, ...)$, for each pair (x, y) of distinct objects, x and y are connected via a chain of binary atoms $\{r_1(x, z_1), r_2(z_1, z_2), ...r_n(z_{n-1}, z_n), r_{n+1}(z_n, y)\} \subseteq A_t$.

If this condition is satisfied, it means that given any object, we can get to any other object by hopping along relations. Everything is connected, even if only indirectly[21].

Note that this notion of spatial unity is rather abstract: the requirement is only that every pair of objects are indirectly connected via some chain of binary relations. Although some of these binary relations might be spatial relations (e.g. “left-of”), they need not all be. The requirement is only that every pair of objects are connected via some chain of binary relations; it does not insist that each binary relation has a specifically “spatial” interpretation.

3.2.2. Conceptual unity

A theory satisfies conceptual unity if every predicate is involved in some constraint, either exclusive disjunction ($\oplus$) or unique existence ($\exists!$). The intuition here is that xor constraints combine predicates into clusters of mutual incompatibility[22].

Definition 14. A theory $\theta = (\phi, I, R, C)$ satisfies **conceptual unity** if for each unary predicate p in ϕ, there is some xor constraint in C of the form $\forall X{:}t, p(X) \oplus q(X) \oplus ...$ containing p; and, for each binary predicate r in ϕ, there is some xor constraint in C of the form $\forall X{:}t_1, \forall Y{:}t_2, r(X, Y) \oplus s(X, Y) \oplus ...$ or some $\exists!$ constraint in C of the form $\forall X{:}t, \exists! Y{:}t_2, r(X, Y)$.

[21] See [A211-5/B255-62].

[22] See [A73-4/B98-9]. See also [*Jäsche Logic* 9:107n].

To see the importance of this, observe that if there are no constraints, then there are no exhaustiveness or exclusiveness relations between atoms. An xor constraint e.g. $\forall X{:}t, on(X) \oplus \mathit{off}(X)$ both rules out the possibility that an object is simultaneously *on* and *off* (exclusiveness) and also rules out the possibility that an object of type t is neither *on* nor *off* (exhaustiveness). It is exhaustiveness which generates states that are *determinate*, in which it is guaranteed every object of type t is e.g. either *on* or *off*. It is exclusiveness which generates *incompossibility* between atoms, e.g. that $on(a)$ and $\mathit{off}(a)$ are incompossible. Incompossibility, in turn, is needed to constrain the scope of the frame axiom (see Section 3.1.1 above). Without incompossibility, *all* atoms from the previous time-step would be transferred to the next time-step, and the set of true atoms in the sequence $(S_1, S_2, ...)$ would grow monotonically over time: $S_i \subseteq S_j$ if $i \leq j$, which is clearly unacceptable. The purpose of the constraint of conceptual unity is to collect predicates into groups, to provide determinacy in each state, and to ground the incompossibility relation that constrains the way information is propagated between states[23].

3.2.3. Static unity

In our effort to interpret the sensory sequence, we construct various ground atoms. These need to be grouped together, somehow, into states (sets of atoms). But what determines how these atoms are grouped together into states?

Treating a set A of ground atoms as a state is (i) to insist that A satisfies all the constraints in C and (ii) to insist that A is closed under the static rules in R. If A does not satisfy the constraints, it is not a coherent and determinate representation; it is "less even than a dream"[24]. This motivates the following definition:

[23]"A judgement is nothing other than the way to bring given cognitions to the objective unity of apperception [B141]".

[24]See [A112].

Definition 15. A theory $\theta = (\phi, I, R, C)$ satisfies **static unity** if every state $(A_1, A_2, ...)$ in $\tau(\theta)$ satisfies all the constraints in C and is closed under the static rules in R.

Note that, from the definition of the trace in Definition 11, all the states in $\tau(\theta)$ are automatically closed under the static rules in R.

3.2.4. *Temporal unity*

Given a set of states, we need to unite these elements in a *sequence*. According to the fourth and final condition of unity, the only thing that can unite states in a sequence is a set of *causal rules*. These causal rules are *universal* in two senses: they apply to all object tuples, and they apply at all times. A causal rule $\alpha_1 \wedge ... \wedge \alpha_n \ni\!\!\!- \alpha_0$ fixes the temporal relation between the atoms $\alpha_1, ..., \alpha_n$ (which are true at t) and the atom α_0 (which is true at $t + 1$). According to Kant[25], the *only thing* that can fix the succession relation between states is the universal causal rule.

Imagine that, instead, we posit a finite sequence of states extensionally (rather than intensionally via initial conditions and rules). Here, our alternative "interpretation" of the sensory sequence S is just a finite S' where $S \sqsubseteq S'$. This S' is *arbitrary* because it is not generated by rules that confer on it the "dignity of necessity"[26]. In a unified interpretation, by contrast, the states are united in a sequence by being necessitated by universal causal rules. The above discussion motivates the following:

Definition 16. A sequence $(A_1, A_2, ...)$ of states satisfies **temporal unity** with respect to a set $R_{\ni\!\!\!-}$ of causal rules if, for each $\alpha_1 \wedge ... \wedge \alpha_n \ni\!\!\!- \alpha_0$ in $R_{\ni\!\!\!-}$, for

[25]See [B233-4].

[26]See [A91/B124].

Unity	**Element**	**Medium of unification**	**Unifier**
Spatial unity	Object	Space	Binary predicates
Conceptual unity	Predicate	The “space of reasons”	Constraints
Static unity	Atom	The temporary moment	Constraints and static rules
Temporal unity	State	The temporal sequence	Causal rules

Table 1: The four conditions of unity.

each ground substitution σ, for each time-step t, if $\{\alpha_1\sigma, ..., \alpha_n\sigma\} \subseteq A_t$ then $\alpha_0\sigma \in A_{t+1}$. Note that, from the definition of the trace in Definition 11, the trace $\tau(\theta)$ *automatically* satisfies temporal unity.

3.2.5. The four conditions of unity

To recap, the trace of a theory is a sequence of sets of atoms. The four types of element are objects, predicates, sets of atoms, and sequences of sets of atoms. Each of the four types of element has its own form of unity:

1. *Spatial unity*: objects are united in space by being connected via chains of relations
2. *Conceptual unity*: predicates are united in the “space of reasons” by constraints
3. *Static unity*: atoms are united in a state by jointly satisfying constraints and static rules
4. *Temporal unity*: states are united in a sequence by causal rules

Since temporal unity is automatically satisfied from the definition of a trace in Definition 11, we are left with only three unity conditions that need to be explicitly checked: spatial unity, conceptual unity, and static unity. A trace partially satisfies static unity since the static rules are automatically enforced by Definition 11; but the constraints are not necessarily satisfied.

3.2.6. *Making sense of sensory input*

Bringing the above definitions together, we are ready to define the conditions under which a theory counts as "making sense" of a sensory sequence:

Definition 17. A theory $\theta = (\phi, I, R, C)$ counts as a **unified interpretation** of S if θ is an interpretation of S (i.e. $S \sqsubseteq \tau(\theta)$) and θ satisfies the conditions of spatial unity, conceptual unity, and static unity. (It automatically satisfies temporal unity from Definition 11).

Note that both checking spatial unity and checking static unity require checking every time-step, and the trace is infinitely long. However, as long as the trace repeats at some point, Theorem 2 ensures that we need only check the finite portion of the trace until we find the first repetition (the first $i \neq j$ such that $\tau(\theta)_i = \tau(\theta)_j$). Since the set G_ϕ of ground atoms is finite, we are guaranteed to eventually reach a repetition.

Example 6. Consider again the sequence $S_{1:6}$ from Example 1. Let $\phi = (T, O, P, V)$ be the frame from Example 3. Let $\theta = (\phi, I, R, C)$ be the theory from Example 5. We shall verify that θ is indeed a unified interpretation of S. First, to verify that θ is an interpretation of S, observe that $\tau(\theta) = (A_1, A_2, A_1, A_2, A_1, A_2, ...)$ where:

$$\begin{aligned} A_1 &= \{on(a), green(b), next(a, b)\} \\ A_2 &= \{off(a), green(b), next(a, b)\} \end{aligned}$$

Clearly, $S \sqsubseteq \tau(\theta)$.

Next, we verify that θ satisfies the unity conditions. *Spatial unity*: at each time-step, objects a and b are directly connected via the $next(a, b)$ atom. *Conceptual unity*: *on* and *off* are constrained by $\forall X{:}t_1, on(X) \oplus off(X)$; *green* and *red* are constrained by $\forall Y{:}t_2, green(Y) \oplus red(Y)$; finally, *next* is constrained by $\forall X{:}t_1, \exists! Y{:}t_2, next(X, Y)$. *Static unity*: we need to check each constraint holds at each time-step for every substitution. We provide one example. To verify that e.g. $A_1 \models_\sigma \forall X{:}t_1, on(X) \oplus off(X)$, where $\sigma = \{X/a, Y/a\}$, note that $on(a) \in A_1$ but $off(a) \notin A_1$. $\triangleleft$

Example 7. Figure 1 shows four theories attempting to explain a sensory sequence S. Each of the four theories satisfies the Kantian unity conditions, but only theory (**d**) covers S. $\triangleleft$

In our search for unified interpretations of sensory sequences, we are particularly interested in *parsimonious* interpretations. To this end, we define the cost of a theory[27]:

Definition 18. Given a theory $\theta = (\phi, I, R, C)$, the **cost** of θ is

$$|I| + \sum \left\{ \; n+1 \mid \alpha_1 \wedge \ldots \wedge \alpha_n \circ \alpha_0 \in R, \circ \in \{\rightarrow, \ni\!\!\!\!-\} \; \right\}$$

Here, $cost(\theta)$ is just the total number of ground atoms in I plus the total number of unground atoms in the rules of R.

The key notion of this paper is the apperception task.

Definition 19. The input to an apperception task is a triple (S, ϕ, C) consisting of a sensory sequence S, a suitable frame ϕ, and a set C of (well-typed) constraints such that (i) each predicate in G_S appears in some constraint in C and (ii) S can be extended to satisfy C: there exists a sequence S' covering S such that $S'_i \models_\sigma c$ for each $c \in C$, for each ground substitution σ, and each state S'_i in S'.

Given such an input triple (S, ϕ, C), the **apperception task** is to find a lowest cost theory $\theta = (\phi', I, R, C')$ such that ϕ' extends ϕ, $C' \supseteq C$, and θ is a unified

[27]Note that this simple measure of cost does not depend on the constraints in C or the frame ϕ. There are various alternative more complex definitions of *cost*. We could, for example, use the Kolmogorov complexity [69] of θ: the size of the smallest program that can generate θ. Or we could use Levin complexity [70] and also take into account the log of the computation time needed to generate $\tau(\theta)$, up to the point where the trace first repeats.

C	$\forall X : t, on(X) \oplus \mathit{off}(X)$			
R				
I				
$\tau(\theta)$				
S_t	$on(a)$	$\mathit{off}(a)$	$on(a)$	$\mathit{off}(a)$
	t_1	t_2	t_3	t_4

(a) Empty theory

C	$\forall X : t, on(X) \oplus \mathit{off}(X)$			
R				
I	$on(a)$			
$\tau(\theta)$	$on(a)$	$on(a)$	$on(a)$	$on(a)$
S_t	$on(a)$	$\mathit{off}(a)$	$on(a)$	$\mathit{off}(a)$
	t_1	t_2	t_3	t_4

(b) One initial condition

C	$\forall X : t, on(X) \oplus \mathit{off}(X)$			
R	$on(X) \ni \mathit{off}(X)$			
I	$on(a)$			
$\tau(\theta)$	$on(a)$	$\mathit{off}(a)$	$\mathit{off}(a)$	$\mathit{off}(a)$
S_t	$on(a)$	$\mathit{off}(a)$	$on(a)$	$\mathit{off}(a)$
	t_1	t_2	t_3	t_4

(c) One rule

C	$\forall X : t, on(X) \oplus \mathit{off}(X)$			
R	$\mathit{off}(X) \ni on(X)$ $on(X) \ni \mathit{off}(X)$			
I	$on(a)$			
$\tau(\theta)$	$on(a)$	$\mathit{off}(a)$	$on(a)$	$\mathit{off}(a)$
S_t	$on(a)$	$\mathit{off}(a)$	$on(a)$	$\mathit{off}(a)$
	t_1	t_2	t_3	t_4

(d) Two rules

Figure 1: Four candidate theories attempting to explain the sequence (S_1, S_2, S_3, S_4) where $S_1 = S_3 = \{on(a)\}$ and $S_2 = S_4 = \{\mathit{off}(a)\}$. In each sub-figure, we show at the top the theory θ composed of constraints C (fixed), rules R, and initial conditions I; below, we show the trace of the theory, $\tau(\theta)$, and the state sequence(S_1, S_2, S_3, S_4). When the trace at time t fails to be a superset of the state S_t, we color the state S_t in red. Sub-figure **(a)** shows the initial theory, with empty initial conditions and rules. This fails to explain any of the sensory states. In **(b)**, we add one initial condition. The atom $on(a)$ persists throughout the time series because of the frame axiom. In **(c)**, we add one causal rule. This changes $on(a)$ at t_1 to $\mathit{off}(a)$ at t_2. But $\mathit{off}(a)$ then persists because of the frame axiom. In **(d)**, we add another causal rule. At this point, the trace $\tau(\theta)$ covers the sequence, and we have found a unified interpretation.

interpretation of S. Here, frame $\phi' = (T', O', P', V')$ **extends** $\phi = (T, O, P, V)$ if $T \subseteq T', O \subseteq O', P \subseteq P', V \subseteq V'$, and the κ mappings are extended conservatively.

