Revealing One-to-Many Event Relationships in Event Knowledge Graphs

Alessio Giacché[1], Sara Pettinari[2], and Lorenzo Rossi[1]

[1] School of Science and Technology, University of Camerino, Italy
alessio.giacche@studenti.unicam.it, lorenzo.rossi@unicam.it
[2] Gran Sasso Science Institute, L'Aquila, Italy sara.pettinari@gssi.it

Abstract. Object-centric process mining is recognized to overcome the limitations of traditional process mining by offering approaches for the analysis of processes with multiple case notions such as collaborations. In this regard, event knowledge graphs are an effective tool for gathering, manipulating, and visualizing event and entity relations. Current approaches focus on inferring correlations between events and objects and directly-follows relationships between events correlated to the same object. However, object-to-object relations may hide one-to-many relations between events essential for understanding the actual flow among processes. In this paper, we propose an approach to reveal these one-to-many causal relationships in an event knowledge graph. Specifically, we define when two events are causally related and extend the standard approach of event knowledge graphs construction to reveal them. We assess the approach using two case studies.

Keywords: Collaborations · Object-Centric Process Mining · Event Knowledge Graph · One-to-Many Relationships

1 Introduction

Process mining collects well-established techniques for analyzing event logs produced during the execution of processes [2]. By its nature, process mining techniques target the analysis of event logs from the perspective of a single case, i.e., an identifier shared by the events that belong to the same process instance. However, many real-life event logs contain different case notions since the events and the associated activities are shared by more than one process [1]. For instance, in an order handling process, a single order can trigger as many pick-up processes as there are items in the order. Therefore, some of the events in the log could be related to a unique order identifier as a case notion, while others could also refer to an item identifier. This creates a one-to-many relationship between orders and items, which must be considered in analyzing the processes.

In recent years, Object-Centric Process Mining (OCPM) has emerged from traditional process mining to address convergence and divergence issues in real-life process analysis [1]. OCPM has been proposed to connect processes not based on the same case notion but to explore and filter the behavior recorded in the logs considering different classes of objects and their interaction [4]. So far, most of the work has been done in providing process models capable of capturing event-to-object and object-to-object interactions, whereas less work also investigated the representation of event-to-event relationships [5].

In this context, an Event Knowledge Graph (EKG) is a flexible and expressive event data model to capture different aspects of the system behavior [11]. It enables the representation of the correlation between events and objects, and the relations between objects [18], while inferring the directly-follows relationships of events correlated with the same object. This approach establishes directly-follows relationships between events by considering each object individually. However, when multiple objects are involved, their interrelations can affect the event relationships. These relationships can vary in nature, ranging from one event triggering many others to multiple events impacting a single event.

This paper discusses causal event relationships and presents an approach to reveal them on EKGs. The approach leverages domain knowledge to identify the object types impacting event relationships. Using this information, the approach constructs the EKG which reveals the causal relation between events correlated to the identified objects. Finally, the nodes and relationships in the resulting EKG can be aggregated to discover a multi-entity Directly-Follows Graph (DFG) [12] representing the underlying system.

The rest of the paper is organized as follows. Sec. 2 introduces the main EKGs' concepts. Sec. 3 motivates the necessity of revealing causal event relationships through two case studies and the current state of the art. Sec. 4 presents the approach for revealing the causal relationships on EKGs. Sec. 5 discusses and applies the proposed approach in the two case studies. Finally, Sec. 6 concludes the paper by discussing the results, and touches upon future directions.

2 Background

An EKG is built on the concepts of events, entities (i.e., objects), and relations, which are interconnected to represent the analyzed system [12]. It is a labeled property graph with a limited set of node and relationship labels. In an EKG, event nodes store at least the *activity* name and its *timestamp*, and entities store at least a property defining its *type*. Event-to-event relations are defined via the directly-follows relationship, while, event-to-entity relations are defined via the correlation relationship. Formally, given the set Λ of labels, and the set $Attr$ of property names over a value domain Val, an EKG is defined as follows.

