On the Conditional Logic of Simulation Models

Duligur Ibeling[1], **Thomas Icard**[2]
[1] Department of Computer Science, Stanford University
[2] Department of Philosophy, Stanford University
duligur@stanford.edu, icard@stanford.edu

Abstract

We propose analyzing conditional reasoning by appeal to a notion of intervention on a simulation program, formalizing and subsuming a number of approaches to conditional thinking in the recent AI literature. Our main results include a series of axiomatizations, allowing comparison between this framework and existing frameworks (normality-ordering models, causal structural equation models), and a complexity result establishing NP-completeness of the satisfiability problem. Perhaps surprisingly, some of the basic logical principles common to all existing approaches are invalidated in our causal simulation approach. We suggest that this additional flexibility is important in modeling some intuitive examples.

1 Introduction and Motivation

Much of intelligent action and reasoning involves assessing what *would* occur (or *would have* occurred) under various non-actual conditions. Such hypothetical and counterfactual (broadly, *subjunctive*) conditionals are bound up with central topics in artificial intelligence, including prediction, explanation, causal reasoning, and decision making. It is thus for good reason that AI researchers have focused a great deal of attention on conditional reasoning (see, e.g., [Ginsberg, 1986; Delgrande, 1998; Friedman *et al.*, 2000; Pearl, 2009; Bottou *et al.*, 2013], among many others).

Two broad approaches to subjunctive conditionals have been especially salient in the literature. The first, originating in philosophy [Stalnaker, 1968; Lewis, 1973], takes as basic a "similarity" or "normality" ordering on possibilities, and evaluates a claim 'if φ then ψ' by asking whether ψ is true in (e.g., all) the most normal φ possibilities. The second approach, associated with the work of Judea Pearl, takes as basic a causal "structural equation" model (SEM), and evaluates conditionals according to a defined notion of *intervention* on the model. These two approaches are in some technical and conceptual respects compatible [Pearl, 2009], though they can also be shown to conflict on some basic logical matters [Halpern, 2013]. Both capture important intuitions about conditional reasoning, and both have enjoyed successful applications in AI research.

In this article we propose a third approach to conditionals, which captures a different intuition, and which can already be seen as implicit in a growing body of work in AI, as well as in cognitive science. This approach takes as basic the notion of a *simulation model*, that is, a *program* for simulating the transformation from one state of the world to another, or for *building up* or *generating* a world from a partial description of it. Simulation models have been of interest since the earliest days of AI [Newell and Simon, 1961]. A recent tradition, coming out of work on statistical relational models, has proposed building complex generative models using rich and expressive programming languages, typically also incorporating probability (e.g., [Pfeffer and Koller, 2000; Milch *et al.*, 2005; Goodman *et al.*, 2008; de Raedt and Kimmig, 2015]). Such languages have also been used for modeling human reasoning, including with counterfactuals [Goodman *et al.*, 2015].

Simulation models have an obvious causal (and more general *dependence*) structure, and it is natural to link conditionals with this very structure. We can assess a claim 'if φ then ψ' by *intervening* on the program to ensure that φ holds true throughout the simulation, and asking whether ψ holds upon termination. This is conceptually different from the role of intervention in structural equation models, where the post-intervention operation is to find solutions to the manipulated system of equations. As we shall see, this conceptual difference has fundamental logical ramifications.

This more procedural way of thinking about subjunctive conditionals enjoys various advantages. First, there is empirical evidence suggesting that human causal and conditional reasoning is closely tied to mental simulation [Sloman, 2005]. Second, there are many independent reasons to build generative models in AI (e.g., minimizing prediction error in classification; see [Liang and Jordan, 2008]), making them a common tool. Thus, opportunistically, we can expect to have such models readily available (perhaps unlike normality orderings or even structural equation models).

Related to this second point, many of the generative models that are currently being built using deep neural networks fit neatly into our approach, even though we can often only use them as black boxes (see, e.g., [Mirza and Osindero, 2014; Kocaoglu *et al.*, 2017], etc.). We know how to intervene on these programs (i.e., controlling input), and how to read off a result or prediction—that is, we can observe what conditional claims they embody—even though we may not understand all the causal details of the learned model. Some authors have recently argued that certain kinds of counterfactual analysis in particular establish an appropriate standard for *interpretability* for these models [Wachter *et al.*, 2018].

Our contribution in this article is threefold: (1) we propose a general semantic analysis of conditional claims in terms of program executions, subsuming all the aforementioned application areas; (2) we establish completeness theorems for a propositional conditional language with respect to (four different classes of) programs, allowing a comparison with alternative approaches at a fundamental logical level; (3) we establish NP-completeness of the satisfiability problem for these logical systems. Before turning to these details, we explain informally what is distinctive about the resulting logic.

2 Conditional Logics

The literature on conditional logic is extensive. We focus here on the most notable differences between the systems below and more familiar systems based on either world-orderings or SEMs. We will be using a notation inspired by dynamic logic (also used by [Halpern, 2000]), whereby $[\alpha]\beta$ can loosely be read as, 'if α were true, then β would be true.' Understanding the complete logic of a given interpretation can be of both theoretical and practical interest. In the causal setting, for instance, a complete set of axioms may give the exact conditions under which some counterfactual quantity is (not) identifiable from statistical data [Pearl, 2009].

One of the bedrock principles of conditional reasoning is called *Cautious Monotonicity* [Kraus *et al.*, 1990], or sometimes the *Composition* rule [Pearl, 2009]. This says that from $[A](B \wedge C)$ we may always infer $[A \wedge B]C$. While there are known counterexamples to it in the literature—it fails for some probabilistic and possibilistic interpretations [Dubois and Prade, 1991] and in standard versions of default logic [Makinson, 1994]—the principle is foundational to both world-ordering models and SEMs. By contrast, in our setting, holding B fixed during the simulation may interrupt the sequence of steps leading to C being made true. Here is a simple example (taken from [Icard, 2017]):

Example 1. If Alf were ever in trouble (A), the neighbors Bea and Cam would both like to help (B and C, respectively). But neither wants to help if the other is already helping. Imagine the following scenario: upon finding out that Alf is in trouble, each looks to see if the other is already there to help. If not, then each begins to prepare to help, eventually making their way to Alf but never stopping again to see if the other is doing the same. If instead, e.g., Cam initially sees Bea already going to help, Cam will not go. One might then argue that the following both truly describe the situation: 'If Alf were in trouble, Bea and Cam would both go to help' and 'If Alf were in trouble and Bea were going to help, Cam would not go to help'.

