Petri Nets Enable Causal Reasoning in Dynamical Systems

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Dynamical systems, e.g. economic systems or biomolecular signaling networks, are processes comprised of states that evolve in time. Causal models represent these processes, and support causal queries inferring outcomes of system perturbations. Unfortunately, Structural Causal Models, the traditional causal models of choice, require the system to be in steady state and don't extend to dynamical systems. Recent formulations of causal models with a compatible dynamic syntax, such as Probability Trees, lack a semantics for representing both states and transitions of a system, limiting their ability to fully represent the system and ability to encode the underlying causal assumptions. In contrast, Petri Nets are well-studied models of dynamical systems, with the ability to encode states and transitions. However, their use for causal reasoning has so far been under-explored. This manuscript expands the scope of causal reasoning in dynamical systems by proposing a causal semantics for Petri Nets. We define a pipeline constructing a Petri Net model and calculating the fundamental causal queries: conditioning, interventions, and counterfactuals. A novel aspect of the proposed causal semantics is an unwrapping procedure, which allows for a dichotomy of Petri Net models when calculating a query. On one hand, a base Petri Net model visually represents the system, implicitly encodes the traces defined by the system, and models the underlying causal assumptions. On the other hand, an unwrapped Petri Net explicitly represents traces, and answers causal queries of interest. We demonstrate the utility of the proposed approach on a case study of a dynamical system where Structural Causal Models fail.

1 Introduction

Dynamical systems are processes composed of states that evolve in time. Such systems are of great interest in many fields including economics, systems biology etc., where causal queries: conditioning, interventions, and counterfactuals [7] are of importance.

Structural Causal Models [7], the traditional causal models of choice, only address fundamental causal queries when the dynamical system is in steady-state. This restriction is reasonable when the answer to the causal query does not depend upon the history of the values of the variables. When the *history* of variables is pertinent [10], structural causal models fail to distinguish between variables that represent events where the systems transitions from one state to another, and variables that represent the state of the system.

This issue extends to recently proposed causal semantics for dynamical systems that only represent dependencies between states or dependencies between transitions [10, 2, 4]. Such models display a tension between transparency of causal assumptions and fidelity to the underlying system.

Submitted to *A causal view on dynamical systems* workshop at NeurIPS 2022. Do not distribute.

In contrast, Petri Nets are well-studied models of dynamical systems, capable of interpretable representation of the relationship between event transitions and states of the system. So far Petri Nets have been primarily used for discrete event system modeling [6], and more recently to model dynamical systems of chemical reaction networks [12] and biological signaling networks [11]. To the best of our knowledge, there is currently no formal causal semantics developed for Petri Nets based on interventions and counterfactuals.

The contributions of this manuscript are as follows:

- We propose to expand the scope of causal reasoning in dynamical systems, by defining a Causal Petri Net model, and by providing an algorithm for its construction from a given dynamical system.
- We show that the proposed Causal Petri Net leverages the power of representing states and transitions, allowing the Causal Petri Net to bypass the choice made by previous work. Thus the proposed model can completely and symbolically model the dynamical system, while outlining the underlying causal assumptions.
- We define an interpretable causal semantics over the proposed model. To this end we use a novel unwrapping procedure, which allows us to compactly calculate queries of interest. We provide concrete algorithms for computing the fundamental queries of conditioning, interventions and counterfactuals.

2 Background

2.1 Prior work in causal models for dynamical systems

A Probability Tree is a simple model for representing processes. Their semantics are self-explanatory: a node in the tree corresponds to a potential state of the process. An arrow indicates probabilistic transitions between the nodes, but does not support variables representing the space of transitions. Algorithms for causal reasoning with Probability Trees [2] were recently proposed. However, as Judea Pearl pointed out in his criticisms against this model [8], its purely numerical representation of the edges (and hence transitions) make the model unable to explicate the underlying causal assumptions apart from temporal order.

In contrast, rule-based models utilized by Laurent et al. [4] are a powerful way to manage the combinatorial complexity of dynamical systems, using event transitions as variables. However, they have difficulties modeling the potential states of a system, and necessitate a causal semantics requiring pure simulation. Furthermore they can only use ad-hoc visualizations of traces thus similar to Probability Trees struggle to explicate causal assumptions.

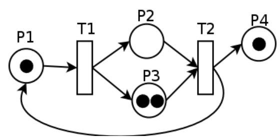
Other models, including Generalized Structural Equations models [9], Causal Constraints models [1] and CP-Logic [13] are similar to Causal Probability Trees and rule-based models, in that they lack the distinction between states and transitions, and thus the ability to encode their causal dependencies in a graphical structure.

Situation Calculus causal models are capable of representing both events and states, but at the cost of requiring second-order logic to answer causal queries [3]. In this manuscript we are interested in a causal semantics of dynamical systems that only requires propositional logic to answer causal queries [5].

2.2 Petri Nets

Petri Nets, illustrated in Fig 1, are bipartite directed multi-graphs. They consist of places P (circles) that model potential states, and transitions T (rectangles) that model potential changes and events of the system. The directed arcs connect places to transitions and vice versa. Places contain movable objects called *tokens* (small black circles in Fig 1), representing the actual state of the system. The weights of the arcs correspond to the movement of tokens during the transitions along the arcs (weights equal to one are not shown).

Definition 2.1 (Petri Net)**.** *A tuple $PN := (P, T, F)$, consisting of a place set P, a transition set T, and a flow function $F : (P, T) \cup (T, P) \to \mathbb{R}$. F takes as input directed edges and outputs a weight of the edge. The weight determines the input and output of tokens when a transition is fired.*



(a)

Figure 1: A Petri Net tuple P, T, F with places $P : \{P1, P2, P3, P4\}$, transitions $T : \{T1, T2\}$ and F denoted by the arcs connecting places and transitions, each with a weight of one (not shown).

A transition is *enabled* if the places connected into the transition have tokens greater than or equal to the weight of the edge. An enabled transition *fires* by consuming the input tokens, and outputting tokens equal to the weight of the outgoing edges into the connected places. A sequence of transition firings is called a *firing sequence*. If no transition is enabled, the Petri Net is said to be in *deadlock*. The token distribution is called a *marking* or equivalently a state, and describes a system's state of the Petri Net. The full set of markings is called the *marking set* M_{PN}. The *state graph* is a graph with each possible marking forming the nodes, and the directed edges are the transitions that connect them.

In the following we refer to a marking as a set of tuples of the form: $[(p, r)]$ where p is a place and r is the number of tokens in that place. The marking set of Fig 1 is $[(P1, 1), (P3, 2), (P4, 1)]$ We denote $PN(m)$ a Petri Net PN set to marking m. We denote $Enabled(PN(m))$ the set of enabled transitions of PN set to marking m.