Definition 1 (EKG). *An EKG is a graph $G = (N, R, \lambda, \#)$ where N is a set of nodes, $R \subseteq N \times N$ is a set of relationships, $\lambda : N \cup R \to \Lambda$ is a function assigning a label to nodes and relationships, and $\# : (N \cup R) \times Attr \not\to Val$ is a partial function assigning properties (attribute-value pairs) to nodes and relationships.*

We adopt the notation $n.a$ as shorthand for $\#(n, a)$. Given an EKG, we distinguish two node sets, $N = N^{Event} \cup N^{Ent}$ where N^{Event} and N^{Ent} are respectively the set of event and entity nodes. Similarly, we distinguish two sets

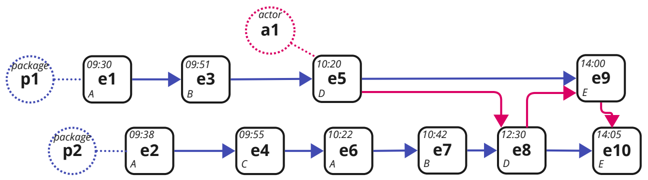


Fig. 1: EKG example

of relations, $R \subseteq R^{corr} \cup R^{df}$, where $R^{corr} = \{r : \lambda(r) = corr\} \subseteq N^{Event} \times N^{Ent}$ is the set of correlations between events and entities, and $R^{df} = \{r : \lambda(r) = df\} \subseteq N^{Event} \times N^{Event}$ is the set of directly-follows relationships. For the sake of simplicity, we say that $r = (e, a) \in R^{corr}$ if $a \in e.\mathcal{A}$; $(e, A) \in R^{corr}$ to mean that $(e, a') \in R^{corr}$, $\forall a' \in A$ with $a'.type = \mathcal{A}$; and $r = (e_i, e_j) \in R^{df}$ with respect to a (we write $r \in R^{df}_a \subseteq R^{df}$ for convenience) if $(e_i, a), (e_j, a) \in R^{corr} \wedge e_i.time < e_j.time \wedge (\nexists\ e : (e, a) \in R^{corr} \wedge e_i.time < e.time < e_j.time)$.

Fig. 1 shows an EKG example. Nodes are represented with rectangles; entities as circles; and correlation and directly-follows relationships as dashed and solid edges. For convention, the correlations are depicted only between an entity and the first event in time correlated to it, and directly-follows relationships over a specific entity are colored uniquely. For example, the relations in R^{df}_{a1} form the trace of events $\langle e_5, e_8, e_9, e_{10} \rangle$. EKGs allow for their manipulation and navigation through queries, facilitating the extraction of desired insights. Among the other manipulations, *aggregation* enables grouping multiple nodes and relations sharing common properties [11]. This produces a summarized view of event data therefore resulting essential to construct a multi-entity DFG, that stores in each directly-follows relationship the corresponding entity type.

3 Motivation

This section presents two case studies exhibiting collaborative patterns [16], showing the need to represent causal event relationships. Following this, current research relevant to this topic is reviewed.

3.1 Case Studies

The first collaborative scenario represents an order-handling process. It has been synthetically generated and keeps track of the ordering procedures in which each order can be composed of several items. The system receives an order, breaks it down into individual items, and prepares them for packing and shipping. Therefore, each order hierarchically involves several sub-processes for each item. An excerpt of the event log of the order-handling system is depicted in Tab. 2a. Here the log stores as entity type the order identifier, i.e., *order*, and the item identifier, i.e., *item*, composing an order. Constructing an EKG from this log would enable the visualization of distinct processes for each order and item. Focusing on the

