Root Cause Analysis for Fulfillment Decisions

D. Gueorguiev 8/6/2023

Table of Contents

Root Cause Analysis for Fulfillment Decisions 1
Preliminaries 1
Notation 1
Assumptions 2
Events 3
Directed Follow Graphs 6
Event Relationships and Causality 4
Directed Causal Graphs 11
Problem Statement for Root Cause Analysis of Fulfillment Decisions 12
Algorithm For Root Cause Analysis 12
Appendix: Probabilistic Temporal Logic 12
Bibliography 21
Downloadable Links for the Bibliography 21

Preliminaries

Before we can formulate the problem statement and the algorithm providing a solution, we need to start with a set of notational conventions and definitions.

Notation

$A, B, \dots, Z$ - with capital Latin letters we will denote *scalar quantities* which are either essential algorithm parameters or constants which will not change during the algorithm execution; for example, *number of feasible nodes for the current bundle* (scalar constant) will be denoted with N and *inventory for given SKU on given node* (algorithm parameter) will be denoted with I. Graphs will also be denoted with capital Latin letters for historical reasons.
$a, b, \dots, z$ – with small Latin letters we will denote *variable/unknown (integral or not) quantities*, not necessarily scalar. For example, with x we can denote the number of order-lines fulfilled at a given node.
$\alpha, \beta, \dots, \omega$ – with small Greek letters we will denote *variable/unknown (integral or not) quantities*, not necessarily scalar.
$\mathcal{A}, \mathcal{B}, \dots, \mathcal{Z}$ – with capital Script letters we will denote a *set* (ordered or unordered) of quantities of the same type; for example, with $\mathcal{S}$ we will denote the set of SKUs in some bundle of some order
$\mathrm{A}, \mathrm{B}, \dots, \Omega$ – with capital Greek letters we will denote a *concept*, *logical statement* or a *logical expression* of *logical terms / statements* which is adorned with *semantic meaning*. In case of a logical statement, the latter can be either true or false depending on the context. The capital Epsilon letter E will be reserved to denote an event type or event of interest. For instance, E_0 will denote the event of type "*an order has been received*".
$\mathfrak{A}, \mathfrak{B}, \dots, \mathfrak{Z}$ - with capital Fraktur letters we will denote a *map* over several arguments where at least one of those arguments is of type logical expression, a logical statement or a set of logical statements. For example, $\mathfrak{N}(\Delta, \mathcal{E})$ denotes graph representation of the concept Δ by the set of events $\mathcal{E}$.
$\mathbb{A}, \mathbb{B}, \dots, \mathbb{Z}$ - with double struck Latin capital letters we will denote standard number sets. For example
$\mathbb{C}$ - the set of complex numbers

$\mathbb{N}$ - the set of natural numbers
$\mathbb{R}$ - the set of the real numbers
$\mathbb{Z}$ - the set of integer numbers

Reserved letters for quantities, sets and concepts:
B_t – number of bundles in the order t.
o_t – order received at moment t.
$b_i(o_t)$ – the i-th bundle of order o_t; alternatively, denoted as $b_{i,t}$.
$\mathcal{S}_i(o_t)$ or $\mathcal{S}_{i,t}$ - the set of SKUs for the i-th bundle will be denoted with $\mathcal{S}_i$.
$x_{i,t}$- denotes some quantity x related to the i-th bundle of the t-th order.
$y_{s,j}$ – denotes some quantity y related to the SKU s at node j e.g., inventory for SKU s at node j.

G - directed graph
$\mathcal{V}(G)$ - the vertex set of the directed graph G
$\mathcal{A}(G)$ – the arc set of the directed graph G

$\omega(\mathrm{E}_a|\mathrm{E}_b)|_{\mathcal{D}}$ – denotes the *relative frequency of occurrence* of the event E_a given event E_b with the dataset $\mathcal{D}$
$S(\mathrm{E}_b \leadsto \mathrm{E}_a|\mathcal{K})$ – denotes *Average Degree Of Causal Significance* (*ADCS*) of event E_b for event E_a given the background contexts $\mathcal{K}$

$\mathrm{E}_a \prec \mathrm{E}_b$ denotes the statement that event E_b follows event E_a
$\mathrm{E}_a < \mathrm{E}_b$ denotes the statement that event E_a precedes event E_b

$\mathfrak{R}(\Delta)$ – denotes graph representation of the concept Δ
$\mathfrak{R}(\Delta, \mathcal{E})$ - denotes complete representation of the concept Δ with the event set $\mathcal{E}$
$\mathfrak{S}: \mathcal{E} \times \mathcal{E} \to \{0,1\}$ – denotes static dependency map
$\mathfrak{A}: \mathcal{E} \times \mathcal{E} \to \{0,1\}$ – denotes static association map

Reserved symbols for relations and operations
$\wedge$ - denotes logical conjunction
$\vee$ - denotes logical disjunction
$\neg$ - denotes logical negation
$\prec$ - denotes *follows in time* relation between two concepts
$<$ - denotes *is reachable* relation between two concepts
$\rightarrowtail$ - denotes *static dependency* between two concepts
$\looparrowright$ - denotes *dynamic dependency* between two concepts
$\rightleftarrows$ - denotes *association (static, dynamic)* between two concepts
$\nrightarrow^{\epsilon}$ – denotes *ϵ-spurious cause* relation between two concepts
$\leftrightsquigarrow$ - denotes *causal association* between two concepts
$\rightsquigarrow$ - *prima facie* causal relation between two concepts
$\leadsto$ - denotes Eells causal relation between two concepts

$\lhd$ - denotes *matching* between directed follow graph (DFG) and a concept

Assumptions

All orders can be ordered in an increasing sequence of moments in time $t_1, t_2, \dots, t_o, t_{o+1}, \dots$. That is, we assume that no two orders will arrive at the same moment in time. Thus, the time t will take the form of a discrete variable on the natural numbers i.e., $t \in \mathbb{N}$. Therefore, any order will be uniquely identified by a subscript $t \in \mathbb{N}$.

Definition: *Atomic Proposition*

A basic proposition (or *atom*) which cannot be represented as a set of other atoms connected using conjunction $\wedge$, disjunction $\vee$, negation $\neg$, implication $\therefore$ and equivalence $\Leftrightarrow$.

Events

Definition: *Event*
The word *Event* will be used to denote a *specific kind of* an *event* which is relevant for the causal analysis. *Event* can be viewed as *a template* from which a specific event can be *instantiated*. We will denote each event with capital Greek letter. Where it will be clear from the context, we will use interchangeably the word "*event*" to denote either *Event of specific kind* or an *Event instance*.

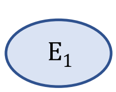
Definition: *Parameters of Event*
Each event has a *set of parameters* which will be denoted with $\mathcal{P}$. The set of parameters $\mathcal{P}$ of an event E together with the semantic description $\mathfrak{S}$ of the event uniquely identify the event. One can think of the semantic description $\mathfrak{S}$ as sort of *"semantic" template* (or *predicate* from some first order logic) identifying this event type. The template parameters will be given obviously with the parameter set $\mathcal{P}$ which is an ordered set. Thus, each event is defined with the pair $(\mathfrak{S}, \mathcal{P})$. An Event Instance additionally to $\mathfrak{S}$ and $\mathcal{P}$ is given a specific value v for each $p \in \mathcal{P}$. We will denote the value space of an Event Instance with $\mathcal{V}$. Thus, an event instance is defined with the triplet $(\mathfrak{S}, \mathcal{P}, \mathcal{V})$.

Note: Every event additionally to its standard parameter set $\mathcal{P}$ will have an implicit timestamp parameter τ which will always be present without regard of the nature of the event. We will not include explicitly the timestamp among the event parameters unless it is necessary in order to define uniquely the event instance.

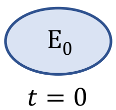
We define the following *Events* which are relevant for the analysis of Fulfillment decisions causing splits:

Set of events for analysis of the cause of splits in Fulfillment Decisions
E_0- order O_t is received. Event parameters: t
E_1- the i-th bundle of order O_t is being processed. Event parameters: t, i
E_2- SKU s in the i-th bundle of order O_t is being processed. Event parameters: t, i, s
E_3- node j has sufficient inventory for SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_4- node j has sufficient capacity for SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_5- node j is shipping eligible for SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_6- node j is deprioritized; node has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_7- node j is turned on; node has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_8- node j is turned off; node has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_9- node j is soft capacity; node has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_{10}- service level sl for node j is overridden; node j has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j, sl
E_{11}- carrier c for node j is overridden; node j has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j, c
E_{12}- backlog days d for node j is overridden; node has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j, d
E_{13}- node j is with depleted inventory; node j has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_{14}- node j is with depleted capacity; node has SKU s in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_{15}- node j contains an orphan for SKU s which is in $\mathcal{S}_i$ with order O_t. Event parameters: t, i, s, j
E_{split} – splitting decision is made; that is, at least for one bundle i the SKUs in $\mathcal{S}_i$ are fulfilled by more than one node with order O_t. Event parameters: $t, B, i_1, i_2, \dots, i_k$

We will visualize an event with an ellipse and a Capital Greek letter denoting the event. For example:



We will visualize an Event instance with an ellipsis and will use a Capital Greek letter to denote the specific kind of event which it is an instance of. We will attach a set of labels where each label will represent an *atomic proposition* with pertinent semantic information for this instance. For example, in case of E_0 we can have:



Thus, the parameter set of E_0 is given with $\mathcal{P}_0 = \{t\}$. Since $t \in \mathbb{N}$ the value space for the parameters of the event instances of E_0 is given with $\mathcal{V}_0 = \mathbb{N}$.