3 Methods

In this section we formally walk through the entire pipeline of calculating a causal query using Petri Nets. We first show how to construct a Petri Net model from data. Next, we define semantic structures to draw meaning from the model. Finally, we calculate the fundamental causal queries: conditioning, interventions, and counterfactuals. Proofs of lemmas somewhere are available in Supplementary Materials.

Throughout this section we use the classic Firing Squad toy example outlined in Fig 2. We evaluate the "Kamchatka" counterfactual query [8], proposed by Pearl in his argument against Probability Trees [2]. The query illustrates that Firing Squad doesn't describe a state of Kamchatka, i.e. the Captain has no influence on the Prisoner given the actions of the riflemen: $P(Prisoner_{Do(Riflemen=NoShoot)} = Alive|Captain = Signal, Prisoner = Dead)$.

We walk through this query in a manner to be illustrative to how Petri Nets should be applied to dynamic causal modeling. For clarity, we introduce concepts such as initialization of the agents (e.g., setting the riflemen to be on "standby") and tracking of the agents (i.e., at any given time we model all relevant agents, such that the Prisoner is viewed as alive unless otherwise shot). However these concepts are not strictly necessary, and a more classic treatment of the Firing Squad example is provided in Supplementary Materials.

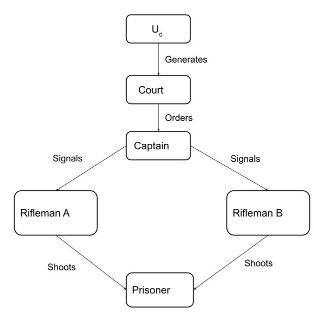


Figure 2: Firing Squad toy example. The court orders or doesn't order the prisoner's death based on exogenous variable U_c, which is assumed to be uniform. If the court gives the order the captain gives a signal, otherwise there is no signal. Each riflemen (A and B) see the signal and shoot the prisoner, otherwise they don't shoot. The prisoner dies if either riflemen shoot.

3.1 Dynamical systems of consideration

Definition 3.1 (Discrete Dynamical System)**.** *A set of discrete random variables* $X : \{X_1, ..., X_n\}$*, where the possible states is defined to be* $S := \{(X_{i_1} = x_1, ..., X_{i_k} = x_k)|i_1, ..., i_k \in \{1, ...n\}, x_i \in dom(X_i)\}$ *denotes the product space of these variables, a set of possible initial states* $S_0 \subseteq S$*, and a family of functions* $F : S \to S$ *with an optional corresponding probability mass for each function* $\Delta(f)$.

We define a dynamical system as comprised of: variables, any combination of realizations of the variables determines the *possible* states of the system, a set of initial states determines the space of initializations and a set of functional relations governs the state changes, usually through time, along with a probability mass associated with these functions determining the probability they occur if able. Here we represent a dynamic system as $M := (X, S, S_0, F, \Delta)$. A formal breakdown of examples using this definition is in Supplementary Materials.

A dynamical system M implicitly defines various traces of states. Here traces are simply a sequence of states beginning with a valid initial state, connected by the defined family of functions of M. This motivates a more flexible definition of an event with respect to these traces. We define an event as a set of states over M where the event is said to occur within a trace if each state occurs in this trace.

Definition 3.2 (Traces defined by M)**.** *A sequence of states $t := [s_0^t, ..., s_k^t]$ where $s_0^t \in S_0, s_i^t \in S$. Furthermore $\exists f \in F$ such that $f(s_i^t) = s_{i+i}^t$.*

Definition 3.3 (Events over M)**.** *A set of states $e := \{s_1^e, ..., s_n^e\}$ where e is said to occur in a given trace t defined by M if for each $s_e \in E$ there exists some $s_t \in t$ s.t. $s_e \subseteq s_t$.*

3.2 Defining and constructing causal Petri Nets

To represent the inherent stochasticity of dynamical systems, we define a Probabilistic Petri Net by imbuing a Petri Net tuple with a probability mass distribution over the transitions. Given a the set of enabled transitions at a given marking, we normalize the probability mass to determine the probability over the transitions. As a standard Petri Net inherently has a uniform distribution over its transitions, in the following we assume without loss of generality that all Petri Nets are Probabilistic Petri Nets. We also define a useful "Tree-Like" property of a Petri Net, applicable when its structure mirrors that of a graphical tree.

Definition 3.4 (Probabilistic Petri Net)**.** *A tuple $PN := (P, T, F, \Delta)$ and $\Delta : T \to R$ is the firing probability mass function of the transitions.*

Definition 3.5 (Tree-like Petri Net)**.** *A Petri Net is tree-like iff each transition has at most one input and one output arc.*

We note that if a Tree-like Petri Net is probabilistic then, given a place r, we can calculate the probability $P(c|r)$ of an ancestor c to occur. This is obtained by multiplying the unique firing sequence of transitions connecting c and r. Moreover, this procedure allows us to derive a distribution $P_r(l) \forall l \in Leaves(r)$ over all the leaves of a root place r.

We can now define the Causal Petri Net. The definition parallels Judea Pearl's method of separating the exogenous and endogenous variables of a system. Define PN_U the portion of the model representing the exogenous part of the system, and PN_M the portion representing the endogenous part. PN_U consists of the one place (referred as the Root) and transitions which determine the initial marking of PN_M.

Definition 3.6 (Causal Petri Net model)**.** *A Petri Net tuple $PN : (P, T, F)$ composed of three disjoint parts (PN_U, PN_M, F_c), where $PN : (P, T, F)$, $PN_U : (P_U, T_U, F_U)$ are Petri Net tuples and F_c is a flow function $(T_U, P_M) \to R$. We then define the elements of our Causal Petri Net: $P = P_U \cup P_M$, $T = T_U \cup T_M$ and $F = F_U \cup F_M \cup F_c$, furthermore P_U is a singleton set, called Root.*

3.2.1 Design choices in the construction of causal Petri Nets

The very first step of the pipeline is constructing the model itself, from a dynamical system $M : (X, S, S_0, F, \Delta)$. This is done by enumerating all the variables X described by the system along with their possible values, represented by the places of the Petri Net. F will be represented by the transitions, where the flow function will be determined by the coefficients of the functions. We capture S_0 with the exogenous Petri Net PN_U. This is done by Alg 1.

