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 3

Events 3

Event Relationships 4

Directed Follow Graphs 7

Causal association between events 11

Directed Causal Graphs 13

Problem Statement for Root Cause Analysis of Fulfillment Decisions 13

Algorithm For Root Cause Analysis 13

Examples 14

Appendix A: Probabilistic Causality Frameworks in the Literature .... Error! Bookmark not defined.

Review on Good's, Reichenbach's and Suppes' Causality theories presented in Salmon's essays .... **Error! Bookmark not defined.**

Review on Reichenbach's and Suppes' Causality frameworks presented in Otte's thesis ... **Error! Bookmark not defined.**

Overview on Causality Foundations in Samantha Kleinberg's work .... **Error! Bookmark not defined.**

Appendix B: Logic Systems: Modal Logic, Computation Tree Logic, Probabilistic Temporal Logic 14

Review on Computation Tree Logic and Specification Language in (E.M. Clarke, 1983) ..... **Error! Bookmark not defined.**

Definitions and Review on Probabilistic Real Time Computation Tree Logic (PCTL) in (Hansson & Jonsson, 1994) .... **Error! Bookmark not defined.**

Definitions and Review on Probabilistic Temporal Logic in (Kleinberg, Causality, Probability, and Time, 2012) .... **Error! Bookmark not defined.**

Examples of PTL .... **Error! Bookmark not defined.**

Bibliography **........ 14**

Downloadable Links for the Bibliography .... Error! Bookmark not defined.

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, $\mathfrak{R}(\Delta, \mathcal{E})$ 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 $\mathcal{D}$
$S(\mathrm{E}_b \rightsquigarrow \mathrm{E}_a|\mathcal{K})$ – 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 in time* event E_a
$\mathrm{E}_a \precsim \mathrm{E}_b$ denotes the statement that event E_b *generally follows* event E_a (either in time or via static association)
$\mathrm{E}_a < \mathrm{E}_b$ denotes the statement that event E_a *precedes in time* event E_b
$\mathrm{E}_a \leqslant \mathrm{E}_b$ denotes the statement that event E_a is *generally reachable from* event E_b (either in time or via static association)
$\mathrm{E}_a \leq \mathrm{E}_b$ denotes the statement that event E_a is *reachable in time* from 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 $\mathcal{E}$
$\mathfrak{S}: \mathcal{E} \times \mathcal{E} \rightarrow \{0,1\}$ – denotes static dependency map
$\mathfrak{A}: \mathcal{E} \times \mathcal{E} \rightarrow \{0,1\}$ – 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
$\precsim$ - denotes *generally follows* relation between two concepts
$\leq$ - denotes *is reachable in time* relation between two concepts
$\preccurlyeq$ - denotes *generally 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
$\curvearrowright$ - *prima facie* causal relation between two concepts
$\rightsquigarrow$ - 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 $t_1, t_2, \ldots, t_o, t_{o+1}, \ldots$. 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 $\Longleftrightarrow$.

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 $(\mathfrak{S}, \mathcal{P}, \mathcal{V})$.

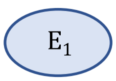
Note: Every event additionally to its standard parameter set $\mathcal{P}$ will have an implicit timestamp parameter τ which will always be present without regard of the nature of the event. We will not include explicitly the timestamp among the event parameters unless it is necessary in order to define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

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

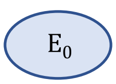
Set of events for analysis of the cause of splits in Fulfillment Decisions
E_0- order O_t is received. Event parameters: t
E_1- the i-th bundle of order O_t is being processed. Event parameters: t, i
E_2- SKU s in the i-th bundle of order O_t is being processed. Event parameters: t, i, s
E_3- node j 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: $t, B, i_1, i_2, \ldots, i_k$

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:



$t = 0$

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 $\mathrm{E}_3, \ldots, \mathrm{E}_9, \mathrm{E}_{13}, \mathrm{E}_{14}$ 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 $m \in \{3,4, \ldots ,9,13,14\}$

For
//TODO: finish this

Event Relationships

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 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.