Let us consider the event E_1: $\mathcal{P}_1 = \{t, i\}$. Since $t, i \in \mathbb{N}$ the value space for the parameters of the event instances of E_1 is given with $\mathcal{V}_1 = \mathbb{N} \times \mathbb{N}$.

For E_2 we have accordingly $\mathcal{P}_2 = \{t, i, s\}$ and $\mathcal{V}_2 = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$.

For $\mathrm{E}_3, \dots, \mathrm{E}_9, \mathrm{E}_{13}, \mathrm{E}_{14}$ we have accordingly $\mathcal{P}_m = \{t, i, s, j\}$ and $\mathcal{V}_m = \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$. Here $m \in \{3,4, \dots ,9,13,14\}$

For
//TODO: finish this

Event Relationships and Causality

Definition: Event E_b *follows in time* Event E_a
We say that event E_b *follows in time* E_a (denoted with $\mathrm{E}_a \prec \mathrm{E}_b$) if event E_b has occurred in time *after* event E_a and there does not exist a third event E_c which occurs *after* E_a and *before* E_b. The *follows in time* relation is a strong partial order. The follows in time relation satisfies the conditions:
Follows in time relation implies the timestamp associated with E_b is newer than that associated with E_a i.e. $\tau_b > \tau_a$. See the **Note** on the event parameters.

Definition: Event E_b *is reachable from* Event E_a
We say that event E_b *is reachable from* E_a (denoted with $\mathrm{E}_a < \mathrm{E}_b$) if event E_b has occurred in time *after* event E_a. *Is reachable from* relation implies that the timestamp associated with E_b is newer than that associated with E_a i.e. $\tau_b > \tau_a$. See the **Note** on the event parameters.

For example, let the event E_8 denotes the node j being turned off, and event E_1 denotes order O_t received at time t. Then $\mathrm{E}_8 < \mathrm{E}_0$ can be interpreted as "node j was turned off prior to receiving order O_t".

Lemma: *is reachable from* relation is the transitive closure of the *follows in time* relation.

Definition: Static (*semantic) dependency between events* E_a and E_b
We say that event E_b is static dependent on E_a (denoted with $\mathrm{E}_a \rightarrowtail \mathrm{E}_b$) if each instance of E_b can exists only *in the context* of some instance of E_a for *any* order data set $\mathcal{D}$. That is, removing an instance of E_a in $\mathcal{D}$ will remove all instances of E_b underneath E_a from the event tree for any chosen $\mathcal{D}$.

if there is a static dependency then we can define a map (called *static dependency map*) $\mathfrak{S}: \mathcal{E} \times \mathcal{E} \to \{0,1\}$ such that $\mathfrak{S}(\mathrm{E}_a, \mathrm{E}_b) = 1$ when $\mathrm{E}_a \rightarrowtail \mathrm{E}_b$.

Example of static (semantic) dependency-
Let the event E_0 denote the statement that an order O_t with two bundles was received. Let the event E_1 denotes the statement that the current bundle being processed is the second bundle for order O_t. Then $\mathrm{E}_0 \rightarrowtail \mathrm{E}_1$ for the order O_t.

Example of absence of static (semantic) dependency
Let the event E_0 denote the statement that an order O_t with two bundles was received.
Let the event E_{split} denotes the statement that a splitting decision for both bundles in order O_t is made; that is, the SKUs in both bundles are fulfilled by more than one node with order O_t.
Let the event E_{15} denotes the statement that node j contains an orphan for SKU s which is in $\mathcal{S}_i$ with order O_t.
Then we can write $\mathrm{E}_0 \rightarrowtail \mathrm{E}_{split}$; that is, the splitting decision for order O_t can be reached only after order O_t has been received. However, there is an absence of static dependency between E_{15} and E_{split}. The relevant for $\mathrm{E}_{split}(\mathrm{O}_t)$ instance of event $\mathrm{E}_{15}(t')$ could have been received in an earlier time t' than that of order O_t; that is, $t' < t$. Removing the instance of E_0 corresponding to order O_t has no impact on the existence of $\mathrm{E}_{15}(t')$.

Lemma: Static (semantic) dependency implies *reachable* relation
That is, $\mathrm{E}_a \rightarrowtail \mathrm{E}_b \therefore \mathrm{E}_a < \mathrm{E}_b$. Note that the *reachable* relation between E_a and E_b will hold <u>for all pairs</u> of instances of those events.

Definition: Static (semantic) descendant
We say that the event E_c is a static descendant of another event E_a if there exist a chain of events such that: $\mathrm{E}_a \rightarrowtail \mathrm{E}_{b_1} \rightarrowtail \mathrm{E}_{b_2} \rightarrowtail \cdots \rightarrowtail \mathrm{E}_{b_k} \rightarrowtail \mathrm{E}_c$ for some $k \in \mathbb{N}$ or if $\mathrm{E}_a \rightarrowtail \mathrm{E}_c$.

Lemma: *Static (semantic) dependency* defines a directed follow graph of statically associated events
Refer to the DFG shown for **Example 1** as an illustration.

Definition: *Dynamic dependency* between events E_a and E_b
We say that event E_b is dynamic dependent on E_a (denoted with $\mathrm{E}_a \nrightarrow \mathrm{E}_b$) if all of the following is true:

- $\mathrm{E}_a < \mathrm{E}_b$
- removing an instance of E_a *may* trigger the removal of an event which is static dependent of E_b or removal of E_b itself.

//TODO: Finish this

Example of dynamic dependency-
In the previous Example of absence of static (semantic) dependency we considered three events - E_0 (order has been received), E_{15} (node contains an orphan for SKU in the order), E_{split} (split fulfillment decision has been made). Clearly, there is some kind of dependency between event E_{15} and E_{split} as triggering of event E_{15} at a time t' earlier than the time the split decision is made can potentially impact the E_{split} instance. Clearly, this dependency is not static as event E_{15} at a time t' later than the time the split decision is made hence this is not a static dependency. Thus, the relation between E_{15} and E_{split} matches the definition of dynamic dependency.

Definition: Event E_b is *associated (statically, dynamically*) with Event E_a
We say that event E_b is *associated* (*statically*) with E_a (denoted with $\mathrm{E}_a \rightleftarrows \mathrm{E}_b$) if each instance of E_b is (static, dynamic) descendant of some instance of E_a or vice versa. That is, if the
//TODO: Finish this

In order to find how a set of events are associated statically we can define a map (called *static association*) $\mathfrak{A}: \mathcal{E} \times \mathcal{E} \rightarrow \{0,1\}$ such that $\mathfrak{A}(\mathrm{E}_a, \mathrm{E}_b) = 1$ when $\mathrm{E}_a \rightleftarrows \mathrm{E}_b$.

Discussion on static association map

How can we define the *static association* map $\mathfrak{A}$? The answer can be found in the definition of *Parameters of Event* given earlier. We have the parameter spaces of the two events - $\mathcal{P}_a$ and $\mathcal{P}_b$ and the value sets $\mathcal{V}_a$ and $\mathcal{V}_b$ of the corresponding event instances. Note that $\mathcal{P}_a = \{p_a(1), p_a(2), \dots, p_a(P_a)\}$ where $P_a = |\mathcal{P}_a|$. Here $\mathcal{V}_a$ denotes the cartesian product of the value sets for each parameter $p \in \mathcal{P}_a$ of event E_a; thus, we have:

$$\mathcal{V}_a = \mathcal{V}_a(1) \times \mathcal{V}_a(2) \times \cdots \times \mathcal{V}_a(P_a).$$

In general, the map $\mathfrak{A}$ should be defined over the cartesian product of the event tuples $(\mathfrak{S}_a, \mathcal{P}_a, \mathcal{V}_a) \times (\mathfrak{S}_b, \mathcal{P}_b, \mathcal{V}_b)$.
Let us consider this question from the context of our Fulfillment Decision **Example 1**.
Clearly, we expect that the E_0 instance and all E_1 instances under the parent E_0 instance are statically associated. We expect that each E_1 instance and all E_2 instances which are children of the current E_1 instance are statically associated as well. Let us denote the E_0 instance in this example with $\mathrm{E}_0|_{t=0}$. Let us denote the three instances of E_1 with $\mathrm{E}_1|_{t=0,i=1}$, $\mathrm{E}_1|_{t=0,i=2}$, and $\mathrm{E}_1|_{t=0,i=3}$. Then we obviously we have:

$$\begin{cases} \mathrm{E}_0|_{t=0} \rightleftarrows \mathrm{E}_1|_{t=0,i=1} \\ \mathrm{E}_0|_{t=0} \rightleftarrows \mathrm{E}_1|_{t=0,i=2} \\ \mathrm{E}_0|_{t=0} \rightleftarrows \mathrm{E}_1|_{t=0,i=3} \end{cases} \quad (1)$$

The relation (1) represents the fact that each child is statically associated to its parent.
Additionally, we can write:

$$\begin{cases} \mathrm{E}_1|_{t=0,i=1} \rightleftarrows \mathrm{E}_1|_{t=0,i=2} \\ \mathrm{E}_1|_{t=0,i=1} \rightleftarrows \mathrm{E}_1|_{t=0,i=3} \\ \mathrm{E}_1|_{t=0,i=2} \rightleftarrows \mathrm{E}_1|_{t=0,i=3} \end{cases} \quad (2)$$

The relation (2) represents the fact that the children of the same parent are statically associated.