Lemma 3.1. *Every trace t of the system M has a corresponding firing sequence $\{m\}$ in its constructed Petri Net, and every firing sequence $\{m\}$ in Petri Net corresponds to a trace in M.*

Throughout our manuscript we will represent places as large rounded rectangles and transitions as smaller rectangles. We apply Alg 1 to the Firing Squad example shown in Fig 3 (a) . We see that all the values of the variables have been initialized as places, with a twist. We introduced the "StandBy" value representing the initialization state of the agents described by the example, as the relations are properly read as "Given the Captain's signal Rifleman A then fires" implying the Riflemen existed in a state of not having made a decision initially. This gives a powerful meaning to the tokens as they effectively track the state of all the agents in the system. This additionally showcases itself in the

Algorithm 1: ConstructPN

Input: M

A dynamical system $M := (X, S, S_0, F)$

Output: A Causal Petri Net tuple $PN : (PN_U, PN_M, F_c)$ modeling the input

Initialize two empty Petri Net tuples $PN_U : (P_U, T_U, F_U), PN_M : (P_M, T_M, F_M)$

We create a Place: $(X_j = x_j^i) \in P_M$ for all i, j

for *Each* $f \in F$*:* $f(x_i) = x_o$ *where* $x_i, x_o \in X$ **do**

  Create a transition t in T_M

  Let the inputs being the places corresponding to x_i, with $F_M(x_i, t)$ corresponding to the coefficients

  Let output places being the places corresponding to x_o, with $F_M(t, x_o)$ corresponding to the coefficients

Create a place $Root \in P_U$

for *Each* $s_0 \in S_0$ **do**

  create a transition t in T_U

  Initialize one input arc $F_U(Root, t) = 1$

  Let output places being the places corresponding to s_0, with $F_c(Root, s_0)$ corresponding to the coefficients

Return: $PN : (P_U \cup P_M.T_U \cup T_M, F_U \cup F_M \cup F_c)$

bi-directionality of many of the transition arrows as often the relation doesn't necessarily *consume* the agent. When Rifleman A sees the captains signal, the signal doesn't go away as other Riflemen can still read it (in this example B). Thus it is natural that this relation doesn't consume the token in the place corresponding to the Captains Signal. While these idiosyncrasies aren't necessary (the model will effectively model the example without) they do serve as a example of the differences in dynamic modeling. We note the separation of models in Fig 3 (a), with the two transitions in the exogenous Petri Net corresponding to the two court orders.

3.3 Unwrapping of a causal Petri Net for query calculation

Often we want to reason about the overall states the Petri Net model can take as well as the connections between them (through the transitions that can occur). However we don't want to enumerate all possible states the Petri Net has but rather a subsection of it. This unwrapping process is described by Alg 2. The Unwrapping Algorithm returns a tree-like Petri Net where the leaves label firing sequences where an event of interest occurs, and where they cannot occur. This procedure will prove critical in evaluating most queries over any Petri Net.

Algorithm 2: UnwrapPN

Input: (PN, M, E)

A causal Petri Net tuple $PN = (P_U, P_M, F_c)$, the dynamical system M, and a set of events E over M

Output: A Tree-Like Petri Net tuple PN_s, with root place corresponding to P_U and leaves corresponding to when E occurs or can't occur

1 Create an empty Petri Net tuple $PN_s : (P_s, T_s, F_s)$

2 Add a place $Root \in P_s$ which corresponds to the marking in PN with a token in root

3 Let m be $Root \in P_s$

4 $\forall e \in E$ if $m \in e$ we pop the corresponding element in e

5 We stop if an event $e \in E$ is empty, if m is a child of itself, or if we are in deadlock

6 **for** *Each* $t \in Enabled(PN(m))$ **do**

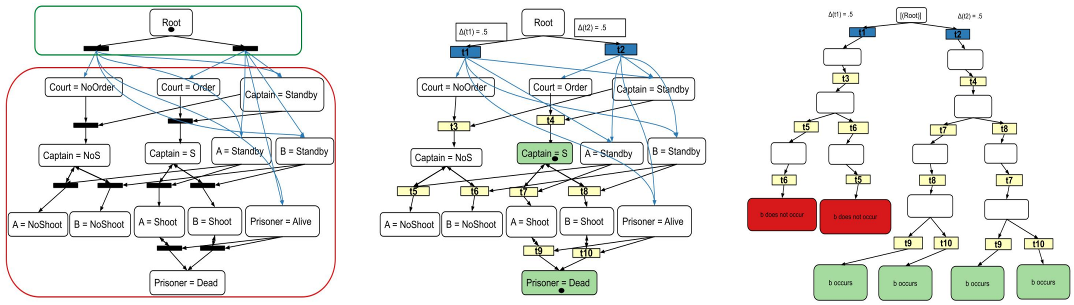
7   Let m_t denote the marking of PN when t fires

8   Append a transition t to T_s

9   Append an incoming arc $F((m, t)) = 1$ and an outgoing arc $F((t, m_t)) = 1$

10 Repeat steps 4-9 for each child place of m in PN_s

11 **Return:** PN_s



(a) (b) (c)

Figure 3: Construction and Abduction on a Petri Net. (a) The constructed Petri Net of the Firing Squad toy example. The green box outlines the exogenous Petri Net PN_U, the red box the endogenous Petri Net PN_M and the blue arcs outline the connection arcs, F_c. We condition on the event $b : \{s_1 = [(Captain = Signal, 1)], s_2 = [(Prisoner = Dead, 1)]\}$ (b) The Petri Net with the places corresponding to the event highlighted in green. The transitions in the exogenous Petri Net are highlighted in blue, and their probabilities are shown. (c) The *outline* of the unwrapped model in the forward simulation step. Leaves where b occurs are in green. Leaves where b doesn't are in red. This allows us to calculate the desired distribution over the exogenous states, using Bayes rule: $P(\Delta(t1)|b), P(\Delta(t2)|b) = (0, 1)$ as desired. A fully labeled version of this Petri Net is in Supplementary Materials.

3.4 Forward simulation on a causal Petri Net

Given a Causal Petri Net model $PN : (PN_U, PN_M, F_c)$ a fundamental query is to calculate the distribution over some set of events E over our dynamical system. We utilize the unwrapping algorithm, and then return the distribution over the leaves. This is outlined in Alg 2

Algorithm 3: FSimPN

Input: $(PN, M, E))$
A Causal Petri Net tuple $PN = (P_U, P_M, F_c)$, the dynamical system M and a set of events E over M
Output: A probability distribution $P(e) \forall e \in E$

Initialize P to map everything to 0
$PN_T \leftarrow UnwrapPN(PN, M, E)$
For each $e \in E$ set $P(e)$ to be the sum of all paths from root to leaf where e occurs
Return: P

Algorithm 4: ConditionPN

Input: (PN, M, a, b)
A Causal Petri Net tuple PN, and events a, b over M
Output: The conditional $P(a|b)$

$PN_s \leftarrow UnwrapPN(PN, M, b)$
Mark nodes where event a occurs over PN_s
Calculate $P(a), P(b)$ over PN_s
Let PN_a be a subtree with root where a occurs
$P(b|a)$ is equal to $P(b)$ over PN_a
$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$
Return: $P(a|b)$

Lemma 3.2. *The probability of event e in M is equal to the probability returned by $FSimPN(PN, e)$*

