Counterfactual Resimulation for Causal Analysis of Rule-Based Models

Jonathan Laurent[1], **Jean Yang**[1], **Walter Fontana**[2],
[1] Carnegie Mellon University
[2] Harvard Medical School
{jonathan.laurent, jyang2+}@cs.cmu.edu, walter_fontana@hms.harvard.edu

Abstract

Models based on rules that express local and heterogenous mechanisms of stochastic interactions between structured agents are an important tool for investigating the dynamical behavior of complex systems, especially in molecular biology. Given a simulated trace of events, the challenge is to construct a causal diagram that explains how a phenomenon of interest occurred. Counterfactual analysis can provide distinctive insights, but its standard definition is not applicable in rule-based models because they are not readily expressible in terms of structural equations. We provide a semantics of counterfactual statements that addresses this challenge by sampling counterfactual trajectories that are probabilistically as close to the factual trace as a given intervention permits them to be. We then show how counterfactual dependencies give rise to explanations in terms of relations of enablement and prevention between events.

1 Introduction

Rule-based modeling languages for molecular biology, such as Kappa [Danos *et al.*, 2007a] and BioNetGen [Harris *et al.*, 2016], or organic chemistry, such as Mød [Andersen *et al.*, 2016], can be used to write mechanistic models of complex reaction systems. These approaches consider entities that have a structure, and make a distinction between the transformation of a structure fragment (a pattern) specified by a rule and the reaction resulting from the application of the rule to a combination of entities contextualizing the fragment. The structure of bio-molecular entities is represented as a graph and a rule is a graph-rewrite directive with a rate constant that determines its propensity to apply. The stochastic simulation of a rule collection generates a time series of rule applications—henceforth events—that might reach a state of interest in processes like the assembly of a molecular machine, the activation of a transcription factor, or the synthesis of a specific compound.

While rule-based models provide compactness, transparency, and the ability of handling combinatorial complexity, the perhaps most significant advantage lies in their suitability for causal analysis. This is because such analysis proceeds at the level of rules, not reactions, thereby avoiding contamination with context that defines a reaction yet is irrelevant to the application of the underlying rule. Due to the concurrent nature of events, it is typically far from obvious how a sequence attained a particular outcome. Biologists often refer to a causal account or explanation as a "pathway", but have no formal framing for it.

Prior work in causal analysis [Danos *et al.*, 2012; Danos *et al.*, 2007a] takes advantage of rule structure to (i) compress a simulation trace into a minimal subset of events that are necessary and jointly sufficient to replicate the outcome of interest and (ii) highlight causal influences between events, exposing the extent of concurrency. Such analysis is performed on a sample of traces to the outcome, thus recovering the salient pathways as those that are statistically favored by the dynamics. This approach, however, suffers from two drawbacks. First, the focus on necessity in step (i) neglects events that are kinetically critical (in that they dramatically increase the probability of observing the outcome), yet are not logically necessary for achieving it. Second, step (ii) is limited to a narrow notion of causal influence that we may call *enablement*. Put simply, an event a (directly) enables event b, if a modifies the state of the world so as to satisfy the requirements for b to occur. This positively tinted version of influence is blind to the ubiquitous role of inhibitory interactions in molecular biology. Indeed, an event a may cause an event b without (transitively) enabling it, but instead by preventing another event c that would have prevented b. Clearly, uncovering such an explanatory narrative is challenging because it involves an event, c in this case, that did *not* occur in a simulation trace.

We here propose an approach that complements the existing causal analysis of event series generated from rule-based models by using counterfactual reasoning to answer questions of the kind: *Had event e_1 not occurred, would event e_2 have happened?* Our contributions may be summarized as follows.

1. We provide a semantics for counterfactual statements in the context of rule-based models, where the standard definition of counterfactuals based on structural equations [Pearl, 2009; Halpern, 2016] does not apply.

2. We show how such statements can be evaluated by sampling *counterfactual traces* that are meant to probabilistically "hug" a given (factual) trace as much as an external

intervention permits them to. To this end, we introduce an algorithm to generate counterfactual traces and provide an efficient implementation for the Kappa language.

3. We show how counterfactual dependencies between events can be systematically explained in terms of enablement and prevention relations that are more in line with biological reasoning.

2 Motivating example

We provide some background on Kappa and introduce a toy example motivating the need for counterfactual reasoning in analyzing the causal structure of simulation traces.

2.1 Some Background on Kappa

Proteins are complex molecular machines that reversibly tag one another with molecular flags and reversibly bind each other to form transient associations. In this way, proteins come to have "state" that can control the interactions they engage in. In Kappa, a protein is modeled as an abstract *agent* with a name that designates its *type* and a signature of distinguishable *sites* at which it can be tagged or bound by other agents. A site can bind at most one agent at a time and must be in a definite state.

In our illustrative example, we consider two types of agents for which we use biological nomenclature. One type, S, is a substrate that receives a tag known as a phosphate group in a phosphorylation interaction. The other, K, is a kinase that phosphorylates S. Agents of both types feature a binding site at which they can bind one another and a site that can be in one of two possible states: *unphosphorylated* or *phosphorylated*. Thus, agents of type K also have a phosphorylation state, but for the sake of simplicity we will have them acquire it "spontaneously".

A *mixture* is a multiset of agents whose state at each site is fully specified. A mixture represents the state of a system and can be thought of as a (potentially large) graph consisting of many connected components. In a mixture, agents of the same type are distinguished by a unique global identifier.

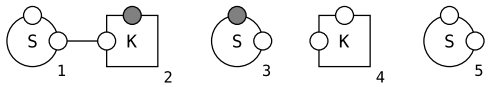

Figure 1: A mixture graph. Sites are here identified by their position on agents instead of a name. A grey site indicates a phosphorylated state. Numbers are global agent identifiers in the mixture.

Interactions between agents are modeled by local rewriting *rules*. A rule r is defined by a triple (L_r, R_r, λ_r), where L_r is the left-hand side specifying a pattern (the pre-condition), R_r the right-hand side (or post-condition), and λ_r a firing rate. The basic idea is that the "location" at which the mixture matches L_r is rewritten in place by R_r, changing the state of the system. (Technically, a rule also requires a function from agents in L_r to agents in R_r to make the rewrite unambiguous.)