The example trades on a temporal ambiguity about when Bea is going to help, and it can be blocked simply by time-indexing variables. However, following a common stance in the literature [Halpern, 2000; Pearl, 2009], we maintain that requiring temporal information always be made explicit is excessively stringent. Furthermore, in line with our earlier remarks about black box models, we may often be in a situation where we simply do not understand the internal temporal and causal structure of the program. To take a simple example, asking a generative image model to produce a cityscape might result in images with clouds and blue skies, even though a request to produce a cityscape with a blue sky might not result in any clouds. We would like a framework that can accommodate conditional theories embodied in artifacts like these.

Our completeness results below (Thm. 1) show that the logic of conditional simulation is strictly weaker than any logic of structural equation models (as established in [Halpern, 2000]) or of normality orderings (as, e.g., in [Lewis, 1973]). The conditional logic of *all* programs is very weak indeed. At the same time, some of the axioms in these frameworks can be recovered by restricting the class of programs (e.g., the principle of *Conditional Excluded Middle*, valid on structural equation models and on some world-ordering models [Stalnaker, 1968], follows from optional axiom F below). We view this additional flexibility as a feature. However, even for a reader who is not convinced of this, we submit that understanding the logic of this increasingly popular way of thinking about conditional information is valuable.

Prior Work. The notion of intervention introduced below (Defn. 1) is different from, but inspired by, the corresponding notion in SEMs [Meek and Glymour, 1994; Pearl, 2009]. The logical language we study in this paper, restricting antecedents to conjunctive clauses but closing off under Boolean connectives, follows [Halpern, 2000].

Interestingly, prior to any of this work, [Balkenius and Gärdenfors, 1991] studied conditionals interpreted specifically over certain classes of neural networks, using a definition of "clamping a node" similar to our notion of intervention. They also observed that some of the core principles of non-monotonic logic fail for that setting. (See in addition [Leitgeb, 2004] for further development of related ideas.)

3 Syntax

Let X be a set of atoms $X_1, X_2, X_3, \ldots$ and let $\mathcal{L}_{\text{prop}}$ be the language of propositional formulas over atoms in X closed under disjunction, conjunction, and negation. Let $\mathcal{L}_{\text{int}} \subset \mathcal{L}_{\text{prop}}$ be the language of purely conjunctive, ordered formulas of unique literals, i.e., formulas of the form $l_{i_1} \wedge \ldots \wedge l_{i_n}$, where $i_j < i_{j+1}$ and each l_{i_j} is either X_{i_j} or $\neg X_{i_j}$. Each formula in $\mathcal{L}_{\text{int}}$ will specify an *intervention* by giving fixed values for a fixed list of variables. We also include the "empty" intervention $\top$ in $\mathcal{L}_{\text{int}}$. Given $\varphi \in \mathcal{L}_{\text{prop}}$, $\varphi' \in \mathcal{L}_{\text{int}}$ is the $\mathcal{L}_{\text{int}}$-*equivalent* of φ if φ is a propositionally consistent, purely conjunctive formula over literals and φ' results from a reordering of literals and deletion of repeated literals in φ. For example, the $\mathcal{L}_{\text{int}}$-equivalent of $\neg X_2 \wedge X_1 \wedge X_1$ is $X_1 \wedge \neg X_2$. Let $\mathcal{L}_{\text{cond}}$ be the language of formulas of the form $[\alpha]\beta$ for $\alpha \in \mathcal{L}_{\text{int}}, \beta \in \mathcal{L}_{\text{prop}}$. We call such a formula a *subjunctive conditional*, and call α the *antecedent* and β the *consequent*. The overall causal simulation language $\mathcal{L}$ is the language of propositional formulas over atoms in $X \cup \mathcal{L}_{\text{cond}}$ closed under disjunction, conjunction, and negation. For $\alpha, \beta \in \mathcal{L}$, $\alpha \rightarrow \beta$ abbreviates $\neg\alpha \vee \beta$, and $\alpha \leftrightarrow \beta$ denotes $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$. We use $\langle\alpha\rangle$ for the dual of $[\alpha]$, i.e., $\langle\alpha\rangle\beta$ abbreviates $\neg[\alpha](\neg\beta)$.

4 Semantics

We now define the semantics of $\mathcal{L}$ over causal simulation models. A *causal simulation model* is a pair $(\mathsf{T}, \mathbf{x})$ of a Turing machine T and tape contents represented by a *state description* $\mathbf{x} = \{x_n\}_{n\in\mathbb{N}}$, which specifies binary[1] values for all tape variables, only finitely many of which can be nonzero. Running T on input $\mathbf{x}$ yields a new state description $\mathbf{x}'$ as output, provided the execution halts. We say $\mathbf{x} \models X_i$ iff $x_i = 1$ in $\mathbf{x}$. Satisfaction $\mathbf{x} \models \varphi$ of $\varphi \in \mathcal{L}_{\text{prop}}$ is then defined in the familiar way by recursion. For X-atoms we define $(\mathsf{T}, \mathbf{x}) \models X_i$ iff $\mathbf{x} \models X_i$. Toward a definition of satisfaction for subjunctive conditionals, we now define an *intervention* (in the same way as in [Icard, 2017]):

Definition 1 (Intervention)**.** An intervention $\mathcal{I}$ is a computable function mapping a machine T to a new machine $\mathcal{I}(\mathsf{T})$ by taking a set of values $\{x_i\}_{i\in I}$, $I \subseteq \mathbb{N}$ a finite index set, and holding fixed the value of each X_i to x_i throughout the execution of T. That is, $\mathcal{I}(\mathsf{T})$ first sets each X_i to x_i, then runs T while ignoring any write to any X_i.