3.5 Conditioning and abduction over a causal Petri Net

Consider a system M and its causal Petri Net $PN : (PN_U, PN_M, F_c)$. Conditioning revolves around calculating a probability of the form $P(a|b)$, where a, b are events over M. We calculate the conditional using Bayes rule, to get the probabilities we consider the unwrapped Petri Net PN_s over event b. We loop through PN_s and mark all nodes where a occurs. We can now calculate $P(b)$ and $P(a)$ by summing the probabilities of all paths from the Root place. We now consider any node where a occurs, and sum the probabilities of all paths from this node to nodes where b occurs to get $P(b|a)$. Applying Bayes rule we get $P(a|b)$ as desired. This is formalized in Alg 4

Lemma 3.3. *The probability of $P(a|b)$ in M is equal to the probability returned by $ConditionPN(PN, M, a, b)$*

Abduction is the act of inferring the distribution of the exogenous variables given some event b. Thus it is closely related to conditioning, where we condition on an event and infer the distribution over the exogenous states. We simply apply the conditioning algorithm over each marking mapped from PN_U to PN_M and note that the probability of each exogenous marking is simply it's associated transition probability. Abduction is formalized in Alg 5. We can now perform the abduction step of the Kamchatka counterfactual query, conditioning the Constructed Causal Petri Net on the event: $(Captain = Signal, Prisoner = Dead)$ shown in Fig 3 (b)-(c).

Algorithm 5: AbductionPN

Input: (PN, M, b)
A Causal Petri Net tuple $PN : (PN_U, PN_M, F_c)$, b is an event over M
Output: An updated $PN_U^b : (P, T, F, \Delta^b)$ where Δ^b is the inferred probability distribution over the exogenous markings

Let Δ be the transition probabilities of PN_U
Initialize Δ^b
Let $\Delta(a)$ denote the transition probability of exogenous marking a
$PN_s \leftarrow UnwrapPN(PN, M, b)$
Calculate $P(b)$ over the leaves in PN_s
for *each child u of Root in PN_s* **do**
    Set $P(u)$ to $\Delta(u)$
    Calculate $P(b|u)$ over the leaves of the subtree with u as the root
    $\Delta^b(u) = \frac{P(b|u)P(u)}{P(b)}$
Return: $PN_U^b := (P, T, F, \Delta^b)$

Algorithm 6: CounterfactualPN

Input: (M, Q)
A system M, A counterfactual query Q over M: Given X, what is the probability of Y had we done Z
Output: The probability distribution $P(Y_{Do(Z)}|X)$

$PN : (PN_U, PN_M, F_c) \leftarrow ConstructPN(M)$
$PN_U^X \leftarrow AbductionPN(PN, M, X)$
$PN_M^Z \leftarrow PN_M$ after the intervention Z
Return: $FSim(PN_{count} := (PN_U^X, PN_M^Z, F_c), M, Y)$

3.6 Interventions on a causal Petri Net

We now define an intervention on a Petri Net, which is simply an extension of DAG mutilation to this model.

Definition 3.7 (Intervention on a Petri Net)**.** *Consider any Causal Petri Net tuple PN : (PN_U, PN_M, F_c), we define an intervention over the tuple PN_M : (P, T, F). An intervention is a mapping of the form $I : PN \rightarrow PN_I =: (P, T_I, F_I)$ where $T_I = (T \backslash T_r) \cup T_a$, with T_r, T_a be the set of transitions we remove and add respectively, $F_I : (T_I, P) \cup (P, T_I) \rightarrow R$ where R is the set of real numbers.*

Given an Intervened Petri Net PN with intervention $DO(Z)$, we have that $P(Y)$ is equal to $P(Y_{DO(Z)})$. We apply our definition to create the Intervened Petri Net corresponding to $DO(riflemen = NoShoot)$ in the Kamchatka query shown in Fig 4 (a).

3.7 Counterfactuals on a causal Petri Net

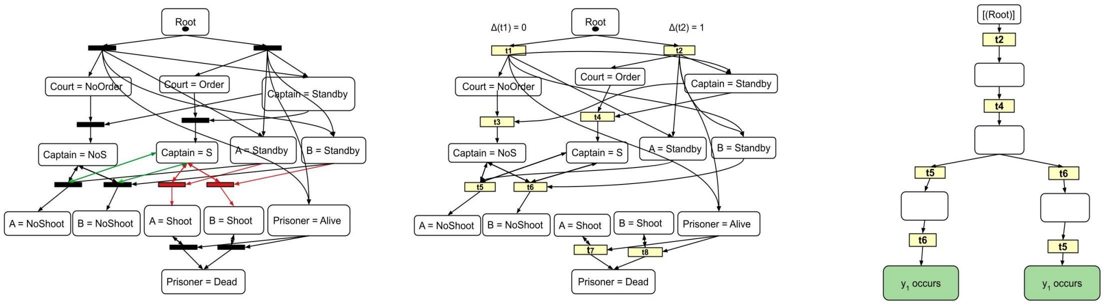
A counterfactual query is a statement of the form: *"Given that X happened, would Y have happened had we done Z"*. Consider a system M and it's constructed causal Petri Net PN. Following [7] we perform counterfactuals in three steps of abduction, action and prediction:

Abduction: We run the abduction algorithm with Causal Petri Net $PN : (PN_U, PN_M, F_c)$, and X as the event to get the updated Exogenous Petri Net PN_U^X

Action: Convert Z to an intervention over PN_M to get PN_M^Z

Simulation: We run the Forward Simulation algorithm with $PN = (PN_U^X, PN_M^Z, F_c)$ over event Y to get $P(Y)$ which is equivalent to the counterfactual query of interest: $P(Y_{Do(Z)}|X)$ as the probability is calculated over the Petri Net with intervention $DO(Z)$ conditioned on X.

Putting these steps together we get Alg 6. Using this Algorithm we can now fully evaluate the Kamchatka query shown in Fig 4.



(a) (b) (c)

Figure 4: Kamchatka intervention and counterfactual query on a Petri Net. (a) The Petri Net from Fig. 3 a. after intervention $DO(Riflemen = NoShoot)$. The deleted elements are highlighted red, while the added elements are green. (b) The intervened net with the abducted distribution over the exogenous states. (c) Denotes the *outline* of the forward simulation with event set $Y := \{\{y_1 : s_1 = [(Prisoner = Alive, 1)]\}, \{y_2 : s_2 = [(Prisoner = Dead, 1)]\}\}$ where y_1, y_2 are singleton events. We highlighted leaves where the prisoner is alive as green, and leaves where the prisoner dies as red. We can now calculate the desired probability $P(Prisoner_{Do(Riflemen=NoShoot)} = Alive|Captain = Signal, Prisoner = Dead) = 1$ as part of our algorithm's output. Thus we are not in a state of "Kamchatka" as expected. For the fully enumerated Petri Net of (c) with the places fully labeled readers can consult the supplementary section.

