This submission is mostly a copy of "Scott Viteri ELK contest submissions #1", with highlighted changes as per Mark Xu's request over email.

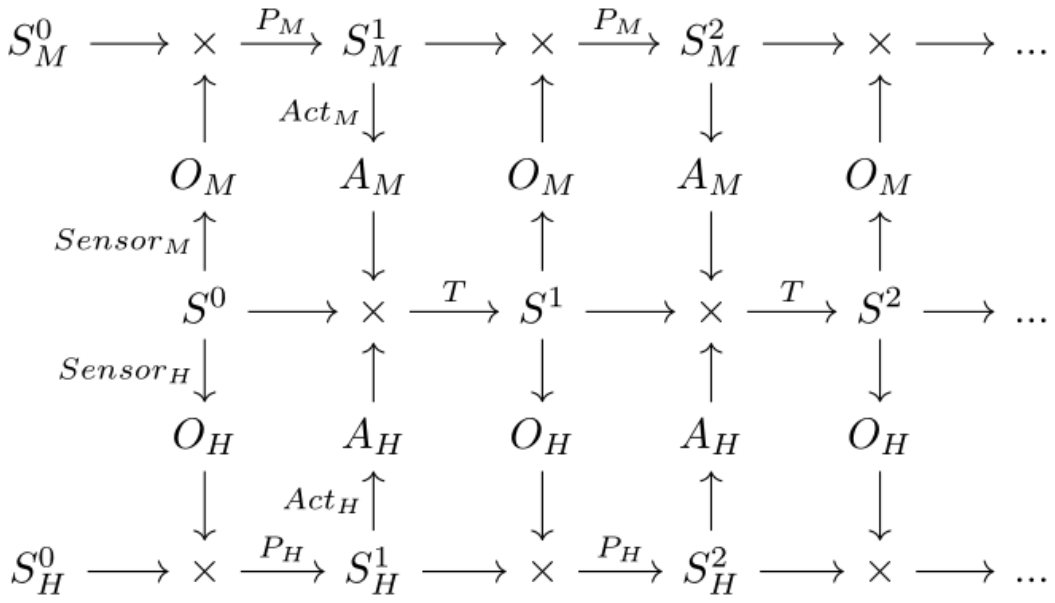
Abstraction, Agents, and ELK

Definitions

We start with a few definitions.
Let S be the set of world states, and assume a world transition function $T : S \times A \rightarrow S$, where A is a set of actions.
We define an agent X to be a 4-tuple of functions $Sensor_x$, $Predict_x$, $Extract_x$, and Act_x, where:

- $Sensor_x$ is a surjective "abstraction" function of type $S \rightarrow O_x$, where O_x is a type for X's abstracted observations.
- $Predict_x$ is a prediction function of type $S_x \times O_x \rightarrow S_x$, where S_x represents the type of X's internal knowledge of the world.
- $Extract_x$ is the agent's interface with the outside world of type $S_x \rightarrow O_x$ (i.e., what it expects the world to look like based on its internal knowledge).
- $Action_x$ is a function from S_x to A_x, where A_x denotes X's contribution to actions on the world. If there are multiple agents, we assume that an action A contains one action from each agent.

Using these definitions, we can model a simple instance of the ELK problem. Suppose there are two agents: a sophisticated AI called M and a human agent H. Initially, these agents and the world are in states $S^0\square$, $S^0\square$, and S^0, respectively. At each timestep, both agents observe the current state of the world with their sensors, update their worldviews using their prediction function, produce an action, and then apply that action to the current world state, resulting in a new world state. This process is depicted in the figure below.



