arXiv:2010.12237v1 [cs.AI] 23 Oct 2020

Algorithms for Causal Reasoning in Probability Trees

Tim Genewein[*] **Tom McGrath**[*] **Grégoire Delétang**[*] **Vladimir Mikulik**[*]
Miljan Martic, Shane Legg, Pedro A. Ortega[†]
DeepMind
London, UK

Abstract

Probability trees are one of the simplest models of causal generative processes. They possess clean semantics and—unlike causal Bayesian networks—they can represent context-specific causal dependencies, which are necessary for e.g. causal induction. Yet, they have received little attention from the AI and ML community. Here we present concrete algorithms for causal reasoning in discrete probability trees that cover the entire causal hierarchy (association, intervention, and counterfactuals), and operate on arbitrary propositional and causal events. Our work expands the domain of causal reasoning to a very general class of discrete stochastic processes.

1 Introduction

The formal treatment of causality is one of the major developments in AI and ML during the last two decades [1–5]. Causal reasoning techniques have found their way into many ML applications [6], and more recently also into RL [7–11], fairness [12–14], and AI safety [15, 16], to mention some. This broad adoption rests on the greater explanatory power delivered by causal models than by those based on statistical dependence alone [5, 17].

Causal dependencies can be described using the language of *causal Bayesian networks* (CBNs) and *structural causal models*, which elegantly tie together graphical properties with conditional independence relations and effects of interventions [1, 2]. While very versatile, CBNs have well-known limitations: the collection of conditional independence properties they can express are very specific [18], requiring the causal relations among random variables to form a partial order.

Here we work with an alternative representation: *discrete probability trees*, sometimes also called *staged tree models* [19–22]. A probability tree is one of the simplest models for representing the causal generative process of a random experiment or stochastic process (Figure 1). The semantics are self-explanatory: each node in the tree corresponds to a potential state of the process, and the arrows indicate both the probabilistic transitions and the causal dependencies between them. Unlike CBNs, probability trees can model context-specific causal dependencies (see e.g. Figure 1c). However, probability trees do not explicitly represent conditional independencies, and thus, when a distribution and its causal relations admit a representation both as a probability tree and a CBN, the latter is more compact.

In spite of their simplicity and expressiveness, probability trees have not enjoyed the same attention as CBNs and structural causal models in the statistical and machine learning literature. In this work, we attempt to remedy this through the following contributions. Focusing on finite probability trees, our work is the first to provide concrete algorithms for (a) computing minimal representations of *arbitrary events* formed through propositional calculus and causal precedences; and (b) computing the *three fundamental operations* of the causal hierarchy [2], namely conditions, interventions, and

[*]Equal contribution
[†]Correspondence to {timgen|mcgrathtom|gdelt|vmikulik|pedroortega}@google.com

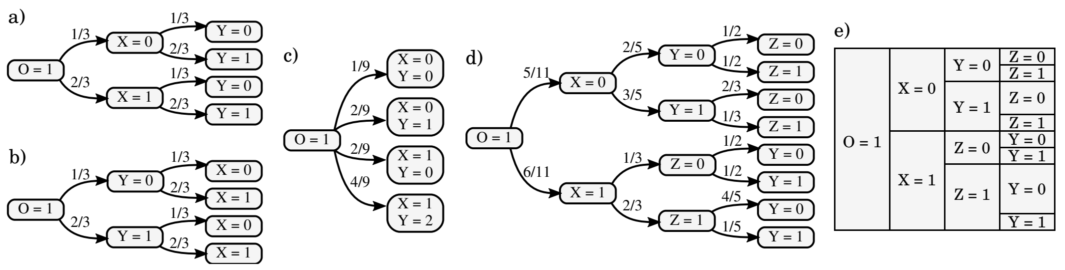

Figure 1: Probability trees. Panels (a), (b), and (c) show the same joint distribution over X and Y. They differ in that (a) assumes $X \to Y$, (b) assumes $Y \to X$, whereas (c) does not assume a causal dependency ($X \not\to Y \wedge Y \not\to X$). Panel (d) will be our running example throughout the paper. It is a probability tree where $Y \to Z$ when $X = 0$ and $Z \to Y$ when $X = 1$. Such conditional causal dependencies cannot be expressed using a CBN. Panel (e) shows a probability tree mass diagram, which is an alternative representation of the probability tree (d) that emphasizes the probability mass of events (encoded as the height of a box). By convention we bind $O = 1$ (as in omega "Ω" for a sample space) at the root node.

counterfactuals. We also provide an interactive tutorial available online[3], containing many concrete examples.

Formal definition. While probability trees can be abstractly axiomatized or described as colored graphs [19, 23, 21], here we adopt a recursive definition that is closer to a computational implementation. We define a *node* $n \in \mathcal{N}$ in the tree as a tuple $n = (u, \mathcal{S}, \mathcal{C})$ where: $u \in \mathbb{N}$ is a unique numerical identifier for the node within the tree; $\mathcal{S}$ is a list of statements such as '$X = 0$' and '$W =$ rainy'; and $\mathcal{C}$ is a (possibly-empty) ordered set of transitions $(p, m) \in [0, 1] \times \mathcal{N}$ where p is the transition probability to the child node m. We will represent statements such as '$X = 1$' as a tuple $(X, 1) \in \mathcal{X} \times \mathcal{V}_X$ where $\mathcal{X}$ is the set of variables and $\mathcal{V}_X$ is the range of the variable X. Obviously, the transition probabilities must sum up to one. The root is the unique node with no parents, and a leaf is a node with an empty set of transitions. A *(total) realization* in the probability tree is a path from the root to a leaf, and its probability is obtained by multiplying the transition probabilities along the path; and a *partial realization* is any connected sub-path within a total realization. When entering a node, the process binds the listed random variables to definite values.