3.7.1 Counterfactuals on an individual trace

Often in dynamic systems, we are interested in a counterfactual over a specific trace simulated by the system. The query is of the form *"Given a sequence of states, X, connected by the system's functions, what is the probability of Y had we done Z"*. This is computed in a similar manner as the traditional counterfactual query. The only difference is that for this conditioning we replace PN_U with a singular transition to the marking in X just before the intervention. This method is showcased in the case study below.

4 Counterfactual Resimulation case study

Motivation We now work through a problem in which SCMs were unable to represent [4], requiring the authors to repeatedly resimulate the system to answer a counterfactual query. Furthermore, the rule-based model was unable to visually represent the dynamical system, shown in Fig 5 (a), requiring multiple ad-hoc partial representations. We show how the Causal Petri Net can both represent and answer causal queries over this system.

Problem description This problem follows from the description outlined in [4]. In this setting we have a set of reactions consisting of some input molecules, some output molecules and a probability of occurrence. Fig 5 (a) illustrates the reactions used by the authors.

Query The query Laurent et al.[4] explored was a classic counterfactual query. Given trace: $b, u, pk, b, p, u*$, would we get a bounded and phosphorylated p molecule, either $(pSK, pSKp)$, had pk not occurred.

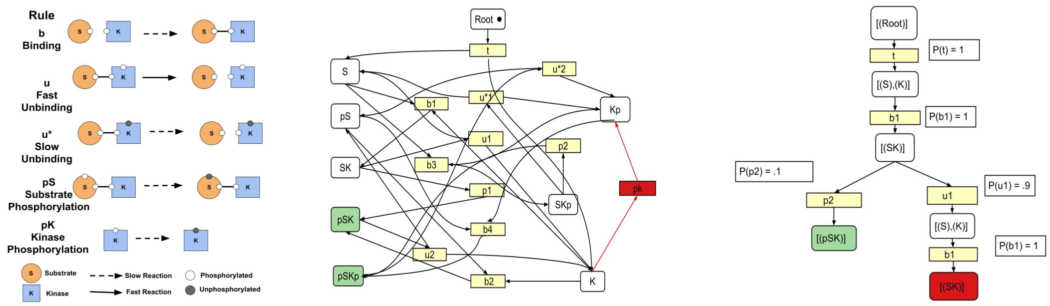
Calculation As we are calculating the counterfactual w.r.t. a single trace we apply our trace algorithm shown in 5. We get precisely the authors findings that the target occurs with very low probability, .1, had pK not occurred.

5 Discussion

5.1 Benefits of the causal Petri Net model

The base model fully and symbolically models the system and does not need a predefined stop point for construction. The constructed Causal Petri Net implicitly encodes all possible traces in our defined dynamical system in a symbolic manner, proven by Lemma 3.1. We can explicitly see this Kamchatka query calculation in Fig 4 (b). where the fact that the captain has no effect on the prisoner is shown through the disconnect from the captain and the place signifying the prisoners death.

Unwrapped models share the causal assumptions of the base model. The power of the unwrapping procedure is that the unwrapped Petri Net only generates traces which the base model can generate. Which means the defined forward simulation algorithm 3 gives us correct probabilities, proven by



(a) (b) (c)

Figure 5: Counterfactual Resimulation case study. (a) Denotes the dynamical system. We have a series of reactions labeled on the left, with their input and output molecules on the right. Dotted arrows indicate a low probability reaction and solid arrows indicate a high probability reaction, specific values aren't specified in the original paper and aren't of fundamental importance. (b) The Causal Petri Net model of the dynamical system. We let a probability mass of $p_{low} = .1$ denote a low probability reaction while a probability of $p_{high} = .9$ a high probability one. The elements highlighted in red denote the intervention of removing pk. The places highlighted green showcase the target molecules. (c) Shows the forward simulation step. We have that the conditioned trace corresponds to the following firing sequence: $m_x = b_1, u_1, pK, b_3, p_2$, as the intervention is removing pk, the marking up to the point of intervention is the marking after u_1 fires: $[(S,1),(K,1)]$. We then forward simulate, where $[(S,1),(K,1)]$ is the initial marking, with event set: $Target := \{e_1 : \{s_{e_1} = [(pSK,1)]\}, e_2 : \{s_{e_2} = [(pSKp,1)]\}\}$. We get the petri net shown, the leaves where the target occurs are highlighted green and the others are highlighted red. We can now calculate: $P(target_{DO(noPK)}|m_x) = .1$. Note we showed the normalized probability of occurance, which we denoted with P and not the probability mass, denoted by Δ, which is why $P(b1) = 1$.

Lemma 3.2. Thus the causal assumptions of the base model are in effect for the unwrapped model. Therefore the unwrapped Petri Nets can be safely used for query calculation and the base model visually represents the actual model in effect.

The unwrapped model makes a distinction between all possible states and states which can occur in a given causal query, making it space efficient Due to the unwrapping procedure outlined in Alg 2, the number of states explicitly unwrapped is dependent on the query. This means the base model can be defined independently of any given query which allows for compact representation. Furthermore traces irrelevant to a query (ones where the target states cannot be reached) do not get unrolled in the unwrapping procedure, saving space. Seen in Fig 5 (c), our counterfactual forward simulation Petri Net requires very few states to be unrolled. Formalizing the extent of this would be of interest in future work.

The causal semantics of a Petri Net do not necessitate simulation. Through the unwrapped Petri Nets we are able to answer causal queries directly on the level of the dynamical systems variables functions, as we were able to calculate the causal query Laurent et al. computed but without the need for simulation (Fig 5). This has clear advantages in the case of working with very low probability traces and conditionals.

5.2 Future work and closing remarks

Laurent et al. [4] stated that ideally there would be a principled approach to gluing together the explanatory accounts of the dynamical system of interest, which would summarize the causal structure of the system. We believe that Petri Nets serve as such a model. There remains many directions for improvement and formalization. Herein we only considered a discrete dynamical system, future work can potentially utilize Colored Petri Nets to extend the domain to continuous systems.

References

[1] T. Blom, S. Bongers, and J. M. Mooij. Beyond structural causal models: Causal constraints models. In *Uncertainty in Artificial Intelligence*, pages 585–594. PMLR, 2020.

[2] T. Genewein, T. McGrath, G. Déletang, V. Mikulik, M. Martic, S. Legg, and P. A. Ortega. Algorithms for causal reasoning in probability trees. *arXiv*, 2020.

[3] M. Hopkins and J. Pearl. Causality and counterfactuals in the situation calculus. *Journal of Logic and Computation*, 17(5):939–953, oct 2007.

