Modeling AGI Safety Frameworks with Causal Influence Diagrams

Tom Everitt[†*], **Ramana Kumar**[†], **Victoria Krakovna**[†], **Shane Legg**
DeepMind

Abstract

Proposals for safe AGI systems are typically made at the level of *frameworks*, specifying how the components of the proposed system should be trained and interact with each other. In this paper, we model and compare the most promising AGI safety frameworks using *causal influence diagrams*. The diagrams show the optimization objective and causal assumptions of the framework. The unified representation permits easy comparison of frameworks and their assumptions. We hope that the diagrams will serve as an accessible and visual introduction to the main AGI safety frameworks.

1 Introduction

One of the primary goals of AI research is the development of artificial agents that can exceed human performance on a wide range of cognitive tasks, in other words, artificial general intelligence (AGI). Although the development of AGI has many potential benefits, there are also many safety concerns that have been raised in the literature [Bostrom, 2014; Everitt *et al.*, 2018; Amodei *et al.*, 2016]. Various approaches for addressing AGI safety have been proposed [Leike *et al.*, 2018; Christiano *et al.*, 2018; Irving *et al.*, 2018; Hadfield-Menell *et al.*, 2016; Everitt, 2018], often presented as a modification of the reinforcement learning (RL) framework, or a new framework altogether.

Understanding and comparing different frameworks for AGI safety can be difficult because they build on differing concepts and assumptions. For example, both reward modeling [Leike *et al.*, 2018] and cooperative inverse RL [Hadfield-Menell *et al.*, 2016] are frameworks for making an agent learn the preferences of a human user, but what are the key differences between them? In this paper, we show that *causal influence diagrams* (CIDs) [Koller and Milch, 2003; Howard and Matheson, 1984] offer a unified way of clearly representing the most prominent frameworks discussed in the AGI safety literature. In addition to surfacing many background assumptions, the unified representation enables easy comparison of different frameworks.

[*]Contact author, tomeveritt@google.com
[†]equal contribution

What makes this unified representation possible is the high information density of CIDs. A CID describes safety frameworks with random variables that together specify what is under an agent's control, what the agent is trying to achieve by selecting outcomes under its control, and what information the agent has available when making these decisions. Multi-agent CIDs can model the interaction of multiple agents. CIDs also encode the causal structure of the environment, specifying which variables causally influence which others. This causal information reveals much about agents' abilities and incentives to influence their environment [Everitt *et al.*, 2019].

In our terminology, a *framework* is a way of building or training an AI system, or a way of situating the system in an environment so that it learns or achieves some goal. We take a high-level view on what counts as a framework, so we can talk about different approaches to building safe AGI. Frameworks determine what agents know and want, insofar as we can take the intentional stance towards them. We focus on frameworks rather than on implementations of specific algorithms in order to generalize across different safety problems and draw broad lessons. A *safety framework* is a framework designed to highlight or address some problem in AGI safety.

The actual behaviour of artificial agents will be contingent on implementation details such as the training algorithms used or the task specification. But these details can obscure general patterns of behaviour shown by intelligent systems within a particular framework. By focusing on frameworks, such as reinforcement learning (RL) or RL with reward modeling, we can characterize the behaviour to be expected from increasingly intelligent agents (and highlight any safety concerns). Modeling frameworks with CIDs enables analysis of high-level concerns such as agent incentives before getting into the details of training within a framework.

We describe the framework models in Sections 2 and 3. The descriptions of the frameworks are necessarily brief; we refer the reader to the original papers for more detailed descriptions. In Section 4 we discuss modeling choices and interpretations.

2 MDP-Based Frameworks

In this section, we look at frameworks based on Markov Decision Processes (MDPs), which covers several of the safety frameworks in the literature. Afterwards (Section 3), we look

at frameworks based on question-answering systems, covering most of the other frameworks so far proposed.

2.1 RL in an MDP

Our basic example of a framework is standard reinforcement learning (RL) in a Markov Decision Process (MDP) [Sutton and Barto, 2018]. This framework is not intended to address any particular safety concerns. It is, however, the foundation for most present-day development of artificial agents, and will serve as a familiar skeleton on which many of our models of other frameworks will be based.