Similarly, we continue with writing the static association relations involving the instances of E_2

$$\begin{cases} \mathrm{E}_1|_{t=0,i=1} \rightleftarrows \mathrm{E}_2|_{t=0,i=1,s=1} \\ \mathrm{E}_1|_{t=0,i=1} \rightleftarrows \mathrm{E}_2|_{t=0,i=1,s=2} \end{cases} \quad \begin{cases} \mathrm{E}_1|_{t=0,i=2} \rightleftarrows \mathrm{E}_2|_{t=0,i=2,s=1} \\ \mathrm{E}_1|_{t=0,i=2} \rightleftarrows \mathrm{E}_2|_{t=0,i=2,s=2} \\ \mathrm{E}_1|_{t=0,i=2} \rightleftarrows \mathrm{E}_2|_{t=0,i=2,s=3} \end{cases} \quad \begin{cases} \mathrm{E}_1|_{t=0,i=3} \rightleftarrows \mathrm{E}_2|_{t=0,i=3,s=1} \\ \mathrm{E}_1|_{t=0,i=3} \rightleftarrows \mathrm{E}_2|_{t=0,i=3,s=2} \\ \mathrm{E}_1|_{t=0,i=3} \rightleftarrows \mathrm{E}_2|_{t=0,i=3,s=3} \\ \mathrm{E}_1|_{t=0,i=3} \rightleftarrows \mathrm{E}_2|_{t=0,i=3,s=4} \end{cases} \quad (3)$$

Additionally, all E_2 instances of the same parent are statically associated with each other - we write this as:

$\mathrm{E}_2|_{t=0,i=m,s=p} \rightleftarrows \mathrm{E}_2|_{t=0,i=m,s=q}$ $\forall m = 1,2,3$ and $\forall p, q \in \mathcal{I}(\mathcal{S}_m)$ (4)
Here $\mathcal{I}(\mathcal{S}_m)$ denotes the index set of $\mathcal{S}_m$.

Also, all E_2 instances are associated with their grandparent $\mathrm{E}_0|_{t=0}$ which is expressed with:

$\mathrm{E}_0|_{t=0} \rightleftarrows \mathrm{E}_1|_{t=0,i=m,s=p}$ $\forall m = 1,2,3$ and $\forall p, q \in \mathcal{I}(\mathcal{S}_m)$ (5)

Let us construct the map $\mathfrak{A}$ for the sets (1)-(5). We start with (1):
//TODO: finish this

Directed Follow Graphs

We use *Directed Follow Graphs* (DFG) to depict order fulfillment scenarios which we are interested to capture.

Each node of the DFG will represent an *Event instance* of interest. We use an arc (or a directed edge) to denote a *follow in time* relation between two Event instances. Each arc has a label with a counter which counts how many times the current arc connecting a pair of events has been seen in the data log given specific dataset.

Definition: *Labeled Directed Follow Graph (LDFG)*
We extend the concept of Directed Follow Graph (DFG) by introducing a set of labels to each node and to each arc. Each label represents an atomic proposition which is relevant to the specific node or to specific arc.
We use LDFGs to represent the follow relationships between event types and event instances for a given dataset of orders.

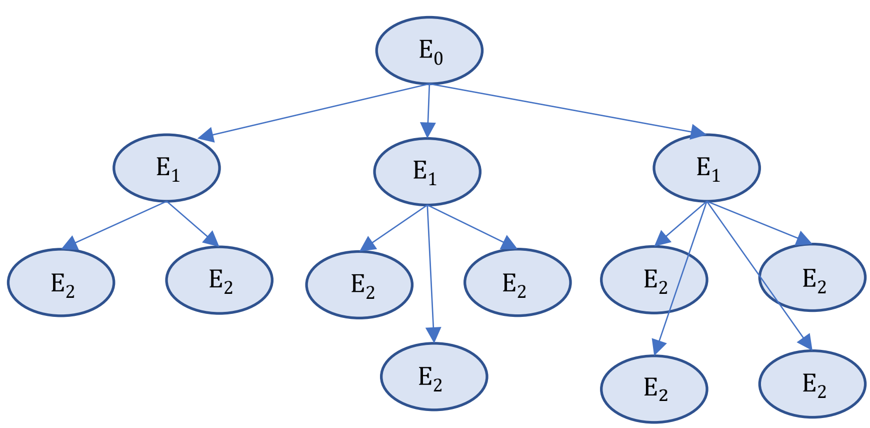
Discussion on how DFG is constructed

Let us consider a given order data set $\mathcal{D}$ and let us assume we have Fulfillment Optimization engine processing the set of orders $\mathcal{D}$ sequentially thereby generating order metrics events. Let us assume we have a parsing engine which combs through the order metrics created after the Fulfillment Optimization engine run. This parsing engine parses the events which it is configured to recognize and assembles the DFG instance based on the parsed events data. Let us denote with $\mathrm{E}_l, l = 1..M$ the events which the parsing engine is configured to recognize. Per our definition of *Parameters of Event* given earlier each event type E_l is represented by the pair $(\mathfrak{S}_l, \mathcal{P}_l)$ where $\mathfrak{S}_l$ is the template of the event which together with the parameter set $\mathcal{P}_l$ uniquely identifies this type of event. Let us denote with $\mathcal{V}_l$ the ordered set of values which correspond to each parameter $p_l \in \mathcal{P}_l$ for all instances of E_l generated using $\mathcal{D}$. Since each event instance has a timestamp, we can construct DFG from the parsed events. Each arc between two event types E_a and E_b will be labeled with the final count showing how many times this pair of events have been seen in a follow relation $\mathrm{E}_a \prec \mathrm{E}_b$.

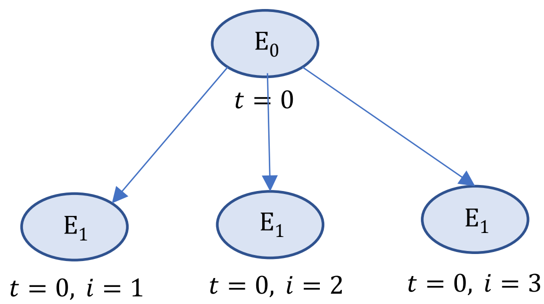
For example:

Example 1:

An order o_t for $t = 0$ has 3 bundles. The first bundle has three SKUs – s_1, s_2, s_3, the second bundle has two SKUs – s_4 and s_5, the third bundle has 4 SKUs – s_6, s_7, s_8, s_9.
Let us denote with E_0 the Event that an order with 3 bundles have been received. Also, we will denote with an instance of event type E_1 each of the bundles of the order o_t. We denote with an instance of event type E_2 each of the SKUs in each bundle of the order O_t
We depict this scenario with the following DFG:



Definition: *Directed Follow Graph Instance (DFGI)*
DFGI is a directed graph G_i in which each node is a specific *event instance (or a token)* of an event type and each arc denotes a *follow in time relation* $\prec$ between the event instances. Each node (event instance/token) is labeled with the value set of parameters for this event type. For example:



Definition: *Aggregated Directed Follow Graph* (*ADFG*)
The ADFG G corresponding to DFGI G_i can be obtained by replacing each event instance by its corresponding event type and replacing a multi-set of arcs leaving an event instance of type E_a and entering event instance of type E_b with a single arc labeled with the corresponding instance count.

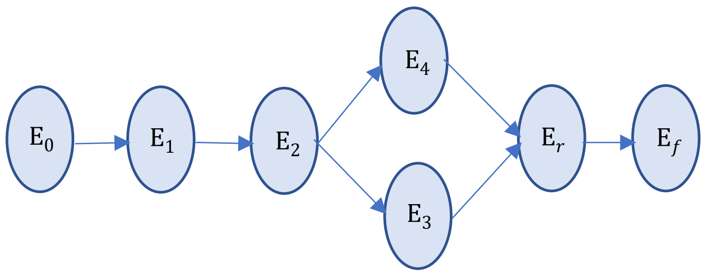
Definition: *Frequency Count* $f(\mathrm{E}_a, \mathrm{E}_b)$ of pair of events E_a and E_b – this is the number f of DFG instances D_i in which E_b directly follows E_a i.e., $\mathrm{E}_a \prec \mathrm{E}_b$.