Any $\alpha \in \mathcal{L}_{\text{int}}$ uniquely specifies an intervention, which we denote as $\mathcal{I}_\alpha$: each literal in α gives a tape variable to hold fixed, and the literal's polarity tells us to which value it is to be fixed. Now we define $(\mathsf{T}, \mathbf{x}) \models [\alpha]\beta$ iff for all halting executions of $\mathcal{I}_\alpha(\mathsf{T})$ on $\mathbf{x}$, the resulting tape satisfies β. Note that for deterministic machines, this means either $\mathcal{I}_\alpha(\mathsf{T})$ does not halt on $\mathbf{x}$, or the unique resulting tape satisfies β. The definition also implies that $(\mathsf{T}, \mathbf{x}) \models \langle\alpha\rangle\beta$ iff there exists a halting execution of $\mathcal{I}_\alpha(\mathsf{T})$ on $\mathbf{x}$ whose result satisfies β. Having now defined $(\mathsf{T}, \mathbf{x}) \models \varphi$ for atoms $\varphi \in X \cup \mathcal{L}_{\text{cond}}$, $(\mathsf{T}, \mathbf{x}) \models \varphi$ for complex $\varphi \in \mathcal{L}$ is defined by recursion.

Interestingly, as revealed by Prop. 1, model checking in this setting is difficult, while satisfiability (or validity) for notable classes of machines is decidable (Thm. 2).

Proposition 1. If $\alpha \wedge \beta$ is propositionally consistent, then it is undecidable whether $(\mathsf{T}, \mathbf{x}) \models \langle\alpha\rangle\beta$.

Proof Sketch. Under a suitable encoding of natural numbers on the variable tape, the class $\mathcal{T}_\alpha = \{\mathcal{I}_\alpha(\mathsf{T}) : \mathsf{T} \in \mathcal{T}\}$, where $\mathcal{T}$ is the class of all machines, gives an enumerable list of all the partial recursive functions, with T computably recoverable from $\mathsf{T}' \in \mathcal{T}_\alpha$. Moreover, $H_\beta = \{\mathsf{T} \in \mathcal{T}_\alpha : \mathsf{T}$ halts on input $\mathbf{x}$ with output $\mathbf{x}' \models \beta\}$ is extensional and $\varnothing \subsetneq H_\beta \subsetneq \mathcal{T}_\alpha$, so by the Rice-Myhill-Shapiro Theorem it is undecidable. If we could decide whether $(\mathsf{T}, \mathbf{x}) \models \langle\alpha\rangle\beta$, this would allow us to decide whether $\mathsf{T}' = \mathcal{I}_\alpha(\mathsf{T}) \in H_\beta$. $\square$

A second limitative result is that we cannot have strong completeness (that is, completeness relative to arbitrary sets of assumptions), since by Prop. 2 we do not have compactness. On the other hand, our axiom systems (Defn. 3) are weakly complete (complete relative to finite assumption sets).

Proposition 2. The language $\mathcal{L}$ interpreted over causal simulation models is not compact.

[1]The present setting can be easily generalized to the arbitrary discrete setting, indeed without changing the logic. See [Icard, 2017].

Proof. Let $f : \mathbb{N} \to \mathbb{N}$ be any uncomputable total function such that $f(n) \neq n$ for all n and consider $\Omega = \{\neg X_n : n \in \mathbb{N}\} \cup \{\langle X_n\rangle X_{f(n)} : n \in \mathbb{N}\} \cup \{[X_n]\neg X_m : m, n \in \mathbb{N}$ with $m \neq n, m \neq f(n)\}$. If $(\mathsf{T}, \mathbf{x})$ satisfies every $\varphi \in \Omega$, we could compute $f(n)$ by intervening to set X_n to 1, and checking which other variable X_m is set to 1. As f is total and $f(n) \neq n$, we could always find such $m = f(n)$. So Ω is unsatisfiable. But it is easily seen that every finite subset of Ω is satisfiable. $\square$

5 Axiomatic Systems

We will now identify axiomatic systems (Defn. 3) that are *sound* and *complete* with respect to salient classes (Defn. 2) of causal simulation models, by which we mean that they prove all (completeness) and only (soundness) the *generally valid principles* with respect to those classes.

Definition 2. Let $\mathcal{M}$ be the class of all causal simulation models $(\mathsf{T}, \mathbf{x})$, where T may be non-deterministic. Let $\mathcal{M}_{\text{det}}$ be the class of models with deterministic T, and let $\mathcal{M}^{\downarrow}$ be the class of models with non-deterministic T that halt on all input tapes and interventions. Also let $\mathcal{M}^{\downarrow}_{\text{det}} = \mathcal{M}_{\text{det}} \cap \mathcal{M}^{\downarrow}$.

Definition 3. Below are two rules and four axioms.[2]

PC.	Propositional calculus (over the atoms of $\mathcal{L}$)
RW.	From $\beta \to \beta'$ infer $[\alpha]\beta \to [\alpha]\beta'$
R.	$[\alpha]\alpha$
K.	$[\alpha](\beta \to \gamma) \to ([\alpha]\beta \to [\alpha]\gamma)$
F.	$\langle\alpha\rangle\beta \to [\alpha]\beta$
D.	$[\alpha]\beta \to \langle\alpha\rangle\beta$

AX denotes the system containing axioms R and K and closed under PC and RW. AX_{det} is AX in addition to axiom F, $\mathsf{AX}^{\downarrow}$ is AX in addition to axiom D, and $\mathsf{AX}^{\downarrow}_{\text{det}}$ is the system combining all of these axioms and rules.

For the remainder of this article, fix $\mathcal{M}^{\dagger}$ to be one of the classes $\mathcal{M}$, $\mathcal{M}_{\text{det}}$, $\mathcal{M}^{\downarrow}$, or $\mathcal{M}^{\downarrow}_{\text{det}}$, and let $\mathsf{AX}^{\dagger}$ be the respective deductive system of Defn. 3. Then:

Theorem 1. $\mathsf{AX}^{\dagger}$ is sound and complete for validities with respect to the class $\mathcal{M}^{\dagger}$.

Proof. The soundness of PC, RW, R, and K is straightforward. If $\mathcal{M}^{\dagger}$ is $\mathcal{M}_{\text{det}}$ (or $\mathcal{M}^{\downarrow}_{\text{det}}$), any $\mathfrak{M} \in \mathcal{M}^{\dagger}$ has at most one halting execution, so a property holding of one execution holds of all and F is sound. If $\mathcal{M}^{\dagger}$ is $\mathcal{M}^{\downarrow}$ (or $\mathcal{M}^{\downarrow}_{\text{det}}$), then any $\mathfrak{M}$ has at least one halting execution, so a property holding of all holds of one, and D is sound.