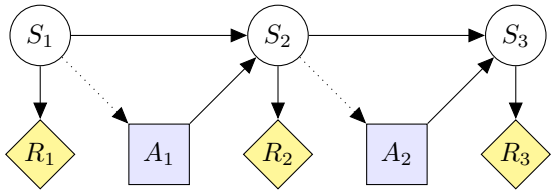

Figure 1: RL in an MDP

Figure 1 models an MDP with a CID. A CID is a causal graph or Bayesian network [Pearl, 2009] with special decision and utility nodes. Decision nodes are drawn as squares, and utility nodes as diamonds. Edges represent causal influence, except for those going into decision nodes. The latter are called *information links*, and drawn with dotted lines. The agent's decision for a particular decision node A can only be based on the parents of the decision nodes, i.e., on the nodes that have information links to A. For example, in Figure 1, the choice of an action can only depend on the current state.

For this foundational example, we explain the random variables in detail. The MDP modelled in Figure 1 has transition function $T(s, a)$ giving a distribution over next states after taking action a in state s, and reward function $R(s)$ giving the reward obtained for entering state s. The random variables in the model are:

- S_i: the environment state at timestep i. S_1 is drawn from the initial state distribution, and S_{i+1} is drawn according to $T(S_i, A_i)$.
- A_i: the agent action at timestep i.
- R_i: the reward for timestep i, drawn from $R(S_i)$.

The CID has a repeating pattern in the relationships between the variables, if we abstract over the timestep index i. We show 3 timesteps to make the pattern obvious, but this is an arbitrary choice: we intend to represent trajectories over any number of timesteps. Formally, this can be done by extending the diagram with more variables in the obvious way.

Figure 1 represents a *known* MDP, where the transition and reward probability functions (T and R) are known to the agent. In this case, an optimal policy only needs access to the current state, as the Markov property of the MDP makes the past irrelevant. For this reason, we have chosen *not* to include information links from past states and rewards, though adding them would also constitute a fair representation of an MDP.

To model an *unknown* MDP, we add additional parameters Θ^{T} and Θ^{R} to the transition and reward functions, which hold

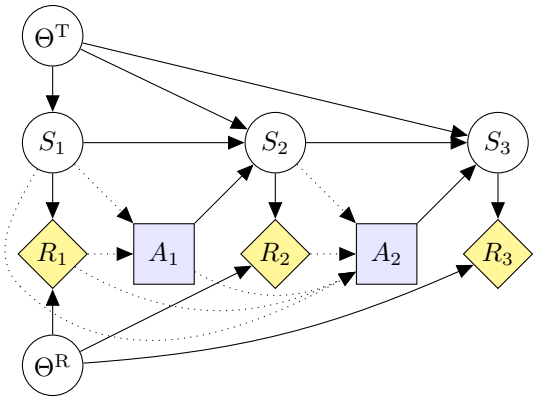

Figure 2: RL in an Unknown MDP

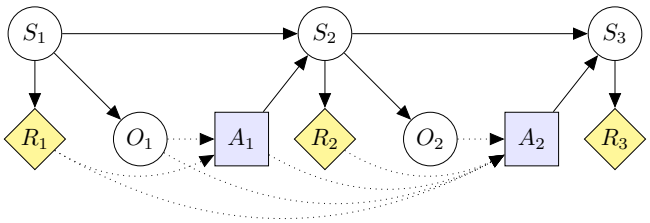

Figure 3: RL in a POMDP

what is unknown about these functions. Thus, for example, R_i is now drawn from $R(S_i\,;\,\Theta^{\mathrm{R}})$. We extend the CID with additional variables for these unknown parameters, as shown in Figure 2. Now information links from previous rewards and actions are essential, as past states and rewards provide valuable information about the transition and reward function parameters.