2 Events

We have seen how to represent the realizations and causal dependencies of a random experiment using a probability tree. Here we show how to represent and calculate events.

An *event* is a collection of total realizations. We can describe events using using propositions about random variables (e.g. '$X = 0$', '$Y = 1$'). For instance, the event '$X = 0$' is the set of all total realizations that traverse a node with the statement '$X = 0$'. Furthermore, we can use logical connectives of negation (NOT, $\neg$), conjunction (AND, $\wedge$), and disjunction (OR, $\vee$) to state composite events, such as '$\neg(X = 0 \wedge Y = 1)$'. In addition, we can also use precedence (PREC, $\prec$) for describing events that meet a causal condition.

2.1 Min-cuts

We can represent events using *cuts*—a collection of nodes with probabilities summing up to one that are mutually exclusive [19]. In particular, we will focus our attention on *min-cuts* (minimal cuts). A min-cut is a minimal representation of an event in terms of the nodes of a probability tree. The min-cut of an event collects the smallest number of nodes in the probability tree that resolves whether an event has occurred or not. In other words, if a realization hits a node in the min-cut, then we know for sure whether the event has occurred[4].

[3] `https://github.com/deepmind/deepmind-research/causal-reasoning`

[4] In measure theory, a similar notion to a min-cut would be the smallest algebra that renders an event measurable.

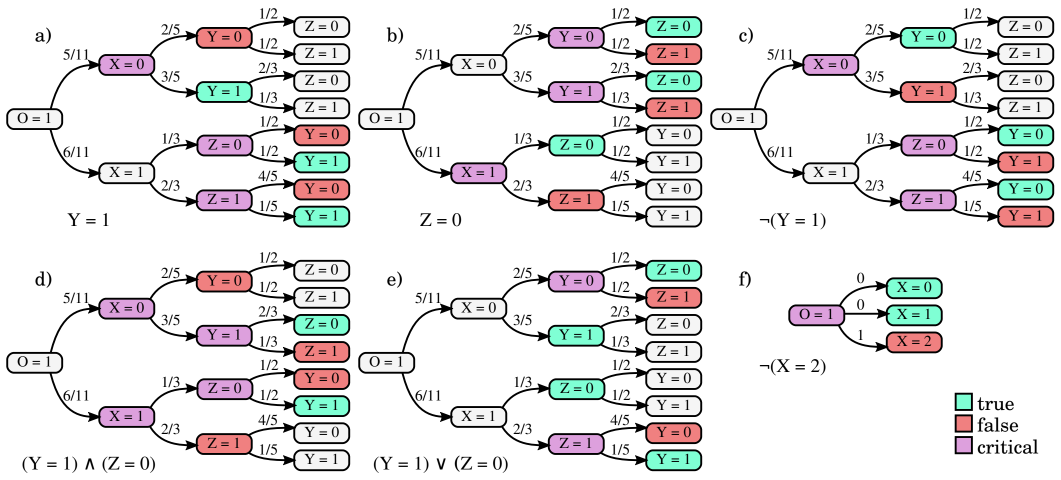

Figure 2: Min-cuts and critical sets. Panels (a)–(e) are min-cuts (red and green) and critical sets (purple) for different events within the probability tree of Figure 1d. They correspond to: (a) $Y = 1$; (b) $Z = 0$; (c) the negation $\neg(Y = 1)$; (d) the conjunction $Y = 1 \wedge Z = 0$; and (e) the disjunction $Y = 1 \vee Z = 0$. Negations swap true and false sets; conjunctions pick, for each realization, the first and last occurrence of a false and true node respectively; and disjunctions invert the selection order of conjunctions. Panel (f) shows the min-cut for the event $\neg(X = 2)$. Notice that the true set contains both $X = 0$ and $X = 1$, which can never occur probabilistically. This example illustrates that min-cuts are determined by the logical truth and not by the probabilistic truth of events.

Our custom notion of min-cuts furthermore distinguishes between the nodes that render the event true from the nodes that render the event false. More precisely, we describe an event using a cut $\delta = (\mathcal{T}, \mathcal{F})$, where the true set $\mathcal{T}$ and the false set $\mathcal{F}$ contain all the nodes where the event becomes true or false respectively. Figure 2 shows examples of min-cuts for *simple events* (defined here as events that bind a single random variable to a constant), negation, conjunction, and disjunction.

The (mostly recursive) pseudo-algorithms for computing the min-cuts are listed in Figure 3. They compute the min-cuts for simple events (PROP), negations (NEG), conjunctions (AND), and disjunctions (OR), returning a pair $\delta = (\mathcal{T}, \mathcal{F})$ of a true and a false set of unique node identifiers. These algorithms assume that every simple event (e.g. $X = 1$) is *well-formed*, that is, every total realization must contain one and only one node within its path where the simple event is either true ($X = 1$) or false ($X \neq 1$). Or, in other words, every random variable must be bound to a unique value by the time the stochastic process terminates at a leaf. If simple events are well-defined, then so are their compositions through the Boolean operators.

It is worth pointing out that we distinguish between probabilistic and logical truth. A min-cut for a given event is determined solely through the skeleton of the probability tree without taking into account the transition probabilities (see the example in Figure 2f).

Once a min-cut for an event has been determined, then the probability of said event is given by the sum of the probability of the true nodes. Probability distributions, expectations, etc. are then determined from these probabilities in the obvious manner.