In our toy model, agents are subject to the rules depicted in Figure 2. Rule b states that kinases and substrates can bind,

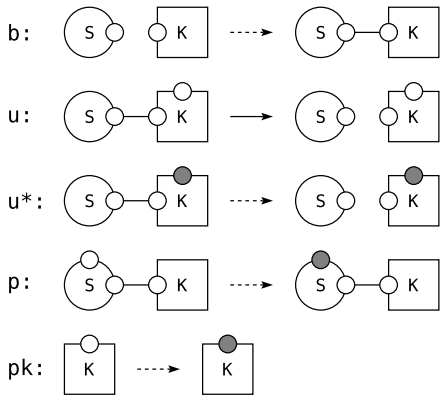

Figure 2: A motivating toy model. Sites not mentioned in a rule are left unchanged by it. Firing rates are not specified, but dotted (solid) arrows indicate *slow* (*fast*) reactions ($\lambda_u \gg \lambda_{u^*} \approx \lambda_p$).

provided their requisite binding sites are free (unbound). Note that L_b is a pattern: It omits mentioning the sites that carry phosphorylation state, which are therefore not considered when matching the mixture to L_b. Rules u and u^* define unbinding events that depend on the phosphorylation state of the kinase K. The distinction is motivated by kinetics: Rule u fires at a much faster rate than u^*. Rule p specifies that a substrate can be phosphorylated when it is bound to a kinase. For the sake of simplicity, we model the phosphorylation of a kinase as a spontaneous process (rule pk).

By virtue of the λ_r, the rules of a model, together with an initial mixture, constitute a dynamical system that we describe shortly. We do so in a slightly nonstandard way by introducing the auxiliary concept of an *event template*. This is to prepare for the insight in section 3.1 that the stochastic and deterministic aspects of a model's dynamics can be cleanly separated, thus enabling counterfactual analysis.

An event template is a pair (r, ξ) where r is a rule and ξ a function from agents in L_r to global identifiers. We say that (r, ξ) is *realizable* in mixture m if the global identifiers assigned by ξ exist in m and the agents bearing them match L_r. In this case, we write $m \vdash (r, \xi)$ and call ξ an *embedding* of L_r into m: $\textsc{Emb}_r(m) \triangleq \{\xi : m \vdash (r, \xi)\}$. Whenever $m \vdash (r, \xi)$, we write $m \cdot (r, \xi)$ the mixture obtained by realizing (r, ξ), i.e. by rewriting the agents with identifiers in the codomain of ξ into R_r. The realization of an event template at a particular time creates an (actual) *event*, formally defined as a pair (e, t) with e an event template and t its time of realization.

A model induces a continuous-time Markov chain (CTMC) over the set of reachable mixtures, where state m transitions to state $m \cdot (r, \xi)$ at rate λ_r for every event template (r, ξ) that is realizable in m. The rate of leaving state m by an application of rule r is called the *activity* $\alpha_r(m)$ of rule r in mixture m and is equal to the product of the rule's firing rate by the number of embeddings of L_r into m: $\alpha_r(m) = \lambda_r |\textsc{Emb}_r(m)|$. For example, in Figure 1, rule b has activity $2\lambda_b$ and rule u has activity 0. The *total activity* of a mixture is defined as $\alpha(m) = \sum_r \alpha_r(m) = \sum_r \lambda_r |\textsc{Emb}_r(m)|$.

The CTMC induced by a model can be simulated with the *Doob-Gillespie algorithm* [Gillespie, 1977], which loops over the following steps: (1) draw a time interval δ to the next event from an exponential distribution with parameter $\alpha(m)$ and increment the simulated system time by δ, (2) draw a rule r with probability $\alpha_r(m)/\alpha(m)$ and (3) pick uniformly an embedding $\xi \in \text{EMB}_r(m)$ of L_r into m and realize the event template (r, ξ). This algorithm is efficiently implemented for rule-based models in Kappa as described in [Danos *et al.*, 2007b; Boutillier *et al.*, 2017]. It outputs a sequence of events, called a *trace*.

2.2 Where Existing Analysis Falls Short

Consider our toy model and assume, for the sake of illustration, an initial mixture I with only a single kinase and a single substrate whose sites are unbound and unphosphorylated. We then ask: Starting from I, *how* is p typically achieved? We are not merely looking for an account of reachability but for a causal explanation, i.e. a collection of events connected by causal influences.

For example, a simulation might produce the following trace (events are labeled by the rules that induced them):

$$b,\ u,\ pk,\ b,\ p,\ u^*,\ \cdots \tag{1}$$

Current techniques [Danos *et al.*, 2012; Danos *et al.*, 2007a] generate a causal account by first computing a *sub-trace* of (1) that is (i) *valid* in the sense that each of its events can be triggered in turn starting from the initial mixture and (ii) *minimal* in the sense that none of its valid sub-traces features p. The relations of enablement among events in the minimal sub-trace yield a directed acyclic graph. Although trivial in our toy example, minimization is an NP-hard problem. Carrying out this approach on (1), one notes that the first occurence of b sets the necessary conditions for p, but these are subsequently undone by u only for the second occurrence of b to re-introduce them. This illustrates why minimization compresses a trace into events that are *necessary* for the outcome—which requires eliminating futile cycles. In our case, the causal account for p starts with the initial condition, symbolized by the *init* event, and skips to the last b before the u. Figure 3 depicts the resulting explanation graph, whose arrows denote enablement as defined informally in section 1 and formally in section 4.

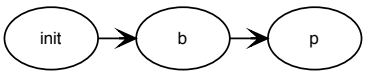

Figure 3: A causal explanation for p in trace (1). Events are labelled by the rules that induced them. The *init* node corresponds to a special event that sets the mixture to its initial state.

The problem is that the explanation depicted in Figure 3 fails to recognize the critical role of pk in the original trace. Given the rules of the model, one notes that p is slow and the average time that K remains bound to S depends on the phosphorylation state of K. The kinase K is phosphorylated in event pk, causing the complex between K and S to be sticky, giving the slow phosphorylation p a chance to occur. It seems reasonable to assert that p would probably not have happened had pk not happened, since the opportunity for p would have been curtailed by a fast unbinding event. Thus, pk should be part of the explanation, although it neither enables b nor p directly (both rules are independent of K's phosphorylation state). Reasoning of this kind is *counterfactual* [Lewis, 1974; Pearl, 2009].