Definition: *DFG Representation* of a concept Δ over an event set $\mathcal{E}$
We say that DFG is a representation of Δ if the graph constructed with the events in $\mathcal{E}$ models *semantically* the internal structure of Δ.
For example, the DFG shown in *Example 1* is DFG representation of the order O_t. The DFG G representing O_t will be denoted with $G = \mathfrak{R}(\mathrm{O}_t)$ or shortly $G(\mathrm{O}_t)$.

Definition: *Complete Representation* of a concept Δ over an event set $\mathcal{E}$
We say that the DFG G is a *complete representation* of the concept Δ (e.g., fulfillment order, fulfillment decision) from the event set $\mathcal{E}$, denoted with $G = \mathfrak{R}(D, \mathcal{E})$, *iff* there does not exist DFG G_1such that $G_1 = \mathfrak{R}(D, \mathcal{E})$ with $\mathcal{V}(G) \subset \mathcal{V}(G_1) \subseteq \mathcal{E}$ and $\mathcal{A}(G) \subseteq \mathcal{A}(G_1)$.

Definition: *Order Fulfillment Decision*
The process of fulfilling the order which can be viewed as a set of events relevant to the decision which was made. The events are pairwise related by the *follow in time* ($\prec$) relation. The events are represented by DFG over some set of events $\mathcal{E}$.
For example:

Example 2
An order O_t with a single bundle b_1 and single SKU s_1 with unit quantity has been received. Let us define the following event set $\mathcal{E} = \{\mathrm{E}_0, \mathrm{E}_1, \mathrm{E}_2, \mathrm{E}_3, \mathrm{E}_4, \mathrm{E}_r, \mathrm{E}_f\}$: The event "order O_t has been received" will be denoted with E_0. The event "The order bundle is being processed" will be denoted with E_1. The event "SKU s_1 is being processed" will be denoted with E_2. The event "node n_j has inventory for SKU s_1" will be denoted with E_3. The event "node n_j has capacity for SKU s_1" will be denoted with E_4. The event "Reward for node n_j has been calculated" will be denoted with E_r. The event "Fulfilling node has been chosen" will be denoted with E_f.
This is visualized as:



Definition: *DFG Matching* of an order fulfillment decision
Let Δ_t denotes the fulfillment decision of order O_t. Let G denotes some DFG. We say that the DFG G *matches* Δ_t (denoted with $G \lhd \Delta_t$) if G is a representation of Δ_t over some set of events $\mathcal{E}$.
//TODO: Finish this

Causal association between events

What does it mean that certain event types can be associated causally with each other? Let us consider two event types - E_a and E_b. Per our definition of *Parameters of Event* given earlier the event E_a is characterized with the pair $(\mathfrak{S}_a, \mathcal{P}_a)$ where $\mathfrak{S}_a$ is the template of the event which together with the parameter set $\mathcal{P}_a$ uniquely identifies this type of event. Similarly, we will consider another event type E_b represented by $(\mathfrak{S}_b, \mathcal{P}_b)$. Now let us consider a given order data set $\mathcal{D}$ and let us assume we have Fulfillment Optimization engine processing the set of orders $\mathcal{D}$ sequentially thereby generating order metrics events. Let us denote with $\mathcal{V}_a$ the ordered set of values which correspond to each parameter $p_a \in \mathcal{P}_a$ for all instances of E_a generated using $\mathcal{D}$. Similarly, with $\mathcal{V}_b$ we denote the ordered set of values which correspond to each E_b parameter and generated for all instances of E_a using order data set $\mathcal{D}$. For an instance of E_a we will denote the values of the instance parameters with v_a. Thus, for each instance of E_a in $\mathcal{D}$ (denoted as $E_a|_{\mathcal{D}}$) we have $v_a \in \mathcal{V}_a$. Similarly, for $E_b|_{\mathcal{D}}$ we have $v_b \in \mathcal{V}_b$.

Definition: *Causal association between events* E_a and E_b – Given the dataset $\mathcal{D}$ we say that E_a and E_b are *associated (causally)* if one of the following is true:

- both E_a and E_b are *causes* of another event E_c in $\mathcal{D}$ //do we need this?
- both E_a and E_b are *caused* by another event E_c in $\mathcal{D}$ //do we need this?
- either E_a *causes* E_b or E_b *causes* E_a
- there is a semantic association between E_a and E_b

Note: we denote causal association between the events E_a and E_b with the symbol $\leftrightsquigarrow$ i.e. $E_a \leftrightsquigarrow E_b$.
For example, a *prima facie causal association* implies that all causal relationships in its definition are *prima facie causes* (defined in the paragraph below).

Definition: *Conditional probability of an event*
Let us consider the event type E_a. Per our definition of *Parameters of Event* given earlier the event E_a is characterized with the pair $(\mathfrak{S}_a, \mathcal{P}_a)$ where $\mathfrak{S}_a$ is the template of the event which together with the parameter set $\mathcal{P}_a$ uniquely identifies this type of event. Similarly, we will consider another event type E_b represented by $(\mathfrak{S}_b, \mathcal{P}_b)$. Now let us consider a given order data set $\mathcal{D}$ and let us assume we have Fulfillment Optimization engine processing the set of orders $\mathcal{D}$ sequentially thereby generating order metrics events.
Let us run the Fulfillment engine with the given order set $\mathcal{D}$ and we find that in A out of the N instances in which event E_a has occurred there has been an instance of E_b *associated with* each instance of E_a.
Then given the data set $\mathcal{D}$ the relative frequency of occurrences of E_a given E_b is obtained as:

$\omega(\mathrm{E}_a|\mathrm{E}_b)|_{\mathcal{D}} = \frac{A}{N}$ (6)

We say that the relative frequency given $\mathcal{D}$ is an estimate for the conditional probability $P(\mathrm{E}_b|\mathrm{E}_a)$ i.e.

$P(\mathrm{E}_b|\mathrm{E}_a)|_{\mathcal{D}} \sim \omega(\mathrm{E}_a|\mathrm{E}_b)|_{\mathcal{D}}$ (7)

Definition: Event E_a is *prima facie cause* of Event E_b
Given the data set $\mathcal{D}$ let us denote with $\mathrm{E}_b|_{\mathcal{D}}$ the set of instances of E_b which follow the set of instances of E_a, denoted with $\mathrm{E}_a|_{\mathcal{D}}$. That is, $\forall\ \mathrm{E}_b|_{\mathcal{D}} \exists\ \mathrm{E}_a|_{\mathcal{D}} s.t. \mathrm{E}_a|_{\mathcal{D}} < \mathrm{E}_b|_{\mathcal{D}}$.
We say that event E_a is a *prima facie cause* of event E_b (denoted with $\mathrm{E}_a \rightsquigarrow \mathrm{E}_b$) *iff*:
the sets $\mathrm{E}_a|_{\mathcal{D}}$ and $\mathrm{E}_b|_{\mathcal{D}}$ are non-empty
and
$P(\mathrm{E}_b|\mathrm{E}_a)|_{\mathcal{D}} > P(\mathrm{E}_b)|_{\mathcal{D}}$ (8)

Lemma: *Prima facie* cause between event E_a and event E_b implies dynamic dependency between the two events
That is, $\mathrm{E}_a \rightsquigarrow \mathrm{E}_b \therefore \mathrm{E}_a \nleftrightarrow \mathrm{E}_b$.

Definition: Event E_b is *ϵ-spurious cause* of an Event E_a
Let us consider the event type E_a given with its template $\mathfrak{S}_a$ and parameter space $\mathcal{P}_a$.
Let us consider another event type E_b given with its template $\mathfrak{S}_b$ and parameter space $\mathcal{P}_b$.
Given the data set $\mathcal{D}$ we denote with $\mathcal{E}_c$ the set of all events with which E_a is associated such that $\mathrm{E}_c|_{\mathcal{E}_c(\mathcal{D})} < \mathrm{E}_b|_{\mathcal{D}}$.