2.2 Mechanisms and critical sets

For every event, we can also identify the nodes where the very next transition determines whether a given event will not occur. These are the *critical nodes*. Formally, given the min-cut for an event, we define the *critical set* as the set of all nodes that are parents of a node in the false set. They are highlighted in purple in Figure 2. The critical set can be thought of as the collection of mechanisms in the tree that we need to manipulate in order to bring about a given event. Critical nodes are the *Markov blankets*: all the variables bound within a path from the root to the critical node constitute the "exogenous variables" for the mechanisms downstream. They are (implicitly) used in defining the three operations of the causal hierarchy [2] presented later.

```
function PROP($n, s$)
    ▷ $n$ : Node, $s$ : Statement
    Let $n = (u, \mathcal{S}, \mathcal{C})$ and $s = (X_s, V_s)$
    ▷ Base case, find $s$ within node:
    for all $(X, V) \in \mathcal{S}$ do
        if $X_s = X$ then
            if $V_s = X$ then
                return $(\{u\}, \varnothing)$
            else return $(\varnothing, \{u\})$
    if $\mathcal{C} = \varnothing$ then
        error: $s$ cannot be resolved.
    ▷ Statement not found, recurse:
    $\delta \leftarrow (\varnothing, \varnothing)$
    for all $c = (p_c, n_c) \in \mathcal{C}$ do
        $\delta_c \leftarrow$ PROP$(n_c, s)$
        Let $\delta_c = (\mathcal{T}_c, \mathcal{F}_c)$ and $\delta = (\mathcal{T}, \mathcal{F})$
        $\delta \leftarrow (\mathcal{T} \cup \mathcal{T}_c, \mathcal{F} \cup \mathcal{F}_c)$
    ▷ Consolidate:
    if $\mathcal{F} = \varnothing$ then $\delta \leftarrow (\{u\}, \varnothing)$
    else if $\mathcal{T} = \varnothing$ then $\delta \leftarrow (\varnothing, \{u\})$
    return $\delta$

function NEG($\delta$: Min-Cut)
    ▷ Switch true and false:
    Let $\delta = (\mathcal{T}, \mathcal{F})$
    return $(\mathcal{F}, \mathcal{T})$
```

```
function AND($n, \delta_1, \delta_2$)
    ▷ $n$: Node; $\delta_1, \delta_2$: Min-Cut
    return AND-R$(n, \delta_1, \delta_2, \bot, \bot)$
function AND-R($n, \delta_1, \delta_2, e_1, e_2$)
    ▷ $n$: Node; $\delta_1, \delta_2$: Min-Cut;
      $e_1, e_2$: Boolean
    Let $n = (u, \mathcal{S}, \mathcal{C})$
    Let $\delta_1 = (\mathcal{T}_1, \mathcal{F}_1)$ and $\delta_2 = (\mathcal{T}_2, \mathcal{F}_2)$
    ▷ Base case:
    if $u \in \mathcal{F}_1 \cup \mathcal{F}_2$ then return $(\varnothing, \{u\})$
    if $u \in \mathcal{T}_1$ then $e_1 \leftarrow \top$
    if $u \in \mathcal{T}_2$ then $e_2 \leftarrow \top$
    if $e_1 \wedge e_2$ then return $(\{u\}, \varnothing)$
    ▷ Recurse:
    $\delta \leftarrow (\varnothing, \varnothing)$
    for all $(p_c, n_c) \in \mathcal{C}$ do
        Let $\delta = (\mathcal{T}, \mathcal{F})$
        $(\mathcal{T}_c, \mathcal{F}_c) \leftarrow$ AND-R$(n_c, \delta_1, \delta_2, e_1, e_2)$
        $\delta \leftarrow (\mathcal{T} \cup \mathcal{T}_c, \mathcal{F} \cup \mathcal{F}_c)$
    ▷ Consolidate:
    if $\mathcal{F} = \varnothing$ then $\delta \leftarrow (\{u\}, \varnothing)$
    else if $\mathcal{T} = \varnothing$ then $\delta \leftarrow (\varnothing, \{u\})$
    return $\delta$

function OR($n$: Node, $\delta_1, \delta_2$: Min-Cut)
    ▷ Use De Morgan's rule:
    return NEG(AND($n$, NEG($\delta_1$), NEG($\delta_2$)))
```

Figure 3: Algorithms for determining min-cuts. All functions calls are *pass-by-value* (that is, deep copies of their arguments). The recursive function PROP(n, s) takes a root node n and a single statement $s = (X_s, V_s)$ and returns the min-cut δ for the event. NEG(δ) computes the min-cut for the negation of δ. AND(n, δ_1, δ_2) computes the conjunction of the min-cuts δ_1 and δ_2 for a probability tree with root n using the auxiliary function AND-R. Finally, OR computes disjunctions.

2.3 Min-cuts for causal events

A statement such as "the event where $Y = 1$ precedes $Z = 0$", written $Y = 1 \prec Z = 0$, cannot be stated logically. Rather, it is a *causal statement* that requires the causal relations provided in the probability tree. Figure 5 illustrates a min-cut for a precedence event. This relation can be combined arbitrarily with the logical connectives to form composite events. Precedences can be computed recursively as shown in the pseudo-code in Figure 4. The function PREC(n, δ_c, δ_e) takes a root n and two min-cuts δ_c and δ_e, one for the cause and the effect respectively, and returns the min-cut for the event where the precedence relation holds.