In section 3, we give a rigorous semantics to this line of reasoning and introduce an algorithm for simulating counterfactual scenarios. In section 4, we show how counterfactual dependencies between events can be systematically explained in terms of a combination of enablement and prevention arrows, leading to the explanation shown in Figure 4.

3 Evaluating Counterfactual Statements

In our example, the counterfactual statement to be assessed is: "Had pk not happened, p would not have happened." Our account in the previous section suggests that pk played a role, but it is also clear that given the stochastic nature of rule firing p could well have happened even in the absence of pk; it just is unlikely. In a stochastic setting, counterfactual statements are not either true or false, but have degrees of likelihood. To assess that likelihood is our task.

Given an original (factual) trace τ, a naive approach might be to sample counterfactual traces, each of which starts with the state of the system attained in τ just before event pk happened, but in which we skip over pk and then run an unconstrained simulation from that point onward. In this approach traces would quickly diverge from the original, distorting the causal role that pk played specifically in it. The question here is not what causal role pk can play in principle, but what role it actually did play in τ. Counterfactual statements are undetachable from the context in which they are formulated.

Pearl's standard account of counterfactuals [Pearl, 2009] is based on performing *"surgical interventions"* on a structural equation model (SEM). A SEM features a finite sequence $(x_1, \ldots, x_n)$ of variables, each associated to a *functional equation* of the form $x_i = f_i(x_1, \ldots, x_{i-1}, u_i)$, where f_i is a deterministic function and u_i a random variable. Ideally, each f_i defines an independent and autonomous physical mechanism. This is partially enforced by the requirement that the u_i must be mutually independent. Given some observation e, the probability of the counterfactual statement "had x_j been equal to a, ψ would have been true" is evaluated following a three-step process: (*abduction*) compute the distribution p_e of values for $\vec{u}$ given observation e, then (*action*) intervene in the model by replacing the defining equation for x_j by "$x_j = a$" and finally (*prediction*) compute the probability that ψ is true in this new model when $\vec{u}$ is distributed according to p_e.

Because of their dynamic nature, rule-based models are not readily expressible in terms of structural equations. However, Pearl's construction generalizes to our setting, assuming a *structural* refinement of Kappa's probabilistic semantics where deterministic causal mechanisms are separated from stochastic aspects.

3.1 A Refined Semantics for Kappa

The factual trace to which a counterfactual statement is tied includes random contingencies. These contingencies reflect the stochasticity underlying the specific sequence and timing of events, as described in section 2.1. Executing a counterfactual experiment requires that we separate this randomness from the deterministic action of rules in order to properly condition on the randomness of the factual trace. We propose such a decomposition, which is motivated by a standard construction in physics that justifies the Doob-Gillespie algorithm used in simulating reaction systems [Gillespie, 1977].

Our refined semantics reconceptualizes the CTMC induced by a model as follows: (i) Consider all possible event templates (r, ξ), where ξ maps into a large enough set of global identifiers. For each event template, imagine an independent Poisson process in which a bell rings at time intervals drawn independently from an exponential distribution with parameter λ_r. These Poisson processes are all gathered in a random variable Σ. A realization of Σ is called a *schedule* and it features a sequence of ringing times for every bell. (ii) With every schedule σ, we associate a unique trace $\mathcal{T}(\sigma)$ that is generated as follows: starting with the initial mixture and moving through time, whenever a bell rings, its associated event template e is realized (by transforming the current mixture m into $m \cdot e$) if and only if $m \vdash e$. For example, if the current mixture m is given as in Figure 1 and the bell linked to *"apply rule* b *on substrate* 3 *and kinase* 4*"* rings, a bond is created between these two agents. In contrast, the bell linked to *"apply rule* b *on substrate* 1 *and kinase* 2*"* would have no effect. (iii) Finally, the dynamic behavior of our model is captured by the random trace $T \triangleq \mathcal{T}(\Sigma)$, which can also be sampled efficiently using the Doob-Gillepsie algorithm introduced in section 2.1. Note that the assumption underlying this whole construction is entirely contained in the existence and definition of $\mathcal{T}$. Given $\mathcal{T}$, the statistical properties of Σ (including the independence of our metaphorical bells) are consequences of Kappa's original probabilistic semantics.

Intuitively, Σ determines when the opportunity for a reaction happens and between which molecules. It plays the same role as the random vector $\vec{u}$ in a SEM. In contrast, $\mathcal{T}$ is a deterministic function that controls whether a reaction can occur when given the opportunity and what it does when it occurs. It corresponds to the f_i in a SEM and is likewise the target of interventions.

3.2 A Semantics for Counterfactuals

We define an intervention ι as a predicate $\text{BLOCKED}_\iota[\cdot]$ ranging over events. The purpose of the predicate is to act as a filter preventing the occurrence of selected events. Given a predicate ψ over traces, we write the statement *"Had intervention* ι *happened in trace* τ*,* ψ *would have been true"* as $\tau \models [\iota]\,\psi$, borrowing a notation from [Halpern, 2016].

For an intervention ι and a schedule σ, we define the altered trace $\mathcal{T}_\iota(\sigma)$ much in the same way as $\mathcal{T}(\sigma)$, but also requiring $\text{BLOCKED}_\iota[(e,t)]$ to be false for e to be realized when its bell rings at time t. Then, we define $T_\iota \triangleq \mathcal{T}_\iota(\Sigma)$. Given an observed trace τ, an intervention ι and a predicate ψ, the probability of $\tau \models [\iota]\,\psi$ can now be defined according to Pearl's three-step strategy: (*abduction*) condition the distribution of Σ by the observation that $\mathcal{T}(\Sigma) = \tau$, then (*action*) alter the behavior of $\mathcal{T}$ with intervention ι and (*prediction*) consider the probability that ψ holds on $\mathcal{T}_\iota(\Sigma)$. This results in the following definition.

Definition 1 (Semantics of counterfactual statements)**.** *For* τ *an observed trace,* ι *an intervention and* ψ *a predicate on traces, the probability of the counterfactual statement "had intervention* ι *happened in trace* τ*, predicate* ψ *would have been true" is defined as:*

$$\mathbf{P}\,(\,\tau \models [\iota]\,\psi\,) \triangleq \mathbf{P}\,(\,\psi(T_\iota) \mid T = \tau\,)\,.$$

Following Definition 1, we estimate the probability of the counterfactual statement $\tau \models [\iota]\,\psi$ by sampling instances of the random variable $T_\iota \mid \{T = \tau\}$. These are called *counterfactual traces*. Intuitively, they give an account of what else trace τ could have been, had intervention ι happened.