Let E_b is an event such that:

- it is not necessarily in $\mathcal{E}_c$: $\mathrm{E}_b \notin \mathcal{E}_c$
- E_a is reachable from E_b i.e. $\mathrm{E}_b|_{\mathcal{D}} < \mathrm{E}_a|_{\mathcal{D}}$

Then we say that E_b is ϵ-spurious cause of an Event E_a *iff*

- $P(\mathrm{E}_b \wedge \mathrm{E}_c)|_{\mathcal{D}} > 0$
- $|P(\mathrm{E}_a|\mathrm{E}_b \wedge \mathrm{E}_c) - P(\mathrm{E}_a|\mathrm{E}_c)| < \epsilon$ over $\mathcal{D}$
- $P(\mathrm{E}_a|\mathrm{E}_b \wedge \mathrm{E}_c) \geq P(\mathrm{E}_a|\mathrm{E}_c)$ over $\mathcal{D}$

We denote ϵ-spurious cause with $\mathrm{E}_c \nrightarrow^{\epsilon} \mathrm{E}_a$

Definition: Event E_b is *Suppe's* cause of an Event E_a (a.k.a. *Suppe's causality*)
We define $\mathcal{E}_c$ and E_b as in the definition of *ϵ-spurious cause*.
Given the data set $\mathcal{D}$ we denote with $\mathcal{E}_c$ the set of all events with which E_a is associated such that $\mathrm{E}_c|_{\mathcal{E}_c(\mathcal{D})} < \mathrm{E}_b|_{\mathcal{D}}$.
Let E_b is an event such that:

- it is not necessarily in $\mathcal{E}_c$: $\mathrm{E}_b \notin \mathcal{E}_c$
- E_a follows E_b i.e., $\mathrm{E}_b|_{\mathcal{D}} < \mathrm{E}_a|_{\mathcal{D}}$

Then we say that E_b Suppe's cause of an Event E_a *iff*

- $P(\mathrm{E}_b \wedge \mathrm{E}_c)|_{\mathcal{D}} > 0$
- $P(\mathrm{E}_a|\mathrm{E}_b \wedge \mathrm{E}_c) > P(\mathrm{E}_a|\mathrm{E}_c)$ over $\mathcal{D}$ (*Suppe*'s causal relation hypothesis)

We denote *Suppe*'s causality relation with $\mathrm{E}_b \rightsquigarrow \mathrm{E}_a$

Definition: Event E_b is *Eells* cause of an Event E_a (a.k.a. *Eells' causality*)
Let us consider the event type E_a given with its template $\mathfrak{S}_a$ and parameter space $\mathcal{P}_a$.
Let us consider another event type E_b given with its template $\mathfrak{S}_b$ and parameter space $\mathcal{P}_b$.
Given the data set $\mathcal{D}$ we denote with $\mathcal{E}_c$ the set of all events with which E_a is causally associated such that $\mathrm{E}_c|_{\mathcal{E}_c(\mathcal{D})} < \mathrm{E}_a|_{\mathcal{D}}$.
Additionally, we define the following causal *background contexts* $\mathcal{K} = \{K_1, K_2, \dots, K_n\}$. Those are formed by holding fixed the set of all factors causally associated with E_a. For instance, given a set of three associated with E_a events $\{\mathrm{E}_{c,1}, \mathrm{E}_{c,2}, \mathrm{E}_{c,3}\}$ one possible background context will be $\{\mathrm{E}_{c,1}, \neg\mathrm{E}_{c,2}, \mathrm{E}_{c,3}\}$

Let E_b is an event such that:

- it is not necessarily in $\mathcal{E}_c$: $\mathrm{E}_b \notin \mathcal{E}_c$
- E_a is reachable from E_b i.e., $\mathrm{E}_b < \mathrm{E}_a$

Then we say that E_a is *Eells*-caused by Event E_b *iff*

$P(\mathrm{E}_a|K_i \wedge \mathrm{E}_b) \neq P(\mathrm{E}_a|K_i \wedge \neg\mathrm{E}_b)\ \forall\ K_i \in \mathcal{K}\ s.t. K_i < \mathrm{E}_a$ over $\mathcal{D}$ ( )

We denote *Eells*-causality with $\mathrm{E}_b \rightsquigarrow \mathrm{E}_a$

Definition: *Average Degree Of Causal Significance* (*ADCS*) of event E_b for event E_a in given context
The *Average Degree Of Causal Significance* (*ADCS*) of event E_b for event E_a given the background contexts $\mathcal{K}$ is defined as:

$$S(\mathrm{E}_b \rightsquigarrow \mathrm{E}_a|\mathcal{K}) = \sum_{i=1}^{n} P(K_i)[P(\mathrm{E}_a|K_i \wedge \mathrm{E}_b) - P(\mathrm{E}_a|K_i \wedge \neg\mathrm{E}_b)]$$

We use the Latin capital letter S to denote *ADCS* from the Lat. *significatio* (significance).

Lemma:
Static dependency implies *Eells* causality. However, *Eells* causality does not imply static dependency.
That is, if $\mathrm{E}_a \rightarrowtail \mathrm{E}_b$ then it is true $\mathrm{E}_a \rightsquigarrow \mathrm{E}_b$. However, if $\mathrm{E}_a \rightsquigarrow \mathrm{E}_b$ it does not necessarily follow that $\mathrm{E}_a \rightarrowtail \mathrm{E}_b$.

Example of *Eells*-causality-

Let the event E_0 denote the statement that an order O_t with one bundle was received. Let the event E_1 denotes the statement that the capacity feasible nodes for order O_t are node i and node j.
//TODO: finish this

Note: the *caused by* $\leadsto, \rightsquigarrow, \twoheadrightarrow, \rightarrowtail$ relations do **not** impose a total order; that is, for **every** pair of events E_a and E_b it does **not** follow that either $\mathrm{E}_a \leadsto \mathrm{E}_b$ or $\mathrm{E}_b \leadsto \mathrm{E}_a$ is true. Therefore, a set of events cannot be visualized as an ordered sequence; instead, we will use *Directed Causal Graph* for the purpose.

Directed Causal Graphs

Definition: *Directed Causal Graph (DSG)*
A directed graph in which each node represents an Event Type, and each arc represents causal relation. Each arc is labeled with causal significant factor, a real number between 0 and 1, describing how significant is the causal relationship between the two event types.

Problem Statement for Root Cause Analysis of Fulfillment Decisions

The goal of the RCA algorithm applied to Fulfillment Optimization events is to understand and analyze causal relationship between predefined set of events based on the order metrics payloads. Each detected causal relationship will be assigned a significance factor which will indicate based on the supplied dataset how significant was this causal relationship inferred from the dataset and the configured set of events $\mathcal{E}$. Thus, the result of a single RCA algorithm run with a given dataset $\mathcal{D}$ will be a Directed Causal Graph instance G, where the vertex set will be a subset of the events set i.e., $\mathcal{V}(G) \subseteq \mathcal{E}$. Each arc will represent a causal relationship between the connected events, and it will be labeled with a causal significance factor $S(\mathrm{E}_a, \mathrm{E}_b) \in [0,1]$ (abbrev. $S_{a,b}$).
For instance, for the set of events shown earlier (see *Set of events for analysis of the cause of splits in Fulfillment Decisions*) we can have the following output of the RCA algorithm:

//TODO: finish this

Algorithm For Root Cause Analysis

Brief description of the RCA algorithm

1. Choose a set $\mathcal{E}$ of events of interest. $\mathrm{E}_\alpha \in \mathcal{E}, \alpha \in \mathcal{I}$
2. Compile order sequence $o_1, o_2, \dots, o_t, \dots$ from the given events dataset $\mathcal{D}$
3. Using the given dataset $\mathcal{D}$ create Directed Follow Graph instances G_t for each order o_t in the dataset.
4. From the created $G_t, t = 1,2, \dots$ construct Aggregated Directed Follow Graph G with the set of events of interest $\mathrm{E}_\alpha \in \mathcal{E}, \alpha \in \mathcal{I}$
5.
6. Using Eell's definition of causality calculate the Average Degree of Causal Significance (ADCS) $S_{a,b}$ for each pair of nodes in G using the already calculated in 3. frequency counts $f_{a,b}$ for each pair of events $(\mathrm{E}_a, \mathrm{E}_b)$ in G.
7. Given a minimum significance level S_{min} construct Directed Causal Graph (DCG) G_C using G and $S_{a,b}$ for each pair of events in $\mathcal{V}(G)$ such that every arc $a - b$ in G_C will have significance factor $S_{a,b}$ larger or equal to S_{min}.

//TODO: finish this

Appendix A: Probabilistic Temporal Logic

Definitions and Review on Probabilistic Temporal Logic in (Kleinberg, Causality, Probability, and Time, 2012)

Probabilistic Temporal Logic (PTL) is a tool for state machine model checking which is a more complete alternative of the Labeled DFG defined earlier. A somewhat reduced subset of Probabilistic Temporal Logic is defined with the help of *Kripke* structures. With randomness introduced *Kripke* structure is roughly equivalent to a Discrete Time Markov Chain, and it is just another tool to validate specific first order logic statements relevant for RCA against our process model.

Definition A1: *Kripke structure*

Let $\mathcal{P}$ be a set of atomic propositions. A *Kripke* structure M over $\mathcal{P}$ is defined as the tuple $M = \{\mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathfrak{L}\}$ where

- $\mathcal{S}$ is a finite set of states
- $\mathcal{S}_0 \subseteq \mathcal{S}$ is the set of initial states
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a total transition relation, such that $\forall s \in \mathcal{S}, \exists\, s' \in \mathcal{S}\ s.t. (s, s') \in \mathcal{R}$
- $\mathfrak{L} : \mathcal{S} \to 2^{|\mathcal{P}|}$ is a function which labels each state with a set of atomic propositions that are true within it.