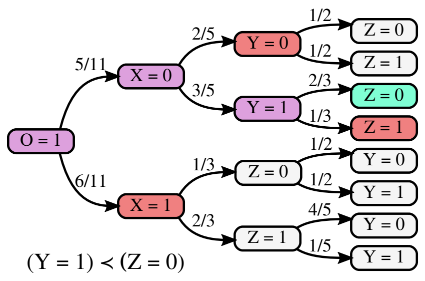

Figure 5: Min-cut for a causal event

```
function PREC($n, \delta_c, \delta_f$)
    ▷ $n$: Node; $\delta_c, \delta_e$: Min-Cut
    return PREC-R$(n, \delta_c, \delta_e, \bot)$
function PREC-R($n, \delta_c, \delta_e, f$)
    ▷ $n$: Node; $\delta_c, \delta_e$: Min-Cut;
      $f$: Boolean
    Let $n = (u, \mathcal{S}, \mathcal{C})$
    Let $\delta_c = (\mathcal{T}_c, \mathcal{F}_c)$ and $\delta_e = (\mathcal{T}_e, \mathcal{F}_e)$
    ▷ Base case:
    if $f = \bot$ then
        if $u \in \mathcal{T}_e \cup \mathcal{F}_e \cup \mathcal{F}_c$ then
            return $(\varnothing, \{u\})$
        if $u \in \mathcal{T}_c$ then $f \leftarrow \top$
    else if $f = \top$ then
        if $u \in \mathcal{T}_c$ then return $(\{u\}, \varnothing)$
        if $u \in \mathcal{F}_c$ then return $(\varnothing, \{u\})$
    ▷ Recurse:
    $\delta \leftarrow (\varnothing, \varnothing)$
    for all $(p_c, n_c) \in \mathcal{C}$ do
        Let $\delta = (\mathcal{T}, \mathcal{F})$
        $(\mathcal{T}_c, \mathcal{F}_c) \leftarrow$ PREC-R$(n_c, \delta_c, \delta_e, f)$
        $\delta \leftarrow (\mathcal{T} \cup \mathcal{T}_c, \mathcal{F} \cup \mathcal{F}_c)$
    ▷ Consolidate:
    if $\mathcal{F} = \varnothing$ then $\delta \leftarrow (\{u\}, \varnothing)$
    else if $\mathcal{T} = \varnothing$ then $\delta \leftarrow (\varnothing, \{u\})$
    return $\delta$
```

Figure 4: Precedence min-cut algorithm

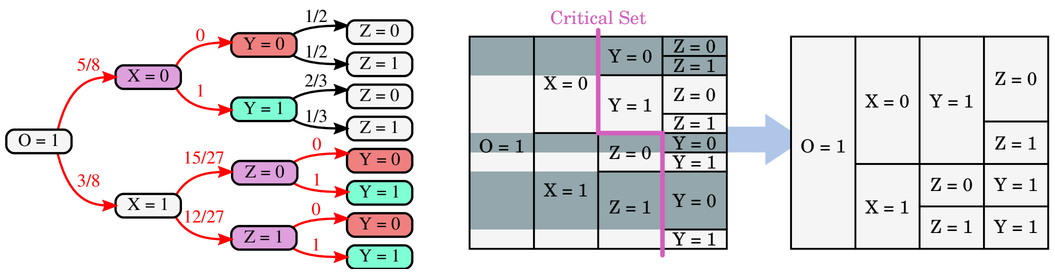

Figure 6: Seeing $Y = 1$. Conditioning on an event proceeds in two steps: first, we remove the probability mass of the realizations passing through the false set of the event's min-cut; then we renormalize the probabilities. We can do this recursively by aggregating the original probabilities of the true set. The left panel shows the result of conditioning the probability tree in Figure 1d on the event $Y = 1$, which also highlights the modified transition probabilities in red. The right panel shows the same operation in a probability mass diagram.

3 Conditions

In a probability tree, conditioning is the act of updating the transition probabilities after an event is revealed to be true. This operation allows answering questions of the form

$$P(A \mid B)$$

that is, "what is the probability of the event A given that the event B is true?" Here, A could e.g. be an event occurring downstream (prediction) or upstream (inference) of the event B. The operation is illustrated in Figure 6.

Computationally, conditioning acts as a filter: given an event B, it removes all the probability mass from the total realizations that are within the event $\neg B$, renormalizing the remaining probability mass globally. This operation can be performed recursively. Given a probability tree and an event's min-cut, we recursively descend the tree depth-first until reaching a node within the min-cut. If the node is within the true set, we compute its probability and return it; otherwise we return zero. Then, during the recursive ascent, we equate the transition probabilities with returned probabilities normalized by their aggregate sum. This sum is then returned upstream and the process is repeated.

A special case occurs when we condition on an event with probability zero, or more generally, when there exists a node where every next transition leading into the true set has probability zero. Such transitions cannot be normalized by dividing by a normalizing constant. In this case, measure theory requires us to choose a *version* of the conditional probability. Many choices exist, but for concreteness, here we have adopted a uniform distribution (see Figure 8).