Note that the input to an apperception task is more than just a sensory sequence S. It also contains a frame ϕ and a set C of constraints. It might be objected: why make things so complicated? Why not simply let the input to an apperception task be just the sequence S, and ask the system to produce some theory θ satisfying the unity conditions such that $S \sqsubseteq \tau(\theta)$? The reason that the input needs to contain types ϕ and constraints C to supplement S is that otherwise the task is severely under-constrained, as the following example shows.

Example 8. Suppose our sequence is $S = (\{on(a)\}, \{off(a)\}, \{on(a)\}, \{off(a)\}, \{on(a)\}, \{off(a)\})$. If we are not given any constraints (such as $\forall X : t, on(X) \oplus off(X)$), if we are free to construct any ϕ and any set C of constraints, then the following interpretation $\theta = (\phi, I, R, C)$ will suffice, where $\phi = (T, O, P, V)$:

$$\begin{array}{rcl} T & = & \{t\} \\ O & = & \{a{:}t\} \\ P & = & \{on(t), off(t), p(t), q(t)\} \\ V & = & \{X{:}t\} \end{array}$$

and I, R, C are defined as:

$$I = \left\{ \begin{array}{c} on(a) \\ off(a) \end{array} \right\} \quad R = \left\{ \quad \right\} \quad C = \left\{ \begin{array}{l} \forall X{:}t,\ on(X) \oplus p(X) \\ \forall X{:}t,\ off(X) \oplus q(X) \end{array} \right\}$$

Here we have introduced two latent predicates p and q which are incompatible with *on* and *off* respectively. But in this interpretation, *on* and *off* are not incompatible with each other, so the degenerate interpretation (where both are true at all times) is acceptable. This shows the need for including constraints on the input predicates as part of the task formulation. ◁

The apperception task can be generalized to the case where we are given as input, not a single sensory sequence S, but a set of m such sequences.

Definition 20. Given a set $\{S^1, ..., S^m\}$ of sensory sequences, a frame ϕ and constraints C such that each (S^i, ϕ, C) is a valid input to an apperception task as defined in Definition 19, the **generalized apperception task** is to find a lowest-cost theory $(\phi', \{\}, R, C')$ and lowest cost set $\{I^1, ..., I^m\}$ of initial conditions such that ϕ' extends ϕ, $C' \supseteq C$, and for each $i = 1..m$, (ϕ', I^i, R, C') is a unified interpretation of S^i.

3.3. Examples

In this section, we provide a worked example of an apperception task, along with two different unified interpretations. We wish to highlight that there are always many alternative ways of interpreting a sensory sequence, each with different latent information (although some may have higher cost than others).

Example 9. Suppose there are two sensors a and b, and each sensor can be *on* or *off*. Suppose the sensory sequence is $S_{1:7}$ where:

$$\begin{array}{ll}
S_1 = \{on(a), on(b)\} & S_2 = \{off(a), on(b)\} \\
S_3 = \{on(a), off(b)\} & S_4 = \{on(a), on(b)\} \\
S_5 = \{off(a), on(b)\} & S_6 = \{on(a), off(b)\} \\
S_7 = \{\} &
\end{array}$$

Let $\phi = (T, O, P, V)$ where $T = \{sensor\}$, $O = \{a, b\}$, $P = \{on(sensor), off(sensor)\}$, $V = \{X{:}sensor\}$. Let $C = \{\forall X{:}sensor,\ on(X) \oplus off(X)\}$. ◁

We shall show two different unified interpretations of this apperception task.

Example 10. One possible way of interpreting Example 9 is as follows. The sensors a and b are simple state machines that cycle between states p_1, p_2, and

p_3. Each sensor switches between *on* and *off* depending on which state it is in. When it is in states p_1 or p_2, the sensor is on; when it is in state p_3, the sensor is off. In this interpretation, the two state machines a and b do not interact with each other in any way. Both sensors are following the same state transitions. The reason the sensors are out of sync is because they start in different states.

The frame for this first unified interpretation is $\phi' = (T, O, P, V)$, where:

$$\begin{aligned}
T &= \{sensor\} \\
O &= \{a{:}sensor, b{:}sensor\} \\
P &= \{on(sensor), off(sensor), right(sensor, sensor), p_1(sensor), p_2(sensor), p_3(sensor)\} \\
V &= \{X{:}sensor, Y{:}sensor\}
\end{aligned}$$

The three unary predicates p_1, p_2, and p_3 are used to represent the three states of the state machine.

Our first unified interpretation is the tuple (ϕ', I, R, C'), where:

$$I = \left\{ \begin{array}{l} p_2(a) \\ p_1(b) \\ right(a, b) \\ right(b, a) \end{array} \right\} \quad R = \left\{ \begin{array}{l} p_1(X) \ni\!\!\!- \, p_2(X) \\ p_2(X) \ni\!\!\!- \, p_3(X) \\ p_3(X) \ni\!\!\!- \, p_1(X) \\ p_1(X) \rightarrow on(X) \\ p_2(X) \rightarrow on(X) \\ p_3(X) \rightarrow off(X) \end{array} \right\} \quad C' = \left\{ \begin{array}{l} \forall X{:}sensor,\ on(X) \oplus off(X) \\ \forall X{:}sensor,\ p_1(X) \oplus p_2(X) \oplus p_3(X) \\ \forall X{:}sensor,\ \exists! Y{:}sensor\ right(X, Y) \end{array} \right\}$$

The update rules R contain three causal rules (using $\ni\!\!\!-$) describing how each sensor cycles from state p_1 to p_2 to p_3, and then back again to p_1. For example, the causal rule $p_1(X) \ni\!\!\!- \, p_2(X)$ states that if sensor X satisfies p_1 at time t, then X satisfies p_2 at time $t + 1$. We know that X is a sensor from the variable typing information in ϕ'. R also contains three static rules (using $\rightarrow$) describing how the *on* or *off* attribute of a sensor depends on its state. For example, the static rule $p_1(Y) \rightarrow on(X)$ states that if X satisfies p_1 at time t, then X also satisfies *on* at time t.

The constraints C' state that (i) every sensor is (exclusively) either *on* or *off*, that every sensor is (exclusively) either p_1, p_2, or p_3, and that every sensor has another sensor to its *right*. The binary *right* predicate, or something like it, is needed to satisfy the constraint of spatial unity.

In this first interpretation, three new predicates are invented (p_1, p_2, and p_3) to represent the three states of the state machine.. In the next interpretation, we will introduce new invented objects instead of invented predicates.

Given the initial conditions I and the update rules R, we can use our interpretation to compute which atoms hold at which time-step. In this case, $\tau(\theta) = (A_1, A_2, ...)$ where $S_i \sqsubseteq A_i$. Note that this trace repeats: $A_i = A_{i+3}$. We can use the trace to predict the future values of our two sensors at time-step 7, since

$$A_7 = \{on(a), on(b), right(a, b), right(b, a), p_2(a), p_1(b)\}$$

As well as being able to predict future values, we can retrodict past values, or interpolate intermediate unknown values[28]. But although an interpretation provides the resources to "fill in" missing data, it has no particular bias to predicting future time-steps. The conditions which it is trying to satisfy (the unity conditions of Definition 17) do not explicitly insist that an interpretation must be able to predict future time-steps. Rather, the ability to predict the future (as well as the ability to retrodict the past, or interpolate intermediate values) is a *derived* capacity that emerges from the more fundamental capacity to "make sense" of the sensory sequence.

$\triangleleft$

Example 11. There are always infinitely many different ways of interpreting a sensory sequence. Next, we show a rather different interpretation of $S_{1:7}$ from that of Example 10. In our second unified interpretation, we no longer see sensors a and b as self-contained state-machines. Now, we see the states of the sensors as depending on their left and right neighbours. In this new interpretation, we no longer need the three invented unary predicates (p_1, p_2,

[28]This ability to "impute" intermediate unknown values is straightforward given an interpretation. Note that current neural methods for sequence learning are more comfortable predicting future values than imputing intermediate values.

and p_3), but instead introduce a new *object* [71].

Object invention is much less explored than predicate invention in inductive logic programming. But Dietterich et al. [72] anticipated the need for it:

> It is a characteristic of many scientific domains that we need to posit the existence of hidden objects in order to achieve compact hypotheses which explain empirical observations. We will refer to this process as object invention. For instance, object invention is required when unknown enzymes produce observable effects related to a given metabolic network.

Our new frame $\phi' = (T, O, P, V)$ is:

$$\begin{aligned} T &= \{sensor\} \\ O &= \{a{:}sensor, b{:}sensor, c{:}sensor\} \\ P &= \{on(sensor), \mathit{off}(sensor), right(sensor, sensor)\} \\ V &= \{X{:}sensor, Y{:}sensor\} \end{aligned}$$

In this new interpretation, imagine there is a one-dimensional cellular automaton with three cells, a, b, and (unobserved) c. The three cells wrap around: the right neighbour of a is b, the right neighbour of b is c, and the right neighbour of c is a. In this interpretation, the spatial relations are fixed. (There are other interpretations where this is not the case). The cells alternate between on and off according to the following simple rule: if X's right neighbour is on (respectively off) at t, then X is on (respectively off) at $t + 1$.

Note that objects a and b are the two sensors we are given, but c is a new unobserved latent object that we posit in order to make sense of the data. Many interpretations follow this pattern: new latent unobserved objects are posited to make sense of the changes to the sensors we are given.

Note further that part of finding an interpretation is constructing the spatial relation between objects; this is not something we are given, but something we must *construct*. In this case, we posit that the imagined cell c is inserted to the right of b and to the left of a.

We represent this interpretation by the tuple (ϕ', I, R, C'), where:

$$
\begin{aligned}
I &= \left\{ \begin{array}{lll} on(a) & on(b) & off(c) \\ right(a,b) & right(b,c) & right(c,a) \end{array} \right\} \\
R &= \left\{ \begin{array}{l} right(X,Y) \wedge off(X) \ni\!\!\!- off(Y) \\ right(X,Y) \wedge on(X) \ni\!\!\!- on(Y) \end{array} \right\} \\
C' &= \left\{ \begin{array}{l} \forall X{:}sensor,\ on(X) \oplus off(X) \\ \forall X{:}sensor,\ \exists! Y{:}sensor,\ right(X,Y) \end{array} \right\}
\end{aligned}
$$

Here, ϕ' extends ϕ, C' extends C, and the interpretation is unified. ◁

3.4. Properties of interpretations

In this section, we provide some results about the correctness and completeness of unified interpretations. The proofs are in the supplementary material.

Theorem 4. For each sensory sequence $S = (S_1, ..., S_t)$ and each unified interpretation θ of S, for each object x that features in S (i.e. x appears in some ground atom in some state S_i in S), for each state A_i in $\tau(\theta) = (A_1, A_2, ...)$, x features in A_i. In other words, if x features in any state in S, then x features in every state in $\tau(\theta)$.

The proof is straightforward from the definitions of conceptual and static unity and is placed in the supplementary material.

Theorem 4 provides some guarantee that admissible interpretations that satisfy the Kantian conditions will always be acceptable in the minimal sense that they always provide some value for each sensor. This theorem is important because it justifies the claim that a unified interpretation will always be able to support prediction (of future values), retrodiction (of previous values), and imputation (of missing values).

The next theorem is a form of "completeness", showing that every sensory sequence has some admissible interpretation that satisfies the Kantian conditions.

Theorem 5. For every apperception task (S, ϕ, C) there exists some unified interpretation $\theta = (\phi', I, R, C')$ where ϕ' extends ϕ and $C' \supseteq C$.

Proof. First, we define ϕ' given $\phi = (T, O, P, V)$. For each sensor x_i that features in S, $i = 1..n$, and each state S_j in S, $j = 1..m$, create a new unary predicate p^i_j. The intention is that $p^i_j(X)$ is true if X is the i'th object x_i at the j'th time-step. If $\kappa_O(x_i) = t$ then let $\kappa_P(p^i_j) = (t)$. For each type $t \in T$, create a new variable X_t where $\kappa_V(X_t) = t$. Let $\phi' = (T, O, P', V')$ where $P' = P \cup \{p^i_j \mid i = 1..n, j = 1..m\}$, and $V' = V \cup \{X_t \mid t \in T\}$.