As for completeness, it suffices to show that any $\mathsf{AX}^{\dagger}$-consistent φ has a canonical model $\mathfrak{M}_\varphi \in \mathcal{M}^{\dagger}$ satisfying it. Working toward the construction of $\mathfrak{M}_\varphi$, we prove a normal form result (Lem. 1) that elucidates what is required in order

[2]We use the standard names from modal and non-monotonic logic. The *Left Equivalence* rule [Kraus *et al.*, 1990], namely, infer $[\alpha]\beta \leftrightarrow [\alpha']\beta$ from $\alpha \leftrightarrow \alpha'$, is not needed: since antecedents belong to $\mathcal{L}_{\text{int}}$, they are never distinguished beyond equivalence.

to satisfy φ (Lem. 3). We then define simple programming languages (Defn. 4)—easily seen to be translatable into Turing machine code—that we employ to construct a program for $\mathfrak{M}_\varphi$ that meets exactly these requirements.

Lemma 1. Any $\varphi \in \mathcal{L}$ is provably-in-AX (and -$\mathsf{AX}^\dagger$) equivalent to a disjunction of conjunctive clauses, where each clause is of the form

$$\pi \wedge \bigwedge_{i\in I} \big([\alpha_i] \bigvee_{j\in J_i} \beta_j\big) \wedge \bigwedge_{k\in K} \langle\alpha_k\rangle\beta_k \qquad (1)$$

and $\pi \in \mathcal{L}_{\text{prop}}$ while $\beta_j, \beta_k \in \mathcal{L}_{\text{int}}$ for all $j \in J_i$ for all $i \in I$ and for all $k \in K$. We may assume without loss of generality that $\alpha_i \neq \alpha_{i'}$ for distinct $i, i' \in I$.

Proof. Note that provably in AX, $[\alpha](\beta \wedge \gamma) \leftrightarrow [\alpha]\beta \wedge [\alpha]\gamma$ and $\langle\alpha\rangle(\beta \vee \gamma) \leftrightarrow \langle\alpha\rangle\beta \vee \langle\alpha\rangle\gamma$. Use these equivalences and PC and RW to rewrite and get the result. ☐

Given a clause δ as in (1), let $\mathcal{S}_\delta \subset \mathcal{L}_{\text{int}}$ be the set of $\mathcal{L}_{\text{cond}}$-antecedents appearing in δ. Each δ gives rise to a *selection function* $f_\delta : \mathcal{S}_\delta \to \wp(\mathcal{L}_{\text{int}})$ (cf. [Stalnaker, 1968]), obtained (not uniquely) as follows. To give the value of $f_\delta(\alpha)$, suppose that $\alpha = \alpha_k$ for some $k \in K$. If $\alpha = \alpha_i$ for some $i \in I$, then $\alpha \wedge \beta_k \wedge \bigvee_{j\in J_i} \beta_j$ is consistent: otherwise, $[\alpha] \bigvee_{j\in J_i} \beta_j \wedge \langle\alpha\rangle\beta_k$ implies $\langle\alpha\rangle\bot$ which is AX- (and $\mathsf{AX}^\dagger$) inconsistent. Thus for some $j \in J_i$, $\alpha \wedge \beta_k \wedge \beta_j$ is also consistent. In general α may be α_k for multiple $k \in K$. For each such k, we find such a β_j. We then set $f_\delta(\alpha)$ to the set of $\mathcal{L}_{\text{int}}$-equivalents of the $\alpha \wedge \beta_k \wedge \beta_j$, and set $f_\delta(\alpha)$ to the set of $\mathcal{L}_{\text{int}}$-equivalents of the $\alpha \wedge \beta_k$, if $\alpha \neq \alpha_i$ for any $i \in I$. The remaining case is that $\alpha \in \mathcal{S}_\delta$ but $\alpha \neq \alpha_k$ for any $k \in K$; in this case, set $f_\delta(\alpha) = \varnothing$.

Lemma 2. If $\mathsf{AX}^\dagger$ is AX_{det} or $\mathsf{AX}^{\downarrow}_{\text{det}}$ we can assume $f_\delta(\alpha)$ is a singleton (or possibly empty in the case of AX_{det}). If $\mathsf{AX}^\dagger$ is $\mathsf{AX}^{\downarrow}$ or $\mathsf{AX}^{\downarrow}_{\text{det}}$ we can assume that $\varnothing \notin \text{range}(f_\delta)$.

Proof. In AX_{det}, if $\langle\alpha\rangle\beta_1$ and $\langle\alpha\rangle\beta_2$, then because $[\alpha]\beta_1$ and $[\alpha]\beta_2$, and thus $[\alpha](\beta_1 \wedge \beta_2)$, we have $\langle\alpha\rangle(\beta_1 \wedge \beta_2)$. In $\mathsf{AX}^{\downarrow}$ it is always possible to assume that for each $i \in I$ there is some $j \in J_i$ such that $\langle\alpha_i\rangle\beta_j$ appears as a conjunct. So no such α_i will be sent to $\varnothing$. ☐

Lemma 3. Let δ be a disjunct as in (1). Let $\mathfrak{M} \in \mathcal{M}$. Suppose that $\mathfrak{M} \models \pi$, and for all $\alpha \in \mathcal{S}_\delta$ that $\mathfrak{M} \models \langle\alpha\rangle\beta$ for each $\beta \in f_\delta(\alpha)$, that $\mathfrak{M} \models [\alpha] \bigvee_{\beta \in f_\delta(\alpha)} \beta$, and that $\mathfrak{M} \models [\alpha]\bot$ whenever $f_\delta(\alpha) = \varnothing$. Then $\mathfrak{M} \models \delta$.