[4] J. Laurent, J. Yang, and W. Fontana. Counterfactual resimulation for causal analysis of rule-based models. In *International Joint Conferences on Artificial Intelligence*, 2018.

[5] B. Lopes, M. Benevides, and E. H. Haeusler. Propositional dynamic logic for petri nets. *Logic Journal of IGPL*, 22(5):721–736, oct 2014.

[6] T. Murata. Petri Nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, page 541, 1989.

[7] J. Pearl. Causal and counterfactual inference. *The Handbook of Rationality*, page 427, 2021.

[8] J. Pearl, J. Zucker, C. Cinelli, and C. Huston. Kamchatka twitter thread on deep mind's "algorithms for causal reasoning in probability trees", nov 2020.

[9] S. Peters and J. Y. Halpern. Causal modeling with infinitely many variables. *CoRR*, abs/2112.09171, 2021.

[10] K. Sachs, S. Itani, J. Fitzgerald, B. Schoeberl, G. P. Nolan, and C. J. Tomlin. Single timepoint models of dynamic systems. *Interface Focus*, 3:20130019, 2013.

[11] A. Sackmann, M. Heiner, and I. Koch. Application of Petri Net based analysis techniques to signal transduction pathways. 7:482, 2006.

[12] K. Trares, J. Ackermann, and I. Koch. The canonical and non-canonical NF- pathways and their crosstalk: A comparative study based on Petri Nets. *Bio Systems*, 211:104564, 2022.

[13] J. Vennekens, M. Bruynooghe, and M. Denecker. Embracing events in causal modelling: Interventions and counterfactuals in CP-logic. In T. Janhunen and I. Niemelä, editors, *Logics in artificial intelligence*, volume 6341, page 313. Springer, 2010.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.2.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section A.3.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Classic Firing Squad walkthrough

We present a walkthrough of the Kamchatka Counterfactual Query as done in the paper but in a much more traditional firing squad layout. This is to show that the changes we made (while relatively superficial) were by no means a necessity and that the Petri Net model is flexible enough to handle multiple encodings.

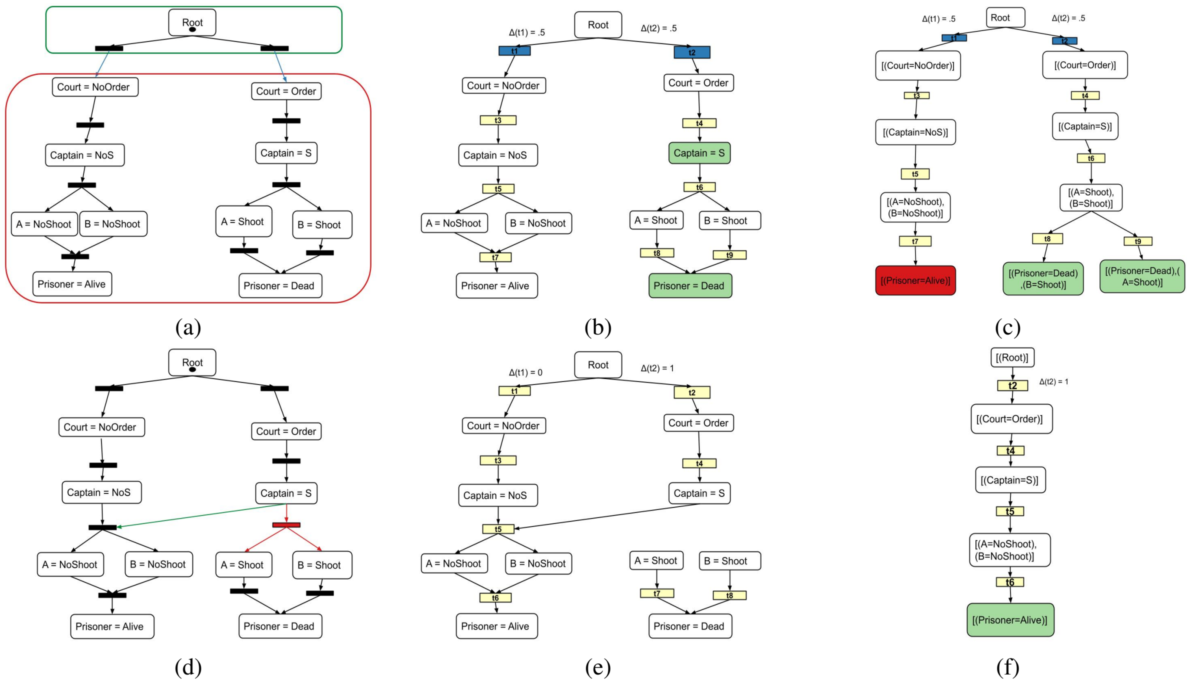


Figure 6: [OV: Rephrase the caption as in Fig 3, to clearly label each subfigure] Kamchatka Counterfactual walkthrough: We have the constructed Petri Net shown in (a), we then perform abduction over the event $E := \{e_1 = [(Captain = S, 1)], e_2 = [(Prisoner = Dead, 1)]\}$ on the model in (b) with the exogenous transition probabilities highlighted to get the unwrapped Petri Net in (c) this allows us to perform Bayes to get the updated probabilities: $\Delta_1 = 0, \Delta_1 = 1$. We now perform the intervention $DO(Riflemen = NoShoot)$ which gives us the Petri Net model in (d) with the deleted elements in red and the added elements in green, we now initialize the updated counterfactual Petri Net model shown in (e) and we perform forward simulation in (f) to get that $P(Prisoner = Alive_{DO(Riflemen=NoShoot)} | (Captain = Signal, Prisoner = Dead)) = 1$ getting the same result as in our paper's variation and showing we aren't in a state of Kamchatka as desired.

A.2 Dynamic model encoding of examples

A.2.1 Classic Firing Squad

Random variables:

$$X_{court} : \{Order, NoOrder\}$$
$$X_{captain} : \{S, NoS\}$$
$$X_A : \{Shoot, NoShoot\}$$
$$X_B : \{Shoot, NoShoot\}$$
$$X_{prisoner} : \{Alive, Dead\}$$

Initial states:

$$(X_{court} = NoOrder)$$
$$(X_{court} = Order)$$

Functions and their corresponding probability mass:

$$f_1((X_{court} = NoOrder)) = (X_{captain} = NoS), \Delta(f_1) = 1$$
$$f_2((X_{court} = Order)) = (X_{captain} = S), \Delta(f_2) = 1$$
$$f_3((X_{captain} = NoS)) = (X_A = NoShoot, X_B = NoShoot), \Delta(f_3) = 1$$
$$f_4((X_{captain} = S)) = (X_A = Shoot, X_B = Shoot), \Delta(f_4) = 1$$
$$f_5((X_A = NoShoot, X_B = NoShoot)) = (X_{prisoner} = Alive), \Delta(f_5) = 1$$
$$f_6((X_A = Shoot)) = (X_{prisoner} = Dead), \Delta(f_6) = 1$$
$$f_7((X_B = Shoot)) = (X_{prisoner} = Dead), \Delta(f_7) = 1$$