The *follows in time* relation is a strong partial order which satisfies the conditions:

- Irreflexivity: $\mathrm{E}_a \nprec \mathrm{E}_a$
- Asymmetry: if $\mathrm{E}_a \prec \mathrm{E}_b$ then $\mathrm{E}_b \nprec \mathrm{E}_a$
- Transitivity: if $\mathrm{E}_a \prec \mathrm{E}_b$ and $\mathrm{E}_b \prec \mathrm{E}_c$ then $\mathrm{E}_a \prec \mathrm{E}_c$

Note that the *follows in time* relation is not guaranteed to hold for every pair of event instances – that is, if E_a and E_b are event instances then it can be true that both $\mathrm{E}_a \nprec \mathrm{E}_b$ and $\mathrm{E}_b \nprec \mathrm{E}_a$. This would be the case if E_a and E_b have the same timestamp or at least one of them is does not have timestamp specified in its parameter set.

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 $\mathrm{E}_{15}(t')$.

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*) $\mathfrak{A}: \mathcal{E} \times \mathcal{E} \to \{0,1\}$ such that $\mathfrak{A}(\mathrm{E}_a, \mathrm{E}_b) = 1$ when $\mathrm{E}_a \rightleftarrows \mathrm{E}_b$. The construction and depiction of static association map will be discussed in **Example 1** below.

Definition: Event E_c *is (generally) reachable from* Event E_a
We say that event E_c *is (generally) reachable from* E_a (denoted with $\mathrm{E}_a \preccurlyeq \mathrm{E}_c$) if event E_c has occurred in time *after* event E_b or if there is a static dependency between E_a and E_b ***and*** E_b is reachable from E_a. Any event is reachable from itself, i.e., $\mathrm{E}_a \preccurlyeq \mathrm{E}_a$. Formally,

$$\mathrm{E}_a \preccurlyeq \mathrm{E}_c \Leftrightarrow \exists \mathrm{E}_b\ s.t. (\mathrm{E}_b \prec \mathrm{E}_c\ \vee \mathrm{E}_b \rightarrowtail \mathrm{E}_c) \wedge\ \mathrm{E}_b \preccurlyeq \mathrm{E}_c$$

Note that the *(generally) reachable* relation $\preccurlyeq$ represents *weak partial order* obeying the reflexivity, asymmetry and transitivity conditions.

Definition: Event E_b *is reachable in time from* Event E_a
Is reachable from relation (denoted with $\leq$) implies that the timestamp associated with E_b is newer than or equal to that associated with E_a i.e., $\tau_b \geq \tau_a$. See the **Note** on the event parameters.
Note that the *reachable in time* relation $\leq$ represents *weak partial order* obeying the reflexivity, asymmetry and transitivity conditions.
Definition: (*irreflexive) reachable in time* relation (denoted with $<$) can be defined similarly to its reflexive counterpart postulating that in order an event E_b to be (irreflexively) reachable in time from E_a the corresponding timestamp must be **newer** than the timestamp of the other event i.e. $\tau_b > \tau_a$. The *reachable in time* relation $<$ represents *strong partial order* obeying the irreflexivity, asymmetry and transitivity conditions.

We will use both definitions in different occasions.

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

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.

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 $\prec$ between two Event instances E_a and E_b or static (semantic) dependency $\rightarrow$ between E_a and E_b. Each arc is marked with label which indicates what kind of relation it represents. In this document we draw all arcs which represent *follow-in-time* relation with red color and all arcs which represent *static dependency* with blue color.
Each *follow-in-time* 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 *follow-in-time* 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-in-time* relation $\mathrm{E}_a \prec \mathrm{E}_b$.