Proof. We show that $\mathfrak{M}$ satisfies every conjunct in (1); satisfaction of π is given. For conjuncts $\langle\alpha_k\rangle\beta_k$, for $k \in K$, suppose first that $\alpha_k \neq \alpha_i$, for any $i \in I$. Then $f_\delta(\alpha_k) = \{\alpha_k \wedge \beta_{k'} : k' \in K \text{ such that } \alpha_k = \alpha_{k'}\}$. If $\mathfrak{M} \models \langle\alpha_k\rangle(\alpha_k \wedge \beta_{k'})$ then $\mathfrak{M} \models \langle\alpha_{k'}\rangle\beta_{k'}$ for all such k'. Thus suppose $\alpha_k = \alpha_i$ for some $i \in I$. Again by the construction of f_δ, we have $f_\delta(\alpha_k) = \{\alpha_k \wedge \beta_{k'} \wedge \beta_{j_{k'}} : k' \in K \text{ such that } \alpha_k = \alpha_{k'}\}$ for some $j_{k'}$ where each $j_{k'} \in J_i$. Then $\mathfrak{M} \models \langle\alpha_k\rangle(\alpha_k \wedge \beta_{k'} \wedge \beta_{j_{k'}})$ implies $\mathfrak{M} \models \langle\alpha_k\rangle\beta_{k'}$ for each such k'. To see that $\mathfrak{M} \models [\alpha_i] \bigvee_{j\in J_i} \beta_j$ for each i such that $\alpha_i = \alpha_k$, by the assumption, we have $\mathfrak{M} \models [\alpha_k] \bigvee_{j \in J_i'} (\alpha_k \wedge \beta_{k'} \wedge \beta_j)$ for some $J_i' \subseteq J_i$. Generalizing the disjunction to J_i and distributing it through shows that $\mathfrak{M} \models [\alpha_k] \bigvee_{j\in J_i} \beta_j$. Finally, for conjuncts $[\alpha_i] \bigvee_{j\in J_i} \beta_j$ where $\alpha_i \neq \alpha_k$ for any $k \in K$, we have $f_\delta(\alpha_i) = \varnothing$ so that $\mathfrak{M} \models [\alpha_i]\bot$. But then $\mathfrak{M} \models [\alpha_i]\beta$ for any β whatsoever, so that such conjuncts are satisfied. ☐

Definition 4. Let PL be a programming language whose programs are the instances of ⟨*prog*⟩ in the following grammar:

⟨*const*⟩ ::= ‘`0`’ | ‘`1`’ ⟨*var*⟩ ::= X_1 | X_2 | … | X_n | …

⟨*cond*⟩ ::= ⟨*var*⟩ ‘`=`’ ⟨*const*⟩ | ⟨*var*⟩ ‘`=`’ ⟨*var*⟩
| ⟨*var*⟩ ‘`!=`’ ⟨*var*⟩ | ⟨*cond*⟩ ‘`&`’ ⟨*cond*⟩

⟨*assign*⟩ ::= ⟨*var*⟩ ‘`:=`’ ⟨*const*⟩ | ⟨*var*⟩ ‘`:=`’ ⟨*var*⟩ | ⟨*var*⟩ ‘`:= !`’ ⟨*var*⟩

⟨*branches*⟩ ::= ⟨*prog*⟩ | ⟨*branches*⟩ ‘`or`’ ⟨*branches*⟩

⟨*prog*⟩ ::= ‘’ | ⟨*assign*⟩ | ⟨*prog*⟩ ‘`;`’ ⟨*prog*⟩ | ‘`loop`’
| ‘`if`’ ⟨*cond*⟩ ‘`then`’ ⟨*prog*⟩ ‘`else`’ ⟨*prog*⟩ ‘`end`’
| ‘`choose`’ ⟨*branches*⟩ ‘`end`’

PL_{det} will denote the same language except that PL_{det} excludes `choose`-statements, $\mathsf{PL}^{\downarrow}$ is identical but for excluding `loop`-statements, and $\mathsf{PL}^{\downarrow}_{\text{det}}$ is identical but for excluding both `choose`- and `loop`-statements.

A program in any of these languages may be “compiled” to the right type of Turing machine in an obvious way (`loop` represents an unconditional infinite loop). For the remainder of the article, fix $\mathsf{PL}^\dagger$ to be the programming language of Defn. 4 corresponding to the choice of $\mathcal{M}^\dagger$.

With the normal form result and suitable languages in hand, we proceed to construct the canonical model $\mathfrak{M}_\varphi = (\mathsf{T}_\varphi, \mathbf{x}_\varphi)$ for φ. $\mathfrak{M}_\varphi$ need only satisfy a consistent clause δ as in (1). Intuitively, $\mathfrak{M}_\varphi$ will satisfy $\mathcal{L}_{\text{prop}}$-atoms in δ via a suitable tape state $\mathbf{x}_\varphi$ (existent as δ and *a fortiori* π is consistent), and will satisfy each $\mathcal{L}_{\text{cond}}$-atom by dint of a branch in T_φ, conditional on the antecedent, in which the consequent is made to hold. We now write the $\mathsf{PL}^\dagger$-code of such a T_φ.

Suppose we are given δ, and that for each $\alpha \in \mathcal{S}_\delta$ we have code `HoldsFromIntervention`(α) defining a condition that is met iff the program is currently being run under an intervention that fixes α to be true. Then consider a PL-program P_φ that contains one `if`-statement for each $\alpha \in \mathcal{S}_\delta$, each executing if `HoldsFromIntervention`(α) is met. In the body of the `if`-statement for α, P_φ has a `choose`-statement with one branch for each $\beta \in f_\delta(\alpha)$. The branch for each β consists of a sequence of assignment statements guaranteed to make β hold, call this `MakeHold`(β), clearly existent since each β is satisfiable. If $f_\delta(\alpha)$ is a singleton, this body contains only `MakeHold`(β); if $f_\delta(\alpha) = \varnothing$, then this body consists of a single `loop`-statement. If T_φ is the machine corresponding to P_φ, and $\mathbf{x}_\varphi$ is a tape state satisfying π, then $\mathfrak{M}_\varphi \models \langle\alpha\rangle\beta$ for each $\beta \in f_\delta(\alpha)$, as the program has a halting branch with `MakeHold`(β); also, $\mathfrak{M}_\varphi \models [\alpha] \bigvee_{\beta \in f_\delta(\alpha)} \beta$ as there are no other halting executions. If $f_\delta(\alpha) = \varnothing$, then $\mathfrak{M}_\varphi \models [\alpha]\bot$, since under an α-fixing intervention the program reaches a loop-statement and has no halting executions. So by Lem. 3, we have that $\mathfrak{M}_\varphi$ satisfies δ. And thus φ. To see that $\mathfrak{M}_\varphi \in \mathcal{M}^\dagger$, apply Lem. 2: in $\mathsf{AX}^\downarrow$, $\varnothing \notin \text{range}(f_\delta)$ so we have no loops in P_φ and $\mathfrak{M}_\varphi \in \mathcal{M}^\downarrow$. In AX_{det}, we have no choose-statements, so $\mathfrak{M}_\varphi \in \mathcal{M}_{\text{det}}$; in $\mathsf{AX}^\downarrow_{\text{det}}$, we have neither loop- nor choose-statements, and $\mathfrak{M}_\varphi \in \mathcal{M}^\downarrow_{\text{det}}$.