Second, we define $\theta = (\phi', I, R, C')$. Let the initial conditions I be:

$$\left\{\ p^i_1(x_i) \mid i = 1..n\ \right\}$$

Let the rules R contain the following causal rules for $i = 1..n$ and $j = 1..m - 1$ (where x_i is of type t):

$$p^i_j(X_t) \ni p^i_{j+1}(X_t)$$

together with the following static rules for each unary atom $q(x_i) \in S_j$:

$$p^i_j(X_t) \rightarrow q(X_t)$$

and the following static rules for each binary atom $r(x_i, x_k) \in S_j$ (where x_i is of type t and x_k is of type t'):

$$p^i_j(X_t) \wedge p^k_j(Y_{t'}) \rightarrow r(X_t, Y_{t'})$$

We augment C to C' by adding the following additional constraints. Let P_t be the unary predicates for all objects of type t:

$$P_t = \left\{\ p^i_j \mid \kappa_O(x_i) = t, j = 1..m\ \right\}$$

Let $P_t = \{p'_1, ..., p'_k\}$. Then for each type t add a unary constraint:

$$\forall X_t : t, p'_1(X_t) \oplus ... \oplus p'_k(X_t)$$

It is straightforward to check that θ as defined satisfies the constraint of conceptual unity, and that the constraints C' are satisfied by each state in the trace $\tau(\theta)$.

To satisfy spatial unity, add a new "world" object w of a new type t_w and for each type t add a relation $part_t(t, t_w)$ and a constraint $\forall X : t, \exists! Y : t_w, part_t(X, Y)$. For each object x of type t, add an initial condition atom $part_t(x, w)$ to I. Thus, all the conditions of unity are satisfied, and θ is a unified interpretation of S. $\square$

Note that the interpretation provided by Theorem 5 is degenerate and unilluminating: it treats each object entirely separately (failing to capture any regularities between objects' behaviour) and treats every time-step entirely separately (failing to capture any laws that hold over multiple time-steps). This unilluminating interpretation provides an *upper bound* on the complexity needed to make sense of the sensory sequence.

4. The Apperception Engine

In this section, we describe our Apperception Engine. Given as input an apperception task (S, ϕ, C), the engine searches for a frame ϕ' and a theory $\theta = (\phi', I, R, C')$ where ϕ' extends ϕ, $C' \supseteq C$ and θ is a unified interpretation of S.

Definition 21. A **template** is a structure for circumscribing a large but finite set of theories. It is a frame together with constants that bound the complexity of the rules in the theory. Formally, a template χ is a tuple $(\phi, N_{\rightarrow}, N_{\ni}, N_B)$ where ϕ is a frame, $N_{\rightarrow}$ is the max number of static rules allowed in R, $N_{\ni}$ is the max number of causal rules allowed in R, and N_B is the max number of atoms allowed in the body of a rule in R.

Each template χ specifies a large (but finite) set of theories that conform to χ. Let $\Theta_{\chi,C} \subset \Theta$ be the subset of theories (ϕ, I, R, C') in Θ that conform to χ and where $C' \supseteq C$.

Our method, presented in Algorithm 1, is an anytime algorithm that enumerates templates of increasing complexity. For each template χ, it finds the

$\theta \in \Theta_{\chi,C}$ with lowest cost (see Definition 18) that satisfies the conditions of unity. If it finds such a θ, it stores it. When it has run out of processing time, it returns the lowest cost θ it has found from all the templates it has considered.

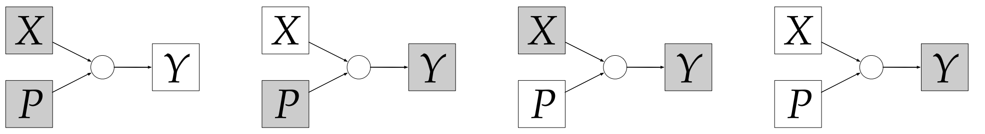
Note that the relationship between the complexity of a template and the cost of a theory satisfying the template is not always simple. Sometimes a theory of lower cost may be found from a template of higher complexity. This is why we cannot terminate as soon as we have found the first theory θ. We must keep going, in case we later find a lower cost theory from a more complex template.

Algorithm 1: The Apperception Engine algorithm in outline

```
input : $(S, \phi, C)$, an apperception task
output: $\theta^*$, a unified interpretation of $S$

1  $(s^*, \theta^*) \leftarrow (max(float), nil)$
2  foreach template $\chi$ extending $\phi$ of increasing complexity do
3      $\theta \leftarrow \text{argmin}_\theta\{cost(\theta) \mid \theta \in \Theta_{\chi,C}, S \sqsubseteq \tau(\theta), unity(\theta)\}$
4      if $\theta \neq nil$ then
5          $s \leftarrow cost(\theta)$
6          if $s < s^*$ then
7              $(s^*, \theta^*) \leftarrow (s, \theta)$
8          end
9      end
10     if exceeded processing time then
11         return $\theta^*$
12     end
13 end
```

The two non-trivial parts of this algorithm are the way we enumerate templates (line 3), and the way we find the lowest-cost theory θ for a given template χ (line 4). We describe how to find the lowest-cost theory in the next sub-section. We describe template iteration in detail in the supplementary material, but in



(a) Deduction (b) Abduction (c) Induction (d) Joint abduction and induction

Figure 2: The varieties of inference. Here, shaded elements are given, and unshaded elements must be generated.

outline: we enumerate templates of increasing complexity in such a way that every possible template is guaranteed to be visited eventually. To do this, we enumerate pairs (T, n) using the standard diagonalization procedure, where T is the number of types, and n is the current number of tuples to enumerate; for each (T, n) pair, we enumerate n 6-tuples of the form $(O, P, V, N_{\rightarrow}, N_{\ni}, N_B)$.

4.1. *Finding the best theory from a template*

The most complex part of Algorithm 1 is line 4:

$$\theta \leftarrow \underset{\theta}{\operatorname{argmin}}\{cost(\theta) \mid \theta \in \Theta_{\chi,C}, S \sqsubseteq \tau(\theta), unity(\theta)\}$$

Here, we search for a theory θ with the lowest cost (see Definition 18) such that θ conforms to the template χ and includes the constraints in C, such that $\tau(\theta)$ covers S, and θ satisfies the conditions of unity. In this sub-section, we explain in outline how this works.

Our approach combines abduction and induction to generate a unified interpretation θ. See Figure 2. Here, $X \subseteq \mathcal{G}$ is a set of facts (ground atoms), $P : \mathcal{G} \rightarrow \mathcal{G}$ is a procedure for generating the consequences of a set of facts, and $Y \subseteq \mathcal{G}$ is the result of applying P to X. If X and P are given, and we wish to generate Y, then we are performing *deduction*. If P and Y are given, and we wish to generate X, then we are performing *abduction*. If X and Y are given, and we wish to generate P, then we are performing *induction*. Finally, if only Y is given, and we wish to generate both X and P, then we are jointly performing abduction and induction. This is what the Apperception Engine does[29].

Our method is described in Algorithm 2. In order to jointly abduce a set I (of initial conditions) and induce sets R and C (of rules and constraints), we implement a Datalog$^{\ni}$ interpreter in ASP. This interpreter takes a set I of atoms (represented as a set of ground ASP terms) and sets R and C of rules and constraints (represented again as a set of ground ASP terms), and computes the trace of the theory $\tau(\theta) = (S_1, S_2, ...)$ up to a finite time limit.

Concretely, we implement the interpreter as an ASP program π_τ that computes $\tau(\theta)$ for theory θ. We implement the conditions of unity as ASP constraints in a program π_u. We implement the cost minimization as an ASP program π_m that counts the number of atoms in each rule and in each initialisation atom in I, and uses an ASP weak constraint [63] to minimize this total. Then we generate ASP programs representing the sequence S, the initial conditions, the rules and constraints. We combine the ASP programs together and ask the ASP solver (`clingo` [73]) to find a lowest cost solution. (There may be multiple solutions that have equally lowest cost; the ASP solver chooses one of the optimal answer sets). We extract a readable interpretation θ from the ground atoms of the answer set. In the supplementary material, we explain how Algorithm 2 is implemented in ASP.

5. Experiments

To evaluate the generality of our system, we tested it in a variety of domains: elementary (one-dimensional) cellular automata, drum rhythms and nursery tunes, sequence induction IQ tasks, multi-modal binding tasks, and occlusion tasks. These particular domains were chosen because they represent a diverse range of tasks that are simple for humans but are hard for state-

[29]In this respect, our system is similar to XHAIL [51]. But there are a number of differences. First, our program P contains causal rules and constraints as well as standard Horn clauses. Second, our conclusion Y is an infinite sequence $(S_1, S_2, ...)$ of sets, rather than a single set. Third, we add additional filters on acceptable theories in the form of the Kantian unity conditions.

Algorithm 2: Finding the lowest cost θ for sequence S and template χ.

```
input  : $S$, a sensory sequence
input  : $\chi = (\phi, N_{\rightarrow}, N_{\ni}, N_B)$, a template
input  : $C$, a set of constraints on the predicates of the sensory sequence
output: $\theta$, the simplest unified interpretation of $S$ that conforms to $\chi$
1  $\pi_S \leftarrow$ gen_input($S$)
2  $\pi_I \leftarrow$ gen_inits($\phi$)
3  $\pi_C \leftarrow$ gen_constraints($\phi, C$)
4  $\pi_R \leftarrow$ gen_rules($\phi, N_{\rightarrow}, N_{\ni}, N_B$)
5  $\Pi \leftarrow \pi_\tau \cup \pi_u \cup \pi_m \cup \pi_S \cup \pi_I \cup \pi_C \cup \pi_R$
6  $A \leftarrow$ clingo($\Pi$)
7  if satisfiable($A$) then
8      $\theta \leftarrow$ extract($A$)
9      return $\theta$
10 end
11 return nil
```

of-the-art machine learning systems. The tasks were chosen to highlight the difference between mere perception (the classification tasks that machine learning systems already excel at) and apperception (assimilating information into a coherent integrated theory, something traditional machine learning systems are not designed to do). Although all the tasks are data-sparse and designed to be easy for humans but hard for machines, in other respects the domains were chosen to maximize diversity: the various domains involve different sensory modalities, and some sensors provide binary discriminators while others are multi-class.

Our code and datasets are publicly available. See the supplementary material for details.

Our experiments (on the prediction task) are summarized in Table 2. Note that our accuracy metric for a single task is rather exacting: the model is accurate (boolean) on a task iff *every* hidden sensor value is predicted correctly[30]. It does not score any points for predicting most of the hidden values correctly. As can be seen from Table 2, our system is able to achieve good accuracy across all five domains.

In Table 3, we display Cohen's kappa coefficient [74] for the five domains. If a is the accuracy and r is the chance of randomly agreeing with the actual classification, then the kappa metric $\kappa = \frac{a-r}{1-r}$. Since our accuracy metric for a single task is rather exacting (since the model is accurate only if *every* hidden sensor value is predicted correctly), the chance r of random accuracy is very low. For example, in the ECA domain with 11 cells, the chance of randomly predicting correctly is 2^{-11}. Similarly, in the music domain, if there are 8 sensors and each can have 4 loudness levels, then the chance of randomly predicting

[30]The reason for using this strict notion of accuracy is that, as the domains are deterministic and noise-free, there is a simplest possible theory that explains the sensory sequence. In such cases where there is a correct answer, we wanted to assess whether the system found that correct answer exactly – not whether it was fortunate enough to come close while misunderstanding the underlying dynamics.

Domain	**Tasks (#)**	**Memory (MBytes)**	**Input size (bits)**	**Held out size (bits)**	**Accuracy (%)**
ECA	256	473.2	154.0	10.7	97.3%
Rhythm & music	30	2172.5	214.4	15.3	73.3%
Seek Whence	30	3767.7	28.4	2.5	76.7%
Multi-modal binding	20	1003.2	266.0	19.1	85.0%
Occlusion	20	604.3	109.2	10.1	90.0 %

Table 2: Results for prediction tasks on five domains. We show the mean information size of the sensory input, to stress the scantiness of our sensory sequences. We also show the mean information size of the held-out data. Our metric of accuracy for prediction tasks is whether the system predicted *every* sensor's value correctly.