3.3 An Example

Let us illustrate our definitions by manually sampling a counterfactual trace for the example trace τ given in (1) and the intervention ι that consists in blocking every application of rule pk: $\text{BLOCKED}_\iota[((r,\xi),t)] = (r = pk)$. For this, we must draw an instance of T_ι, conditioned on the observation $T = \tau$.

Let us assume that $T = \tau$. Then, the first event of T_ι has to coincide with the first event of τ (namely b). Indeed, suppose that (e,t) belongs to T_ι, with t prior to the time of the first event of τ. Thus, e is scheduled in Σ at time t and realizable in the initial mixture, which is shared between T_ι and T. As a consequence, (e,t) also belongs to T and therefore to τ, which is a contradiction. Continuing this line of reasoning, T_ι and τ must coincide until an event of τ is blocked by ι.

After pk is blocked in T_ι, the current mixtures in T and T_ι start diverging (the kinase is phosphorylated in the former and unphosphorylated in the latter). We call these mixtures *factual mixture* and *counterfactual mixture*, respectively. The next event to happen in τ is the second binding event b. We argue that it also has to be the next event to happen in T_ι. Indeed, the only way an event (e,t) can happen in T_ι before b while not happening in T is if e is realizable in the counterfactual mixture and not in the factual one. This is only true if e is an instance of rule pk and applications of this rule – if scheduled – would be blocked by ι anyway.

After b happens in both T_ι and τ, the event template associated with rule u (fast unbinding) becomes realizable in the counterfactual mixture, but not in the factual one. Therefore, the observation $T = \tau$ provides no useful information about whether or not u is scheduled in Σ before p happens in τ. In fact, the probability that this is not the case is exactly equal to $\exp(-\lambda_u \delta)$, where δ is the length of the time interval between b and p in τ. Given the rates in our model, δ is typically of the order of $(\lambda_{u^*} + \lambda_p)^{-1}$. Therefore, $\lambda_u \delta \gg 1$ and it is very likely that a fast unbinding event happens in T_ι before p happens in τ, preventing p to happen in T_ι. This gives us the following counterfactual trace:

$$b,\ u,\ \cancel{pk},\ b,\ \boldsymbol{u},\ \cancel{p},\ \cancel{u^*},\ \cdots \tag{2}$$

where events that are striked out are events of τ that do not appear in T_ι and events in bold are proper to T_ι.

3.4 The Counterfactual Resimulation Algorithm

We introduce Algorithm 1, a variation of the Doob-Gillespie algorithm, to sample a counterfactual trace efficiently given a reference trace τ and an intervention ι. We call it *counterfactual resimulation*, since it works by going through every event of τ, resimulating only those parts of τ that are affected by ι. In particular, when ι is the trivial intervention ($\text{BLOCKED}_\iota[\cdot] = \text{false}$), it returns τ.

This algorithm relies on a modified notion of activity we call *divergent activity*. We define the set of *divergent embeddings* of the left-hand side of a rule r into mixture m and relative to m_0 as $\text{EMB}'_r(m, m_0) \triangleq \text{EMB}_r(m) \setminus \text{EMB}_r(m_0)$. Equivalently, a divergent embedding is an embedding whose codomain features a *divergent site*, that is, a site whose state differs across m and m_0. The divergent activity of a rule r in mixture m relative to m_0 is then the product $\lambda_r|\text{EMB}'_r(m, m_0)|$. The *total divergent activity* of the system, $\alpha'(m, m_0)$, is the sum of all divergent activities. Finally, we use the notation $\tau[t]$ to refer to the mixture at time t in τ.

Algorithm 1 Counterfactual resimulation

Input: a model, a reference trace τ and an intervention ι
Output: an instance of $T_\iota \mid \{T = \tau\}$ (counterfactual trace)

1. $t \leftarrow 0$
2. $m \leftarrow$ initial mixture
3. **while** $t < t_{\text{end}}$ **do**
4. $\quad m_0 \leftarrow \tau[t]$
5. $\quad (e_\text{f}, t_\text{f}) \leftarrow$ first event of τ in time interval (t, ∞)
6. $\quad \alpha' \leftarrow \sum_r \lambda_r|\text{EMB}'_r(m, m_0)|$
7. $\quad$ draw $\delta \sim \text{EXP}(\alpha')$
8. $\quad t_\text{c} \leftarrow t + \delta$
9. $\quad$ **if** $t_\text{c} < t_\text{f}$ **then**
10. $\quad\quad$ draw a rule r with prob. $\propto \lambda_r|\text{EMB}'_r(m, m_0)|$
11. $\quad\quad$ draw a divergent embedding $\xi \in \text{EMB}'_r(m, m_0)$
12. $\quad\quad e \leftarrow (r, \xi)$
13. $\quad\quad t \leftarrow t_\text{c}$
14. $\quad$ **else**
15. $\quad\quad e \leftarrow e_\text{f}$
16. $\quad\quad t \leftarrow t_\text{f}$
17. $\quad$ **if** $\neg\, \text{BLOCKED}_\iota[(e, t)]$ and $m \vdash e$ **then**
18. $\quad\quad$ update m to $m \cdot e$ and log event (e, t)

The role and relevance of the concept of divergent activity in counterfactual resimulation is summarized by the following proposition, where $\tau \cap I = \emptyset$ is a shortcut for "no event of trace τ occurs in the time interval I".