But how do we know it is possible to write code HoldsFromIntervention(α) by which the program can tell whether it is being run under an α-fixing intervention? For any tape variable, we may try to toggle it. If the attempt succeeds, then the variable is not presently fixed by an intervention. If not, then the present execution is under an intervention fixing the variable. Thus, we first try to toggle each *relevant* variable. Let N be the maximum index i of any atom X_i appearing in φ. Listing 1—call it IsIntervened(X_i)— performs the toggle check for X_i and records the result in X_{i+N}. It uses X_{i+2N} as a temporary variable and ultimately leaves the value of X_i unchanged.

Listing 1: IsIntervened(X_i)

```
$X_{i+N}$ := $X_i$;
$X_i$ := ! $X_i$;
$X_{i+2N}$ := $X_i$;
if $X_{i+2N}$ = $X_{i+N}$ then $X_{i+N}$ := 1
else $X_{i+N}$ := 0 end;
$X_i$ := ! $X_i$;
```

If IsIntervened(X_i) has already been run for all $1 \le i \le N$, HoldsFromIntervention(α) simply checks that exactly those variables appearing in α have been marked as intervened on, and that these have the correct values. If α is the $\mathcal{L}_{\text{int}}$-equivalent of $\neg X_{i_1} \wedge \ldots \wedge \neg X_{i_k} \wedge X_{i_{k+1}} \wedge \ldots \wedge X_{i_n}$, code for HoldsFromIntervention(α) is given in Listing 2.

Listing 2: HoldsFromIntervention(α)

```
$X_{i_1}$ = 0 & ... & $X_{i_k}$ = 0 &
    $X_{i_{k+1}}$ = 1 & ... & $X_{i_n}$ = 1 &
$X_{i_1+N}$ = 1 & ... & $X_{i_k+N}$ = 1 &
    $X_{i_{k+1}+N}$ = 1 & ... & $X_{i_n+N}$ = 1
```

Completing the description of the code of P_φ adumbrated earlier, P_φ consists of, in order:

1. One copy of IsIntervened(X_i) for each $1 \le i \le N$.
2. For each $\alpha \in \mathcal{S}_\delta$, an if-statement with condition HoldsFromIntervention(α), whose body is:
   (a) a choose-statement with a branch for each $\beta \in f_\delta(\alpha)$, with body MakeHold($\beta$), if $|f_\delta(\alpha)| \ge 2$;
   (b) a MakeHold(β)-snippet, if $|f_\delta(\alpha)| = 1$;
   (c) or a single loop-statement if $f_\delta(\alpha) = \varnothing$.

□

Note that P_φ never reads or writes a variable X_i for $i > 3N$, and the relevant $\mathsf{PL}^\dagger$-operations may be implemented with bounded space, so that we have the following Corollary:

Corollary 1. Let $\mathcal{M}^{\dagger,\text{fin}}$ be the class of *finite state machine* restrictions of $\mathcal{M}^\dagger$, i.e. those $(\mathsf{T}, \mathbf{x}) \in \mathcal{M}^\dagger$ where T uses only boundedly many tape variables, for any input and intervention. Then Thm. 1 holds also for $\mathcal{M}^{\dagger,\text{fin}}$. □

6 Computational Complexity

In this section we consider the problem SIM-SAT(φ) of deciding whether a given $\varphi \in \mathcal{L}$ is satisfiable in $\mathcal{M}^\dagger$. Although by Prop. 1, it is in general undecidable whether a given *particular* simulation model satisfies a formula, we show here that it is decidable whether a given formula is satisfied by *any* model. In fact, reasoning in this framework is no harder than reasoning in propositional logic:

Theorem 2. SIM-SAT(φ) is NP-complete in $|\varphi|$ (where $|\varphi|$ is defined standardly).

Proof. We clearly have NP-hardness as propositional satisfiability can be embedded directly into $\mathcal{L}$-satisfiability. To see that satisfiability is NP, we guess a $\mathfrak{M}$ and check whether $\mathfrak{M} \models \varphi$. $\mathcal{M}^\dagger$ is infinite, and the checking step is undecidable by Prop. 1. So how could such an algorithm work? The crucial insight is that we may limit our search to a finite class $\mathcal{M}^\dagger_\varphi$ of models that are similar to the canonical $\mathfrak{M}_\varphi$ (Lem. 5). Moreover, a nice property of the canonical T_φ is that it wears its causal structure on its sleeves: one can read off the effect of any intervention from the code of P_φ, and P_φ has polynomial size in $|\varphi|$ (implied by Lem. 4). Models in $\mathcal{M}^\dagger_\varphi$ will share this property, guaranteeing that the checking step can be done in polynomial time. We will now make $\mathcal{M}^\dagger_\varphi$ precise and prove these claims. Let $\mathcal{S}_\varphi \subset \mathcal{L}_{\text{int}}$ denote the set of $\mathcal{L}_{\text{cond}}$-antecedents appearing in φ. For $C \in \mathbb{N}$, define $\mathsf{PL}^\dagger_{\varphi,C} \subset \mathsf{PL}^\dagger$ as the fragment of programs whose code consists of:

1. One copy of IsIntervened(X_i) (Listing 1), for each $1 \le i \le N$, followed by
2. at most one copy of an if-statement with condition HoldsFromIntervention(α) (Listing 2) for each $\alpha \in \mathcal{S}_\varphi$, whose body is one and only one of the following options, (a)–(c):
   (a) a choose-statement with at most $C|\varphi|$ branches, each of which has a body consisting of a single sequence of assignments, which may only be to variables X_i for $1 \le i \le N$;
   (b) a single sequence of assignment statements, only to variables X_i for $1 \le i \le N$;
   (c) a single loop-statement.

   However, if $\mathsf{PL}^\dagger = \mathsf{PL}_{\text{det}}$, (a) is not allowed; if $\mathsf{PL}^\dagger = \mathsf{PL}^\downarrow$, (c) is not allowed; and if $\mathsf{PL}^\dagger = \mathsf{PL}^\downarrow_{\text{det}}$, neither (a) nor (c) is allowed.