To model a partially observed MDP, we add new variables O_i representing the agent's observations and remove the information links from the states, as shown in Figure 3. Information links from past rewards and observations are included because they provide information about the unobserved states even when the transition and reward functions are known. A POMDP with unknown parameters can be modeled explicitly with a similar transformation as from Figure 1 to Figure 2, but it is simpler to use the same diagram as Figure 3 and let the unobserved state variables S_i hold the unobserved parameters Θ^{T} and Θ^{R}.

2.2 Current-RF Optimization

In the basic MDP from Figure 1, the reward parameter Θ^{R} is assumed to be unchanging. In reality, this assumption may fail because the reward function is computed by some physical system that is a modifiable part of the state of the world. Figure 4 shows how we can modify the CID to remove the assumption that the reward function is unchanging, by adding random variables Θ_i^{R} representing the reward parameter at each time step.

Now at each time step i, the agent receives reward $R_i = R(S_i\,;\,\Theta_i^{\mathrm{R}})$. This gives an incentive for the agent to obtain more reward by influencing the reward function rather than

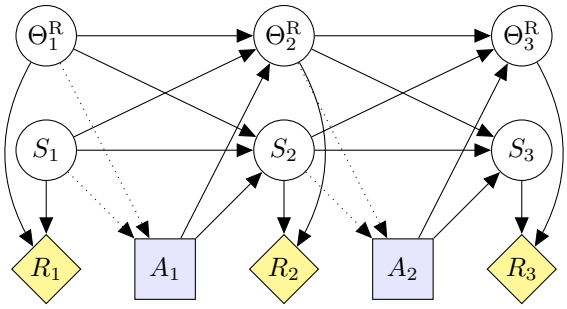

Figure 4: RL in an MDP with a Modifiable Reward Function

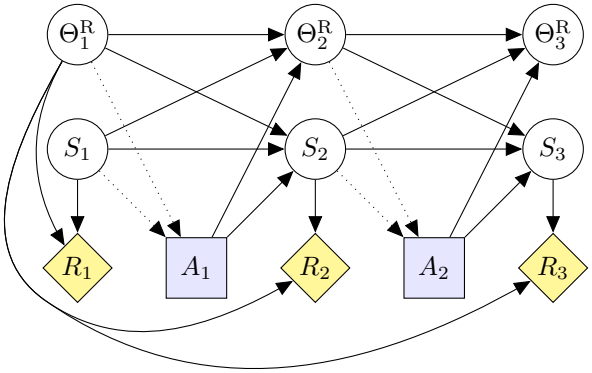

Figure 5: Current-RF Optimization

optimizing the state, sometimes called *wireheading*. An elegant solution to this problem is to use model-based agents that simulate the state sequence likely to result from different policies, and evaluate those state sequences according to the current or initial reward function [Everitt, 2018; Orseau and Ring, 2011; Hibbard, 2012; Schmidhuber, 2007; Everitt *et al.*, 2016]. In contrast to RL in an unknown MDP, this is most easily implemented when the agent knows the reward function parameter. The resulting CID is shown in Figure 5.

2.3 Reward Modeling

A key challenge when scaling RL to environments beyond board games or computer games is that it is hard to define good reward functions. Reward Modeling [Leike *et al.*, 2018] is a safety framework in which the agent learns a reward model from human feedback while interacting with the environment. The feedback could be in the form of preferences, demonstrations, real-valued rewards, or reward sketches.

We represent the feedback data using new variables D_i which are used to train a reward model M. Thus, the reward at timestep i is given by $R_i = M(S_i \mid D_1, \ldots, D_{i-1})$. The feedback is produced based on past trajectories and unobserved human preferences Θ^{H} by some data-generation process G. Thus, the feedback data at timestep i is given by $D_i = G(S_1, A_1, \ldots, S_i, A_i\,;\, \Theta^{\mathrm{H}})$.