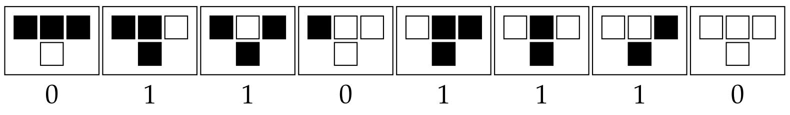


Figure 3: Updates for Rule 110. The top row shows the context: the target cell together with its left and right neighbour. The bottom row shows the new value of the target cell given the context. A cell is black if it is on and white if it is off.

correctly is 4^{-8}. Because the chance of random accuracy is so low, the kappa metric closely tracks the accuracy.

5.1. Elementary cellular automata

An Elementary Cellular Automaton (ECA) [75, 76] is a one-dimensional Cellular Automaton containing a circular array of cells. Each cell can be either *on* or *off*. The state of a cell depends only on its previous state and the previous state of its left and right neighbours.

Figure 3 shows one set of ECA update rules[31]. Each update specifies the new value of a cell based on its previous left neighbour, its previous value,

[31]This particular set of update rules is known as Rule 110. (110 is the decimal representation of the binary 01101110 update rule, as shown in Figure 3). This rule has been shown to be Turing-complete [76].

Domain	**Accuracy** (a)	**Random agreement** (r)	**Kappa metric** (κ)
ECA	97.3%	0.00048	0.972
Rhythm & music	73.3%	0.00001	0.732
Seek Whence	76.7%	0.16666	0.720
Multi-modal binding	85.0%	0.00003	0.849
Occlusion	90.0 %	0.03846	0.896

Table 3: Cohen's kappa coefficient for the five domains. Note that the chance of random agreement (r) is very low because we define accuracy as correctly predicting *every* sensor reading. When r is very low, the κ metric closely tracks the accuracy.

and its previous right neighbour. The top row shows the values of the left neighbour, previous value, and right neighbour. The bottom row shows the new value of the cell. There are 8 updates, one for each of the 2^3 configurations. In the diagram, the leftmost update states that if the left neighbour is *on*, and the cell is *on*, and its right neighbour is *on*, then at the next time-step, the cell will be turned *off*. Given that each of the 2^3 configurations can produce *on* or *off* at the next time-step, there are $2^{2^3} = 256$ total sets of update rules.

Given update rules for each of the 8 configurations, and an initial starting state, the trajectory of the ECA is fully determined. Figure 4 shows the state sequence for Rule 110 above from one initial starting state of length 11.

In our experiments, we attach sensors to each of the 11 cells, produce a sensory sequence, and then ask our system to find an interpretation that makes sense of the sequence. For example, for the state sequence of Figure 4, the

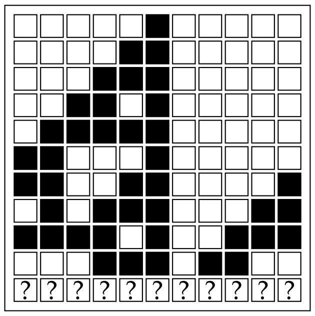


Figure 4: One trajectory for Rule 110. Each row represents the state of the ECA at one time-step. In this prediction task, the bottom row (representing the final time-step) is held out.

sensory sequence is $(S_1, ..., S_{10})$ where:

$$\begin{aligned}
S_1 &= \{off(c_1), off(c_2), off(c_3), off(c_4), off(c_5), on(c_6), off(c_7), off(c_8), off(c_()), off(c_{10}), off(c_{11})\} \\
S_2 &= \{off(c_1), off(c_2), off(c_3), off(c_4), on(c_5), on(c_6), off(c_7), off(c_8), off(c_()), off(c_{10}), off(c_{11})\} \\
S_3 &= \{off(c_1), off(c_2), off(c_3), on(c_4), on(c_5), on(c_6), off(c_7), off(c_8), off(c_()), off(c_{10}), off(c_{11})\} \\
S_4 &= \{off(c_1), off(c_2), on(c_3), on(c_4), off(c_5), on(c_6), off(c_7), off(c_8), off(c_()), off(c_{10}), off(c_{11})\} \\
S_5 &= \{off(c_1), on(c_2), on(c_3), on(c_4), on(c_5), on(c_6), off(c_7), off(c_8), off(c_()), off(c_{10}), off(c_{11})\} \\
&...
\end{aligned}$$

Note that we do *not* provide the spatial relation between cells. The system does not know that e.g. cell c_1 is directly to the left of cell c_2.

The input contains frame ϕ and initial constraints C, where:

$$\begin{aligned}
\phi &= \left\{ \begin{array}{l} T = \{cell\} \\ O = \{c_1{:}cell, c_2{:}cell, ..., c_{11}{:}cell\} \\ P = \{on(cell), off(cell)\} \\ V = \{X{:}cell, Y{:}cell, Z{:}cell\} \end{array} \right\} \\
C &= \left\{ \forall X{:}cell, on(X) \oplus off(X) \right\}
\end{aligned}$$

We applied our interpretation learning method to all $2^{2^3} = 256$ ECA rule-sets. For Rule 110 (see Figure 4), it found the following interpretation (ϕ', I, R, C'), where ϕ' extends ϕ with a binary relation $r(cell, cell)$, and:

$$I = \left\{ \begin{array}{llll} \mathit{off}(c_1) & \mathit{off}(c_2) & \mathit{off}(c_3) & \mathit{off}(c_4) \\ \mathit{off}(c_5) & \mathit{on}(c_6) & \mathit{off}(c_7) & \mathit{off}(c_8) \\ \mathit{off}(c_9) & \mathit{off}(c_{10}) & \mathit{off}(c_{11}) & r(c_1, c_{11}) \\ r(c_2, c_1) & r(c_3, c_2) & r(c_4, c_3) & r(c_5, c_4) \\ r(c_6, c_5) & r(c_7, c_6) & r(c_8, c_7) & r(c_9, c_8) \\ r(c_{10}, c_9) & r(c_{11}, c_{10}) & & \end{array} \right\}$$

$$R = \left\{ \begin{array}{l} r(X, Y) \wedge on(X) \wedge \mathit{off}(Y) \ni on(Y) \\ r(X, Y) \wedge r(Y, Z) \wedge on(X) \wedge on(Z) \wedge on(Y) \ni \mathit{off}(Y) \end{array} \right\}$$

$$C' = \left\{ \begin{array}{l} \forall X{:}cell,\ on(X) \oplus \mathit{off}(X) \\ \forall X{:}cell,\ \exists! Y\ r(X, Y) \end{array} \right\}$$

The two update rules are a very compact representation of the 8 ECA updates for Rule 110 in Figure 3: the first rule states that if the right neighbour is on, then the target cell switches from off to on, while the second rule states that if all three cells are on, then the target cell switches from on to off. Here, the system uses $r(X, Y)$ to mean that cell Y is immediately to the right of cell X. Note that *the system has constructed the spatial relation itself.* It was not given the spatial relation r between cells. It was given only the sensor readings of the 11 cells and constructed the spatial relationship r between the cells in order to make sense of the data.

Results. Given the 256 ECA rules, all with the same initial configuration, we treated the trajectories as a prediction task and applied our system to it. Our system was able to predict 249/256 correctly. In each of the 7/256 failure cases, the Apperception Engine found a unified interpretation, but this interpretation produced a prediction which was not the same as the oracle.

5.2. Drum rhythms and nursery tunes

We also tested our system on simple auditory perception tasks. Here, each sensor is an auditory receptor that is tuned to listen for a particular note or

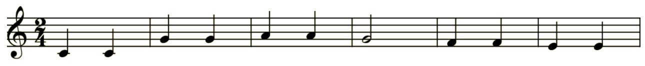


Figure 5: Twinkle Twinkle Little Star tune

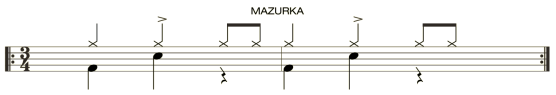


Figure 6: Mazurka rhythm

drum beat. In the tune tasks, there is one sensor for *C*, one for *D*, one for *E*, all the way to *HighC*. (There are no flats or sharps). In the rhythm tasks, there is one sensor listening out for bass drum, one for snare drum, and one for hi-hat. In both cases, each sensor can distinguish four loudness levels, between 0 and 3. When a note is pressed, it starts at max loudness (3), and then decays down to 0. Multiple notes can be pressed simultaneously.

For example, the *Twinkle Twinkle* tune generates the following sensor readings (assuming 8 time-steps for a bar):

$$\begin{aligned}
S_1 &= \{v(s_c, 3), v(s_d, 0), v(s_e, 0), v(s_f, 0), v(s_g, 0), v(s_a, 0), v(s_b, 0), v(s_{c*}, 0)\} \\
S_2 &= \{v(s_c, 2), v(s_d, 0), v(s_e, 0), v(s_f, 0), v(s_g, 0), v(s_a, 0), v(s_b, 0), v(s_{c*}, 0)\} \\
S_3 &= \{v(s_c, 3), v(s_d, 0), v(s_e, 0), v(s_f, 0), v(s_g, 0), v(s_a, 0), v(s_b, 0), v(s_{c*}, 0)\} \\
S_4 &= \{v(s_c, 2), v(s_d, 0), v(s_e, 0), v(s_f, 0), v(s_g, 0), v(s_a, 0), v(s_b, 0), v(s_{c*}, 0)\} \\
S_5 &= \{v(s_c, 1), v(s_d, 0), v(s_e, 0), v(s_f, 0), v(s_g, 3), v(s_a, 0), v(s_b, 0), v(s_{c*}, 0)\} \\
S_6 &= \{v(s_c, 0), v(s_d, 0), v(s_e, 0), v(s_f, 0), v(s_g, 2), v(s_a, 0), v(s_b, 0), v(s_{c*}, 0)\} \\
&\ldots
\end{aligned}$$

We tested our system on some simple rhythms (*Pop Rock*, *Samba*, etc.) and tunes (*Twinkle Twinkle*, *Three Blind Mice*, etc).

On the first two bars of *Twinkle Twinkle*, it finds an interpretation with 6 rules and 26 initial atoms. One of the rules states that when sensor *S* satisfies predicate p_1, then the value of the sensor S is set to the *max* loudness level:

$$p_1(S) \wedge max(L) \wedge v(S, L_2) \ni v(S, L)$$

The following rule states that when sensor S satisfies p_2, then the value decays:

$$p_2(S) \wedge succ(L, L2) \wedge v(S, L_2) \ni v(S, L)$$

Clearly, p_1 and p_2 are exclusive unary predicates used to determine whether a note is currently being pressed or not.

The next rule states that when the finger F satisfies predicate q_1, then the note which the finger is on becomes pressed:

$$q_1(F) \wedge part(F, S) \wedge p_2(S) \ni p_1(S)$$

Here, the system is using q_1 to indicate whether or not the finger is down. It uses the other predicates $q_2, q_3, ...$ to indicate which state the finger is in (and hence which note the finger should be on), and the other rules to indicate when to transition from one state to another.

Results. Recall that our accuracy metric is stringent and only counts a prediction as accurate if *every* sensor's value is predicted correctly. In the rhythm and music domain, this means the Apperception Engine must correctly predict the loudness value (between 0 and 3) for each of the sound sensors. There are 8 sensors for tunes and 3 sensors for rhythms.

When we tested the Apperception Engine on the 20 drum rhythms and 10 nursery tunes, our system was able to predict 22/30 correctly. The complexities of the interpretations are shown in Table 4. Note that the interpretations found are large and complex programs by the standards of state-of-the-art ILP systems. In *Mazurka*, for example, the interpretation contained 16 update rules with 44 body atoms. In *Three Blind Mice*, the interpretation contained 10 update rules and 34 initialisation atoms making a total of 44 clauses.

In the 8 cases where the Apperception Engine failed to predict correctly, this was because the system failed to find a unified interpretation of the sensory

Task	# static rules	# cause rules	# body atoms	# inits	# clauses	complexity
Twinkle Twinkle	2	4	9	26	32	45
Eighth Note Drum Beat	4	8	29	13	25	62
Stairway to Heaven	4	8	30	13	25	63
Three Blind Mice	2	8	17	34	44	69
Twist	4	12	40	16	32	84
Mazurka	4	12	44	14	30	86

Table 4: The complexity of the interpretations found for rhythm and tune prediction tasks

given. It wasn't that the system found an interpretation which produced the wrong prediction. Rather, in the 8 failure cases, it was simply unable to find a unified interpretation within the memory and time limits. In the ECA tasks, by contrast, the system always found some unified interpretation for each of the 256 tasks, but some of these interpretations produced the wrong prediction.