e_id	activity	time	order	item
e1	receive order	21:32:23	O1	⊥
e2	start order	21:42:23	O1	⊥
e3	picking item	21:47:34	O1	i1
e4	picking item	21:47:45	O1	i2
e5	picking item	21:47:49	O1	i3
e6	out-of-stock	21:49:49	O1	i3
e7	item available	21:57:34	O1	i1
e8	item available	21:57:45	O1	i2
e9	picking completed	22:01:18	O1	i1
e10	picking completed	22:01:50	O1	i2
e11	item available	13:02:00	O1	i3
e12	picking item	13:05:04	O1	i3
e13	item available	13:09:28	O1	i3
e14	picking completed	13:14:51	O1	i3
e15	create pack	13:25:31	O1	⊥
e16	ship order	13:32:31	O1	⊥

(a)

e_id	activity	time	msg	robot
e1	takeoff	15:33:22	⊥	drone
e2	explore	15:33:25	⊥	drone
e3	weed_found	15:34:45	⊥	drone
e4	weed_position!	15:34:50	wp_1	drone
e5	weed_position?	15:35:06	wp_1	tractor_1
e6	weed_position?	15:35:07	wp_1	tractor_2
e7	tractor_position!	15:35:31	tp_2	tractor_1
e8	tractor_position?	15:35:32	tp_2	drone

(b)

Fig. 2: Excerpt of order-handling (a) and robotic system (b) event logs

order entity type, inferring the directly-follows relationships only based on order identifier and the time would create a trace like $\langle e_1, e_2, e_3, e_4, e_5, \ldots, e_{15}, e_{16}\rangle$. This implies that the activities of $O1$, but correlated to different items, would be related via a directly-follow relationship, as in the case of e_3 and e_4. However, since each item follows a unique process, the order process flow should depict a partial ordering of events. Therefore, the actual order process should start by initializing the order with $\langle e_1, e_2\rangle$, then it triggers the item flows, for instance for item $i1$ it performs $\langle e_3, e_7, e_9\rangle$, and concludes the order shipment with $\langle e_{15}, e_{16}\rangle$. Notably, this scenario also serves as a running example throughout this paper.

The second collaborative scenario is from the robotic domain [7]. The scenario consists of one drone and two tractors cooperating to identify and remove weed grass in farmland. The system workflow relies on the direct interaction among the robots via messages-exchange. Specifically, the drone identifies weed grass areas and broadcasts their location to the tractors. Tractors share their positions back, and the drone notifies the nearest one to cut the weed grass. An excerpt of the event log of the robotic system is depicted in Tab. 2b. Here the log stores as entity types the robot identifier, i.e., *robot*, and the message identifier, i.e., *msg*, generated during robots' interaction. Constructing an EKG from this log would enable the visualization of distinct processes for each robot and each message flow. Focusing on the *msg* entity type, inferring the directly-follows relationships solely based on the message identifier and the time would lead to a trace like $\langle e_4, e_5, e_6\rangle$. This implies that the message-sending event of the *drone* directly-follows the message-receiving event of *tractor_1*, which in turn directly-follows the message-receiving event of *tractor_2*. However, this sequence does not accurately reflect the actual system behavior. Since the drone sends a broadcast message, the trace of the entity wp_1 should reflect the partial order among events where the message-sending event of the *drone* directly-follows both message-receiving events, i.e., $\langle e_4, e_5\rangle$ and $\langle e_4, e_6\rangle$.

3.2 State of the Art

To discover the causal relationship between two events, i.e., when one event occurs as a result of another [10], it is important to consider the concept of partial ordering. This concept implies that when two events are causally unrelated, their chronological order can be disregarded, and a partial ordering can be used instead [1]. The partial order concept has been investigated with traditional process mining techniques to depict causal relationships between events [14]. With the advent of OCPM, a partial order of events has been obtained by inferring directly-follow relationships based on the object shared between two sequential events [3, 15]. Thus, for each object, the corresponding trace represents a total order of events [12]. However, depending on the requirements and characteristics of different scenarios, these approaches may lead to incorrect and misleading connections. Differently from these approaches and in line with the idea of depicting one-to-many and many-to-one relationships among events, causal event models based on object types that can trigger different object types have been investigated in [20,21]. The authors introduce and define the Causal Event Graph (CEG) data structure. This structure, along with its aggregated version, represents event causalities through many-to-many relationships tied to the corresponding objects. However, the CEG data structure only models events and excludes objects and their relationships.