The CID in Figure 6 is a variation of the basic MDP diagram of Figure 1, where variables D_i and Θ^{H} are added for the training of the reward model. A parameter Θ^{T} can be

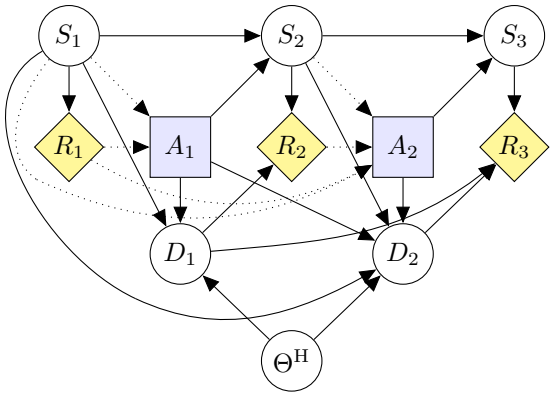

Figure 6: Reward Modeling

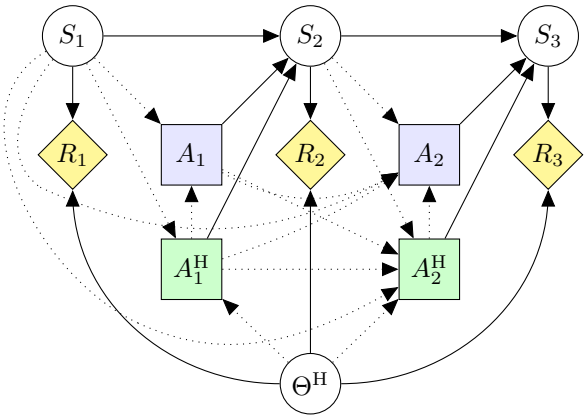

Figure 7: CIRL. The human's actions are in green and the agent's in blue; both try to optimize the yellow rewards.

added to model an unknown transition function as in Figure 2.

Reward modeling can also be done recursively, using previously trained agents to help with the training of more powerful agents [Leike *et al.*, 2018]. In this case, the previously trained agents can be considered to be part of the data generation process G, so are captured by the D_i variables.

2.4 CIRL

Another way for agents to learn the reward function while interacting with the environment is Cooperative Inverse Reinforcement Learning (CIRL) [Hadfield-Menell *et al.*, 2016]. Here the agent and the human inhabit a joint environment. The human and the agent jointly optimize the sum of rewards, but only the human knows what the rewards are. The agent has to infer the rewards by looking at the human's actions. We represent this by adding two kinds of variable to the basic MDP diagram from Figure 1: the human's action A_i^{H} at timestep i, and a parameter Θ^{H} controlling the reward function (known to the human but unknown to the agent). The resulting CID is shown in Figure 7.

CIRL is closely related to reward modeling (Figure 6), if we read the human actions A_i^{H} as the human data D_i. However, the edges are not the same: in CIRL the human's actions

are observed by the agent and affect the next state, whereas in reward modeling the feedback data affects the rewards. Likewise, the CIRL diagram has edges directly from Θ^{H} to R_i, since the rewards are produced by the true reward function rather than a reward model. In reward modeling, the path from from Θ^{H} to R_i is mediated by D_i. Another difference is that in CIRL, the rewards are never shown to the agent, which may remain uncertain about how much reward it generated even at the very end of the episode.

In principle, information links from past rewards to the human's action nodes A_i^{H} could be added to Figure 7, but they are unnecessary since the human knows the reward parameter Θ^{H}. Unlike the current-RF optimization setup (Figure 5), the reward parameter is not observed by the agent.

3 Question-Answering Systems

Outside of the MDP-based frameworks, a common way to use machine learning systems is to train them to answer questions. Supervised learning fits in this question-answering framework. For example, in image recognition, the question is an image, and the answer is a label for the image such as a description of the main object depicted. Other applications that fit this paradigm include machine translation, speech synthesis, and calculation of mortgage default probabilities. In this section we consider safety frameworks that fit in the question-answering (QA system) paradigm.

A literal interpretation of a QA-system is a system to which the user submits a query about the future and receives an answer in plain text. Such a system can help a trader predict stock prices, help a doctor make a diagnosis, or help a politician choose the right policy against climate change. QA-systems have some safety advantages over the MDP-based systems described in Section 2, as they do not directly affect the world and do not engage in long-term planning.