5.3. "Seek Whence" and C-test sequence induction IQ tasks

Hofstadter introduced the **Seek Whence**[32] domain in [24]. The task is, given a sequence $s_1, ..., s_t$ of symbols, to predict the next symbol s_{t+1}. Typical "Seek Whence" tasks include[33]:

- **b, b, b, c, c, b, b, b, c, c, b, b, b, c, c, ...**
- **a, f, b, f, f, c, f, f, f, d, f, f, ...**
- **b, a, b, b, b, b, b, c, b, b, d, b, b, e, b, ...**

Hofstadter called the third sequence the "theme song" of the Seek Whence project because of its difficulty. There is a "perceptual mirage" in the sequence

[32]The name is a pun on "sequence". See also the related Copycat domain [77].

[33]Hofstadter used natural numbers, but we transpose the sequences to letters, to bring them in line with the Thurstone letter completion problems [78] and the C-test [79].

because of the sub-sequence of five b's in a row that makes it hard to see the intended pattern: $(b, x, b)^*$ for ascending x.

It is important to note that these tasks, unlike the tasks in the ECA domain or in the rhythm and music domain (where the sensor readings are produced by an oracle), have a certain subjective element. There are always many different ways of interpreting a finite sequence. Given that these different interpretations will provide different continuations, why privilege some continuations over others?

When Hernández-Orallo designed the C-test [79, 80, 25, 81], one of his central motivations was to address this "subjectivity" objection via the concept of *unquestionability*. If we are given a particular programming language for generating sequences, then a sequence $s_{1:T}$ is **unquestionable** if it is not the case that the smallest program π that generates $s_{1:T}$ is rivalled by another program π' that is almost as small, where π and π' have different continuations after T time-steps.

Consider, for example, the sequence **a, b, b, c, c, ...** This sequence is highly questionable because there are two interpretations which are very similar in length (according to most programming languages), one of which parses the sequence as $(a), (b, b), (c, c, c), (d, d, d, d), ...$ and the other of which parses the sequence as a sequence of pairs $(a, b), (b, c), (c, d), ...$ Hernández-Orallo generated the C-test sequences by enumerating programs from a particular domain-specific language, executing them to generate a sequence, and then restricting the computer-generated sequences to those that are unquestionable.

For our set of sequence induction tasks, we combined sequences from Hofstadter's "Seek Whence" dataset (transposed from numbers to letters) together with sequences from the C-test. The C-test sequences are unquestionable by construction, and we also observed (by examining the size of the smallest interpretations) that Hofstadter's sequences were unquestionable with respect to Datalog$^{\ni}$. This goes some way to answering the "subjectivity" objection[34].

[34]Some may still be concerned that the definition of unquestionability is relative to a particular

There have been a number of attempts to implement sequence induction systems using domain-specific knowledge of the types of sequence to be encountered. Simon et al [82] implemented the first program for solving Thurstone's letter sequences [78]. Meredith [83] and Hofstadter [24] also used domain-specific knowledge: after observing various types of commonly recurring patterns in the "Seek Whence" sequences, they hand-crafted a set of procedures to detect the patterns. Although their search algorithm is general, the patterns over which it is searching are hand-coded and domain-specific.

Hernández-Orallo et al [25] identify a limitation of the above approaches: "In fact, for most approaches the system does not learn to solve the problems but it is *programmed* to solve the problems. In other words, the task is hard-coded into the program and it can be easier to become 'superhuman' in many specific tasks, as happens with chess, draughts, some kinds of planning, and many other tasks. But humans are not programmed to do intelligence tests." If solutions to sequence induction or IQ tasks are to be useful in general models of cognition, it is essential that we do not provide domain-specific solutions to those tasks. What we want is a *general-purpose domain-agnostic* perceptual system that can solve sequence induction tasks "out of the box" *without hard-coded domain-specific knowledge* [84].

The Apperception Engine described in this paper is just such a general-purpose domain-agnostic perceptual system. We tested it on 30 sequences (see the supplementary material for details), and it got 76.6% correct (23/30 correct, 3/30 incorrect and 4/30 timeout).

We illustrate our system on the "theme song" of the Seek Whence project: **b, a, b, b, b, b, b, c, b, b, d, b, b, e, b, b, ...** Let the sensory sequence be $S_{1:16}$

domain-specific language, and the Kolmogorov complexity of a sequence depends on the choice of language. Hernández-Orallo [81] discusses this issue at length.

where:

$$\begin{array}{ll} S_1 = \{value(s, l_b)\} & S_2 = \{value(s, l_a)\} \\ S_3 = \{value(s, l_b)\} & S_4 = \{value(s, l_b)\} \\ S_5 = \{value(s, l_b)\} & S_6 = \{value(s, l_b)\} \\ S_7 = \{value(s, l_b)\} & S_8 = \{value(s, l_c)\} \\ S_9 = \{value(s, l_b)\} & S_{10} = \{value(s, l_b)\} \end{array}$$

The input frame ϕ and initial constraints C are:

$$\phi = \left\{ \begin{array}{l} T = \{sensor, letter\} \\ O = \{s{:}sensor, l_a{:}letter, l_b{:}letter, l_c{:}letter, ...\} \\ P = \{value(sensor, letter)\} \\ V = \{X{:}sensor, L{:}letter\} \end{array} \right\}$$

$$C = \left\{ \forall X{:}sensor,\ \exists! L\ value(X, L) \right\}$$

When our system is run on this sensory input, the first few templates are unable to find a solution. The first template that is expressive enough to admit a solution is one where there are three latent objects. The interpretation found is (ϕ', I, R, C') where ϕ' extends ϕ with objects c_1, c_2, c_3, binary predicates $p(cell, letter)$ and $r(cell, cell)$, and unary predicates q_1 and q_2; I, R, and C' are defined as:

$$I = \left\{ \begin{array}{llll} p(c_1, l_b) & p(c_2, l_b) & p(c_3, l_a) & q_1(c_3) \\ q_2(c_1) & q_2(c_2) & r(c_1, c_3) & r(c_3, c_2) \\ r(c_2, c_1) & part(s, c_1) & & \end{array} \right\}$$

$$R = \left\{ \begin{array}{l} part(X, Y) \wedge p(Y, L) \rightarrow value(X, L) \\ r(Y, Y_2) \wedge part(X, Y) \ni\!\!\!- part(X, Y_2) \\ part(X, Y) \wedge q_1(Y) \wedge succ(L, L_2) \wedge p(Y, L) \ni\!\!\!- p(Y, L_2) \end{array} \right\}$$

$$C' = \left\{ \begin{array}{l} \forall X{:}sensor,\ \exists! L\ value(X, L) \\ \forall Y{:}cell,\ \exists! L\ p(Y, L) \\ \forall Y{:}cell\ q_1(Y) \oplus q_2(Y) \\ \forall Y{:}cell,\ \exists! Y_2\ r(Y, Y_2) \end{array} \right\}$$

In this interpretation, the sensor moves between the three latent cell objects in the order $c_1, c_3, c_2, c_1, c_3, c_2, c_1, c_3, c_2, ...$ The two unary predicates q_1 and q_2 are

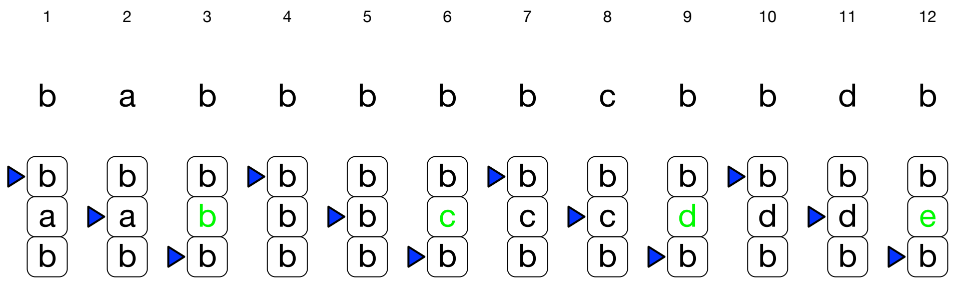


Figure 7: A visualization of the Apperception Engine's interpretation of the "theme song" Seek Whence sequence **b, a, b, b, b, b, b, c, b, b, d, b, b, e, b, b, ...** We show the trace $\tau(\theta) = A_1, A_2, ...,$ of this interpretation for the first 12 time steps. The t'th column represents the state at time t. Each column contains the time index t, the sensor reading S_t, the values of the three cells c_1, c_2, c_3 at time t, and the position of the sensor s at t. The only moving object is the sensor, represented by a triangle, that moves between the three cells from top to bottom and then repeats. Note that the middle cell c_2's value changes when the sensor passes over it; we change the color of the cell letter to indicate when the cell's value has changed.

used to divide the cells into two types. Effectively, q_1 is interpreted as a "cell that increases its letter" while q_2 is interpreted as a "static cell". The static rule states that the sensor inherits its value from the p-value of the cell it is on. The causal rule $r(Y, Y_2) \wedge part(X, Y) \ni part(X, Y_2)$ states that the sensor moves from left to right along the cells. The causal rule $part(X, Y) \wedge q_1(Y) \wedge succ(L, L_2) \wedge p(Y, L) \ni p(Y, L_2)$ states that q_1 cells increase their p-value when a sensor is on them. This is an intelligible and satisfying interpretation of the sensory sequence. See Figure 7.

Results. Given the 30 Seek Whence sequences, we treated the trajectories as a prediction task and applied our system to it. Our system was able to predict 23/30 correctly. For the 7 failure cases, 4 of them were due to the system not being able to find any unified interpretation within the memory and time limits, while in 3 of them, the system found a unified interpretation that produced the "incorrect" prediction. The complexities of the interpretations are shown in Table 5. The first key point we want to emphasize here is that our system

Sequence	# static rules	# cause rules	# body atoms	# inits	# clauses	complexity
abcde...	0	1	1	7	8	10
fafbfcfdf...	1	2	6	7	10	18
babbbbbcbbdbbe...	1	2	6	14	17	25
aababcabcdabcde...	3	5	19	7	15	39
abccddeeefff...	3	5	21	8	16	42
baabbbaaaabbbbb...	3	5	23	7	15	43

Table 5: The complexity of the interpretations found for Seek Whence prediction tasks

was able to achieve human-level performance[35] on these tasks without hand-coded domain-specific knowledge. This is a *general* system for making sense of sensory data that, when applied to the Seek Whence domain[36], is able to solve these particular problems. The second point we want to stress is that our system did not learn to solve these sequence induction tasks after seeing many previous examples[37]. On the contrary: our system had never seen *any* such sequences before; it confronts each sequence *de novo*, without prior experience. This system is, to the best of our knowledge, the first such general system that is able to achieve such a result.

5.4. Binding tasks

We wanted to see whether our system could handle traditional problems from cognitive science "out of the box", without needing additional task-specific information. We used probe tasks to evaluate two key issues: binding and occlusion.

[35]See Meredith [83] for empirical results 25 students on the"Blackburn dozen" Seek Whence problems.

[36]The only domain-specific information provided is the *succ* relation on letters.

[37]Machine learning approaches to these tasks need thousands of examples before they can learn to predict. See for example [85].

The binding problem [86] is the task of recognising that information from different sensory modalities should be collected together as different aspects of a single external object. For example, you hear a buzzing and a siren in your auditory field and you see a insect and an ambulance in your visual field. How do you associate the buzzing with the insect-appearance as aspects of one object, and the siren with the ambulance appearance as aspects of a separate object?

To investigate how our system handles such binding problems, we tested it on the following multi-modal variant of the ECA described above. Here, there are two types of sensor. The light sensors have just two states: black and white, while the touch sensors have four states: fully un-pressed (0), fully pressed (3), and two intermediate states (1, 2). After a touch sensor is fully pressed (3), it slowly depresses, going from states 2 to 1 to 0 over 3 time-steps. In this example, we chose Rule 110 (the Turing-complete ECA rule) with the same initial configuration as in Figure 4, as described earlier. In this multi-modal variant, there are 11 light sensors, one for each cell in the ECA, and two touch sensors on cells 3 and 11. See Figure 8.

Suppose we insist that the frame contains no binary relations connecting any of the sensors together. Suppose there is no relation in the given frame between light sensors, no relation between touch sensors, and no relation between light sensors and touch sensors. Now, in order to satisfy the constraint of spatial unity, there must be some indirect connection between any two sensors. But if there are no direct relations between the sensors, *the only way our system can satisfy the constraint of spatial unity is by positing latent objects, directly connected to each other, that the sensors are connected to*. Thus the latent objects are the intermediaries through which the various sensors are indirectly connected. See the supplementary material for details of the interpretations found.

We ran 20 multi-modal binding experiments, with different ECA rules, different initial conditions, and the touch sensors attached to different cells. The results are shown in Table 6.

l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	t_1	t_2
W	W	W	W	W	B	W	W	W	W	W	0	0
W	W	W	W	B	B	W	W	W	W	W	0	0
W	W	W	B	B	B	W	W	W	W	W	0	0
W	W	B	B	W	B	W	W	W	W	W	3	0
W	B	B	B	B	B	W	W	W	W	W	3	0
B	B	W	W	W	B	W	W	W	W	W	2	0
B	B	W	W	B	B	W	W	W	W	B	1	3
W	B	W	B	B	B	W	W	W	B	B	0	3
B	B	B	B	W	B	W	W	B	B	B	3	3
W	W	W	B	B	B	W	B	B	W	W	2	2
?	?	?	?	?	?	?	?	?	?	?	?	?