The function (relation) $\mathcal{R}$ being a total transition function (relation) means that for every state, there is at least one transition from that state (to itself or to another state). The function (relation) $\mathfrak{L}$ maps states to the truth values of propositions at that state. Since there are $|\mathcal{P}|$ propositions, there are $2^{|\mathcal{P}|}$ possible truth values and $\mathfrak{L}$ maps each state to one of these.
A *path* in a *Kripke* structure is an infinite sequence of states. Precisely, a path π is a sequence of states ($\pi = s_0, s_1, \ldots$) such that for every $i \geq 0$, $(s_i, s_{i+1}) \in \mathcal{R}$. That says that the series of transitions described in the sequence π is possible. The notation π^i is used to denote the *subpath suffix* of the path π starting with state s_i.
To find the properties that are true in such kind of structures we need a formal method for representing the properties to be tested. There are number of temporal logic systems which express (slightly) different sets of formula. We are going to introduce *Computational Tree Logic* (CTL) system which will be used to build upon later and define PTL.
The formulas in CTL are composed of paired *path quantifiers* and *temporal operators*. Path quantifiers describe whether a property holds ***for all paths*** (denoted with the operator A), or ***for some path*** (denoted with operator E), starting at a given state. The temporal operators describe where along the path the properties will hold. For example, if f is some state, then AGf is a valid CTL formula, but $AGFf$ is not, since F is not paired with one of A or E. More formally,

- *Finally* (F) – at some state on the path the property will hold
- *Globally* (G) – the property will hold along the entire path
- *Next* (X) – the property will hold at the next state of the path
- *Until* (U) – applies to two properties, the first one holds in every state along the path until at some state the second property holds
- *Weak Until aka Until or Release* (W)

//Finish this paragraph on CTL

As in CTL, in PTL there are two types of formulas: *path formulas* and *state formulas*. State formulas express properties that must hold within a state, such as being labeled with certain atomic propositions, while path formulas refer to sequences of states along which the formula must hold. The formulas are comprised of atomic propositions $a \in \mathcal{P}$, propositional logical connectives (such as $\neg$, $\vee$, $\wedge$), and the modal operators denoting time and probability. The logic syntax tells how valid PTL formulas are constructed:

1. All atomic propositions are state formulas
2. If $\neg f$ and g are state formulas, so are f, $f \wedge g$, $f \vee g$, and

Examples of PTL

$\mathrm{E}_a <_{\geq p}^{\geq r, \leq s} \mathrm{E}_b$: Event E_b is reachable from event E_a with probability at least p after at least r steps and at most s steps

//Finish the paragraph on PTL Examples

Definition A2: *prima facie* cause expressed with PTL formulas

1. $F_{>0}^{\leq\infty} c$
2. $c <_{\geq p}^{\geq 1, \leq\infty} e$
3. $F_{<p}^{\leq\infty} e$

These conditions mean that 1) a state where c is true will be reached with non-zero probability and 2) the probability of reaching a state where e is true (within the time bounds) is greater after being in a state where c is true (probability $\geq p$) than 3) it is by simply starting from initial state of the system (probability $< p$). When making inferences from data that means that c must occur at least once, and the conditional probability of e given c is greater than the marginal probability of e (usually calculated from frequencies). Since negative (probability lowering) causes can be defined in terms of their complement (so that if c lowers the probability of e, $\neg c$ raises its probability, the definition here is in terms of positive, probability raising causes.

Definition A3: Suppes' definition of *prima facie* cause
An event $B_{t'}$ is a *prima facie* cause of event A_t *iff*:

1. $t' < t$
2. $P(B_{t'}) > 0$
3. $P(A_t|B_{t'}) > P(A_t)$

We should interpret this as being for all t and t' where $t' < t$. That is, the probability of A occurring <u>at any time</u> after B is greater than the marginal probability of A occurring <u>at any time</u>. Thus, the conditions 1-3 do not refer to specific values of t and t' but rather describe the relationship between t and t'. In some cases, these causes may turn out to be false. Even if something meets the criterion of being a *prima facie* cause, this may be due only to common cause of it and the effect. Suppes introduces two ways in which something may be a false, or spurious cause. In each, the idea is that there is some earlier event than the *prima facie* cause that accounts equally well for the effect, so that his other information is known, the spurious cause does not have any influence on the effect.

Definition A4: Suppes' first definition of *spurious cause*
An event $B_{t'}$, a prima facie cause of event A_t, is a *spurious cause* in sense one iff $\exists\, t" < t'$ and $C_{t"}$ such that:

1. $P(B_{t'} \wedge C_{t"}) > 0$
2. $P(A_t|B_{t'} \wedge C_{t"}) = P(A_t|C_{t"})$
3. $P(A_t|B_{t'} \wedge C_{t"}) \geq P(A_t|B_{t'})$

While $B_{t'}$ is a possible cause of A_t, there may be another, earlier, event that has more explanatory relevance to A_t. However, condition 2 of the definition above is very strong and perhaps counterintuitive. It means that there exists an event that completely eliminates the effectiveness of the cause for predicting the effect. One way of relaxing this condition is to find not individual events but rather kinds of events. In Suppes' second definition of spurious causes there will be a set of nonempty sets that cover the full sample space, and which are mutually exclusive (pairwise disjoint). Thus, only one of these sets can be true *and* together they cover all possibilities.

Definition A5: Suppes' second definition of *spurious cause*
An event $B_{t'}$, a prima facie cause of event A_t, is a *spurious cause* in sense two iff there is a partition $\pi_{t"}$ where $t" < t'$ and for every $C_{t"}$ in $\pi_{t"}$:

1. $P(B_{t'} \wedge C_{t"}) > 0$
2. $P(A_t|B_{t'} \wedge C_{t"}) = P(A_t|B_{t'})$

Distinction between these two kinds of spuriousness is made with an example given by (Otte, 1982) on pp63:

For now on I will abbreviate "spurious in sense two" by $spurious_2$ and "spurious in sense one" by $spurious_1$. This definition makes an event $spurious_2$ if the world can be partitioned in such a way that the above conditions are satisfied. Thus, if we can observe a certain kind of event given by the partition, the observation of the later event $B_{t'}$ is uninformative, which makes it a $spurious_2$ cause. Suppes proves that if an event is a $spurious_2$ cause, then it is a $spurious_1$ cause. The converse of this theorem, however, is not necessarily true: it is possible for an event to be a $spurious_1$ cause and not be a $spurious_2$ cause.

As an example of a $spurious_1$ cause, let us take the case of decreasing air pressure causing not only rain but a falling barometer reading. The falling barometer reading is a prima facie cause of rain; given that the barometer reading is dropping, the probability that it will rain rises. Letting A denote rain, B denote a falling barometer reading, and C denote decreasing air pressure, the probability of rain given that the barometer reading, and the air pressure are decreasing, $P(A|B \wedge C)$, is equal to the probability of rain given that the air pressure is decreasing, $P(A|C)$; thus the second condition of the second definition of spurious cause is satisfied. The third condition is likewise satisfied, since the probability of rain given decreasing air pressure and a falling barometer reading is a least as great as the probability of rain given a falling barometer reading, $P(A|B \wedge C) \geq P(A|B)$. Thus, by the second definition a falling barometer reading is a $spurious_1$ cause of rain. The falling barometer reading is a $spurious_2$ cause of rain. If we let π be our partition (decreasing air pressure, non-decreasing air pressure), then

1. $P(BC) > 0$
2. $P(A|BC) = P(A|C)$
3. $P(A|B\neg C) = P(A|\neg C)$

So the falling barometer reading is a $spurious_2$ cause of the rain.

Theorem A1: Assume there is a *Kripke* structure $M = \{\mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathfrak{L}\}$ representing the underlying system governing the occurrences of the events. Then the conditions for causality given in the **Definition A2** for prima facie cause earlier are satisfied if and only if the conditions for causality given by **Definition A3** (**Definition A3**) are satisfied.
<u>*Proof:*</u>
We begin by showing that **Definition A2→Definition A3** and then show that **Definition A3→Definition A2**.

Proposition A1.1: **Definition A2→Definition A3Definition A3**
<u>*Proof:*</u>
Assume that $c = C$, $e = E$ and there is a *Kripke* structure $M = \{\mathcal{S}, \mathcal{S}_0, \mathcal{R}, \mathfrak{L}\}$, representing the underlying system governing the occurrences of these events. Also assume that states in M that satisfy c and e are labeled as such. If $t' < t$ in **Definition A3**, we assume that in M that satisfy c and e are labeled as such. If $t' < t$ in **Definition A3**, we assume that in M there will be at least one transition between an event at t' and one at t. That is, the timescale of M is as fine as that of Suppes and vice versa. Further, we assume that the probabilities of Suppes's formulation and those in M come from the same source and this if represented correctly, $P(E)$ in **Definition A3** is equal to $P(e)$ in **Definition A2**.

<u>*Condition 1*</u>: $P(E_t | C_{t'}) > P(E_t)$

By definition of $F^{\leq\infty}_{<p}e$, the probability $P(E_t)$ of E occurring at any time t is less than p. Recall that the probability of a path is the product of the transition probabilities along the path, and the probability of a set of paths is the sum of their individual path probabilities. For a structure to satisfy this formula, the set of paths from the start state that reach a state where e holds must be less than p, and the probability of reaching a state where e holds in this system is less than p. Thus,

$$P(E_t) < p \quad \text{(A.1)}$$

Now we must show $P(E_t|C_{t'}) > p$. We now show that this conditional probability is greater than or equal to p if:

$$c <^{\geq 1, \leq\infty}_{\geq p} e \quad \text{(A.2)}$$

is satisfied.