Proposition 1 (Property of the divergent activity)**.** *For τ a trace and ι an intervention, let $I = (t, t+\delta)$ be a time interval such that $\tau \cap I = \emptyset$ and $m_0 = \tau[t]$. Then, we have*

$$\mathbf{P}\,\{\, T_\iota \cap I = \emptyset \mid T = \tau,\ T_\iota[t] = m \,\} \;=\; e^{-\alpha'(m, m_0)\cdot\delta}.$$

At every iteration of Algorithm 1, the divergent activity α' determines the probability that an event happens in the counterfactual trace prior to the next event in the factual trace τ (test of line 9). A proof of Proposition 1 is given in Appendix A. It is the main step in establishing:

Theorem 1. *The counterfactual resimulation algorithm correctly samples instances of $T_\iota \mid \{T = \tau\}$.*

3.5 Implementation

There are two challenges in efficiently implementing counterfactual resimulation. The first is a suitable representation for the sets of divergent embeddings $\text{EMB}'_r(m)$ to minimize the cost of their update at each iteration. Since the Kappa simulator solves exactly that problem for the sets of embeddings $\text{EMB}_r(m)$ [Danos *et al.*, 2007b], we leverage most of that infrastructure. The second consists in avoiding excessively many iterations of Algorithm 1 in which time is advanced in tiny increments and the proposed event is rejected. Suppose, for example, that in our toy model pk has a very high firing rate and we wish to block, from a specific time onward, *all* events in which the sole kinase becomes phosphorylated. Upon blocking one occurrence of the event, the same event would want to happen again, and we would keep rejecting it a huge number of times until a different rule fires. More generally, event templates whose realization is bound to be blocked should be removed efficiently before their realization is attempted and not be counted in the system's divergent activity. We solve this problem for a class of interventions we call *regular*. Specifically, an intervention ι is regular if the predicate $\text{BLOCKED}_\iota[((r, \xi), t)]$ can be expressed as a finite disjunction of formulae of the form $(r{=}r') \wedge F(\xi\restriction_c) \wedge (t \in I)$ or $G(r, \xi) \wedge (t{=}t')$, where r' is a rule, t' a time, I a time interval, $\xi\restriction_c$ the restriction of ξ to a single connected component c of $L_{r'}$, and F, G arbitrary predicates. For regular interventions, our implementation is guaranteed to either produce or consume an event at each iteration.

Proposition 2. *Sampling a counterfactual trace for a* regular *intervention can be done in time $\mathcal{O}(n \cdot r \log |m|)$, where n is the sum of the number of events in the reference trace and in the resulting counterfactual trace, r is the number of rules in the model and $|m|$ the size of the reaction mixture.*

For non-regular interventions, there is an additional time-complexity term of $\mathcal{O}(N_\emptyset \cdot r \log |m|)$, where $N_\emptyset$ denotes the number of non-productive iterations of Algorithm 1. In Appendix C, we provide a benchmark of our implementation on a scaled-up version of our toy model. The average slowdown per event compared to the Kappa simulator does not exceed 50% for a variety of interventions. Also, we observe that $N_\emptyset$ is typically small for the type of non-regular intervention that we expect to be most useful in practice.

Returning to our running example, sampling counterfactual traces repeatedly for trace 1 would reveal that, with very high probability, "event p would not have happened, had pk not happened". However, we can go further by using counterfactual traces to *explain* this observation using enablement and prevention arrows.

4 Counterfactuals And Prevention

The diagram shown Figure 4 extends the one in Figure 3 and explains the counterfactual dependency between pk and p in

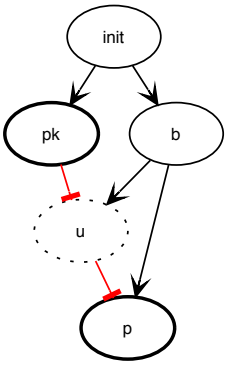

Figure 4: A graphical explanation of the counterfactual dependency between pk and p in trace 1, in terms of enablement and prevention arrows. It is based on the (compressed) counterfactual experiment (τ, ι, τ') where $\tau = (pk, b, p)$, ι blocks pk and $\tau' = (b, u)$.

trace 1. The dotted node corresponds to a *counterfactual event*, which is absent from trace 1. It is related to *factual* events by prevention arrows, shown in red. These arrows can be read as follows: "pk prevents u, which prevents p". In this section, we give a precise semantics to such diagrams and discuss how they can be generated systematically.

As discussed in section 2.2, a causal narrative like in Figure 3 results from a trace after causal compression. Its events are organized in a directed acyclic graphs whose edges are enablement arrows. While enablement is straightforward to define, prevention is trickier because it relates events that happened to events that did not. Our insight is to use counterfactual traces to define prevention as connecting events from a factual trace to events in a cognate counterfactual trace or vice versa.

Notation and diction We use the symbol e, which in previous sections referred to an event template, to directly denote an event. Moreover, when we say that an event *tests* or *modifies* a site, we really mean the tests and actions involved when matching and rewriting, respectively, the underlying rule in the mixture $\tau[t]$. For example, event p in trace 1 tests three sites and modifies one. Finally, an event e occurring at time t is said to be *executable* in trace τ if the associated template is realizable in mixture $\tau[t]$.

4.1 Prevention in Counterfactual Experiments

A *counterfactual experiment* is a triple (τ, ι, τ') for which there exists a schedule σ such that $\tau = \mathcal{T}(\sigma)$ and $\tau' = \mathcal{T}_\iota(\sigma)$. Such triples are produced by counterfactual resimulation.

To formalize enablement and prevention, we need to define some terms used in Kappa to talk about events. Without loss of generality, we shall assume that a site on an agent carries either binding state or tagging state but not both. The *value* of a tagged site in a mixture is its current tag and the value of a binding site is either FREE or BOUND-TO(s), where s identifies another site in the mixture.

Definition 2 (Enablement)**.** *Let τ be a trace and $e, e' \in \tau$ two events. We say that e enables e' if e is the last event before e' that modifies some site to the value it is tested for by e'.*

Definition 3 (Prevention)**.** *Let (τ, ι, τ') be a counterfactual experiment. An event e that occurs at time t in τ is said to prevent an event e' that occurs at time t' in τ' if all of the following hold: (1) $t < t'$; (2) there exists a site s such that e is the last event in τ before t' that modifies the value of s away from what e' tests it for ; (3) there are no events in τ' that modify s during the time interval (t, t'). The same definition holds switching τ and τ'.*

Counterfactual experiments can be represented as directed acyclic graphs like the one in Figure 4. Such a graph features three kinds of nodes: events that are proper to the factual trace (thick solid nodes), events that are proper to the counterfactual trace (dotted nodes) and events that are common to both traces (thin solid nodes).

As illustrated Figure 4, the influence of pk on p in our example is mediated by the counterfactual event u. Such mediating events always exist, as stated by the following theorem.