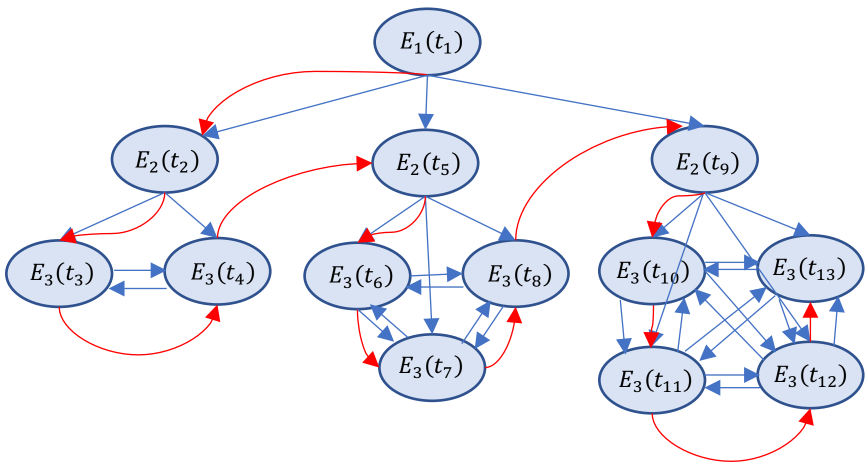
For example:

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_1 the Event that an order with 3 bundles have been received. Also, we will denote with an instance of event type E_2 each of the bundles of the order o_t. We denote with an instance of event type E_3 each of the SKUs in each bundle of the order O_t. We will assume the following time sequence of the event instances:

$$E_1(t_1) \prec E_2(t_2) \prec E_3(t_3) \prec E_3(t_4) \prec E_2(t_5) \prec E_3(t_6) \prec E_3(t_7) \prec E_3(t_8) \prec E_2(t_9) \prec E_3(t_{10}) \prec E_3(t_{11}) \prec E_3(t_{12}) \prec E_3(t_{13})$$

We depict this scenario with the following DFG:



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:

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

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:

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

The relation (1) represents the fact that each child is statically associated to its parent and is statically dependent on the parent event, namely an order with specified number of bundles has been received. This relation is depicted with arrow $\rightarrowtail$ in (1).

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

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

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

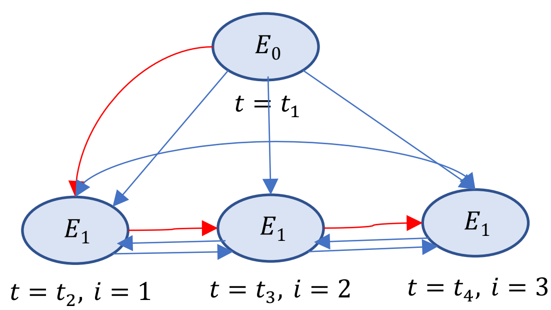
$\mathrm{E}_2|_{t=0,i=m,s=p} \rightleftarrows \mathrm{E}_2|_{t=0,i=m,s=q}\ \forall m = 1,2,3$ and $\forall p, q \in \mathcal{I}(\mathcal{S}_m)$ (4)
Here $\mathcal{I}(\mathcal{S}_m)$ denotes the index set of $\mathcal{S}_m$.

~~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

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 red arc denotes *follow-in-time* relation $\prec$. Each blue arc denotes static (semantic) dependency $\rightarrowtail$ between the event instances. Static association $\rightleftarrows$ between nodes is shown via a pair of blue arcs each pointing to the opposite node. 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 *follow-in-time* 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 in time E_a i.e., $\mathrm{E}_a \prec \mathrm{E}_b$.