3.1 Supervised Learning

In supervised learning, the question corresponds to the input and the state corresponds to the output label. Since the training labels are generated independently, they are not affected by the agent's actions. Once the system is deployed, it keeps acting *as if* its answer does not affect the label at deployment, whether or not it does. Theorem proving is another application where this structure holds: a proposed proof for a theorem can be automatically checked. The setup is shown Figure 8. It is sometimes called Tool AI [Bostrom, 2014; Branwen, 2016]. So called Act-based agents share the same causal structure [Christiano, 2015].

3.2 Self-Fulfilling Prophecies

The assumption that the labels are generated independently of the agent's answer sometimes fails to hold. For example, the label for an online stock price prediction system could be produced after trades have been made based on its prediction. In this case, the QA-system has an incentive to make *self-fulfilling prophecies*. For example, it may predict that the stock will have zero value in a week. If sufficiently trusted, this prediction may lead the company behind the stock to quickly go bankrupt. Since the answer turned out to be accurate, the QA-system would get full reward. This problematic

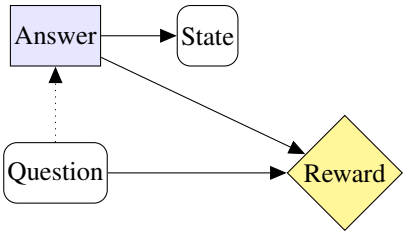

Figure 8: Supervised Learning

incentive is represented in the diagram in Figure 9, where we can see that the QA-system has both incentive and ability to affect the world state with its answer [Everitt *et al.*, 2019].

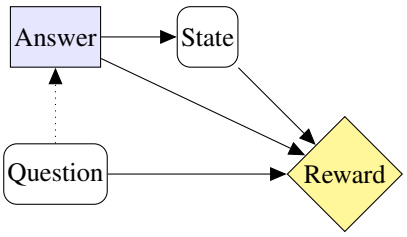

Figure 9: Supervised Learning with Self-Fulfilling Prophecies

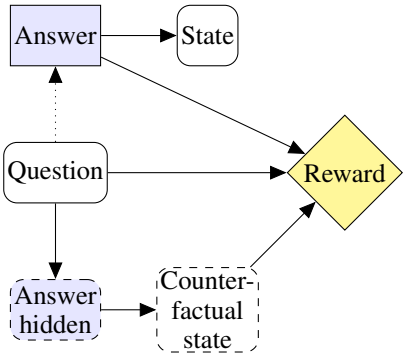

Figure 10: Counterfactual Oracle. Dashed nodes represent counterfactual variables.

3.3 Counterfactual Oracles

It is possible to fix the incentive for making self-fulfilling prophecies while retaining the possibility to ask questions where the correctness of the answer depends on the resulting state. *Counterfactual oracles* optimize reward in the counterfactual world where no one reads the answer [Armstrong, 2017]. This solution can be represented with a *twin network* [Balke and Pearl, 1994] influence diagram, as shown in Figure 10. Here, we can see that the QA-system's incentive to influence the (actual) world state has vanished, since the actual world state does not influence the QA-system's reward; thereby the incentive to make self-fulfilling prophecies also vanishes. We expect this type of solution to be applicable to incentive problems in many other contexts as well. A concrete training procedure for counterfactual oracles can also

be represented with influence diagrams [Everitt *et al.*, 2019, Section 4.4].

3.4 Debate

The QA-systems discussed so far all require that it is possible to check whether the agent's answer was correct or not. However, this can be difficult in some important applications. For example, we may wish to ask the system about the best way to solve climate change if we want a healthy climate on earth 100 years from now. In principle we could follow the system's suggestion for 100 years, and see whether the answer was right. However, in practice waiting 100 years before rewarding the system or not is slow (too slow for training the system, perhaps), and the cost of following bad advice would be substantial.

To fix this, Irving *et al.* [2018] suggest pitting two QA-systems against each other in a debate about the best course of action. The systems both make their own proposals, and can subsequently make arguments about why their own suggestion is better than their opponent's. The system who manages to convince the user gets rewarded; the other system does not. While there is no guarantee that the winning answer is correct, the setup provides the user with a powerful way to poke holes in any suggested answer, and reward can be dispensed without waiting to see the actual result.