Theorem 2 (Completeness of enablement and prevention)**.** *Let (τ, ι, τ') be a counterfactual experiment and e an event that belongs only to τ. Then, there exists an event $\hat{e} \in \tau$ that is blocked by ι and such that there is a directed path from $\hat{e}$ to e with an even number of prevention arrows.*

This theorem states that counterfactual dependencies can always be explained in terms of enablement and prevention relations between individual events. A proof is in Appendix B. This result establishes a bridge between two different visions of causality: the vision dominant in the concurrency community, in which causality is defined predominantly in terms of enablement, in opposition to concurrency [Winskel, 1986], and the vision based on counterfactuals, which is dominant in the causal inference community [Pearl, 2009].

4.2 Compression of Counterfactual Experiments

Counterfactual experiments produced with counterfactual resimulation are usually very large. They typically feature a lot of redundancy, including events that are irrelevant to the outcome of interest or futile cycles as discussed in section 2.2. A compression step is usually necessary before concise causal narratives can be extracted from such experiments. However, the two traces of a counterfactual experiment cannot be compressed separately following the procedure described in section 2.2, as there is no guarantee that the compressed traces can still be generated from a unique schedule to form a valid counterfactual experiment. Instead, compressing a counterfactual experiment consists in extracting a minimal *valid* sub-experiment.

A counterfactual experiment (τ_1, ι, τ_1') is said to be a *sub-experiment* of (τ_2, ι, τ_2') if τ_1 is a sub-trace of τ_2 and τ_1' is a sub-trace of τ_2'. Also, valid counterfactual experiments can be characterized as follows.

Proposition 3. *A triple (τ, ι, τ') is a valid counterfactual experiment if and only if all of the following hold: (1) both τ and τ' are valid traces ; (2) no event of τ' is blocked by ι ; (3) for every event $e \in \tau$ such that $e \notin \tau'$, then either e is not executable in τ' or e is blocked by ι ; (4) for every event $e' \in \tau'$ such that $e' \notin \tau$, then e' is not executable in τ.*

Compressing a counterfactual experiment consists in finding a minimal valid sub-experiment such that (i) the outcome

of interest appears in the factual trace but not in the counterfactual trace and (ii) events that are blocked by ι, or on which the outcome of interest was shown to be counterfactually dependent in previous analyses, are kept in the factual trace. Because these constraints along with the properties featured in Proposition 3 can be encoded as boolean satisfiability constraints, compressing a counterfactual experiment can be done using standard SAT-solving techniques.

Conclusions and Future Work

In this paper, we propose a method for giving meaning to counterfactual statements in mechanistic models of complex physical processes. We modify the continuous-time Monte Carlo algorithm that is used to generate traces in these models so as to sample counterfactual trajectories that stay probabilistically as close to the original (factual) trajectory as an intervention permits them to be. We then construct causal diagrams that explain counterfactual dependencies in terms of enablement and prevention relations between events. Enablement is a standard causal relation between events within factual or within counterfactual traces, whereas prevention shows up as a relation between pairs of events, one in the factual and the other in the counterfactual trace. In particular, prevention can involve events that were not observed in the factual trace. This results in explanatory diagrams that resonate more with the ubiquitous presence of inhibitory interactions in biology and that can capture subtle kinetic aspects of rule-based models. Our completeness result, according to which any counterfactual dependency can be "explained" in terms of enablement and prevention, increases our confidence in this approach to counterfactuals and connects two visions of causality that are typically treated disjointly.

Despite a sound theoretical foundation and an effective implementation, our technique is in need of rigorous practical assessment using large-scale models. Moreover, difficult practical questions remain. Most notably, which counterfactual experiments are worth trying? It is unclear a priori which interventions are informative for traces that include many millions of events. At present, we can only offer tentative directions for future study.

One way to identify interventions worth making without relying on expert knowledge is by developing heuristics, such as recognizing correlations between events in samples of factual traces (or in a single long trace). In our toy model, the occurrence of p is often preceded by the occurrence of pk. This correlation, together with the absence of pk from some initial causal account—such as the one presently achievable in a fully automated fashion based on enablement alone, (Figure 3)—suggests to try a counterfactual experiment on pk. More generally, if (i) a context $\mathcal{C}$ in which an event e occurs is frequently more specific than is required by the left-hand side of the underlying rule and (ii) this observation cannot be explained by the current causal narrative, then a counterfactual experiment in which we block the last event responsible for at least part of $\mathcal{C}$ seems worthwhile in order to assess whether the current causal narrative needs to be updated. Another important practical question is whether interventions should block single events or "knock out" related event templates for a defined timeframe. Our framework accomodates both approaches, as exemplified in Appendix C.

On a more conceptual side, we are investigating principled ways of "gluing" together all explanatory accounts (such as Figure 4) that correspond to different counterfactual experiments. This would summarize the causal structure of a system relative to an outcome of interest. One wonders whether such a summary diagram might constitute a basis for obtaining approximate structural equations for complex mechanistic models. Replacing a rule-based model with such equations could enable targeted statistical analysis to estimate model parameters, for which simulations would be too expensive.

Acknowledgments

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the U. S. Army Research Office under grant numbers W911NF-14-1-0367 and W911NF-17-1-0073. We are grateful to Pierre Boutillier for his help in implementing counterfactual resimulation. We thank Jérôme Feret, Matt Fredrikson, Mickaël Laurent and Jean Krivine for valuable discussions and insights. We thank Thomas Kolokotrones, Ariel Procaccia and the anonymous reviewers for valuable feedback.