The probability p_1 of a transition from state s_1 to state s_2 that labels the edge between them,

$$s_1 \overset{p_1}{\rightarrow} s_2 \,,$$

Is the conditional probability:

$$P(s_{2,t+1}|s_{1,t}) \quad \text{(A.3)}$$

The probability of reaching s_2 one time unit after s_1. Then, for a path:

$$s_1 \overset{p_1}{\rightarrow} s_2 \overset{p_2}{\rightarrow} s_3 \,,$$

we can calculate the probability, given s_1, of reaching s_3 (via s_2) within two time units:

$$P(s_{3,t+2}, s_{2,t+1}|s_{1,t}) = P(s_{3,t+2}|s_{2,t+1}, s_{1,t}) \times P(s_{2,t+1}|s_{1,t}) \quad \text{(A.4)}$$

and since s_3 and s_1 are independent conditioned on s_2 this becomes:

$$P(s_{3,t+2}, s_{2,t+1}|s_{1,t}) = P(s_{3,t+2}|s_{2,t+1}) \times P(s_{2,t+1}|s_{1,t}) \quad \text{(A.5)}$$

Note that the probabilities of the righthand side are simply the transition probabilities from s_1 to s_2, and s_2 to s_3 (since there is one time unit between the states, they can only be reached via single transition).
Thus, the conditional probability is precisely the path probability:

$$P(s_{3,t+2}, s_{2,t+1}|s_{1,t}) = p_2 \times p_1 \quad \text{(A.6)}$$

Then, if we have a set of paths from s_1 to s_3, the conditional probability $P(s_3|s_1)$ is the sum of these path probabilities. For example, we may have the following paths:

$$s_1 \overset{p_1}{\rightarrow} s_2 \overset{p_2}{\rightarrow} s_3$$

$$s_1 \overset{p_3}{\rightarrow} s_4 \overset{p_4}{\rightarrow} s_3$$

In which case:

$$P(s_{3,t+2}|s_{1,t}) = P(s_{3,t+2}, s_{2,t+1}|s_{1,t}) + P(s_{3,t+2}, s_{4,t+1}|s_{1,t}) \quad \text{(A.7)}$$

and from eq. (A.6) this becomes:

$P(s_{3,t+2}|s_{1,t}) = p_2 \times p_1 + p_4 \times p_3$ (A.8)

the sum of the individual path probabilities. Let us now assume that s_1 is labeled with c and s_3 is labeled with e, these are the only c and e states in the system, and there are no other paths between the states taking less than or equal to 2 time units. Then, this probability we have computed is in fact the probability of:

$c <^{\geq 1, \leq 2} e$ (A.9)

since the probability of reaching s_3, during a window of time simply means looking at the set of paths reaching s_3 during that window. Similarly, to find the probability of:

$c <^{\geq 1, \leq \infty} e$ (A.10)

we must consider the set of paths from states labeled with c to those labeled with e that take at least 1 time unit. Since there can be cycles in our graph, calculating the probability associated with a leads-to formula with an infinite upper time bound requires a slightly different method.
//Finish the paragraph the leads-to formula with lower and upper bound

Leads-to with Both Lower and Upper Time Bounds
This paragraph deals with evaluation of *Leads-To* with applied window of time in which c leads to e. We assume a minimum time after c is true before which e is true. Here it is shown that it is possible to add such a lower bound. By Definition:

$f <^{\geq t_1, \leq t_2}_{\geq p} g \equiv AG[f \rightarrow F^{\geq t_1, \leq t_2}_{\geq p} g]$ (A.11)

where $t_1 \leq t_2$. Thus, we are only adding a minimum time to the consequent of the conditional. If we can label states where $F^{\geq t_1, \leq t_2}_{\geq p} g$ is true, then we can proceed as in the algorithm of (Hansson & Jonsson, 1994).

//Finish the paragraph on Leads-to with Both Lower and Upper Bounds taken from the B.2 of (Kleinberg, Causality, Probability, and Time, 2012)

Definitions and Review on Probabilistic Real Time Computation Tree Logic (PCTL) in (Hansson & Jonsson, 1994)

Notation

Assume a finite set A of *atomic propositions*. We use a, a_1, etc. to denote atomic propositions. Formulas in PCTL are built from atomic propositions, propositional logic connectives and operators for expressing time and probabilities. The set of PCTL formulas is divided into *path formulas* and *state formulas*. Their syntax is defined inductively as follows:

- Each atomic proposition is a state formula
- If f_1 and f_2 are state formulas, then so are $\neg f_1$, $(f_1 \wedge f_2)$, $(f_1 \vee f_2)$, $(f_1 \rightarrow f_2)$
- If f_1 and f_2 are state formulas, and t is a nonnegative integer or ∞, then $(f_1 U^{\leq t} f_2)$ and $(f_1 W^{\leq t} f_2)$ are path formulas,
- If f is a path formula and p is a real number with $0 \leq p \leq 1$, then $[f]_{\geq p}$ and $[f]_{>p}$ are state formulas.

We shall use f, f_1, etc. to range over PCTL formulas. Intuitively, state formulas represent properties of states and path formulas represent properties of paths (i.e., sequences of states). The propositional connectives $\neg$, $\vee$, $\wedge$, and $\rightarrow$ have their usual meanings. The operator U is the *(strong) until* operator, and W is the *unless* (or *weak until*) operator. For a given state s, the formulas $[f]_{\geq p}$ and $[f]_{>p}$ express that f holds for a path from s with a probability of at least p and greater than p, respectively.
We shall use $f_1 U_{\geq p}^{\leq t} f_2$ as a shorthand for $[f_1 U^{\leq t} f_2]_{\geq p}$, and use $f_1 W_{\geq p}^{\leq t} f_2$ as a shorthand for $[f_1 W^{\leq t} f_2]_{\geq p}$. Intuitively, $f_1 U_{\geq p}^{\leq t} f_2$ means that there is at least a probability p that both f_2 will become true within t time units and that f_1 will be true from now on until f_2 becomes true. $f_1 W_{\geq p}^{\leq t} f_2$ means that there is at least a probability p that either f_1 will remain true for at least t time units, or that both f_2 will become true within t time units and that f_1 will be true from now on until f_2 becomes true.

PCTL formulas are interpreted over structures that are discrete time Markov chains. A specified initial state is associated with the structure. In addition, for each state there is an assignment of truth values to atomic propositions appearing in a given formula. Formally, a structure is a quadruple $\langle S, s^i, \mathcal{T}, L \rangle$, where

S is a finite set of states, ranged over by s, s_1, etc.,
$s^i \in S$ is an *initial state*,
$\mathcal{T}$ is a *transition probability function*, $\mathcal{T}: S \times S \rightarrow [0,1]$, such that for all s in S we have

$\sum_{s' \in S} \mathcal{T}(s, s') = 1$,

L is a labeling function assigning atomic propositions to states, i.e.,

$L : S \rightarrow 2^A$

Intuitively, each transition is considered to require one *time unit*. We will display structures as transition diagrams, where states (circles) are labeled with atomic propositions and transitions with non-zero probability are represented as arrows labeled with their probabilities (e.g., the arrow going from state s_k to state s_l is labelled with $\mathcal{T}(s_k, s_l)$). The initial state s^i is indicated with an extra arrow.

A path σ from a state s_0 in a structure is an infinite sequence

$s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \rightarrow \cdots$

of states with s_0 as the first state. The n:th state (s_n) of σ is denoted $\sigma[n]$, and the prefix of σ of length n is denoted $\sigma \uparrow n$, i.e.,

$\sigma \uparrow n = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$

For each structure and state s_0 we define a probability measure μ_m on the set of paths from s_0. μ_m is defined on the probability space $\langle X, \mathcal{A} \rangle$, where X is the set of paths starting in s_0 and $\mathcal{A}$ is a sigma-algebra on X generated by sets

$\{\sigma \in X : \sigma \uparrow n = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n\}$

Of paths with a common finite prefix $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$. The measure μ_m is defined as follows: for each finite sequence $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$ of states,

$\mu_m(\{\sigma \in X : \sigma \uparrow n = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n\}) = \mathcal{T}(s_0, s_1) \times \cdots \times \mathcal{T}(s_{n-1}, s_n)$

i.e., the measure of the set of paths σ for which $\sigma \uparrow n = s_0 \to s_1 \to \cdots \to s_n$ is equal to the product $\mathcal{T}(s_0, s_1) \times \cdots \times \mathcal{T}(s_{n-1}, s_n)$. For $n = 0$ we define $\mu_m(\{\sigma \in X : \ \sigma \uparrow 0 = s_0\}) = 1$. This uniquely defines the measure μ_m on all sets of paths in the sigma-algebra $\mathcal{A}$.