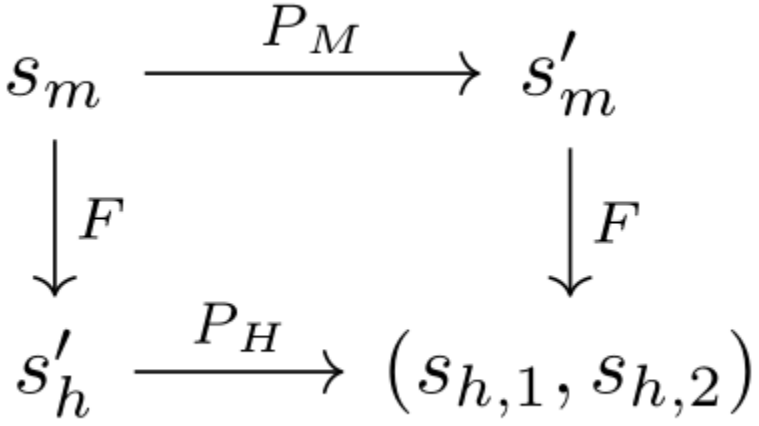
We have the human's utility function $U_H : S_H \to \mathbb{Q}$, which is defined on the human's model of reality. We would like to lift U_H to a version $U_R : S_R \to \mathbb{Q}$ that the machine can use to influence the world in way that is agreeable to the human.

One way to do this is to somehow learn a mapping $F : S_R \to S_H$. Then we can derive $U_R = F \circ U_H$. Naively, we could sample world simulations to create sequences of (S_R, S_H) and use these to learn a mapping. However, this method might lose important *latent* information in M. Namely, we want to include situations where the sequence of world states is sufficiently complicated that the human is now confused about the true state of the world. This could include situations where the sensor has been tampered with, for instance, but only the AI is sophisticated enough to have noticed. If we map from S_R to S_H using plain co-occurrence, we will learn a function that forgets that the sensors are now faulty. Instead, we want a human concept that is semantically as close as possible to the machine concept, not just one that commonly co-occurs with a machine concept.

An additional issue with the previous idea is that the embedding spaces of S_R and S_H are arbitrary, and the machine could use a representation of S_R that creates arbitrary distortions in F by contorting $Predict_R$ edges. Let G_x denote the state machine with nodes as states in S_x and edges given by $Predict_x$. Instead we could use shortest path length in G_h to measure the distance between states.

To this end, we propose learning a ~~reversible~~ function $F : S_R \to S_H$ by minimizing the difference between two paths from a state s_R, one in which the machine's prediction function is used and one in which the human's prediction function is used. Concretely, we propose minimizing $Dist(s_{H1}, s_{H2}) + \lambda|U(s_{H1})-U(s_{H2})|$ where $s_{H1} = F(Predict_R(s_R,o_R))$ and $s_{H2} = Predict_H(F(s_R),o_R)$, Dist is shortest path distance in G_h, and observations o_R and o_H are generated by the same underlying state S. $Predict_R$ is trained in the first place to enact a

function on a finite set of state machine vertices, whether through one-hot encodings and softmax or with rounding of input and output.



Details

- We previously tried to minimize the difference between Predict□ and F' ◦ Predict□ ◦ F, but this provides an extra degree of freedom in F and F', allowing them to memorize and perfectly reconstruct S□. Having a single F disallows this possibility.
- The utility regularization term provides a notion of differential caring about accuracy in predictions that are more utility relevant.
- Structural reasoning can lead to non-unique maps. Imagine S□ = S□ = {Diamond, NoDiamond}. This optimization cannot distinguish between identity and inverting F's. Another example is S□ = S□ = ℤ and Predict□ = Predict□ = λx.x+1, where could learn F = λx.k+x for any integer k. We are assuming that in practice connections between states are sufficiently dense that such exact ties do not occur.
- One might think that F does not need to use the accuracy of Predict□ since the diagram paths are meeting in S□. This cannot be true because F ◦ Predict□ = Predict□ ◦ F implies Predict□ = F^{-1} ◦ Predict□ ◦ F.

Reasoning about the behaviour

It is useful to make a few simplifying assumptions to reason about the behavior of this setup. Suppose that there are no actions, the human and machine have the same observation type, and that S□ and S□ are both 𝕆². Fix a specific background observation, and Predict□ and Predict□'s state machines form directed graphs on 𝕆², which we'll denote G□ and G□. F will map nodes and edges from G□ to G□. Then a self loop in G□ must correspond to a self-loop in G□, and a cycle in G□ must correspond to at least one cycle in G□. In general, I conjecture that this mapping will preserve the global structure of the AI's state machine as much as possible

while using the human edges, and it will prefer to glue machine nodes together rather than tear them apart.