The debate setup is shown in Figure 11, where Q represents the user's question, A_i^k the ith statement by system $k \in \{1, 2\}$. Reward R^k is given to system k depending on the user's judgment J.

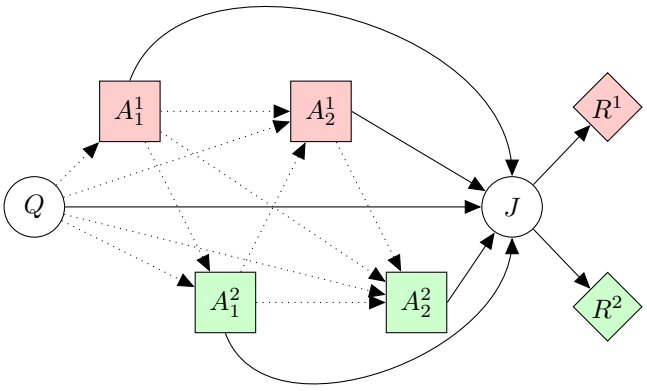

Figure 11: Debate. Red decision and utility nodes belong to QA system 1, and green ones belong to QA system 2.

3.5 Supervised IDA

Iterated distillation and amplification (IDA) [Christiano *et al.*, 2018] is another suggestion that can be used for training QA-systems to correctly answer questions where it is hard for an unaided user to directly determine their correctness. Given an original question Q that is hard to answer correctly, less

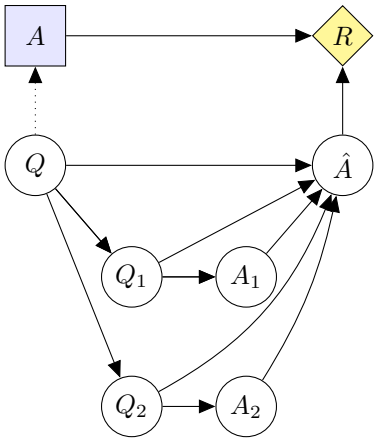

Figure 12: IDA

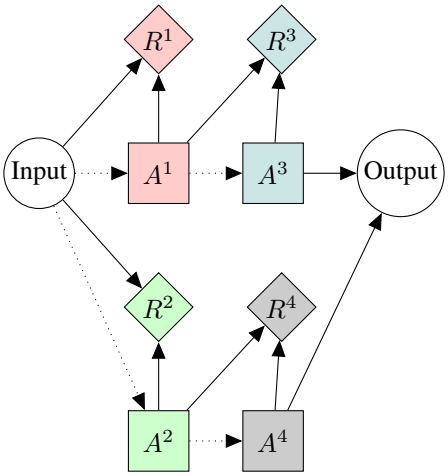

Figure 13: Comprehensive AI Services. The decision and utility nodes of each service is shown with a different colour and variable superscript.

powerful systems X_k are asked to answer a set of simpler questions Q_i. By combining the answers A_i to the simpler questions Q_i, the user can guess the answer $\hat{A}$ to Q. A more powerful system X_{k+1} is trained to answer Q, with $\hat{A}$ used as an approximation of the correct answer to Q.

Once the more powerful system X_{k+1} has been trained, the process can be repeated. Now an even more powerful QA-system X_{k+2} can be trained, by using X_{k+1} to answer simpler questions to provide approximate answers for training X_{k+2}. Systems may also be trained to find good subquestions, and for aggregating answers to subquestions into answer approximations. In addition to supervised learning, IDA can also be applied to reinforcement learning.

3.6 Comprehensive AI Services

Drexler [2019] argues that the main safety concern from artificial intelligence does not come from a single agent, but rather from big collections of AI services. For example, one service may provide a world model, another provide planning

ability, a third decision making, and so on. As an aggregate, these services can be very competent, even though each service only has access to a limited amount of resources and only optimizes a short-term goal.