We define the truth of PCTL formulas for a structure K by a satisfaction relation:

$s \vDash_K f$

which means that the state formula f is true at state s in the structure K. In order to define the satisfaction relation for states, it is helpful to use another relation

$s \vdash_K f$

which means that the path σ satisfies the path formula f in K. The relations $s \vDash_K f$ and $s \vdash_K f$ are inductively defined as follows:

$s \vDash_K a$ iff $a \in L(s)$
$s \vDash_K \neg f$ iff not $s \vDash_K f$
$s \vDash_K f_1 \wedge f_2$ iff $s \vDash_K f_1$ and $s \vDash_K f_2$
$s \vDash_K f_1 \vee f_2$ iff $s \vDash_K f_1$ or $s \vDash_K f_2$
$s \vDash_K f_1 \to f_2$ iff $s \vDash_K \neg f_1$ or $s \vDash_K f_2$
$\sigma \vdash_K f_1 U^{\leq t} f_2$ iff there exist an $i \leq t$ such that $\sigma[i] \vDash_K f_2$ and $\forall j : 0 \leq j < i : \ (\sigma[j] \vDash_K f_1)$
$\sigma \vdash_K f_1 W^{\leq t} f_2$ iff $\sigma \vdash_K f_1 U^{\leq t} f_2$ or $\forall j : 0 \leq j < i : \ (\sigma[j] \vDash_K f_1)$
$s \vDash_K [f]_{\geq p}$ iff the μ_m-measure of the set of paths σ starting in s for which $\sigma \vdash_K f$ is at least p.
$s \vDash_K [f]_{> p}$ iff the μ_m-measure of the set of paths σ starting in s for which $\sigma \vdash_K f$ is greater than p.

We define

$\vDash_K f \equiv s^i \vDash_K f$

where s^i is the initial state of K.

Properties expressible in PCTL

We will present examples of properties that can be expressed in PCTL. First, we discuss some of the facilities of PCTL which makes it suitable for specification of soft and hard deadlines.

The main difference between PCTL and branching time temporal logics such as CTL, is the quantification over paths and the ability to specify quantitative time. CTL allows universal (Af) and existential (Ef) quantification over paths, i.e., one can state that a property should hold for all computations (paths) or that it should hold for some computations (paths). It is not possible to state that a property should hold for a certain portion of the computations, e.g., for at least 50% of the computations. In PCTL, on the other hand, arbitrary probabilities can be assigned to path formulas, thus obtaining a more general quantification over paths. An analogy to universal and existential quantification can in PCTL be defined as:

$Af \equiv [f]_{\geq 1}$
$Ef \equiv [f]_{>0}$

Quantitative time allows us to specify time-critical properties that relate the occurrence of events of a system in real-time. In PCTL it is possible to state that a property will hold continuously during a specific time interval, or that a property will hold sometime during a time interval. Combining this with the above quantification we can define

$G_{\geq p}^{\leq t} f \equiv f W_{\geq p}^{\leq t} false$

$F_{\geq p}^{\leq t} f \equiv true U_{\geq p}^{\leq t} f$

$G_{\geq p}^{\leq t} f$ means that the formula f holds continuously for t time units with a probability of at least p, and $F_{\geq p}^{\leq t} f$ means that the formula f holds within t time units with a probability of at least p.

An important requirement on most real-time and distributed systems is that they should be continuously operating, e.g., every time the controller receives an alarm signal from a sensor the controller should take the appropriate action. We can express such requirements with the following PCTL operators:

$AGf \equiv f W_{\geq 1}^{\leq \infty} false$
$AFf \equiv true U_{\geq 1}^{\leq \infty} f$
$EGf \equiv f W_{>0}^{\leq \infty} false$
$EFf \equiv true\ U_{>0}^{\leq \infty} f$

AGf means that f is always $true$ (in all states that can be reached with non-zero probability), AFf means that a state where f is $true$ will eventually be reached with probability 1, EGf means that there is a non-zero probability for f to be continuously true, and EFf means that there exists a state where f holds which can be reached with non-zero probability.

Definition (*Owicki & Lamport, 1982*)[1]: (unquantified) *leads-to* operator ($a < b$)
Whenever a becomes true, b will eventually hold.

Definition: PCTL quantified *leads-to* operator ($f_1 <_{\geq p}^{\leq t} f_2$):
$f_1 <_{\geq p}^{\leq t} f_2 \equiv AG\left[\left(f_1 \rightarrow F_{\geq p}^{\leq t} f_2\right)\right]$
$f_1 <_{\geq p}^{\leq t} f_2$ means that whenever f_1 holds there is a probability of at least p that f_2 will hold within t time units.

Many modal operators can be derived from the basic PCTL operators. We can for instance define an operator that corresponds to the CTL operator $A[f_1 \cup f_2]$ (E.M. Clarke, 1983) as follows:

$A[f_1 \cup f_2] \equiv f_1 U_{\geq 1}^{\leq \infty} f_2$

As an example, we will specify a mutual exclusion property. Consider two processes (P_1 and P_2) using the same criticial section. The atomic propositions N_i, T_i, and C_i indicates that P_i is in its non-critical, trying, and critical regions, respectively. The mutual exclusion property can be expressed as:

$AG[\neg(C_1 \wedge C_2)]$

This is not sufficient for most *real-time systems* since the property only states that simultaneous access to the critical section must be avoided always under all circumstances. To capture a specific real-time behavior, we can specify that whenever P_1 enters its trying region, it will enter its critical region within 4 time units. This can in PCTL be expressed as:

$T_1 <_{\geq 1}^{\leq 4} C_1$

For some systems, it might be sufficient that the deadline is almost always met (e.g. in 99% of the cases). The relaxed property can be expressed as:

$T_1 <_{\geq 0.99}^{\leq 4} C_1$

---

[1] In (Hansson & Jonsson, 1994) and (Owicki & Lamport, 1982) the symbol $\leadsto$ is used to denote the *leads-to* operator. In this document we use $\leadsto$ to denote *prima facie* cause.

Relaxing the timing requirement might enable a less costly implementation that still shows acceptable behavior. To be on the safe side we could add a strict upper limit to the relaxed property, combining the hard and soft deadlines above. If we assume that we want P_1 to always enter its critical region within 10 time units, and almost always within 4 time units we get the property:

$$(T_1 <_{\geq 1}^{\leq 10} C_1) \wedge (T_1 <_{\geq 0.99}^{\leq 4} C_1)$$

Model Checking in PCTL

In this section we present a model checking algorithm, which given a structure $K = \langle S, s^i, \mathcal{T}, L \rangle$ and a PCTL formula f determines whether $\models_K f$. The algorithm is based on the algorithm for model checking CTL (E.M. Clarke, 1983). It is designed so that when it finishes each state will be labeled with the set of subformulas of f that are $true$ in the state. One can then conclude that $\models_K f$ if the initial state (s^i) is labeled with f.

For each state of the structure, the algorithm uses a variable $label(s)$ to indicate the subformulas that are $true$ in state s. Initially, each state s is labeled with the atomic propositions that are $true$ in s, i.e., $label(s) := L(s), \forall s \in S$. The labeling is then performed starting with the smallest subformula of f that has not yet been labeled and ending with labeling states with f itself. Composite formulas are labeled based on the labeling of their parts. Assuming that we have performed the labeling of f_1 and f_2, the labeling corresponding to negation ($\neg f_1$) and propositional connectives ($f_1 \wedge f_2$,)

//Appendix: Finish the paragraph on PTL Theory

Bibliography

Clarke, E. M., & Schlingloff, B. H. (2001). Model Checking. In A. Robinson, & A. Voronkov, *Handbook of Automated Reasoning* (pp. 1369-1520). Elsevier Science Publishers B.V.

E.M. Clarke, E. E. (1983). Automatic Verification Of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach. *ACM*, 117-126.

Eells, E. (1991). *Probabilistic Causality.* Cambridge UK: Cambridge University Press.

G.E. Hughes, M. C. (1996). *A New Introduction To Modal Logic.* London: Routledge.

Hansson, H., & Jonsson, B. (1994). *A Logic for Reasoning about Time and Reliability.* Uppsala, Sweden: SICS Research Report SICS/R90013.

Houdth, G. V., Depaire, B., & Martin, N. (2022). Root Cause Analysis in Process Mining with Probabilistic Temporal Logic. *ICPM 2021 Workshops* (pp. pp. 73–84). Eindhoven, The Netherlands: LNBIP Volume 433.

Kleinberg, S. (2012). *Causality, Probability, and Time.* Cambridge, UK: Cambridge University Press.

Kleinberg, S., & Mishra, B. (2009). The Temporal Logic Of Causal Structures. *The Conference on Uncertainty in Artificial Intelligence*, (pp. 303-312). Montreal, Canada.

Otte, R. E. (1982). *Probability and Causality, PhD Thesis.* Ann Arbor , MI, 48106: University of Arizona Graduate College, University Microfilm International.

Spirtes, P., Glymour, C., & Sheines, R. (1993). *Causation, Prediction and Search.* New York: Springer Verlag.

Downloadable Links for the Bibliography

(Clarke & Schlingloff, 2001): here

(Eells, 1991): here

(Hansson & Jonsson, 1994): here

(Houdth, Depaire, & Martin, 2022): here

(Kleinberg & Mishra, The Temporal Logic Of Causal Structures, 2009): here

(Spirtes, Glymour, & Sheines, 1993): here
(Otte, 1982): here
(G.E. Hughes, 1996): here
(E.M. Clarke, 1983): here