Experiment. We will test this theoretical model using a cellular automata setup. We will train predictive agents H and M, and then learn the knowledge representation mapping F. We will measure the degree to which machine states can be reconstructed from human states, under various degrees of separation between the sophistication of human and machine models.

Concretely, let we will let the machine and human sensors compute abstractions of the automata cells (e.g., 2 by 2 maximum) and let the Actions correspond to bit flips in arbitrary positions. Both agents' prediction functions can be implemented as neural networks and supervised by runs using some automata transition function (e.g., the Game of Life) and comparing output of the Extract function to observations. The machine can be trained longer to simulate a more in-depth model of reality for the purposes of the experiment, and in reality Predict□ could be supervised by human predictions.

Counterexamples

ELK seems particularly hard when these Bayes nets satisfy the following properties:

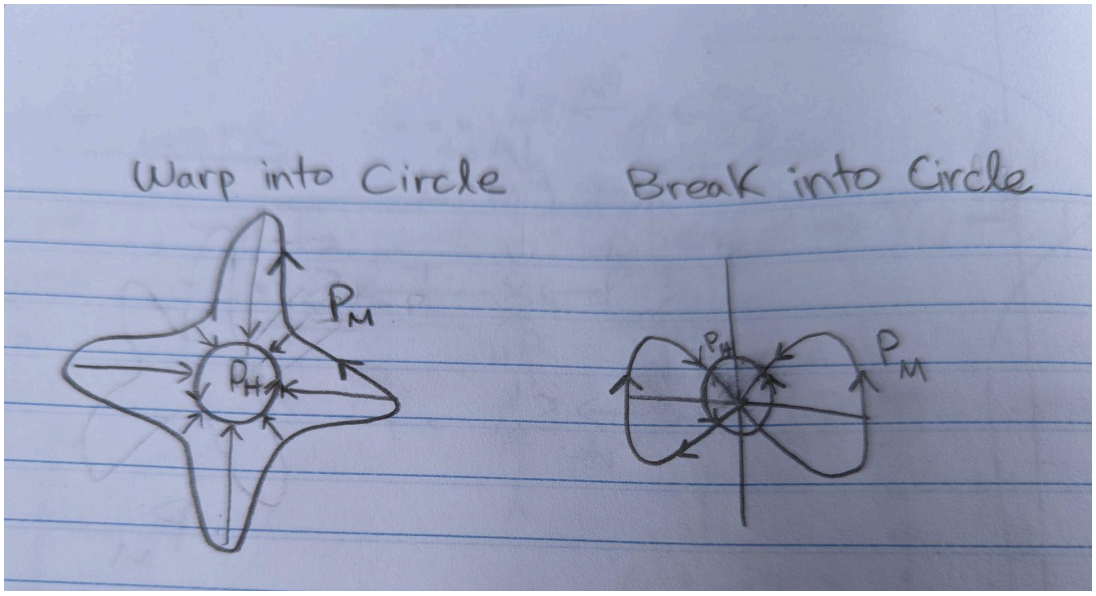
1. There are arbitrarily severe “ontology mismatches” between the predictor’s Bayes net and the human’s Bayes net, such that specifying the direct translation between them can be very complex.

Both S□ and S□ can be represented as vector spaces of potentially different dimensionality. Since S□ is actually the machine’s simulation of the real human state, the work has already been done in translating the state to a fixed language. Now learning the relationship between S□ and S□ as vector spaces could still be arbitrarily complicated, but this learning process can be discharged by appeal to the sophistication of future ML learning algorithms.