Focusing instead on the characteristics of the case studies, the literature proposes several techniques, belonging to traditional process mining, to handle both the hierarchical dependency between processes and the message interactions. For hierarchical processes, there exist approaches for the discovery of multiple instance subprocesses [6,22]. For message interactions, one can refer to approaches for the discovery of collaborative systems [8,17], as well as agent system mining approaches [19]. However, these approaches are purpose-specific and not suited for object-centric event logs.

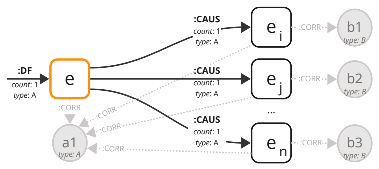
To address this limitation, we propose the use of EKGs to reveal causal event relationships in terms of causality, and related object types.

4 Approach

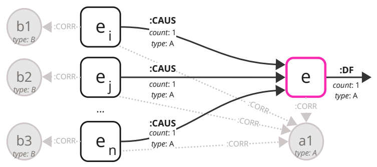
This section presents an approach for creating an EKG that reveals the causal relationship between events related to multiple interrelated objects. Specifically, we define when two events are in a causal relationship over two entities. Moreover, we extend EKG creation guidelines presented in [12] to infer the causal relationship and avoid the wrongly inferred directly-follows relationships. Finally, we applied existing aggregation operations to retrieve causal activity relationships.

4.1 Defining Causal Event Relationship

To reveal causal relationships between events, we start from an EKG with two entity types in a one-to-many relation, we call *trigger* the entity with cardinality 1 and *target* the entity with cardinality N. In a nutshell, two events in an EKG are in a causal relationship if they are correlated to the same entity of the trigger type and to different entities of the target type. Moreover, there must not exist a third event that happens in between the first two events, and that either is correlated to the same entities of one of them.



(a) Opening pattern



(b) Closing pattern

Fig. 3: Causal Event Relationships

Definition 2 (Causal relationship). *Let $G = (N, R, \lambda, \#)$ be an EKG, $\mathcal{A}$ and $\mathcal{B}$ two entity types in a one-to-many relation, e and e_i two events, a and b two entities respectively of type $\mathcal{A}$ and $\mathcal{B}$. We say that $r = (e, e_i)$ with $\lambda(r) =$ "causal" is a causal relation over a, b if the following hold:*

- $(e, a), (e_i, a) \in R^{corr} \;\wedge\; (e_i, b) \in R^{corr}$*; and*
- $(e, B') \in R^{corr}$ *with* $(b \in B' \wedge B' \neq \{b\}) \vee B' = \emptyset$*; and*
- $\nexists\, e_j : (e_j, a), (e_j, b) \in R^{corr} \wedge (e.time < e_j.time < e_i.time \vee e_i.time < e_j.time < e.time)$.

We call $R^{caus}_{a,B'}$ the set of all causal relationships over an entity a of type $\mathcal{A}$ and the set B' of entities of type $\mathcal{B}$. This set of causal relationships creates two kinds of patterns, namely *opening* and *closing* patterns. An **opening pattern** consists of a node, named *opening*, with outgoing causal relationships (Fig. 3a); a **closing pattern** consists of a node, named *closing* with incoming causal relationships (Fig. 3b). Following, we provide a formal definition of opening and closing nodes.