A Proof of Proposition 1

Proposition (Property of the divergent activity)**.** *For τ a trace and ι an intervention, let $I = (t, t+\delta)$ be a time interval such that $\tau \cap I = \emptyset$ and $m_0 = \tau[t]$. Then, we have*

$$\mathbf{P}\left\{ T_\iota \cap I = \emptyset \mid T = \tau,\, T_\iota[t] = m \right\} = e^{-\alpha'(m,m_0)\cdot\delta}.$$

Proof. Given that $T_\iota[t] = m$, no counterfactual event happens in time interval I if and only if no event template that is realizable in mixture m is scheduled in I. Therefore,

$$\begin{aligned} &\mathbf{P}\left\{ T_\iota \cap I = \emptyset \mid T = \tau,\, T_\iota[t] = m \right\} \\ &= \quad \mathbf{P}\left\{ \bigwedge_{m \vdash e} \left(e \notin \Sigma \cap I \right) \mid T = \tau \right\} \\ &= \quad \prod_{m \vdash e} \mathbf{P}\left\{ e \notin \Sigma \cap I \mid T = \tau \right\}. \end{aligned} \tag{3}$$

Let e an event template such that $m \vdash e$. The probability that e has not been scheduled for tentative realization in I given that $T = \tau$ depends on whether or not e is realizable in mixture $m_0 = \tau[t]$. Indeed, we assumed that τ contains no event in time interval I. Therefore, if $m_0 \vdash e$, then e cannot be scheduled in I, without which it would have been realized in τ. Thus,

$$m_0 \vdash e \implies \mathbf{P}\left\{ e \notin \Sigma \cap I \mid T = \tau \right\} = 1. \tag{4}$$

In addition, if $m_0 \nvdash e$, then the observation $\{T = \tau\}$ gives no information on whether or not e has been scheduled in I. Because event templates are scheduled for tentative realization according to Poisson processes, we have:

$$m_0 \nvdash e \implies \mathbf{P}\left\{ e \notin \Sigma \cap I \mid T = \tau \right\} = e^{-\lambda_e \delta} \tag{5}$$

where λ_e is the rate of the rule associated with event template e. Combining (4) and (5) with equation (3),

$$\mathbf{P}\{T_\iota \cap I = \emptyset \mid T = \tau,\, T_\iota[t] = m\} = \exp\Big(-\sum_{m \vdash e,\, m_0 \nvdash e} \lambda_e \cdot \delta\Big)$$

Finally, we can recognize the divergent activity in the exponential above:

$$\begin{aligned}\sum_{m \vdash e,\, m_0 \nvdash e} \lambda_e &= \sum_{(r,\xi)} \lambda_r \mathbf{1}\{m \vdash (r,\xi),\ m_o \nvdash (r,\xi)\} \\ &= \sum_r \lambda_r \sum_\xi \mathbf{1}\{m \vdash (r,\xi),\ , m_o \nvdash (r,\xi)\} \\ &= \sum_r \lambda_r |\textsc{Emb}'_r(m, m_0)| \ = \ \alpha'(m, m_0).\end{aligned}$$

This concludes the proof. □

B Proof of Theorem 2

It is convenient to prove the following result instead, which is slightly more general.

Theorem. *Let (τ, ι, τ') be a counterfactual experiment. If e is an event that belongs only to τ **or** τ', then there exists an event $\hat{e}$ in τ that is blocked by ι and there is a directed path from $\hat{e}$ to e.*

Proof. We prove this theorem by induction on the number of events before e in both τ and τ'. Let's consider $e \in \tau$ such that $e \notin \tau'$. If $\textsc{blocked}_\iota[e]$ is true, then we are done. Otherwise, by item (3) of Proposition 3, e is not executable in τ'. Therefore, there exists a site s such that event e tests s to a different value than it has in $\tau'[t]$. Let t the time of occurrence of e. Moreover, let's define e_0 the last event in τ modifying s that occurs strictly before time t, and e'_0 the last event in τ' modifying s that occurs strictly before time t. These events modify s to different values so they cannot be the same. Therefore, writing t_0 and t'_0 their respective time of occurrence, we have $t_0 \neq t'_0$ with probability one.

- If $t_0 < t'_0$, then we have $e_0 \notin \tau'$ and so we can apply the induction hypothesis on e_0. As a consequence, there exists a path from an event $\hat{e}$ in τ which is blocked by ι to e_0. Moreover, e_0 enables e. Therefore, there is a path from $\hat{e}$ to e.
- If $t'_0 < t_0$, then we have $e'_0 \notin \tau$ and so we can apply the induction hypothesis on e'_0. As a consequence, there exists a path from an event $\hat{e}$ in τ which is blocked by ι to e'_0. Moreover, e'_0 prevents e. Therefore, there is a path from $\hat{e}$ to e.

The same proof holds for when $e \in \tau'$ and $e \notin \tau$, using item (4) of Proposition 3 instead of item (3). □

This result implies Theorem 2. Indeed, we only need to prove that the constructed path from $\hat{e}$ to e contains an even number of prevention arrows, which is true for *any* such path as prevention arrows always go from τ to τ' or the other way around but cannot "stay" within a single trace.

C Benchmark

We assessed the performance of our implementation of counterfactual resimulation using the toy model of Figure 2, but with an initial mixture consisting of a large number of kinase and substrate instances. Although such a simple model is not of biological significance, it is adequate for an initial assessment of performance.

C.1 Experimental protocol

We consider four kinds of intervention defined in terms of an event $e_0 = ((r_0, \xi_0), t_0)$:

1. *Singular block:* This blocks only the specific event e_0:
$$\textsc{blocked}_\iota[e] \triangleq (e = e_0).$$
2. *Template block:* This blocks every realization of the event template (r_0, ξ_0) from time t_0 onward. In formal terms:
$$\textsc{blocked}_\iota[((r,\xi),t)] \triangleq ((r,\xi) = (r_0,\xi_0) \ \wedge\ t \geq t_0).$$
3. *Agent-dependent rule block:* This blocks, from t_0 onward, every event resulting from rule r_0 that tests an agent modified by e_0. For example, we might wish to prevent substrate S_{627} from being phosphorylated by any kinase. In formal terms:
$$\textsc{blocked}_\iota[((r,\xi),t)] \triangleq (r = r_0 \wedge \xi(L_r) \cap M_0 \neq \emptyset \wedge t \geq t_0)$$
where M_0 is the set of agents modified by e_0.
4. *Rule block:* This outright disables rule r_0 from t_0 onward:
$$\textsc{blocked}_\iota[((r,\xi),t)] \triangleq (r = r_0 \ \wedge\ t \geq t_0).$$

All four kinds of intervention might be useful for causal analysis in different settings.