2. The human’s Bayes net is simpler than the predictor’s Bayes net, and the gap can be arbitrarily large.

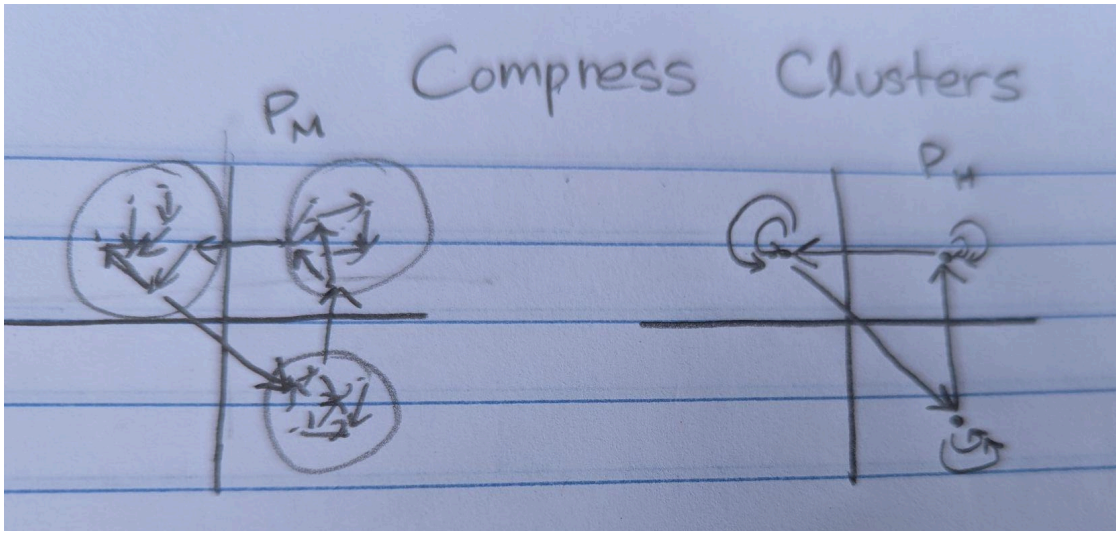
If the human’s state machine is much simpler, many edges just end up getting compressed into identity. If the topology of the AI’s state machine is fundamentally incompatible with what can be expressed using Predict□, then a minimally violating choice must be made. But maybe if the shapes are truly incompatible, local fidelity is the best we can ask for, and maybe this is not

terrible since it matches our state machine up to some number of transitions in expectation.



3. The predictor's Bayes net is arbitrarily large, and individual nodes of the human's Bayes net depend on a constant fraction of the predictor's Bayes net. Therefore directly translating between the two Bayes nets takes arbitrarily much computation.

When many machine states map to a single human state, each internal edge is mapped to the identity transition.



4. It's computationally easy for the human simulator to do good inference in the human's Bayes net—at least good enough inference that it answers questions as well as the direct translator—and the gap between this complexity and the complexity of direct translation can be arbitrarily large. More precisely, a human interrogator can't think of

questions which require the human simulator to do hard inference. However, we can't rule out the possibility that you, the algorithm-designer, will think of some trick they can use to generate hard questions.

It may be difficult to sample pieces of the space where the machine and human models strongly diverge while training F, and it would only make sense that we are mapping the global structure of Predict□ faithfully if we can train with a sufficiently representative sample of machine states.

But since the human is actually the AI's model of a human, the human can be placed in the algorithmic loop, and a model could be trained to propose machine states and where the predictions on S□ and F(S□) currently diverge. In other words, this can be helped with a sufficiently strong active learning procedure.

5. There are situations where a model which is trying to answer a question will *first* be able to predict what (potentially mistaken) belief the human-with-assistance will have about that question before it knows the true answer to the question.

This proposal is bypassing question-asking by learning the translation between ontologies directly.

6. While the predictor's Bayes net is very *large*, it has a short succinct description which is simpler than the human's Bayes net, and that gap is arbitrarily large. Think of simple physical laws that describe arbitrarily computationally expensive simulations, or simple cognitive heuristics that are very widely applicable.

I am visualizating this as a high-dimensional S□ which has a lower dimensional representation. In this case, if anything it should then be easier to map to a lower dimensional S□.