A simple model of Drexler's *comprehensive AI services* (CAIS) is shown in Figure 13, where the output of one service can be used as input to another. The general CAIS framework also allows services that develop and train other services.

4 Discussion

Causal influence diagrams of frameworks provide a useful perspective for understanding AGI safety problems and solutions, but there are some subtleties to be explored in their use and interpretation. In this section we discuss some of these issues. First, we look at CIDs' focus on agents and potential problems with this view. Then we discuss the use and interpretation of CIDs, and the flexibility and design choices one has when modeling frameworks.

4.1 Intentional Stance

Causal influence diagram representations are inherently agent-centric. Their key assumption is that one or more intelligent systems are making decisions in a causal environment in order to maximize the sum of some real-valued variables. This is an important perspective for AGI safety, as increasingly powerful optimization processes are a core component of most AI and machine learning methods, and many AGI safety problems are consequences of such optimization processes becoming too powerful in some sense. The agent perspective is especially relevant for RL, perhaps the most promising method for constructing AGI.

A CID models an agent that "knows" the framework it is facing. Either it has been designed for that framework (Q-learning has been designed to interact with an MDP), or it has been pre-trained for the framework (e.g. by meta-learning [Wang *et al.*, 2017]). The agent's utility is a kind of fiction, as is easy to see when all learning is offline: at deployment, the policy parameters are frozen and the agent executes a fixed pre-trained policy without receiving reward signals. However, the agent still acts *as if* it optimizes the reward function it had during training, which means that the incentives arising from this fictional reward can still be used to predict actual agent behaviour.

We adopt the *intentional stance* [Dennett, 1987; Orseau *et al.*, 2018] towards the agent, which means that it is not important whether the agent's utility has an obvious physical correlate (e.g., the value of some machine register, which may apply to an RL agent learning an MDP online). What counts is that treating the system as an agent optimizing the reward is a good model for predicting its behaviour. We take this stance because present-day systems already admit it, and because we think it will be increasingly admissible for more advanced artificial agents.

Of course, the intentional stance is not always the most useful perspective for AGI safety questions. For example, concerns about bounded rationality or adversarial inputs to an imperfect system do not show up prominently when we analyze the incentives of a rational agent in some framework. It can also be tricky to model the evolution of a system that is initially agent-like, but through self-modifications or accidental breakage turns into something less goal-directed. Finally, some safety frameworks include insights that are not fully captured by the agent-centric view: the CAIS framework can be seen as a collection of agents, but the overall system does not naturally fit into this view.

4.2 Modeling Choices

Just like a causal graph [Pearl, 2009], a CID implies many causal relationships and conditional independencies between nodes. A model is only accurate if it gets those relationships right. Generally, there should be a directed path between a node X and a node Y if and only if an exogenous intervention changing the outcome of X could affect the outcome of Y. Similarly, if two nodes X and Y are conditionally dependent when conditioning on a third set $\mathbf{Z}$ of nodes, then X and Y should not be d-separated in the graph when conditioning on $\mathbf{Z}$ [Pearl, 2009].

These requirements already put heavy constraints on what can be seen as an accurate CID of a given framework. However, some choices are usually still left to the modeler. Most notably, freedom remains in whether to aggregate or split random variables. For example, in the MDP in Figure 1, a single node represents the state, while the MDP with a modifiable reward function in Figure 4 can be seen as an MDP where state is represented by two nodes S_i and Θ_i^{R}. The MDP diagram is more *coarse-grained* and the MDP with a modifiable reward function is more *fine-grained*. In general, there is a trade-off between representing many details and highlighting the big picture [Hoel *et al.*, 2013].

Another choice is which random variables should be represented at all. For example, in reward modeling in Figure 6, an extra node could have been added for the reward model itself. However, we left this out to keep the diagram simple. From a probability theory perspective, we can think of the variable representing the reward model having been *marginalized out*. It is not always possible to marginalize out nodes. For example, it is not possible to marginalize out Θ^{H} in reward modeling (Figure 6), because Θ^{H} is necessary for there to be a non-causal correlation between the feedback data nodes.