The algorithm for conditioning is shown in Figure 7. $\text{SEE}(n, \delta)$ takes a root n of a probability tree and a min-cut δ and returns the root of a new, conditioned probability tree. The logical structure of the original probability tree is preserved; only the transition probabilities upstream of the min-cut change. Finally, conditioning is a commutative operation.

```
1:  function SEE($n$: Node, $\delta$: Min-Cut)
2:      $(n, l, p) \leftarrow$ SEE-R$(n, \delta, 1)$
3:  return $n$
4:  function SEE-R$(n, \delta, q)$
5:      ▷ $n$: Node; $\delta$: Min-Cut; $q$: [0, 1]
6:      Let $n = (u, \mathcal{S}, \mathcal{C})$ and $\delta = (\mathcal{T}, \mathcal{F})$
7:      ▷ Base case:
8:      if $u \in \mathcal{T}$ then
9:          return $(n, 1, p)$
10:     if $u \in \mathcal{F}$ then
11:         return $(n, 0, 0)$
12:     ▷ Recurse:
13:     $\mathcal{D} \leftarrow \varnothing$
14:     $\sigma_l \leftarrow 0$ and $\sigma_p \leftarrow 0$
15:     for all $(p, \tilde{n}) \in \mathcal{C}$ do
16:         $(\tilde{n}, l, p) \leftarrow$ SEE-R$(\tilde{n}, \delta, q \cdot p)$
17:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\tilde{n}, l, p)\}$
18:         $\sigma_l \leftarrow \sigma_l + l$
19:         $\sigma_p \leftarrow \sigma_p + p$
20:     ▷ Normalize:
21:     $\mathcal{C} \leftarrow$ NORMALIZE$(\mathcal{D}, \sigma_l, \sigma_p)$
22:     $n \leftarrow (u, \mathcal{S}, \mathcal{C})$
23:     return $(n, 1, \sigma_p)$

1:  function NORMALIZE$(\mathcal{D}, \sigma_l, \sigma_p)$
2:      if $\sigma_p > 0$ then
3:          $\mathcal{C} \leftarrow \{(\frac{p}{\sigma_p}, n) | (n, l, p) \in \mathcal{D}\}$
4:      else
5:          $\mathcal{C} \leftarrow \{(\frac{l}{\sigma_l}, n) | (n, l, p) \in \mathcal{D}\}$
6:      return $\mathcal{C}$
```

Figure 7: Conditioning algorithm

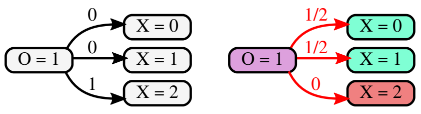

Figure 8: Conditioning on $\neg(X = 2)$.

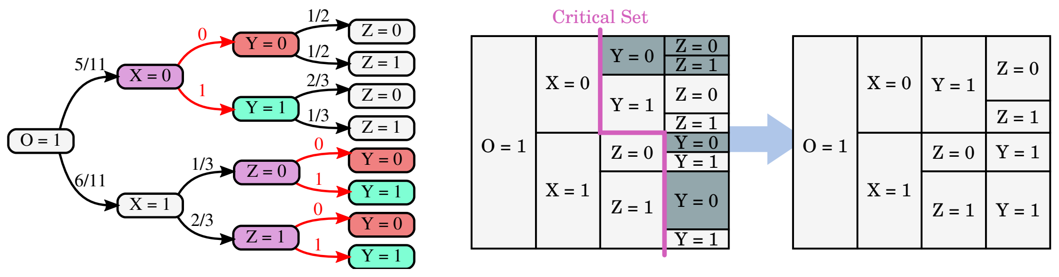

Figure 9: Doing $Y \leftarrow 1 := \text{do}(Y = 1)$. An intervention proceeds in two steps: first, it selects the partial realizations starting in a critical node and ending in a leaf that traverse the false set of the event's min-cut; then it removes their probability mass, renormalizing the probabilities leaving the critical set. The left panel shows the result of intervening the probability tree from Figure 1d with $Y \leftarrow 1$. The right panel illustrates the procedure on the corresponding probability mass diagram.

4 Interventions

In a probability tree, intervening is the act of minimally changing the transition probabilities in order to bring about a desired event with probability one. This operation allows answering questions of the form

$$P(A \mid \text{do}(B))$$

that is, "what is the probability of the event A given that the event B was made true?" The operation is illustrated in Figure 6. The crucial difference to conditions is that interventions only affect realizations downstream of the critical set, whereas conditions also provide information on what happened upstream of the set.

Computationally, interventions are simpler than conditions. We compute interventions using a depth-first search recursive algorithm, which descends down the probability tree until the min-cut of the desired event is reached. Then, we remove the probability mass of the transitions leading into the nodes of the false set and renormalize the transition probabilities into the nodes of the true set. This normalization is local, relative to the subtree rooted at the parent critical node. As in conditioning, intervening on events having nodes with probability zero in the false set requires special treatment, and we settle for the same solution as before.

```
1:  function Do($n$: Node, $\delta$: Min-Cut)
2:      $(n, l) \leftarrow$ Do-R$(n, \delta)$
3:      return $n$
4:  function Do-R($n$: Node, $\delta$: Min-Cut)
5:      Let $n = (u, \mathcal{S}, \mathcal{C})$ and $\delta = (\mathcal{T}, \mathcal{F})$
6:      ▷ Base case:
7:      if $u \in \mathcal{T}$ then return $(n, \top)$
8:      if $u \in \mathcal{F}$ then return $(n, \bot)$
9:      ▷ Recurse:
10:     $\mathcal{D} \leftarrow \varnothing$
11:     $\sigma_l \leftarrow 0$ and $\sigma_p \leftarrow 0$
12:     for all $(p, \tilde{n}) \in \mathcal{C}$ do
13:         $(\tilde{n}, \beta) \leftarrow$ Do-R$(\tilde{n}, \delta)$
14:         if $\beta = \top$ then
15:             $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\tilde{n}, 1, p)\}$
16:             $\sigma_l \leftarrow \sigma_l + 1$
17:             $\sigma_p \leftarrow \sigma_p + p$
18:         else
19:             $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\tilde{n}, 0, p)\}$
20:     ▷ Normalize:
21:     $\mathcal{C} \leftarrow$ Normalize$(\mathcal{D}, \sigma_l, \sigma_p)$
22:     $n \leftarrow (u, \mathcal{S}, \mathcal{C})$
23:     return $(n, \top)$
```

Figure 10: Intervention algorithm

The pseudo-algorithm for computing interventions is shown in Figure 10. The function Do(n, δ) takes a root n and a min-cut δ, and returns the root of a new, intervened probability tree.