7. There are sequences of actions for which the predictor's Bayes net correctly anticipates that tampering will occur yet nothing looks suspicious to the human's Bayes net.

Such a machine state will not map to such a human state because the human states with tampering and non-tampering should have quite differently shaped state neighborhoods. The truth of this claim should scale with the density of connections, which will decrease the odds of having similar neighborhoods with chance. And the density of connections should be high when you also take into account that each state gets an outgoing edge for each possible observation.

8. After tampering has occurred, it is possible that no sequence of further actions reveal that anything is wrong, e.g. the camera and actuators may have been completely disconnected from the world and replaced with a convincing-to-humans simulation.

I think that the presence of such a sophisticated adversary would land you in a very different part of human state space though, so I believe the machine should still be able to learn an F that separates these cases. I am unsure about this.

9. There are sequences of actions leading to undetectable tampering that have no human-comprehensible similarities to sequences of actions leading to detectable tampering; in particular, the early steps of this undetectable tampering cannot be used to perform detectable tampering.

In the process of learning F, we do not use the human's choice of S□, but rather put the human into a particular state of knowledge such that it will correctly predict the next state. In situations where the human would be fooled in even a single prediction step, I believe that this model would learn a "white lie" F(s□) such that Predict□(F(s□)) is still correct.

10. Self-reference

Suppose there was an s□ in S□ that included the belief that "s_x, o_x, and $Predict_x$ are correct where X=me", and "me" is lazily computed. If we can represent such a statement, then neither P□ or P□ will modify s□ or F(s□). However, the loss on this example will still be high insofar as s□ contains other object level information besides this meta-claim, since that content will be updated differently by P□ and P□.

11. Heat Death

I originally thought that F needed to be invertible because otherwise I could get zero loss by mapping each s□ to a single human state that is fixed under Predict□, such as the heat death of the universe. However, the heat death is not actually fixed under all observations – if I think that it is heat death and then I observe a sun, I no longer think we are in heat death, and if I see crazy random bits I will think that I am in a simulation. In fact, if there is a state of human knowledge such that no observation can change it, then the human has a broken epistemology and it makes sense that we learn a degenerate F.

True Counterexamples and Resolutions

1. Embedded Agency

There is likely some way of breaking this proposal by using the fact that Predict□ and Predict□ are actually themselves implemented by the transition function of the universe T.

For instance, this could mean that insofar as running T affects the current state S, running Predict□ or Predict□ also affects S. If this is the case, there is no consistent function for F to learn. In response, here is an alternate setup that I believe works in the setting where agents are part of their environments.

We still have a state S and transition function $T : S \to S$. But now we want to treat an agent as purely an abstraction of the state $A_x : S \to S_x$, where S_x is still a notion of the internal state of an agent. The key idea is that the agent's transition function is now lifted from the universe's transition function: $T_x : S_x \to S_x := A_x^{-1} \cdot T \cdot A_x$. I'll require an invertible abstraction A_x so that T_x is

easily implementable, though I admit this does not capture the notion of an abstraction that throws away information. Now the *Agency* of an abstraction A_x over a sequence of state machine transitions is quantified by how much T_x *keeps the internal state the same*.

Agency : $A_x \rightarrow Q := L_2(\text{map}(L_2, \text{map}(P_x, \text{Prev}\square), \text{Next}\square))$
where $\text{Prev}\square := S^0,...,S^n$, $\text{Next}\square := S^1,...,S^{n+1}$, st $\forall$ k. $T(S\hat{}k) = S^{k+1}$

Now that we a notion of agent, we can use $A\square$ and $A\square$ to denote the machine and human agent-like abstraction functions respectively. Now this proposal is to learn an invertible F st $L_2(F \cdot T\square, T\square \cdot F)$ is minimized. Notice that this can be written purely in terms of T, $A\square$, and $A\square$: $L_2(F \cdot A\square^{-1} \cdot T \cdot A\square, A\square^{-1} \cdot T \cdot A\square \cdot F)$.