Definition 3 (Opening and closing nodes). *Let $G = (N, R, \lambda, \#)$ be an EKG, $e \in N^{Event}$ an event node and a and entity node of type $\mathcal{A}$ such that $(e, a) \in R^{corr}$, B' a set of entities of type $\mathcal{B}$, and $R^{caus}_{a,B'}$ the set of all causal relationships over a and B', we call e:*

- *an opening node if either* $\exists\, e_i, e_j \in N^{Event} : (e, e_i) \in R^{caus}_{a,B'} \wedge (e, e_j) \in R^{caus}_{a,B'}$ ***or*** $(\exists\, e_i \in N^{Event} : (e, e_i) \in R^{caus}_{a,B'}) \wedge (\nexists\, e_k \in N^{Event} : (e_k, e_i) \in R^{caus}_{a,B'})$
- *a closing node if either* $\exists\, e_i, e_j \in N^{Event} : (e_i, e) \in R^{caus}_{a,B'} \wedge (e_j, e) \in R^{caus}_{a,B'}$ ***or*** $(\exists\, e_i \in N^{Event} : (e_i, e) \in R^{caus}_{a,B'}) \wedge (\nexists\, e_k \in N^{Event} : (e_i, e_k) \in R^{caus}_{a,B'})$

Moreover, opening and closing nodes are identified by property *rel* i.e., $e.rel =$ "*opening*" and $e.rel =$ "*closing*" respectively.

4.2 Constructing the EKG

Following, we present the approach for revealing causal relationships in an EKG. Let $G = (N, R, \lambda, \#)$ be an EKG, a be a trigger entity of $\mathcal{A}$ type, and $\mathcal{B}$ a target entity type. The approach consists of applying the following steps for each trigger entity of type $\mathcal{A}$. First, we *combine traces* of events correlated to trigger and target entities. Then, we *reveal opening and closing patterns*. Finally, we *infer directly-follows relationships*.

Combine Traces. To identify the opening and closing patterns for a trigger entity a, we need to retrieve from the graph the set of traces by looking at events correlated with both a and the set of target entities. The function in Algorithm 1 represents the proposed step. It defines an empty dictionary E_a for containing the traces; the set E' of events correlated to a; and the set B' of values for the property $e.\mathcal{B}$ (lines 2 to 4). Then, the function cycles the elements of E' and B' to populate the dictionary E_a that will match each entity of B' with the set of events of E' correlated to that entity (lines 5 to 11).

Algorithm 1 Combine Traces

```
1:  function COMBINE(a)
2:      $E_a \leftarrow [][]$
3:      $E' \leftarrow \{e : (e, a) \in R^{corr}\}$
4:      $B' \leftarrow \{e.\mathcal{B} : e \in E'\}$
5:      for $e \in E'$ do
6:          for $b \in B'$ do
7:              if $e.\mathcal{B} = b$ then
8:                  $E_a[b] \leftarrow E_a[b] \cup \{e\}$
9:              end if
10:         end for
11:     end for
12:     return $E_a$
13: end function
```