Interventions do not alter the structure of the original probability tree; they only change the transition probabilities emanating from the critical set. Furthermore, they are commutative, as is conditioning. However, conditions and interventions do *not* commute. For instance, for the probability tree in Figure 1d,

$$P(X = 0 \mid Y \leftarrow 1; Z = 0) = \frac{5}{8} \neq P(X = 0 \mid Z = 0; Y \leftarrow 1) = \frac{3}{5}$$

where the semicolon ";" denotes the sequential composition of the operators.

It should be stressed that interventions on events are strictly more general than interventions on random variables. Thus, unlike in CBNs, an intervention does not necessarily assign a unique value to a manipulated random variable. Rather, depending on the critical set, the intervention could assign different values to the same random variables in separate branches of the tree, or in fact even manipulate different random variables in every branch (a context-dependent "recipe"), in order to bring about a desired effect.

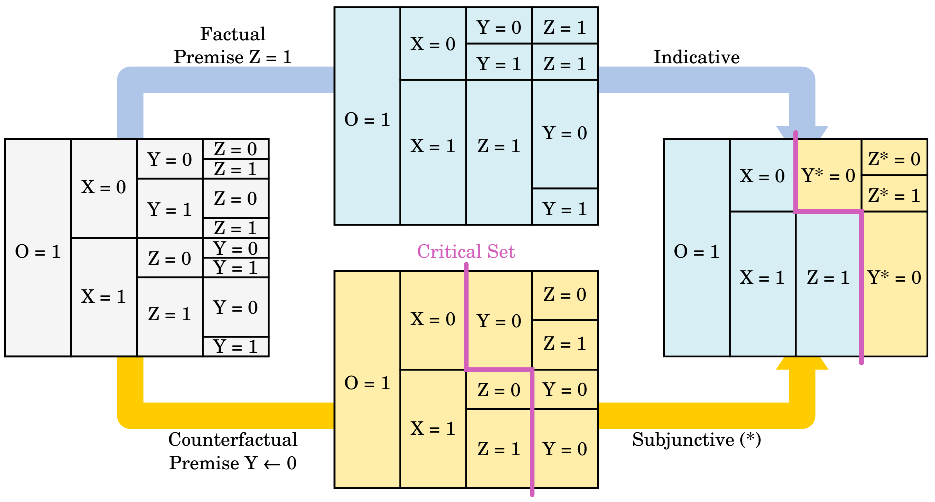

Figure 11: The counterfactual probability tree generated by imposing $Y \leftarrow 0$, given the factual premise $Z = 1$. Starting from a reference probability tree, we first derive two additional trees: a factual premise, capturing the current state of affairs; and a counterfactual premise, represented as an intervention on the reference tree. To form the counterfactual, we slice both derived trees along the critical set of the counterfactual premise. Then, we compose the counterfactual tree by taking the transition probabilities upstream of the slice from the factual premise, and those downstream from the counterfactual premise. The events downstream then span a new scope with copies of the original random variables (marked with "$*$"), ready to adopt new values. In particular note that $Z^* = 0$ can happen in our alternative reality, even though we know that $Z = 1$.

5 Counterfactuals

In a probability tree, a counterfactual is a statement about a subjunctive (i.e. possible or imagined) event that could have happened had the stochastic process taken a different course during its realization. This operation allows evaluating probabilities of the form

$$P(A_C \mid B)$$

that is, "Given that B is true, what would the probability of A be if C *were* true?". Here, A_C denotes the subjunctive event A under the counterfactual assumption that the event C has occurred (i.e. a potential response), and B is the indicative (i.e. factual) assumption.

A counterfactual probability tree is obtained by modifying a reference probability tree through factual statements (i.e. conditioning and intervening), and then spawning a new variable scope from a counterfactual modification, formalized as an intervention. This operation effectively resets the state of the random variables downstream of the intervention to their reference, unbound state. The operation is illustrated in Figure 11.

```
1:  function CF($n, m$: Node, $\delta$: Min-Cut)
2:      $n \leftarrow \textsc{Do}(n, \delta)$
3:      $(n, l) \leftarrow \text{CF-R}(n, m, \delta)$
4:      return $n$
5:  function CF-R($n, m, \delta$)
6:      ▷ $n, m$: Node; $\delta$: Min-Cut
7:      Let $n = (u, \mathcal{S}, \mathcal{C}_n)$
8:      Let $m = (u, \mathcal{S}, \mathcal{C}_m)$
9:      Let $\delta = (\mathcal{T}, \mathcal{F})$
10:     ▷ Base case:
11:     if $u \in \mathcal{T}$ then return $(n, \top)$
12:     if $u \in \mathcal{F}$ then return $(n, \bot)$
13:     ▷ Recurse:
14:     $\alpha \leftarrow \bot, \mathcal{C} \leftarrow \varnothing$
15:     for all $(c, d) \in \textsc{Zip}(\mathcal{C}_n, \mathcal{C}_m)$ do
16:         Let $c = (p, \tilde{n}), d = (q, \tilde{m})$
17:         $(\tilde{n}, \beta) \leftarrow \text{CF-R}(\tilde{n}, \tilde{m}, \delta)$
18:         if $\beta = \bot$ then $\alpha \leftarrow \top$
19:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{(q, \tilde{n})\}$
20:     ▷ Not a critical bifurcation?
21:     if $\alpha = \bot$ then
22:         $n \leftarrow (u, \mathcal{S}, \mathcal{C})$
23:     return $(n, \top)$
```

Figure 12: Counterfactual algorithm

The pseudo-code for the computation of counterfactuals is listed in Figure 12. The function CF(n, m, δ) takes the root n of the reference probability tree, the root m of the factual premise tree, and a min-cut δ for the counterfactual event. It returns a new counterfactual probability tree. The auxiliary function $\textsc{Zip}(\mathcal{A}, \mathcal{B})$ takes to ordered sets $\mathcal{A} = \{a_n\}_{n=1}^{N}$ and $\mathcal{B} = \{b_n\}_{n=1}^{N}$ and returns a new set $\mathcal{C} = \{(a_n, b_n)\}_{n=1}^{N}$ of ordered pairs.