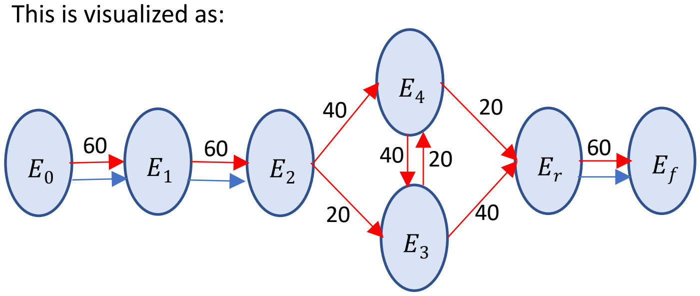
Definition: *DFG Representation* of a concept Δ over an event set $\mathcal{E}$
We say that DFG is a representation of Δ if the graph constructed with the events in $\mathcal{E}$ models *semantically* the internal structure of Δ.
For example, the DFG shown in *Example 1* is DFG representation of the order O_t. The DFG G representing O_t will be denoted with $G = \mathfrak{N}(\mathrm{O}_t)$ 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 $G = \mathfrak{N}(D, \mathcal{E})$, *iff* there does not exist DFG G_1such that $G_1 = \mathfrak{N}(D, \mathcal{E})$ 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 which is depicted by red-colored arcs. Additionally, we depict semantic dependencies () using blue arcs. The events are represented by DFG over some set of events $\mathcal{E}$.
For example:

Example 2
Sixty orders $\mathrm{O}_t, t = 1,2, \ldots, 60$ with a single bundle b_1 and single SKU s_1 with unit quantity have 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 $\mathrm{E}_a|_{\mathcal{D}}$) we have $v_a \in \mathcal{V}_a$. Similarly, for $\mathrm{E}_b|_{\mathcal{D}}$ 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. $\mathrm{E}_a \leftrightsquigarrow \mathrm{E}_b$.
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:

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

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.

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

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

$$P(\mathrm{E}_b|\mathrm{E}_a)|_{\mathcal{D}} > P(\mathrm{E}_b)|_{\mathcal{D}} \quad (8)$$

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* $\mathcal{K} = \{K_1, K_2, \ldots, K_n\}$. 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 $G_t, t = 1,2, \ldots$ 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 the algorithm

Examples

//TODO: finish the examples

Appendix A: Overview of Probabilistic Causality Concepts and Theories

See document here.

Also see notes on (Kleinberg, Causality, Probability, and Time, 2012) here.

Appendix B: Logic Systems: Modal Logic, Computation Tree Logic, Probabilistic Temporal Logic

See document here.

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.

Good, I. J. (1961). A Causal Calculus (I and II). *The British Journal for the Philosophy of Science*, 305-318, 43-51.

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.

Hume, D. (1748). *Philosophical Essays Concerning Human Understanding.* London: Printed for A. Millar, opposite Katharine-Street in the Strand. MDCCXLVII.

Kleinberg, S. (2010). *An Algorithmic Enquiry Concerning Causality.* New York: New York University.

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.

Lewis, D. (1973). *Counterfactuals.* Malden, Massachusetts: Blackwell Publishers.

Lewis, D. (1974). Causation. *Journal of Philosophy*, 556-567.

Mackie, J. (1980). *The Cement of The Universe: A Study of Causation.* New York: Oxford University Press, New York, United States.

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

Reichenbach, H. (1956). *The Direction of Time.* Berkeley and Los Angeles, California: University of California Press.

Salmon, W. C. (1980). Probabilistic Causality. *Pacific Philosophical Quarterly*, pp. 137-153.

Salmon, W. C. (1998). *Causality and Explanation.* Pittsburgh, Pennsylvania: Oxford University Press.
Spirtes, P., Glymour, C., & Sheines, R. (1993). *Causation, Prediction and Search.* New York: Springer Verlag.
Susan Owicki, L. L. (1982). Proving Liveness Properties of Concurrent Programs. *ACM Transactions on Programming Languages and Systems*, 455-495.

Downloadable Links for the Bibliography

(Hume, Philosophical Essays Concerning Human Understanding, 1748): here
(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
(Kleinberg, An Algorithmic Enquiry Concerning Causality, 2010): here
(Reichenbach, 1956): here
(Spirtes, Glymour, & Sheines, 1993): here
(Otte, 1982): here
(G.E. Hughes, 1996): here
(E.M. Clarke, 1983): here
(Salmon, Causality and Explanation, 1998): here
(Salmon, Probabilistic Causality, 1980): here
(Good, 1961): here
(Mackie, 1980): here
(Lewis, Counterfactuals, 1973): here
(Lewis, Causation, 1974): here
(Susan Owicki, 1982): here