Our experimental setup consists of the model in Figure 2 comprising 10^4 substrates and 10^4 kinases. Every agent starts out in an unbound and unphosphorylated state. The simulation is stopped as soon as the system contains more phosphorylated than unphosphorylated substrates. We then proceed as follows. (i) We first generate $n_\tau = 10$ reference traces and record the CPU time T it took the Kappa simulator to generate each of them on a personal computer with a 2.7GHz Intel Core i5 processor and 16GB of random-access memory. For each trace, we also identify the first application of each $r \in \{b, u, u^*, p, pk\}$ and declare it to be the event e_0 underlying the intervention. (ii) We test the 20 possible interventions, based on e_0, that can be formed by combining each of the five rules r with each of the four intervention types. (iii) For each such intervention ι, we generate $n_{\tau',\iota} = 10$ counterfactual traces τ'. (iv) For every counterfactual trace τ', we record the CPU time T' used by our implementation to generate it. We also record the number $N_\emptyset$ of iterations that neither produced a counterfactual event nor consumed a factual event (non-productive cycles). Finally, we define the *slowdown* S of counterfactual resimulation relative to simulation as the ratio of T', normalized by the number of distinct events $|\tau \cup \tau'|$ in the counterfactual experiment (τ, ι, τ'), to T, normalized by the number $|\tau|$ of events in τ:

$$S \triangleq \frac{|\tau|}{|\tau \cup \tau'|} \cdot \frac{T'}{T}.$$

		T' (s)	S	$\lvert\tau\backslash\tau'\rvert$	$\lvert\tau'\backslash\tau\rvert$	$N_\emptyset$
Singular	b	4.70	$1.17 \pm .03$	2.0	0	0
	u	4.70	$1.17 \pm .03$	1.0	1.0	0
	$u*$	4.70	$1.17 \pm .04$	8.3	5.1	0
	p	4.70	$1.17 \pm .03$	1.0	0.5	0
	pk	4.68	$1.17 \pm .03$	8.0	17.6	0
Template	b	4.69	$1.17 \pm .04$	2.0	0	0
	u	5.52	$1.38 \pm .05$	30.3	2.9	0
	$u*$	5.43	$1.35 \pm .04$	30.3	6.9	0
	p	5.50	$1.37 \pm .04$	1.0	0.5	0
	pk	5.41	$1.35 \pm .04$	8.0	17.6	0
Agent	b	5.62	$1.40 \pm .04$	29.6	0.9	0
	u	5.55	$1.38 \pm .04$	30.3	2.9	0
	$u*$	5.45	$1.36 \pm .04$	30.3	6.9	0
	p	5.52	$1.38 \pm .04$	1.0	0	0
	pk	5.42	$1.35 \pm .04$	8.0	17.6	0
Rule	b	3.78	$.94 \pm .03$	1.4e5	0	0
	u	4.43	$1.02 \pm .03$	1.3e5	1.2e4	0
	$u*$	4.82	$1.16 \pm .03$	2.5e4	5.3e3	0
	p	4.82	$1.20 \pm .03$	5.0e3	0	0
	pk	5.19	$1.08 \pm .03$	1.4e4	3.0e4	0

Table 1: A benchmark of counterfactual resimulation. On average, $T = 4.01 \pm .12$ s. In addition, $|\tau| = 1.6\text{e}5 \pm 1.5\text{e}3$.

The average value and standard deviation of these quantities is shown Table 1. Note that each row of the table corresponds to one intervention ι and to a sample set of $n_\tau \times n_{\tau',\iota} = 100$ counterfactual experiments. For every intervention, we also report a measure of how much counterfactual traces differ from their cognate factual trace on average: given a counterfactual experiment (τ, ι, τ'), we write $|\tau \setminus \tau'|$ for the number of events that are proper to τ and $|\tau' \setminus \tau|$ the number of events that are proper to τ' (also called counterfactual events).

C.2 Results

The observed slowdown S never exceeds 50% on average. No intervention produced a non-productive cycle. This is not too surprising, as all the interventions we considered are regular, with the only exception of the "*template block*" for rule b. Although this intervention can produce non-productive cycles in theory, it is highly unlikely for a kinase to bind the same substrate twice in a large mixture. More generally, an intervention ι that is irregular, because $\text{BLOCKED}_\iota[((r, \xi), t)]$ features a conjunction of terms constraining ξ on different connected components of L_r, does not tend to induce many non-productive cycles for a similar reason and, therefore, can often be handled efficiently anyway.

As expected, we observe that the "stronger" the intervention, the bigger the divergence of counterfactual traces from their factual reference trace. Moreover, interventions that only affect a small number of agents in a large mixture do not cause major divergences at the population level. In fact, only the five rule-blocking interventions had a major impact.

References

[Andersen *et al.*, 2016] Jakob L. Andersen, Christoph Flamm, Daniel Merkle, and Peter F. Stadler. A Software Package for Chemically Inspired Graph Transformation. *Lecture Notes in Computer Science*, 9761:73–88, 2016.

[Boutillier *et al.*, 2017] Pierre Boutillier, Thomas Ehrhard, and Jean Krivine. Incremental update for graph rewriting. In *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017*, 2017.

[Danos *et al.*, 2007a] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-Based Modelling of Cellular Signalling, invited paper. In *Proceedings of the Eighteenth International Conference on Concurrency Theory, CONCUR 2007, Lisbon, Portugal*, Lecture Notes in Computer Science, Lisbon, Portugal, 2007.

[Danos *et al.*, 2007b] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable Simulation of Cellular Signaling Networks, invited paper. In *Proceedings of the Fifth Asian Symposium on Programming Systems, APLAS 2007, Singapore*, Singapore, 2007.

[Danos *et al.*, 2012] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, Jonathan Hayman, Jean Krivine, Christopher D Thompson-Walsh, and Glynn Winskel. Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012*, volume 18, pages 276–288, 2012.

[Gillespie, 1977] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[Halpern and Pearl, 2005] JY Halpern and J Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *The British journal for the philosophy of*, 2005.

[Halpern, 2016] Joseph Y Halpern. *Actual causality*. MIT Press, 2016.

[Harris *et al.*, 2016] Leonard A. Harris, Justin S. Hogg, José-Juan Tapia, John A. P. Sekar, Sanjana Gupta, Ilya Korsunsky, Arshi Arora, Dipak Barua, Robert P. Sheehan, and James R. Faeder. BioNetGen 2.2: Advances in rule-based modeling. *Bioinformatics*, 32(21):3366—-3368, 2016.

[Lewis, 1974] David Lewis. Causation. *The journal of philosophy*, 70(17):556–567, 1974.

[Pearl, 2009] Judea Pearl. *Causality*. Cambridge university press, 2009.

[Winskel, 1986] Glynn Winskel. Event structures. In *advanced course on Petri nets*, pages 325–392. Springer, 1986.