6 Discussion

Previous work. As already mentioned in the introduction, the literature on probability trees is limited. Although trees are common for representing games (specifically, extensive-form games [24, 25]) and sequential decision problems [26], it was Shafer's seminal work [19] that first articulated a treatment of causality based on probability trees. Unlike Pearl [2], who grounds the semantics of causal relations on the notion of interventions, Shafer considered causality as a side-effect entirely subsumed under conditional independence. Thus, to the best of our knowledge, event-based interventions were only introduced later independently in [23] and in [27, 21], the former on systems of nested events (whose algebraic closure generates the algebra of the probability space) and the latter on an elegant method characterizing probability trees in terms of interpolating polynomials (where interventions amount to computing derivatives), which also establishes the equivalence classes of probability trees. The formalization of counterfactuals presented here is original and differs from the counterfactuals in [2] in important ways discussed later.

The inability of CBNs (and SCMs) to represent context-specific independencies is well known: the conditional independencies of CBNs are exactly those that are logically equivalent to a collection of the form $X_n \perp\!\!\!\perp \{X_1, \dots, X_{n-1}\} \mid S_n$, where $X_1, \dots, X_N$ is an ordering of the vertices in the graph, and S_n is a subset of $\{X_1, \dots, X_{n-1}\}$ [18]. This has led to the suggestion of using probability trees as an alternative representation for modeling context-specific independencies [28, 20]. In the context of factor graphs, [29] introduced a notation (called "gates") for modeling context-specific independencies along with corresponding inference algorithms. Lastly, since probability trees may be regarded as a formalization of the computational traces of probabilistic programs, context-specific independencies are modeled naturally by probabilistic programming languages [30].

Computational complexity. All of the presented algorithms (except NEG, which has $\mathcal{O}(1)$ time complexity) follow the same pattern: they recursively descend the tree once and then backtrack to the root node. The worst-case scenario occurs when the probability tree is structured like a chain, in which case the time and space complexity of the algorithms is $\mathcal{O}(N)$, where N is the number of nodes in the tree. However, a shortcoming of our algorithms is that they do not exploit the independencies present in a probability tree, which could dramatically reduce the computational complexity. More work is required to understand how to incorporate those.

Counterfactuals. The algorithm for computing counterfactuals generalizes the twin-network construction used in SCMs [2]. To understand this generalization, recall that SCMs are stricter than CBNs, in that SCMs impose constraints on how the uncertainty enters the random experiment (Figures 13a,b). Specifically, the variables of the random experiment—the endogenous variables—are deterministic functions of exogenous variables, which concentrate all the uncertainty. Hence, all the uncertainty of about the random experiment causally precedes the endogenous variables. CBNs make no such assumption: in particular, the uncertainty of any variable could causally depend on its parents.

This difference matters when estimating counterfactuals. In an SCM, any knowledge gained about the exogenous variables transfers to the counterfactual world because it always precedes any counterfactual intervention on the endogenous variables. There is no "endogenous uncertainty"—which would not necessarily transfer.

In contrast, in probability trees, there is no distinction between endogenous and exogenous variables; the uncertainties can originate anywhere along the realization of the random experiment in a context-specific manner—see e.g. how one could model the difference between CBNs and SCMs in Figures 13c,d. The rule for transferring information to the counterfactual world is simple: information upstream of the counterfactual intervention transfers while the one downstream doesn't. This allows for a fine-grained control over the assumptions for estimating counterfactuals.

Interpretation. Our work presents interpretation-agnostic algorithms for causal reasoning. For instance, we can impute both objective and subjective semantics to the probabilities and causal relations in a probability tree. Regardless, not every construction yields statements with well-defined empirical content—e.g. see a critique of counterfactual statements in [31]. Indeed, many of the interpretative limitations and caveats that apply to CBNs extend to probability trees [18], especially

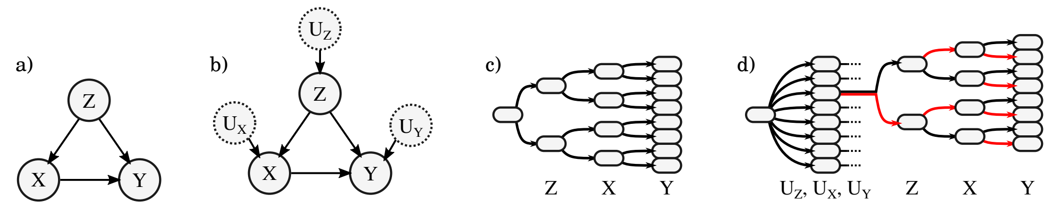

Figure 13: Modeling CBNs versus SCMs as probability trees. a) In a CBN, every random variable has an associated probability distribution conditioned on the parents, but otherwise no further assumptions are made. b) Instead, SCMs assume that the endogenous variables (solid) are deterministic functions of the exogenous random variables (dotted). c) The CBN in (a) can be modeled as a four-stage probability tree, where each layer binds one of the variables in turn. We assumed X, Y, and Z to be binary. d) In contrast, the SCM in (b) requires a five-stage probability tree, placing all the uncertainty in the first transition which binds the triple of exogenous variables (U_Z, U_X, U_Y), followed by deterministic transitions that bind the endogenous variables X, Y, and Z (the red transitions indicate transitions of zero probability).

the reification[5] of the formal elements that are peculiar to probability trees. More work is required in order to understand these.