110

Figure 8: A multi-modal trace of ECA Rule 110 with eleven light sensors (left) $l_1, ..., l_{11}$ and two touch sensors (right) t_1, t_2 attached to cells 3 and 11. Each row represents the states of the sensors for one time-step. For this prediction task, the final time-step is held out.

Domain	**# Tasks**	**Memory**	**Time**	**% Correct**
Multi-modal binding	20	1003.2	2.4	85.0%
Occlusion	20	604.3	2.3	90.0 %

Table 6: The two types of probe task. We show mean memory in megabytes and mean solution time in hours.

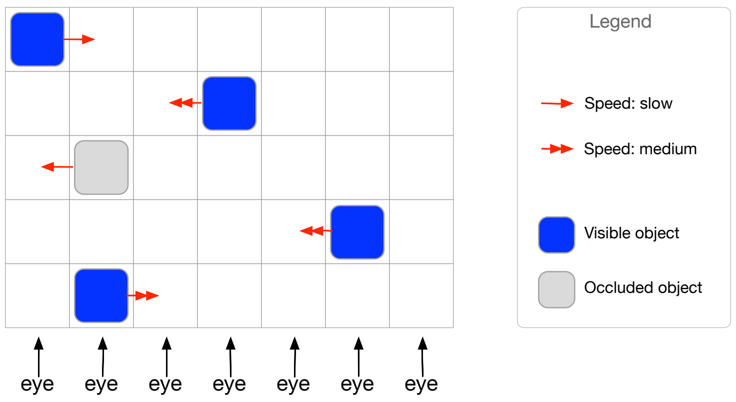


Figure 9: An occlusion task

5.5. *Occlusion tasks*

Neural nets that predict future sensory data conditioned on past sensory data struggle to solve occlusion tasks because it is hard to inject into them the prior knowledge that objects persist over time. Our system, by contrast, is constrained to posit objects that persist over time.

To test our system's ability to solve occlusion problems, we generated a set of tasks of the following form: there is a 2D grid of cells in which objects move horizontally. Some move from left to right, while others move from right to left, with wrap around when they get to the edge of a row. The objects move at different speeds. Each object is placed in its own row, so there is no possibility of collision. There is an "eye" placed at the bottom of each column, looking up. Each eye can only see the objects in the column it is placed in. An object is occluded if there is another object below it in the same column. See Figure 9.

The system receives a sensory sequence consisting of the positions of the moving objects whenever they are visible. The positions of the objects when they are occluded is used as held-out test data to verify the predictions of the model. This is an imputation task. Further details are in the supplementary material.

We generated 20 occlusion tasks by varying the size of the grid, the number of moving objects, their direction and speed. Our system was able to solve these tasks without needing additional domain-specific information. The results are shown in Table 6. In the cases where the Apperception Engine failed to predict correctly, this was because it ran out of time or memory when trying to find a unified interpretation.

6. Empirical comparisons with other approaches

In this section, we evaluate our system experimentally and attempt to establish the following claims. First, we claim the test domains of Section 5 represent a challenging set of tasks. We show that these domains are challenging by providing baselines that are unable to interpret the sequences. Second, we claim our system is general in that it can handle retrodiction and imputation as easily as it can handle prediction tasks. We show in extensive tests that results for retrodicting earlier values and imputing intermediate values are comparable with results for predicting future values. Third, we claim that the various features of the system (the Kantian unity conditions and the cost minimization procedure) are essential to the success of the system. In ablation tests, where individual features are removed, the system performs significantly worse. Fourth, we claim that the particular program synthesis technique we use is efficient when compared with state-of-the-art program synthesis methods. Specifically, we show how ILASP (a state-of-the-art ILP system [87, 88, 89, 90]) is capable of solving some of the smaller tasks, but struggles for the larger tasks.

6.1. Our domains are challenging for existing baselines

To evaluate whether our domains are indeed sufficiently challenging, we compared our system against four baselines[38]. The first **constant** baseline

[38]We also considered using a hidden Markov model (HMM) as a baseline. However, as Ghahramani emphasizes ([37], Section 5), a HMM represents each of the exponential number of propositional states separately, and thus fails to generalize in the way that a first-order rule induction system does. Thus, although we did not test it, we are confident that a HMM would not perform well on our tasks.

always predicts the same constant value for every sensor for each time-step. The second **inertia** baseline always predicts that the final hidden time-step equals the penultimate time-step. The third **MLP** baseline is a fully-connected multilayer perceptron (MLP) that looks at a window of earlier time-steps to predict the next time-step. The fourth **LSTM** baseline is a recurrent neural net based on the long short-term memory (LSTM) architecture [91]. Although the neural network architectures are very different from our system, we tried to give the various systems access to the same amount of information. This means in particular that:

- Since our system interprets the sequence without any knowledge of the other sequences, *we do not allow the neural net baselines to train on any sequences other than the one they are currently given*. Each neural net baseline is only allowed to look at the single sensory sequence it is given. This extreme paucity of training data is unusual for data-hungry methods like neural nets, and explains their weak results. But we stress that this is the only fair comparison, given that the Apperception Engine, also, only has access to a single sequence.
- Since our system interprets the sequence without knowing anything about the relative spatial position of the sensors (it does not know, in the ECA examples, the spatial locations of the cells), we do not give the neural nets a (1-dimensional) convolutional structure, even though this would help significantly in the ECA tasks.

The neural baselines are designed to exploit potential statistical patterns that are indicative of hidden sensor states. In the MLP baseline, we formulate the problem as a multi-class classification problem, where the input consists in a feature representation $\mathbf{x}$ of relevant context sensors, and a feed-forward network is trained to predict the correct state $\mathbf{y}$ of a given sensor in question. In the prediction task, the feature representation comprises one-hot representations for the state of every sensor in the previous two time steps before the hidden sensor. The training data consists of the collection of all observed states in an episode (as potential hidden sensors), together with the respective history before. Samples with incomplete history window (at the beginning of the episode) are discarded.

The MLP classifier is a 2-layer feed-forward neural network, which is trained on all training examples derived from the current episode (thus no cross-episode transfer is possible). We restrict the number of hidden neurons to (20, 20) for the two layers, respectively, in order to prevent overfitting given the limited number of training points within an episode. We use a learning rate of 10^{-3} and train the model using the *Adam* optimiser for up to 200 epochs, holding aside 10% of data for early stopping.

Given that the input is a temporal sequence, a recurrent neural network (that was designed to model temporal dynamics) is a natural choice of baseline. But we found that the LSTM performs only slightly better than the MLP on Seek Whence tasks, and worse on the other tasks. The reason for this is that the extremely small amount of data (a single temporal sequence consisting of a small number of time-steps) does not provide enough information for the high capacity LSTM to learn desirable gating behaviour. The simpler and more constrained MLP with fewer weights is able to do slightly better on some of the tasks, yet both neural baselines achieve low accuracy in absolute terms.

Figure 10 show the results. Clearly, the tasks are very challenging for all four baseline systems. See the supplementary material for further details.

6.2. Our system handles retrodiction and imputation just as easily as prediction

To verify that our system is just as capable of retrodicting earlier values and imputing missing intermediate values as it is at predicting future values, we ran tests where the unseen hidden sensor values were at the first time step (in the case of retrodiction) or randomly scattered through the time-series (in the case of imputation). We made sure that the number of hidden sensor values was the same for prediction, retrodiction, and imputation.

Figure 11 shows the results. The results are significantly lower for retrodiction in the ECA tasks, but otherwise comparable. The reason for retrodiction's

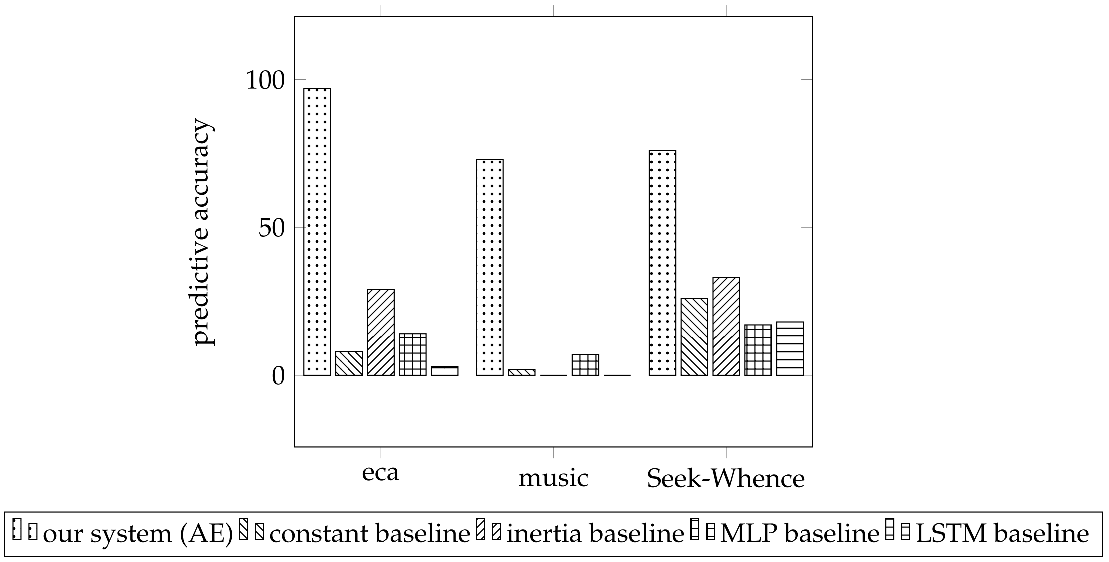


Figure 10: Comparison with baselines. We display predictive accuracy on the held-out final time-step.

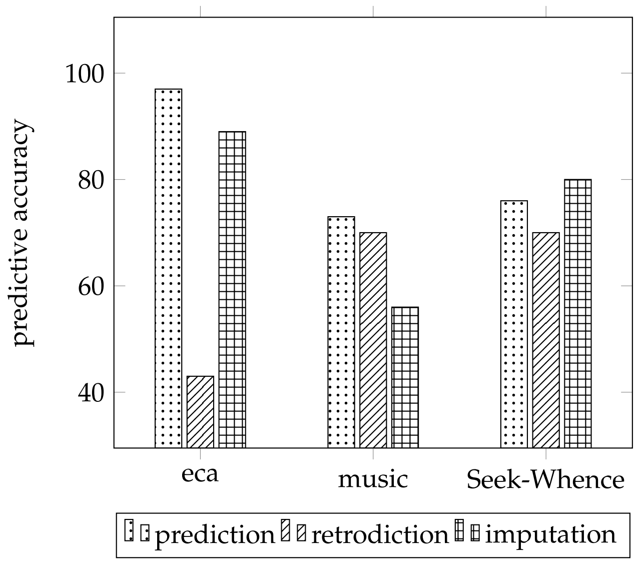


Figure 11: Comparing prediction with retrodiction and imputation. In retrodiction, we display accuracy on the held-out initial time-step. In imputation, a random subset of atoms are held-out; the held-out atoms are scattered throughout the time-series. In other words, there may be different held-out atoms at different times. The number of held-out atoms in imputation matches the number of held-out atoms in prediction and retrodiction.

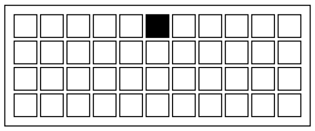


Figure 12: One trajectory for ECA Rule # 0. This trajectory shows how information is lost as we progress through time. Here, clearly, retrodiction (where the first row is held-out) is much harder than prediction (where the final row is held-out).

lower performance on ECA is that for a particular initial configuration there are a significant number (more than 50%) of the ECA rules that wipe out all the information in the current state after the first state transition, and all subsequent states then remain the same. So, for example, in Rule # 0, one trajectory is shown in Figure 12. Here, although it is possible to predict the future state from earlier states, it is not possible to retrodict the initial state given subsequent states.

The results for imputation are comparable with the results for prediction. Although the results for rhythm and music are lower, the results on Seek Whence are slightly higher (see Figure 11).