Lemma 4. The maximum length (defined standardly) of a program in $\mathsf{PL}^\dagger_{\varphi,C}$ is polynomial in $|\varphi|$, and there is a C such that for all φ, we have $\mathsf{P}_\varphi \in \mathsf{PL}^\dagger_{\varphi,C}$, assuming P_φ exists.

Proof. N is $\mathcal{O}(|\varphi|)$, so part 1 of a program is $\mathcal{O}(|\varphi|)$ in length. There are at most $|\mathcal{S}_\varphi|$ if-statements in part 2; consider the body of each one. In case (a) it has $\mathcal{O}(|\varphi|)$ branches, each of which involves assignment to at most N variables, and thus has length $\mathcal{O}(|\varphi|^2)$. In case (b) its length is $\mathcal{O}(|\varphi|)$; in case (c) its length is $\mathcal{O}(1)$. Since $|\mathcal{S}_\varphi|$ is $\mathcal{O}(|\varphi|)$, the total length of part 2 is $\mathcal{O}(|\varphi|^3)$, so that both parts combined are $\mathcal{O}(|\varphi|^3)$. To show the existence of C, it suffices to prove:

any `choose`-statement in the body of an `if`-statement in P_φ has $\mathcal{O}(|\varphi|)$ branches. Now, the number of branches in the `if`-statement for α is $|f_\delta(\alpha)|$, for some consistent δ as in (1). But (1) is a clause of the disjunctive normal form of φ and contains no more $\mathcal{L}$-literals than does φ, which is of course $\mathcal{O}(|\varphi|)$. Since each element of $f_\delta(\alpha)$ arises from the selection of a literal in (1), the number of branches is $\mathcal{O}(|\varphi|)$. □

Henceforth let $\mathsf{PL}^\dagger_\varphi$ denote $\mathsf{PL}^\dagger_{\varphi,C}$ for some C guaranteed by Lem. 4, and call the set of $\mathbf{x}$ where only tape variables X_i with indices $1 \le i \le N$ are possibly nonzero $\mathcal{X}_N$. Let $\mathcal{M}^\dagger_\varphi$ be the class of models $(\mathsf{T}, \mathbf{x})$ where T comes from a $\mathsf{PL}^\dagger_\varphi$-program and $\mathbf{x} \in \mathcal{X}_N$. $\mathcal{M}^\dagger_\varphi$ is finite, and the following Lemma guarantees that we may restrict the search to $\mathcal{M}^\dagger_\varphi$:

Lemma 5. φ is satisfiable with respect to $\mathcal{M}^\dagger$ iff it is satisfiable with respect to $\mathcal{M}^\dagger_\varphi$.

Proof. If φ is satisfiable in $\mathcal{M}^\dagger$, it is $\mathsf{AX}^\dagger$-consistent by soundness, and hence has a canonical $(\mathsf{T}_\varphi, \mathbf{x}_\varphi)$. Without loss of generality take $\mathbf{x}_\varphi$ from Thm. 1 to be in $\mathcal{X}_N$. Then by Lem. 4, $(\mathsf{T}_\varphi, \mathbf{x}_\varphi) \in \mathcal{M}^\dagger_\varphi$, so φ is satisfiable in $\mathcal{M}^\dagger_\varphi$. □

Now with Lem. 5 our algorithm will guess a program $\mathsf{P} \in \mathsf{PL}^\dagger_\varphi$ and a tape $\mathbf{x} \in \mathcal{X}_N$, and verify whether the guessed model $\mathfrak{M} \in \mathcal{M}^\dagger_\varphi$ satisfies φ. We just need to show that the verification step is decidable in polynomial time. Suppose that all negations in φ appear only before $\mathcal{L}$-atoms, since any formula may be converted to such a form in linear time. Further, rewrite literals of the form $\neg[\alpha]\beta$ to $\langle\alpha\rangle\neg\beta$. Then it suffices to show that we can decide in polynomial time whether $\mathfrak{M}$ satisfies a given literal in φ: there are linearly many of these and the truth-value of φ may be evaluated from their values in linear time. For an X-literal X_i or $\neg X_i$, we simply output whether or not $\mathbf{x} \models X_i$. For $\mathcal{L}_{\text{cond}}$-literals with antecedent α, simulate execution of $\mathcal{I}_\alpha(\mathsf{T})$ on $\mathbf{x}$. Because $\mathsf{P} \in \mathsf{PL}^\dagger_\varphi$ and such programs trigger at most one `HoldsFromIntervention`(α) `if`-statement when run under an intervention, we may perform this simulation by checking if there is any `if`-statement for α in P. If so, do one of the following, depending on what its body contains:

(a) If a `choose`-statement, simulate the result of running each branch. Output true iff: either the literal was $[\alpha]\beta$ and every resulting tape satisfies β, or the literal was $\langle\alpha\rangle\beta$ and at least one resulting tape satisfies β.

(b) If an assignment sequence, simulate running it on the current tape, and output true iff the resulting tape satisfies β.

(c) If a `loop`, output true iff the literal is of the $[\alpha]\beta$ form.

This algorithm is correct since we thereby capture all halting executions, given that $\mathsf{PL}^\dagger_\varphi$-programs conform to the fixed structure above. That it runs in polynomial time follows from the polynomial-length bound of Lem. 4. □

7 Conclusion and Future Work

A very natural way to assess a claim, 'if α were true, then β would be true,' is to run a simulation in which α is assumed to hold and determine whether β would then follow. Simulations can be built using any number of tools: (probabilistic) programming languages designed specifically for generative models, generative neural networks, and many others. Our formulation of *intervention on a simulation program* is intended to subsume all such applications where conditional reasoning seems especially useful. We have shown that this general way of interpreting conditional claims has its own distinctive, and quite weak, logic. Due to the generality of the approach, we can validate further familiar axioms by restricting attention to smaller classes of programs (deterministic, always-halting). We believe this work represents an important initial step in providing a foundation for conditional reasoning in these increasingly common contexts.

To close, we would like to mention several notable future directions. Perhaps the most obvious next step is to extend our treatment to richer languages, and in particular to the first order setting. This is pressing for several reasons. First, much of the motivation for many of the generative frameworks mentioned earlier was to go beyond the propositional setting characteristic of traditional graphical models, for example, to be able to handle unknown (numbers of) objects (see [Poole, 2003; Milch *et al.*, 2005]).