It must be noted however, that a subjective interpretation of probability trees puts causal discovery on a firm formal ground. Questions such as “does X cause Y or Y cause X?” can both be articulated (through conditional causal relations) and answered (at least in a Bayesian sense, through observing the effects of interventions) without introducing extraneous formal machinery [20]. Unlike in CBNs, there is no need to reason over a collection of models—a single probability tree suffices.

Features and Limitations. The presented algorithms assume that probability trees are finite. If the random variables are well-formed (Section 2.1), then the algorithms extend without modification to trees with countably many nodes, because the computations of min-cuts and the causal reasoning operations always terminate due to their recursive nature. Extensions to trees with uncountably many nodes require fundamentally different techniques, but they are strictly more general than CBNs in their expressive power.

The focus on events rather than random variables has led us to rethink and generalize the causal reasoning operations. This is especially salient in the algorithms for computing interventions and counterfactuals. The intervention algorithm presented here allows answering reverse-engineering questions like “what are the necessary manipulations in order to bring about an event A?”. Moreover, though the resulting probabilities coincide when applicable, the computation of counterfactuals in probability trees differs significantly from that in CBNs using e.g. the twin network construction [2].

Having said that, the building blocks for constructing events are still limited. In particular, they are limited to propositional logic and one causal relation, namely *precedence* (Section 2.3). We envision the addition of more causal relations, and the extension to first-order logic, to enrich the language of events. In particular, one could also explore links to *temporal logic* [32] and *probabilistic programming* [33].

A system of transformations akin to the *do-calculus* is entirely absent and left for future work. Such a system is necessary for addressing questions about *identification*, i.e. whether a causal effect can be estimated from observation alone [2]. More generally, the precise algebraic properties of the causal operations, such as when conditions and interventions commute, are currently unknown.

Finally, the decision to place uniform conditional probabilities over events with probability zero is arbitrary and only made for the sake of simplicity and concreteness. To encode other versions of the conditional probabilities, one could ban zero-probability transitions and replace them with “vanishingly small” quantities (for instance, polynomials in $\epsilon \approx 0$).

[5]That is, mistaking an abstract idea for a physical thing.

Acknowledgments

We thank Tom Everitt, Ramana Kumar for their comments and Silvia Chiappa for proof-reading.

References

[1] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

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

[3] A Philip Dawid. Statistical causality from a decision-theoretic perspective. *Annual Review of Statistics and Its Application*, 2:273–303, 2015.

[4] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[5] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[6] Bernhard Schölkopf. Causality for machine learning. *arXiv preprint arXiv:1911.10500*, 2019.

[7] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736, 2008.

[8] Pedro A Ortega and Daniel A Braun. A minimum relative entropy principle for learning and acting. *Journal of Artificial Intelligence Research*, 38:475–511, 2010.

[9] Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. In *Advances in Neural Information Processing Systems*, pages 1342–1350, 2015.

[10] Finnian Lattimore, Tor Lattimore, and Mark D Reid. Causal bandits: Learning good interventions via causal inference. In *Advances in Neural Information Processing Systems*, pages 1181–1189, 2016.

[11] Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162*, 2019.

[12] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, pages 4066–4076, 2017.

[13] Junzhe Zhang and Elias Bareinboim. Fairness in decision-making—the causal explanation formula. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] Silvia Chiappa. Path-specific counterfactual fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7801–7808, 2019.

[15] Tom Everitt, Pedro A Ortega, Elizabeth Barnes, and Shane Legg. Understanding agent incentives using causal influence diagrams, part i: single action settings. *arXiv preprint arXiv:1902.09980*, 2019.

[16] Ryan Carey, Eric Langlois, Tom Everitt, and Shane Legg. The incentives that shape behaviour. *arXiv preprint arXiv:2001.07118*, 2020.

[17] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[18] A Philip Dawid. Beware of the DAG! In *Causality: objectives and assessment*, pages 59–86, 2010.

[19] Glenn Shafer. *The art of causal conjecture*. MIT press, 1996.

[20] Pedro A Ortega. Bayesian causal induction. *NIPS Workshop in Philosophy and Machine Learning*, 2011.

[21] Christiane Görgen. *An algebraic characterisation of staged trees: their geometry and causal implications*. PhD thesis, University of Warwick, 2017.

[22] Christiane Görgen, Jim Q Smith, et al. Equivalence classes of staged trees. *Bernoulli*, 24(4A):2676–2692, 2018.

[23] Pedro A Ortega. Subjectivity, bayesianism, and causality. *Pattern Recognition Letters*, 64:63–70, 2015.

[24] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[25] Kevin Leyton-Brown and Yoav Shoham. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis lectures on artificial intelligence and machine learning*, 2(1):1–88, 2008.

[26] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[27] Christiane Görgen and Jim Q Smith. A differential approach to causality in staged trees. In *Conference on Probabilistic Graphical Models*, pages 207–215, 2016.

[28] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. *arXiv preprint arXiv:1302.3562*, 2013.

[29] Tom Minka and John Winn. Gates. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2009.

[30] Sam Witty and David Jensen. Causal graphs vs. causal programs: The case of conditional branching. *First Conference on Probabilistic Programming (ProbProg)*, 2018.

[31] A Philip Dawid. Causal inference without counterfactuals. *Journal of the American statistical Association*, 95(450):407–424, 2000.

[32] Peter Øhrstrøm and Per Hasle. *Temporal logic: from ancient ideas to artificial intelligence*, volume 57. Springer Science & Business Media, 2007.

[33] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.