Algorithm 2 Reveal opening and closing Patterns

```
1:  Input: $E_a$
2:  $first = head(E_a)$
3:  $remaining = E_a \setminus \{first\}$
4:  for $rel \in \{$“opening”, “closing”$\}$ do
5:      $first' =$ FILTER($first$, $remaining$, $rel$)
6:      $e_c =$ IDENTIFY($first'$, $rel$)
7:      REVEAL($e$, $rel$)
8:  end for
9:  function FILTER($tr$, $remaining$, $relation$)
10:     if $relation =$ “opening” then
11:         $tr' \leftarrow \{e \in tr : \nexists\, e_r \in remaining \ s.t.\ e.time < e_r.time\}$
12:     else if $relation =$ “closing” then
13:         $tr' \leftarrow \{e \in tr : \nexists\, e_r \in remaining \ s.t.\ e.time > e_r.time\}$
14:     end if
15:     return $tr'$
16: end function
17: function IDENTIFY($tr$, $relation$)
18:     $e_c \leftarrow null$
19:     if $relation =$ “opening” then
20:         $e_c \leftarrow e \in tr : \nexists\, e'' \in tr \ s.t.\ e''.time > e.time$
21:     else if $relation =$ “closing” then
22:         $e_c \leftarrow e \in tr : \nexists\, e'' \in tr \ s.t.\ e''.time < e.time$
23:     end if
24:     $e_c.rel = relation$
25:     return $e_c$
26: end function
27: function REVEAL($e_c$, $remaining$, $relation$)
28:     $r \leftarrow null$
29:     if $relation =$ “opening” then
30:         for $tr \in remaining$ do
31:             $e_r \leftarrow e \in tr : \nexists\, e'' \in tr \ s.t.\ e''.time < e.time$
32:             $r \leftarrow (e_c, e_r)$
33:         end for
34:     else if $relation =$ “closing” then
35:         for $tr \in remaining$ do
36:             $e_r \leftarrow e \in tr : \nexists\, e'' \in tr \ s.t.\ e''.time > e.time$
37:             $r \leftarrow (e_r, e_c)$
38:         end for
39:     end if
40:     $R^{caus}_{a,B'} \leftarrow R^{caus}_{a,B'} \cup \{r\}$
41: end function
```

Reveal Opening and Closing Patterns. To reveal opening and closing patterns, we use the set of combined traces E_a for the a entity. We consider the *relation* variable that indicates whether we are looking for a *opening* or a *closing*. We extract $first = head(E_a)$ where function $head()$ returns the trace with the event occurring for first in all events of E_a, and the set $remaining = tail(E_a)$ as the set of other traces except for the first one. The opening and closing pattern identification follows a similar path composed of three main steps: (*i*) filter, (*ii*) identify, and (*iii*) reveal. Algorithm 2 depicts the functions to reveal opening and closing patterns. Starting from the FILTER function, it retrieves events from the $first$ trace where their timestamps are less than those in the events of the *remaining* traces (if looking for an opening pattern) or higher than those in remaining events (if looking for a closing pattern) (lines 10 to 14). This function results in the $first'$ set of events, used to identify the opening or closing nodes.

The IDENTIFY function retrieves the node e_c from $first'$ as the one with a higher timestamp (opening) (line 20) or the one with a lower timestamp (closing) (line 22). Finally, it marks the identified node with the corresponding relation (line 24). Finally, the REVEAL function uses the identified node e_c and the set *remaining* to create a causal relationship. For each trace in *remaining*, it gets the event with a lower timestamp (opening) (line 31) or the event with a higher timestamp (closing) (line 36) and creates a causal relationship (lines 32 and 37), via r such that it stores in the *rel* property, the *causal* value. The revealed causal relation is then added to $R^{caus}_{a,B'}$ to include in the relations of the EKG (line 40).

Infer Directly-Follows Relationship. Once the opening and closing patterns are identified and the causal relationships revealed, the last step is inferring the remaining directly-follows relations. Algorithm 3 describes the final step.

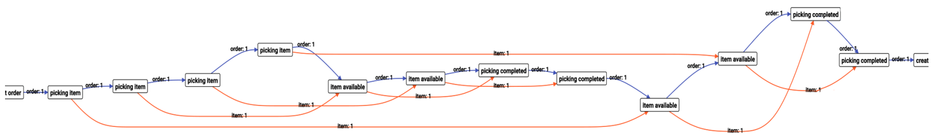
Algorithm 3 Infer R^{df}

```
1: Input: $a$, $B'$
2: function INFER($a$, $B'$)
3:     $R^{df} \leftarrow R^{df}_a \cup \{R^{df}_b, \forall\, b \in B'\}$
4:     $e_{op} \leftarrow e : (e, a) \in R^{corr} \wedge e.rel = \text{“}opening\text{”}$
5:     $e_{cl} \leftarrow e : (e, a) \in R^{corr} \wedge e.rel = \text{“}closing\text{”}$
6:     $R^{df} \leftarrow R^{df} \setminus \{(e_{op}, e_{cl})\}$
7: end function
```