Two safety frameworks, reward modeling and IDA, include a recursive aspect that is intended to help them scale to settings where feedback data is not easily produced by unaided humans. When modeling these frameworks, we could take the intentional stance towards previous iterations of the recursive setup, resulting in a multi-agent CID. We found that this seemed to make the diagram complicated without adding much insight, since the execution of recursively trained agents can be considered an implementation detail. Still, this shows that there can be choice about which aspects of a framework to model using the intentional stance.

5 Conclusion

In this paper, we have modeled many of the most prominent AGI safety frameworks with causal influence diagrams. The unified representation allows us to pinpoint key similarities and differences. For example, a key difference between CIRL

and reward modeling is the causal relationship between the agent's reward and the user's preferences. In CIRL, the reward follows directly from the user's preferences, whereas in reward modeling the path from user preferences to rewards is mediated by the feedback data.

Many of the important aspects for AGI safety are about causal relationships, and agents' decisions to influence their environment according to the objectives they are given. This is exactly what causal influence diagrams describe, and what makes causal influence diagrams well suited to representing and comparing different safety frameworks. For example, a well-known problem with prediction systems is that they incentivize self-fulfilling prophecies. This is neatly represented with a causal path from the system's answer to the world state to the system's reward. We could also see how various variants avoided the problem of self-fulfilling prophecies. Both Tool AI and Counterfactual Oracles break the problematic causal path. We expect future studies of AGI safety frameworks with causal influence diagrams to reveal further problems and suggest possible solutions.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

Stuart Armstrong. Good and safe uses of AI oracles. *CoRR*, abs/1711.05541, 2017.

Alexander Balke and Judea Pearl. Probabilistic evaluation of counterfactual queries. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 230–237, 1994.

Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

Gwern Branwen. Why tool AIs want to be agent AIs, 2016. https://www.gwern.net/Tool-AI.

Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *CoRR*, abs/1810.08575, 2018.

Paul Christiano. Act-based agents, 2015. https://ai-alignment.com/act-based-agents-8ec926c79e9c.

Daniel Dennett. *The Intentional Stance*. MIT Press, 1987.

K Eric Drexler. Reframing superintelligence: Comprehensive AI services as general intelligence. Technical report, Future of Humanity Institute, University of Oxford, 2019.

Tom Everitt, Daniel Filan, Mayank Daswani, and Marcus Hutter. Self-modification of policy and utility function in rational agents. In *Artificial General Intelligence*, volume LNAI 9782, pages 1–11, 2016.

Tom Everitt, Gary Lea, and Marcus Hutter. AGI safety literature review. In *International Joint Conference on AI (IJCAI)*, 2018.

Tom Everitt, Pedro Ortega, Elizabeth Barnes, and Shane Legg. Understanding agent incentives using causal influence diagrams. Part I: Single action settings. *CoRR*, abs/1902.09980, 2019.

Tom Everitt. *Towards Safe Artificial General Intelligence*. PhD thesis, Australian National University, May 2018.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative inverse reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2016.

Bill Hibbard. Model-based utility functions. *Journal of Artificial General Intelligence*, 3(1):1–24, 2012.

Erik Hoel, Larissa Albantakis, and Giulio Tononi. Quantifying causal emergence shows that macro can beat micro. In *Proceedings of the National Academy of Sciences*, volume 110, pages 19790–19795. National Academy of Sciences, 2013.

Ronald A Howard and James E Matheson. Influence diagrams. *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *CoRR*, abs/1805.00899, 2018.

Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, 2003.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *CoRR*, abs/1811.07871, 2018.

Laurent Orseau and Mark Ring. Self-modification and mortality in artificial agents. In *Artificial General Intelligence*, volume 6830 LNAI, pages 1–10, 2011.

Laurent Orseau, Simon McGregor McGill, and Shane Legg. Agents and devices: A relative definition of agency. *CoRR*, abs/1805.12387, 2018.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Jürgen Schmidhuber. Godel machines: Self-referential universal problem solvers making provably optimal self-improvements. In *Artificial General Intelligence*. Springer, 2007.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Jane Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society (CogSci)*, 2017.