A.2.2 Firing Squad

Random variables:

$$X_{court} : \{Order, NoOrder\}$$
$$X_{captain} : \{StandBy, S, NoS\}$$
$$X_A : \{StandBy, Shoot, NoShoot\}$$
$$X_B : \{StandBy, Shoot, NoShoot\}$$
$$X_{prisoner} : \{Alive, Dead\}$$

Initial states:

$$(X_{court} = NoOrder, X_{captain} = Standby, X_A = StandBy, X_B = Standby, X_{prisoner} = Alive)$$
$$(X_{court} = Order, X_{captain} = Standby, X_A = StandBy, X_B = Standby, X_{prisoner} = Alive)$$

Functions and their corresponding probability mass:

$$f_1((X_{court} = NoOrder, X_{captain} = Standby)) = (X_{captain} = NoS), \Delta(f_1) = 1$$
$$f_2((X_{court} = Order, X_{captain} = Standby)) = (X_{captain} = S), \Delta(f_2) = 1$$
$$f_3((X_{captain} = NoS, X_A = Standby)) = (X_{captain} = NoS, X_A = NoShoot), \Delta(f_3) = 1$$
$$f_4((X_{captain} = NoS, X_B = Standby)) = (X_{captain} = NoS, X_B = NoShoot), \Delta(f_4) = 1$$
$$f_5((X_{captain} = S, X_A = Standby)) = (X_{captain} = S, X_A = Shoot), \Delta(f_5) = 1$$
$$f_6((X_{captain} = S, X_B = Standby)) = (X_{captain} = S, X_B = Shoot), \Delta(f_6) = 1$$
$$f_7((X_A = Shoot, X_{prisoner} = Alive)) = (X_A = Shoot, X_{prisoner} = Dead), \Delta(f_7) = 1$$
$$f_8((X_B = Shoot, X_{prisoner} = Alive)) = (X_B = Shoot, X_{prisoner} = Dead), \Delta(f_8) = 1$$

A.2.3 Counterfactual Resimulation

Random variables: We have discrete indicator variables for each molecule $(S, K, pS, Kp, SK, pSK, SKp, pSKp)$

Initial states: (S, K)

Functions and their corresponding probability mass:

$$b1((S, K)) = (SK), \Delta(b1) = p_{low}$$
$$b2((pS, K)) = (pSK), \Delta(b2) = p_{low}$$
$$b3((S, Kp)) = (SKp), \Delta(b3) = p_{low}$$
$$b4((pS, Kp)) = (pSKp), \Delta(b4) = p_{low}$$

$$u1((SK)) = (S, K), \Delta(u1) = p_{high}$$
$$u*1((SKp)) = (S, Kp), \Delta(u*1) = p_{low}$$
$$u2((pSK)) = (pS, K), \Delta(u2) = p_{high}$$
$$u*2((pSKp)) = (pS, Kp), \Delta(u*2) = p_{low}$$

$$p1((S)) = (pS), \Delta(p1) = p_{low}$$
$$p2((SKp)) = (pSKp), \Delta(p2) = p_{low}$$
$$pk((K)) = (Kp), \Delta(pk) = p_{low}$$

A.3 Proofs of included theorems

Lemma A.1. *Every trace t of the system M has a corresponding firing sequence $\{m\}$ in its constructed PN and every firing sequence $\{m\}$ in PN corresponds to a trace in M*

Proof. Let $M := (X, S, S_0, F)$ be a dynamical system and let $PN := (PN_U, PN_M, F_c)$ be the constructed causal Petri Net of M.

Consider a trace $t := [s_0^t, ..., s_k^t]$ of M. We have by construction for every state $s_i^t \in S$ there exists a marking $m_{s_i^t}$ in the marking set of PN. Since $s_0^t \in S_0$ by construction there exists a transition $t_0 \in PN_U$ connecting the root place to the marking $m_{s_0^t}$. For every pair of states s_i^t, s_{i+1}^t in t there must exist a function $f_i \in F$ s.t. $f_i(s_t^t) = s_{i+1}^t$. By construction there must exist a transition $t_i \in PN_M$ connecting the marking $m_{s_i^t}$ to the marking $m_{s_{i+1}^t}$. This means the marking sequence $\{m\} := [m_{s_0^t}, ..., m_{s_k^t}]$ is a valid firing sequence of PN corresponding to the trace t as desired.

Now consider a marking sequence $\{m\} := [m_{s_0^t}, ..., m_{s_k^t}]$ of PN. We have by construction that every marking $m_{s_i^t}$ must have a corresponding state $s_i^t \in S$. We also have that $s_0^t \in S_0$. By definition of a marking sequence for every pair of markings $m_{s_i^t}, m_{s_{i+1}^t}$ there exists a transition t_i in PN s.t. the marking of PN when set or marking $m_{s_i^t}$ will be $m_{s_{i+1}^t}$. By construction there exists a function $f_i \in F$ s.t. $f_i(s_i^t) = s_{i+1}^t$. Thus we have a valid trace $t := [s_0^t, ..., s_k^t]$ of M corresponding to $\{m\}$. □

Lemma A.2. *The probability of event e in M is equal to the probability returned by $FSimPN(PN, e)$*

Proof. We have that the probability that e happens in M is the sum of the probability of all traces where e occurs. Thus if $FSimPN(PN, e)$ considers all the traces we have that it returns the correct result. This is equivalent to the statement that $UnwrapPN(PN, M, e)$ enumerates all the possible traces where e can occur.

We have that $UnwrapPN(PN, M, e)$ considers every possible reachable marking at every step. It only stops a trace if e occurs, if PN reaches a state of deadlock or if PN reaches the same state twice in a trace. We note that if PN reaches deadlock and e hasn't occurred (since if it had we would have stopped earlier) e cannot occur in this trace as no transition (and hence function in M by Lemma 3.1) can be applied. If PN reaches the same state twice the current trace is a loop and since e hasn't occurred it cannot occur. Thus $UnwrapPN(PN, M, e)$ necessarily enumerates all possible traces where e can occur, which means the probability of e in M is equal to the probability returned by $FSimPN(PN, e)$. □

Lemma A.3. *The probability of $P(a|b)$ in M is equal to the probability returned by $ConditionPN(PN, M, a, b)$*

Proof. We have that by 3.2 the probabilities calculated for $P(a), P(b), P(b|a)$ correspond to the probabilities in M. This necessarily means the probability returned by $ConditionPN(PN, M, a, b) = \frac{P(b|a)P(b)}{P(a)}$ is equal to $P(a|b)$ in M from Bayes Theorem. □