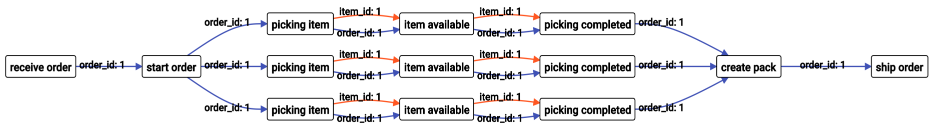
For the a entity and each entity of $\mathcal{B}$ type, a directly-follows relationship of the respective type is inferred following the formalization prescribed by [12] (line 3). Additionally, for the a entity a directly-follows relationship is not inferred between the opening and the closing node (lines 4 to 6). Notably, for each trigger entity, an opening pattern generates as many causal relationships as the number of unique occurrences with the target entity. A closing pattern collects as many causal relationships as the number of unique occurrences with the target entity.

4.3 Revealing Causal Activity Relations

Let $G = (N, R, \lambda, \#)$ be a graph constructed considering causal event relations. To reveal causal activity relations we performed node and relation aggregation as prescribed by the current state-of-the-art [11]. The aggregation operation relies on a generic aggregation function $Agg(class, G) = G^c$ to produce a multi-entity



(a) Without causal relations



(b) With causal relations

Fig. 4: EKGs excerpt of the order-handling case study

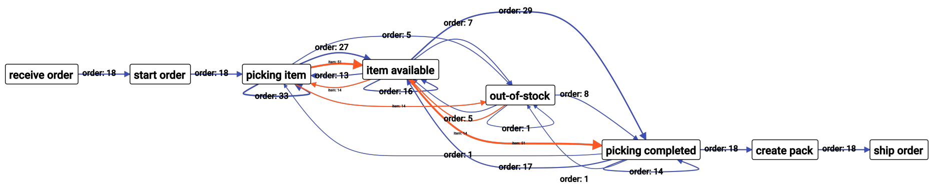
DFG, following the node aggregation property *class* to create *Class* nodes. Usually, the aggregation property refers to the events *activity* name, but it can also be a combination of properties, such as aggregating event nodes with the same *activity* and *order*. Moreover, the event-to-event relationships are aggregated to derive class-to-class relationships. The *type* of the class-to-class relationship is set based on the event-to-event relationship type, and the node causal *rel* information, i.e., opening or closing, is kept. Additionally, the occurrences of each observed event pair are stored as a relationship property *count*. Therefore, after EKG aggregation, the resulting multi-entity DFG shows class nodes and the relations among them, keeping track of the causal activity relations.

5 Proof of Concept

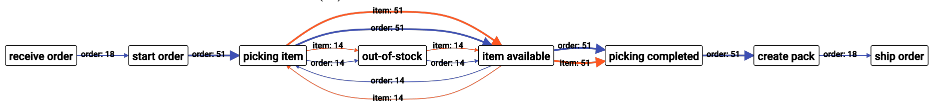
In this section, we show the proposed approach compared to the standard EKG creation of [12]. The approach has been applied to the case studies presented in Sec. 3 by implementing it as a collection of queries[3] in the *Cypher* language, enriched with the trigger and target information selected by the domain expert.