Second, much of the work in conditional logic in AI has dealt adequately with the first order setting by using frameworks based on normality orderings [Delgrande, 1998; Friedman *et al.*, 2000]. It is perhaps a strike against the structural equation approach that no one has shown how to extend it adequately to first order languages (though see [Halpern, 2000] for partial suggestions). In the present setting, just as we have used a tape data structure to encode a propositional valuation, we could also use such data structures to encode first order models. The difficult question then becomes how to understand complex (i.e., arbitrary first-order) interventions. We have begun exploring this important extension.

Given the centrality of probabilistic reasoning for many of the aforementioned tools, it is important to consider the probabilistic setting. Adding explicit probability operators in the style of [Fagin *et al.*, 1990] results in a very natural extension of the system [Ibeling, 2018]. One could also use probability thresholds (see, e.g, [Hawthorne and Makinson, 2007]): we might say $(\mathsf{T}, \mathbf{x}) \models [\alpha]\beta$ just when $\mathcal{I}_\alpha(\mathsf{T})$ results in output satisfying β with at least some threshold probability.

Finally, another direction is to consider additional subclasses of programs, even for the basic propositional setting we have studied here. For example, in some contexts it makes sense to assume that variables are time-indexed and that no variable depends on any variable at a later point in time (as in dynamic Bayesian networks [Dean and Kanazawa, 1989]). In this setting there are no cyclic dependencies, which means we do not have programs like that in Example 1. Understanding the logic of such classes would be worthwhile, especially for further comparison with central classes of structural equation models (such as the "recursive" models of [Pearl, 2009]).

Acknowledgments

Duligur Ibeling is supported by the Sudhakar and Sumithra Ravi Family Graduate Fellowship in the School of Engineering at Stanford University.

References

[Balkenius and Gärdenfors, 1991] Christian Balkenius and Peter Gärdenfors. Nonmonotonic inferences in neural networks. In *Proceedings of KR*, 1991.

[Bottou *et al.*, 2013] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Danis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260, 2013.

[de Raedt and Kimmig, 2015] Luc de Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.

[Dean and Kanazawa, 1989] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2):142–150, 1989.

[Delgrande, 1998] James P. Delgrande. On first-order conditional logics. *Artificial Intelligence*, 105:105–137, 1998.

[Dubois and Prade, 1991] Didier Dubois and Henri Prade. Fuzzy sets in approximate reasoning, Part 1: Inference with possibility distributions. *Fuzzy Sets and Systems*, 40(1):182–224, 1991.

[Fagin *et al.*, 1990] Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[Friedman *et al.*, 2000] Nir Friedman, Joseph Y. Halpern, and Daphne Koller. First-order conditional logic for default reasoning revisited. *ACM Transactions on Computational Logic*, 1(2):175–207, 2000.

[Ginsberg, 1986] Matthew L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986.

[Goodman *et al.*, 2008] Noah D. Goodman, Vikash K. Mansinghka, Daniel Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. In *Proc. 24th UAI*, 2008.

[Goodman *et al.*, 2015] Noah D. Goodman, Joshua B. Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. In Eric Margolis and Stephan Laurence, editors, *The Conceptual Mind: New Directions in the Study of Concepts*. MIT Press, 2015.

[Halpern, 2000] Joseph Y. Halpern. Axiomatizing causal reasoning. *Journal of AI Research*, 12:317–337, 2000.

[Halpern, 2013] Joseph Y. Halpern. From causal models to counterfactual structures. *Review of Symbolic Logic*, 6(2):305–322, 2013.

[Hawthorne and Makinson, 2007] James Hawthorne and David Makinson. The quantitative/qualitative watershed for rules of uncertain inference. *Studia Logica*, 86(2):247–297, 2007.

[Ibeling, 2018] Duligur Ibeling. Causal modeling with probabilistic simulation models. Manuscript, 2018.

[Icard, 2017] Thomas F. Icard. From programs to causal models. In Alexandre Cremers, Thom van Gessel, and Floris Roelofsen, editors, *Proceedings of the 21st Amsterdam Colloquium*, pages 35–44, 2017.

[Kocaoglu *et al.*, 2017] Murat Kocaoglu, Christopher Snyder, Alexandros G. Dimakis, and Sriram Vishwanath. CausalGAN: Learning causal implicit generative models with adversarial training. Unpublished manuscript: https://arxiv.org/abs/1709.02023, 2017.

[Kraus *et al.*, 1990] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(2):167–207, 1990.

[Leitgeb, 2004] Hannes Leitgeb. *Inference on the Low Level: An Investigation Into Deduction, Nonmonotonic Reasoning, and the Philosophy of Cognition*. Kluwer, 2004.

[Lewis, 1973] David Lewis. *Counterfactuals*. Harvard University Press, 1973.

[Liang and Jordan, 2008] Percy Liang and Michael I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *25th ICML*, 2008.

[Makinson, 1994] David Makinson. General patterns in nonmonotonic reasoning. In D. Gabbay et al., editor, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume III, pages 35–110. OUP, 1994.

[Meek and Glymour, 1994] Christopher Meek and Clark Glymour. Conditioning and intervening. *The British Journal for the Philosophy of Science*, 45:1001–1021, 1994.

[Milch *et al.*, 2005] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic models with unknown objects. In *Proc. 19th IJCAI*, pages 1352–1359, 2005.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial networks. Manuscript: https://arxiv.org/abs/1709.02023, 2014.

[Newell and Simon, 1961] Allen Newell and Herbert A. Simon. Computer simulation of human thinking. *Science*, 134(3495):2011–2017, 1961.

[Pearl, 2009] Judea Pearl. *Causality*. CUP, 2009.

[Pfeffer and Koller, 2000] Avi Pfeffer and Daphne Koller. Semantics and inference for recursive probability models. In *Proc. 7th AAAI*, pages 538–544, 2000.

[Poole, 2003] David Poole. First-order probabilistic inference. In *Proc. 18th IJCAI*, 2003.

[Sloman, 2005] Steven A. Sloman. *Causal Models: How We Think About the World and its Alternatives*. OUP, 2005.

[Stalnaker, 1968] Robert Stalnaker. A theory of conditionals. *American Philosophical Quarterly*, pages 98–112, 1968.

[Wachter *et al.*, 2018] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law and Technology*, 2018.