6.3. The features of our system are essential to its performance

To verify that the unity conditions are doing useful work, we performed a number of experiments in which the various conditions were removed, and compared the results. We ran four ablation experiments. In the first, we removed the check that the theory's trace covers the input sequence: $S \sqsubseteq \tau(\theta)$ (see Definition 12). In the second, we removed the check on conceptual unity. Removing this condition means that the unary predicates are no longer connected together via exclusion relations $\oplus$, and the binary predicates are no longer constrained by $\exists!$ conditions. (See Definition 17). In the third ablation test, we removed the check on spatial unity. Removing this condition means allowing objects which are not connected via binary relations. In the fourth ablation test, we removed the cost minimization part of the system. Removing this minimization means that the system will return the first interpretation it

	ECA	**Rhythm & Music**	**Seek Whence**
Full system (AE)	97.3%	73.3%	76.7%
No check that $S \sqsubseteq \tau(\theta)$	5.1%	3.0%	4.6%
No conceptual unity	5.3%	0.0%	6.7%
No spatial unity	95.7%	73.3%	73.3%
No cost minimization	96.7%	56.6%	73.3%

Table 7: Ablation experiments. We display predictive accuracy on the final held-out time-step.

finds, irrespective of size.

The results of the ablation experiments are displayed in Table 7.

The first ablation test, where we remove the check that the generated sequence of sets of ground atoms respects the original sensory sequence ($S \sqsubseteq \tau(\theta)$), performs very poorly. Of course, if the generated sequence does not cover the given part of the sensory sequence, it is highly unlikely to accurately predict the held-out part of the sensory sequence. This test is just a sanity check that our evaluation scripts are working as intended.

The second ablation test, where we remove the check on conceptual unity, also performs poorly. The reason is that without constraints, there are no incompossible atoms. Recall from Section 3.1 that two atoms are incompossible if there is some $\oplus$ constraint or some $\exists!$ constraint that means the two atoms cannot be simultaneously true. But recall from Section 3.1 that the frame axiom allows atoms that were true at the previous time-step to also be true at the next time-step unless the old atoms are incompossible with some new atom: we add α to H_t if α is in H_{t-1} and there is no atom in H_t that is incompossible with α. But if there are no incompossible atoms, then all previous atoms are always added. Therefore, if there are no $\oplus$ and $\exists!$ constraints, then the set of true atoms monotonically increases over time. This in turn means that state information becomes meaningless, as once it becomes true it remains always true, and cannot be used to convey information.

When we remove the spatial unity constraint, the results for the rhythm tasks are identical, but the results for the ECA and Seek Whence tasks are lower. The reason why the results are identical for the rhythm tasks is because the background knowledge provided (the r relation on notes, see Section 5.2) means that the spatial unity constraint is guaranteed to be satisfied. The reason why the results are lower for ECA tasks is because interpretations that fail to satisfy spatial unity contain disconnected clusters of cells (e.g. cells $\{c_1, ..., c_5\}$ are connected by r in one cluster, while cells $\{c_6, ..., c_{11}\}$ are connected in another cluster, but $\{c_1, ..., c_5\}$ and $\{c_6, ..., c_{11}\}$ are disconnected). Interpretations with disconnected clusters tend to generalize poorly and hence predict with less accuracy. The reason why the results are only slightly lower for the Seek Whence tasks is because the lowest cost unified interpretation for most of these tasks also happens to satisfy spatial unity. We have assumed, in deference to Kant, that the spatial unity constraint does useful work. But it is at least conceptually possible that this constraint is not needed in at least some domains. In future work, we shall test the Apperception Engine in a much wider variety of domains, to understand when spatial unity is and is not important[39].

The results for the fourth ablation test, where we remove the cost minimization, are broadly comparable with the full system in ECA and Seek Whence, but are markedly worse in the rhythm / music tasks. But even if the results were comparable in all tasks, there are independent reasons to want to minimize the size of the interpretation. Shorter interpretations are more human-readable, and transfer better to new situations (since they tend to be more general, as they have fewer atoms in the bodies of the rules). On the other hand, it is significantly more expensive to compute the lowest cost theory than it is to just find any unified theory (see the complexity results in the supplementary material). So in some domains, where the difference in accuracy is minimal, the cost minimization step can be avoided.

[39]Some philosophers (e.g. Strawson [92]) have questioned whether the spatial unity constraint is, in fact, necessary.

7. Conclusion

This paper is an attempt to answer a key question of unsupervised learning: what does it mean to "make sense" of a sensory sequence? Our answer is broadly Kantian [93]: making sense means positing a collection of objects that persist over time, with attributes that change over time, according to intelligible laws. As well as providing a precise formalization of this task, we also provide a concrete implementation of a system that is able to make sense of the sensory stream. We have tested the Apperception Engine in a variety of domains; in each domain, we tested its ability to predict future values, retrodict previous values, and impute missing intermediate values. Our system achieves good results across the board.

Of particular note is that it is able to achieve human performance on challenging sequence induction IQ tasks. We stress, once more, that the system was not hard-coded to solve these tasks. Rather, it is a general *domain-independent* sense-making system that is able to apply its general architecture to the particular problem of Seek Whence induction tasks, and is able to solve these problems "out of the box" without human hand-engineered help. We also stress, again, that the system did not learn to solve these sequence induction tasks by being presented with hundreds of training examples[40]. Indeed, the system had never seen a *single* such task before. Instead, it applied its general sense-making urge to each individual task, *de novo*. We also stress that the interpretations produced are human readable and can be used to provide explanations and justifications of the decisions taken: when the Apperception Engine produces an interpretation, we can not only see what it predicts will happen next, but we can also understand *why* it thinks this is the right continuation. We believe these results are highly suggestive, and shows that a sense-making component such as this will be a key aspect of any general intelligence.

[40]Barrett et al [85] train a neural network to learn to solve Raven's progressive matrices from thousands of training examples.

Our architecture, an unsupervised program synthesis system, is a purely symbolic system, and as such, it inherits two key advantages of ILP systems [67]. First, the interpretations produced are *interpretable*. Because the output is symbolic, it can be read and verified by a human[41]. Second, it is very *data-efficient*. Because of the language bias of the Datalog$^{\ni}$ language, and the strong inductive bias provided by the Kantian unity conditions, the system is able to make sense of extremely short sequences of sensory data, without having seen any others.

However, the system in its current form has some limitations that we wish to make explicit. First, the sensory input must be discretized before it can be passed to the system. We assume some prior system has already discretized the continuous sensory values by grouping them into buckets. In future work, we shall describe various systems that can jointly learn to discretize and interpret at once. We are currently evaluating a number of different ways to do this. In one prototype, we have implemented a differentiable version of the Apperception Engine in TensorFlow [94]. We define a loss that is at a global minimum when the theory both satisfies the unity conditions and covers the sensory sequence. In this prototype, searching for unified interpretations can be implemented via stochastic gradient descent. However, neural networks tend to get stuck in local minima when solving continuous relaxations of discrete problems [95], so we are also considering alternative ways of jointly discretizing and finding unified interpretations. In a second prototype, we discretize the input by simulating a binary neural network in ASP. By adding weak constraints penalizing the number of weights, we can also guarantee that the binary neural network (that discretizes the input) is sparse.

Second, our current implementation assumes all causal rules are fully deterministic. In future work, we shall address this limitation by adding exogenous

[41] Large machine-generated programs are not always easy to understand. But machine-generated symbolic programs are certainly easier to understand than the weights of a neural network. See Muggleton et al [23] for an extensive discussion.

actions that affect the state but do not themselves need to be explained. This will allow us to model non-determinism as the result of exogenous activity.

Third, our system does not currently handle noise in the sensory input. All sensory information is assumed to be significant, and the system will strive to find an explanation of *every* sensor reading. There is no room, in the current implementation, for the idea that some sensor readings are inaccurate. In future work, we plan to use Bayesian inference to minimize the combined log size of the interpretation together with the data-reconstruction error. This will allow our system to support noisy input [22].

Fourth, and perhaps most significantly, the enormous size of the search space (see the supplementary material for details) means that our system is currently restricted to small problems. Because of the complexity of the search space, we must be careful to choose domains that do not require too many objects, predicates, or variables. The Apperception Engine is able to synthesize significantly larger programs than state-of-the-art program induction systems. (See Table 4, and Appendix F of the Supplementary Material for a comparison with ILASP). Nevertheless, we are considering a range of alternative approaches to program synthesis that we hope might scale better to much larger problems. This is a challenging open research problem.

We hope in future work to address these limitations. But we believe that, even in its current form, the Apperception Engine shows considerable promise as a prototype of what a general-purpose domain-independent sense-making machine must look like.

Acknowledgements

We are very grateful to Andrew Cropper, Jessica Hamrick, Mark Law, Matko Bošnjak, Murray Shanahan and Nando de Freitas for extensive feedback.

Appendix A. Notation

$\mathcal{T}$	the set of all types
$\mathcal{O}$	the set of all objects
$\mathcal{P}$	the set of all predicates
$\mathcal{V}$	the set of all variables
$\mathcal{G}$	the set of all ground atoms formed from $\mathcal{T}, \mathcal{O}, \mathcal{P}$
$\mathcal{U}$	the set of all unground atoms formed from $\mathcal{T}, \mathcal{V}, \mathcal{P}$
$\mathcal{R}$	all rules of the form $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_{n+1}$ and $\alpha_1 \wedge ... \wedge \alpha_n \ni\!\!\rightarrow \alpha_{n+1}$, where $\alpha_i \in \mathcal{U}$
$\mathcal{C}$	all constraints of the form $\forall X, p_1(X) \oplus ... \oplus p_n(X)$ and $\forall X\ \exists! Y\ p(X, Y)$
S	a sequence of sets of ground atoms: $S \in \left(2^{\mathcal{G}}\right)^*$
S_i	a sequence at index i
$S_{i:j}$	the slice of a sequence from indices i to j inclusive
$\sqsubseteq$	$S \sqsubseteq S'$ if for all indices i of S, $S_i \subseteq S'_i$
(T, O, P, V)	a frame containing types $T \subseteq \mathcal{T}$, objects $O \subseteq \mathcal{O}$, predicates $P \subseteq \mathcal{P}$, and variables $V \subseteq \mathcal{V}$
(ϕ, I, R, C)	a theory consisting of frame ϕ, initial atoms $I \subseteq \mathcal{G}$, rules $R \subseteq \mathcal{R}$ and constraints $C \subseteq \mathcal{C}$
ϕ	a frame of the form (T, O, P, V)
θ	a theory of the form (ϕ, I, R, C)
Φ	the set of all possible frames
Θ	the set of all theories
$\tau : \Theta \rightarrow \left(2^{\mathcal{G}}\right)^*$	the trace function that takes a theory θ and produces an infinite sequence $S \in \left(2^{\mathcal{G}}\right)^*$
$\kappa_O : O \rightarrow T$	the type of an object
$\kappa_P : P \rightarrow T^*$	the arg-types of a predicate
$\kappa_V : V \rightarrow T$	the type of a variable

$G_S \subseteq \mathcal{G}$	all the ground atoms in a sequence S
$G_\phi \subseteq \mathcal{G}$	all the well-typed ground atoms according to frame ϕ
$U_\phi \subseteq \mathcal{U}$	all the well-typed unground atoms according to frame ϕ
$R_\phi \subseteq \mathcal{R}$	all the well-typed static rules and causal rules according to frame ϕ
$C_\phi \subseteq \mathcal{C}$	all the well-typed constraints according to frame ϕ
σ	A ground substitution mapping variables in $\mathcal{V}$ to objects in $\mathcal{O}$
$\Sigma = \mathcal{O}^{\mathcal{V}}$	The set of all ground substitutions
$\Sigma_\phi \subseteq \Sigma$	The set of all ground substitutions compatible with the type information in frame ϕ
X, Y, Z	Object-level variables appearing in constraints and update rules
x, y, z	Meta-level variables ranging over objects in $\mathcal{O}$
α, β	Meta-level variables ranging over atoms (ground or unground)
f	A functor (a structure-preserving map) between theories
$N_{\rightarrow}$	The max number of static rules allowed by the template
$N_{\ni}$	The max number of causal rules allowed by the template
N_B	The max number of atoms in the body of a rule allowed by the template
χ	A template of the form $(\phi, N_{\rightarrow}, N_{\ni}, N_B)$ that defines a large but finite set of theories
$\Theta_{\chi,C}$	The (finite) set of theories (ϕ, I, R, C') that conform to template χ and where $C \subseteq C'$

References

References

[1] J. Oh, X. Guo, H. Lee, R. L. Lewis, S. Singh, Action-conditional video prediction using deep networks in atari games, in: Advances in neural information processing systems, 2015, pp. 2863–2871.

[2] M. Mathieu, C. Couprie, Y. LeCun, Deep multi-scale video prediction beyond mean square error, arXiv preprint arXiv:1511.05440.

[3] M. Watter, J. Springenberg, J. Boedecker, M. Riedmiller, Embed to control: A locally linear latent dynamics model for control from raw images, in: Advances in neural information processing systems, 2015, pp. 2746–2754.

[4] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, K. Kavukcuoglu, Reinforcement learning with unsupervised auxiliary tasks, arXiv preprint arXiv:1611.05397.