A.4 Full-sized images

Fig 3c

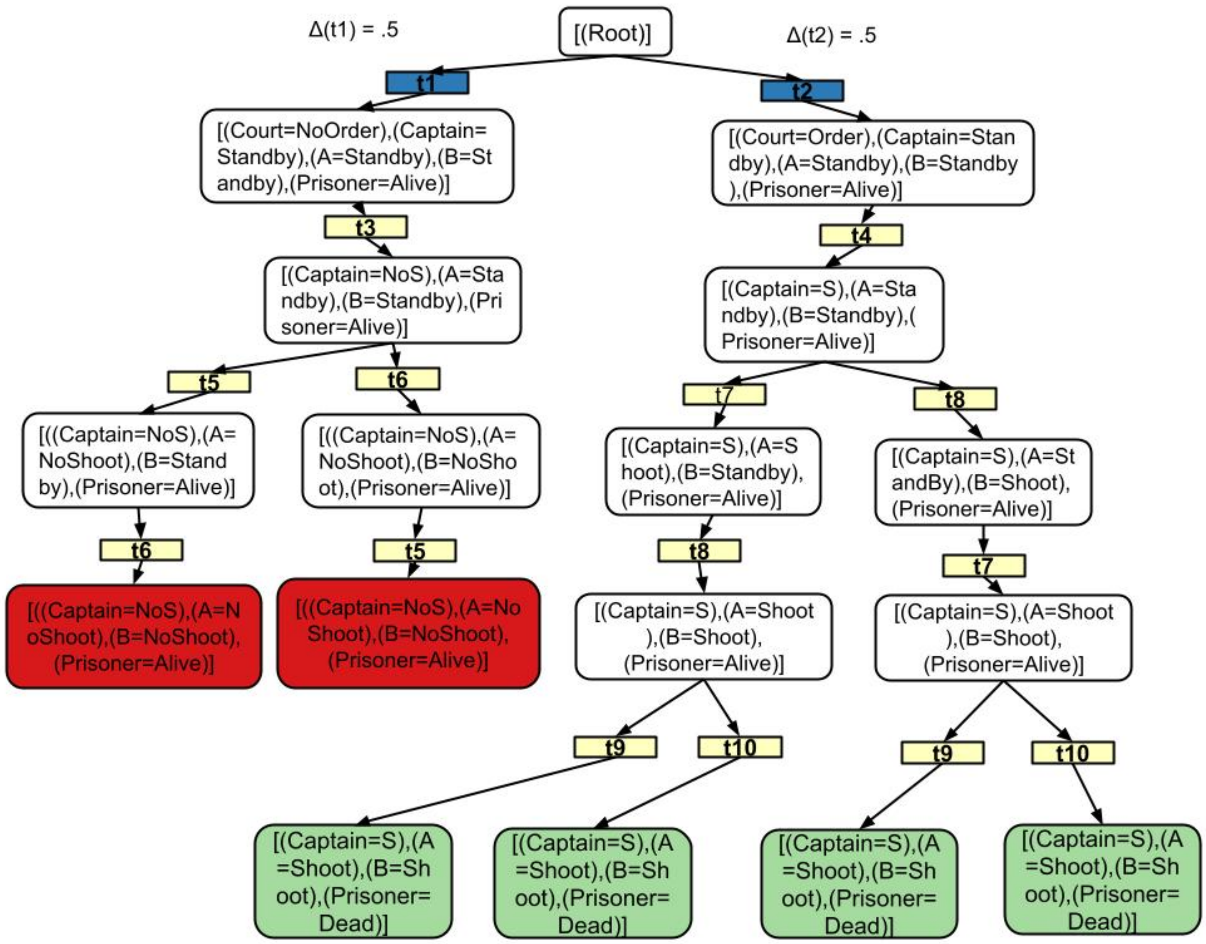


Figure 7: Figure 3c in the main manuscript, with each place fully enumerated.

421 **Fig 4c**

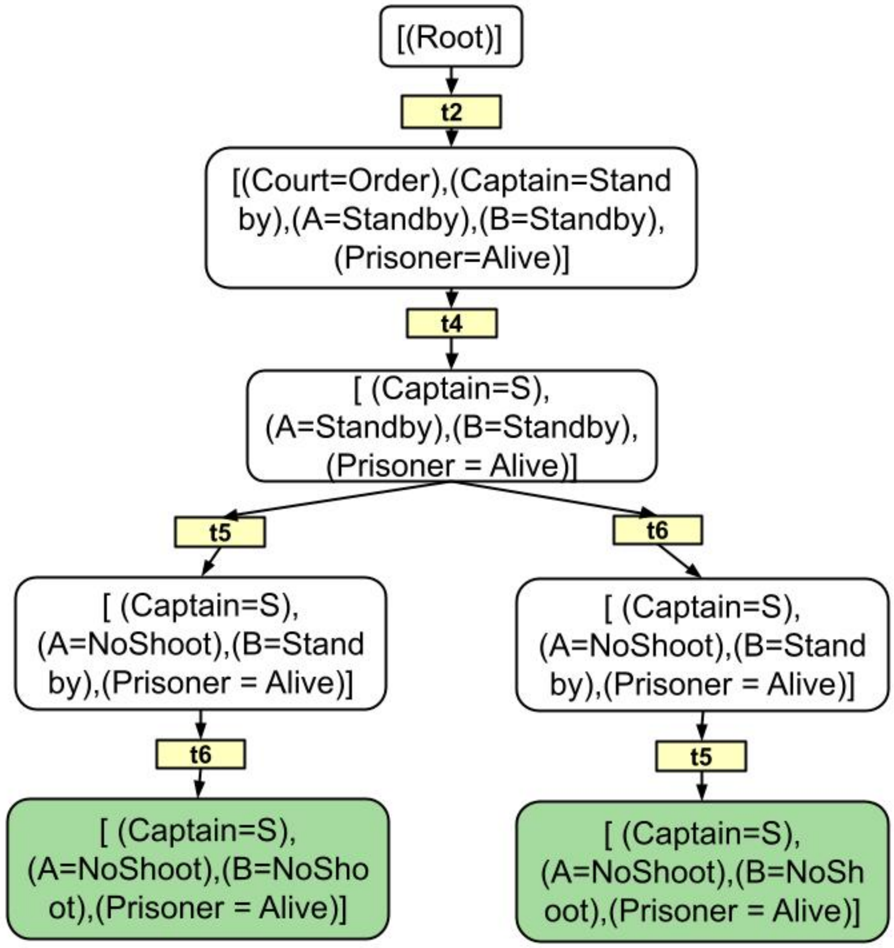


Figure 8: Figure 4c in the main manuscript, with each place fully enumerated.