[5] A. Lerer, S. Gross, R. Fergus, Learning physical intuition of block towers by example, arXiv preprint arXiv:1603.01312.

[6] A. Bhattacharyya, M. Malinowski, B. Schiele, M. Fritz, Long-term image boundary prediction, arXiv preprint arXiv:1611.08841.

[7] C. Finn, S. Levine, Deep visual foresight for planning robot motion, in: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 2786–2793.

[8] T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al., Imagination-augmented agents for deep reinforcement learning, arXiv preprint arXiv:1707.06203.

[9] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, A. Agarwala, Video frame synthesis using deep voxel flow., in: ICCV, 2017, pp. 4473–4481.

[10] S. Chiappa, S. Racaniere, D. Wierstra, S. Mohamed, Recurrent environment simulators, arXiv preprint arXiv:1704.02254.

[11] L. Buesing, T. Weber, S. Racaniere, S. Eslami, D. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, et al., Learning and querying fast generative models for reinforcement learning, arXiv preprint arXiv:1802.03006.

[12] K. Friston, A theory of cortical responses, Philosophical transactions of the Royal Society B: Biological sciences 360 (1456) (2005) 815–836.

[13] K. Friston, The history of the future of the bayesian brain, NeuroImage 62 (2) (2012) 1230–1233.

[14] A. Clark, Whatever next? predictive brains, situated agents, and the future of cognitive science, Behavioral and brain sciences 36 (3) (2013) 181–204.

[15] L. R. Swanson, The predictive processing paradigm has roots in kant, Frontiers in systems neuroscience 10 (2016) 79.

[16] Y. Huang, Y. Cheng, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Z. Chen, Gpipe: Efficient training of giant neural networks using pipeline parallelism, arXiv preprint arXiv:1811.06965.

[17] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.

[18] G. W. Leibniz, G. W. F. von Leibniz, Leibniz: New essays on human understanding, Cambridge University Press, 1996.

[19] I. Kant, Critique of pure reason, trans, Cambridge University Press, 1781.

[20] J. Dewey, Leibniz's New Essays Concerning the Human Understanding: A Critical Exposition, SC Griggs, 1888.

[21] W. James, Talks to Teachers on Psychology and to Students on Some of Life's Ideals, Vol. 12, Harvard University Press, 1983.

[22] K. Ellis, A. Solar-Lezama, J. Tenenbaum, Unsupervised learning by program synthesis, in: Advances in neural information processing systems, 2015, pp. 973–981.

[23] S. H. Muggleton, U. Schmid, C. Zeller, A. Tamaddoni-Nezhad, T. Besold, Ultra-strong machine learning: comprehensibility of programs learned with ilp, Machine Learning 107 (7) (2018) 1119–1140.

[24] D. R. Hofstadter, Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought, Basic books, 1995.

[25] J. Hernández-Orallo, F. Martínez-Plumed, U. Schmid, M. Siebers, D. L. Dowe, Computer models solving intelligence test problems: Progress and implications, Artificial Intelligence 230 (2016) 74–107.

[26] K. J. W. Craik, The nature of explanation, Vol. 445, CUP Archive, 1967.

[27] M. Hegarty, Mechanical reasoning by mental simulation, Trends in cognitive sciences 8 (6) (2004) 280–285.

[28] D. Gentner, A. L. Stevens, Mental models, Psychology Press, 2014.

[29] P. N. Johnson-Laird, Inference with mental models, The Oxford handbook of thinking and reasoning (2012) 134–145.

[30] P. L. Harris, The work of the imagination, Blackwell Publishers Oxford, 2000.

[31] T. Gerstenberg, J. B. Tenenbaum, Intuitive theories, Oxford handbook of causal reasoning (2017) 515–548.

[32] M. Garnelo, K. Arulkumaran, M. Shanahan, Towards deep symbolic reinforcement learning, arXiv preprint arXiv:1609.05518.

[33] L. Kaiser, M. Babaeizadeh, P. Milos, Model based reinforcement learning for atari, arXiv preprint arXiv:1903.00374v2.

[34] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science 362 (6419) (2018) 1140–1144.

[35] J. B. Hamrick, Analogues of mental simulation and imagination in deep learning, Current Opinion in Behavioral Sciences 29 (2019) 8–16.

[36] L. E. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state markov chains, The annals of mathematical statistics 37 (6) (1966) 1554–1563.

[37] Z. Ghahramani, An introduction to hidden markov models and bayesian networks, in: Hidden Markov models: applications in computer vision, World Scientific, 2001, pp. 9–41.

[38] V. Feinberg, A. Wan, I. Stoica, M. Jordan, J. Gonzalez, S. Levine, Model-based value expansion for efficient model-free reinforcement learning, in: Proceedings of the 35th International Conference on Machine Learning (ICML 2018), 2018.

[39] A. Nagabandi, G. Kahn, R. S. Fearing, S. Levine, Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 7559–7566.

[40] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al., Interaction networks for learning about objects, relations and physics, in: Advances in neural information processing systems, 2016, pp. 4502–4510.

[41] M. B. Chang, T. Ullman, A. Torralba, J. B. Tenenbaum, A compositional object-based approach to learning physical dynamics, arXiv preprint arXiv:1612.00341.

[42] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. F. Fei-Fei, J. Tenenbaum, D. L. Yamins, Flexible neural representation for physics prediction, in: Advances in Neural Information Processing Systems, 2018, pp. 8813–8824.

[43] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia, Graph networks as learnable physics engines for inference and control, arXiv preprint arXiv:1806.01242.

[44] A. Bhattacharyya, M. Malinowski, B. Schiele, M. Fritz, Long-term image boundary prediction, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[45] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, in: Advances in Neural Information Processing Systems, 2018, pp. 2455–2467.

[46] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, J. Wu, Reasoning about physical interactions with object-oriented prediction and planning, arXiv preprint arXiv:1812.10972.

[47] A. Zhang, A. Lerer, S. Sukhbaatar, R. Fergus, A. Szlam, Composable planning with attributes, arXiv preprint arXiv:1803.00512.

[48] Z. Xu, Z. Liu, C. Sun, K. Murphy, W. Freeman, J. Tennenbaum, J. Wu, Unsupervised discovery of parts, structure, and dynamics, in: Proceedings of the International Conference on Learning Representations, ICLR 2019, New Orleans, USA, 6-9 May 2019, pp. 1418–1424.

[49] M. Asai, A. Fukunaga, Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[50] M. Asai, Unsupervised grounding of plannable first-order logic representation from images, arXiv preprint arXiv:1902.08093.

[51] O. Ray, Nonmonotonic abductive inductive learning, Journal of Applied Logic 7 (3) (2009) 329–340.

[52] K. Inoue, T. Ribeiro, C. Sakama, Learning from interpretation transition, Machine Learning 94 (1) (2014) 51–79.

[53] N. Katzouris, A. Artikis, G. Paliouras, Incremental learning of event definitions with inductive logic programming, Machine Learning 100 (2-3) (2015) 555–585.

[54] E. Michelioudakis, A. Skarlatidis, G. Paliouras, A. Artikis, Online structure learning using background knowledge axiomatization, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 232–247.

[55] N. Katzouris, A. Artikis, G. Paliouras, Online learning of event definitions, Theory and Practice of Logic Programming 16 (5-6) (2016) 817–833.

[56] E. Michelioudakis, A. Artikis, G. Paliouras, Semi-supervised online structure learning for composite event recognition, arXiv preprint arXiv:1803.00546.

[57] R. Kowalski, M. Sergot, A logic-based calculus of events, in: Foundations of knowledge base management, Springer, 1989, pp. 23–55.

[58] R. Kowalski, Predicate logic as programming language, in: IFIP congress, Vol. 74, 1974, pp. 569–544.

[59] K. R. Apt, Logic programming., Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B) 1990 (1990) 493–574.

[60] J. W. Lloyd, Foundations of logic programming, Springer Science & Business Media, 2012.

[61] M. H. Van Emden, R. A. Kowalski, The semantics of predicate logic as a programming language, Journal of the ACM (JACM) 23 (4) (1976) 733–742.

[62] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming., in: ICLP/SLP, Vol. 88, 1988, pp. 1070–1080.

[63] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, Asp-core-2: Input language format, ASP Standardization Working Group.

[64] R. Kowalski, Logic for problem solving, Vol. 7, Ediciones Díaz de Santos, 1979.

[65] S. Ceri, G. Gottlob, L. Tanca, Logic programming and databases, Springer Science & Business Media, 2012.

[66] P. Alvaro, W. R. Marczak, N. Conway, J. M. Hellerstein, D. Maier, R. Sears, Dedalus: Datalog in time and space, in: International Datalog 2.0 Workshop, Springer, 2010, pp. 262–281.

[67] R. Evans, E. Grefenstette, Learning explanatory rules from noisy data, Journal of Artificial Intelligence Research 61 (2018) 1–64.

[68] B. Longuenesse, Kant and the Capacity to Judge, Princeton: Princeton UP, 1998.

[69] A. N. Kolmogorov, On tables of random numbers, Sankhyā: The Indian Journal of Statistics, Series A (1963) 369–376.

[70] L. A. Levin, Universal sequential search problems, Problemy Peredachi Informatsii 9 (3) (1973) 115–116.

[71] D. C. Dennett, Real patterns, The journal of Philosophy 88 (1) (1991) 27–51.

[72] T. G. Dietterich, P. Domingos, L. Getoor, S. Muggleton, P. Tadepalli, Structured machine learning: the next ten years, Machine Learning 73 (1) (2008) 3.

[73] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Clingo= asp+ control: Preliminary report, arXiv preprint arXiv:1405.3694.

[74] J. Cohen, A coefficient of agreement for nominal scales, Educational and psychological measurement 20 (1) (1960) 37–46.

[75] S. Wolfram, Statistical mechanics of cellular automata, Reviews of modern physics 55 (3) (1983) 601.

[76] M. Cook, Universality in elementary cellular automata, Complex systems 15 (1) (2004) 1–40.

[77] M. Mitchell, Analogy-making as perception: A computer model, MIT Press, 1993.

[78] L. L. Thurstone, T. G. Thurstone, Factorial studies of intelligence., Psychometric monographs.

[79] J. Hernandez-Orallo, N. Minaya-Collado, A formal definition of intelligence, in: Proceedings of International Symposium of Engineering of Intelligent Systems (EIS 98), 1998, pp. 146–163.

[80] J. Hernandez-Orallo, Beyond the turing test, Journal of Logic, Language and Information 9 (4) (2000) 447–466.

[81] J. Hernández-Orallo, The measure of all minds: evaluating natural and artificial intelligence, Cambridge University Press, 2017.

[82] H. A. Simon, K. Kotovsky, Human acquisition of concepts for sequential patterns., Psychological Review 70 (6) (1963) 534.

[83] M. J. E. Meredith, Seek-whence: A model of pattern perception, Tech. rep., Indiana Univ., Bloomington (USA) (1986).

[84] T. R. Besold, O. DE, The artificial jack of all trades: The importance of generality in approaches to human-level artificial intelligence, in: Proceedings of the Third Annual Conference on Advances in Cognitive Systems ACS, 2015, p. 18.

[85] D. G. Barrett, F. Hill, A. Santoro, A. S. Morcos, T. Lillicrap, Measuring abstract reasoning in neural networks, arXiv preprint arXiv:1807.04225.

[86] A. Holcombe, The binding problem, The sage encyclopedia of perception.

[87] M. Law, A. Russo, K. Broda, Inductive learning of answer set programs, in: Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings, 2014, pp. 311–325. `doi:10.1007/978-3-319-11558-0_22`.
URL `https://doi.org/10.1007/978-3-319-11558-0_22`

[88] M. Law, A. Russo, K. Broda, Learning weak constraints in answer set programming, Theory and Practice of Logic Programming 15 (4-5) (2015) 511–525.

[89] M. Law, A. Russo, K. Broda, Iterative learning of answer set programs from context dependent examples, Theory and Practice of Logic Programming 16 (5-6) (2016) 834–848.

[90] M. Law, A. Russo, K. Broda, The complexity and generality of learning answer set programs, Artificial Intelligence 259 (2018) 110–146.

[91] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[92] P. Strawson, The bounds of sense: An essay on Kant's critique of pure reason, Routledge, 2018.

[93] D. J. Chalmers, R. M. French, D. R. Hofstadter, High-level perception, representation, and analogy: A critique of artificial intelligence methodology, Journal of Experimental & Theoretical Artificial Intelligence 4 (3) (1992) 185–211.

[94] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[95] A. L. Gaunt, M. Brockschmidt, R. Singh, N. Kushman, P. Kohli, J. Taylor, D. Tarlow, Terpret: A probabilistic programming language for program induction, arXiv preprint arXiv:1608.04428.