Order-handling. This case study is characterized by a hierarchical behavior where each order is split into several item lines composing it. In this scenario, an event produced by an order triggers multiple flows for item preparation. Therefore, the approach has been assessed considering the *order* entity type as the trigger and the *item* as the target entity type. Fig. 4 shows an excerpt of the EKGs generated without considering causal relationships (Fig. 4a) and with causal relationships (Fig. 4b). As illustrated in Fig. 4a, constructing the flow of an order entity based on temporal order incorrectly links different *picking item* activities, even if they belong to different items. Additionally, it links a *item available* activity to an *picking completed* activity across different items, as shown by looking both at the *order* and the *item* flow. Differently, Fig. 4b reveals the causal relationships between an order and the items composing it. It shows

[3] `https://bitbucket.org/proslabteam/soup/wiki/CausalRelationships`



(a) Without causal relations



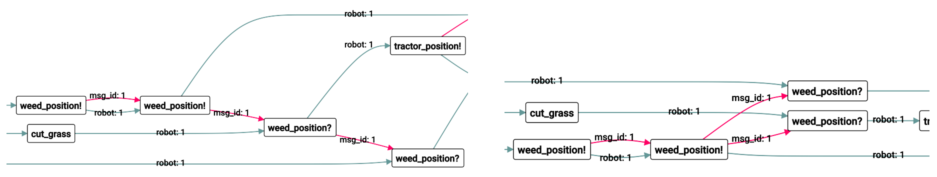
(b) With causal relations

Fig. 5: Multi-entity DFG of the order-handling case study

that the start order activity splits its flow into three item lines, each of which independently executes its process. After a *picking completed* items' flows collide into the order *create pack* activity. Performing the aggregation operation for the *activity* property results in Fig. 5. Specifically, Fig. 5a shows the aggregation of the EKG without considering the causal relationships between events correlated to orders and items. As a result, the order flow only reflects the sequence of activities performed on the orders over time. In contrast, Fig. 5b depicts the multi-entity DFG, which incorporates causal relationships. This view captures the entire process: 18 orders are received and started, generating 51 items. Of these, 14 were initially out-of-stock but were later prepared to assemble an order pack and ship the order.

Robotic System. This case study involves direct communication among robots. In a collaborative system with broadcast communication, a single event that generates a message can trigger multiple message-catching events by different participants. Therefore, the *msg* entity type has been identified as the trigger entity, and the *robot* entity type has been identified as the target entity affected by events correlated with a message. Fig. 6 shows an excerpt of the EKG constructed using the current state-of-the-art method (Fig. 6a) and the proposed approach (Fig. 6b). This excerpt focuses on the interaction between events from the message and robot perspectives. In Fig. 6a, the *msg* flow is constructed by following the temporal order of events correlated to the same message, leading to a one-to-one relationship between events, where two message-receive events, belonging to different robots, are connected. In contrast, Fig. 6b splits the message flow, creating a one-to-many relationship between a message-sending event and two message-receiving events. The structure of the EKG influences its aggregated version, which is used for system analysis. In this case study, events are aggregated following the activity name and the robot identifier to retrieve the collaboration between robots. However, for the sake of presentation, the result of EKG aggregation is in the online documentation.

Summing up, the generated EKGs demonstrate the potential of the proposed approach through two case studies with different characteristics but exhibiting



(a) Without causal relations (b) With causal relations

Fig. 6: EKG excerpt of the robotic case study

collaboration patterns. Specifically, the comparison between the EKGs generated with causal relations and those generated with existing approaches highlights the necessity of considering not only one-to-one event relations but also one-to-many relations between events.

6 Concluding Remarks

We presented an approach for revealing causal relationships between events correlated to objects in one-to-many relations, that extends the guidelines of [12]. The approach has been assessed on two collaborative cases studied, showing its effectiveness in revealing causal relationships over an EKG.

The approach highlights the limitation of constructing event-to-event relations solely based on the shared entity and the temporal occurrence, emphasizing the necessity of representing the partial order of events from an object-centric perspective. Although it has been applied only to pairs of entity types, the approach can be extended to multiple entity pairs. For instance, an order be in a one-to-many relation with items, which can be in a many-to-one relation with packages, which in turn can be related in a many-to-one fashion to an order.

With this work, we aim to focus on the importance of causal relationships between events when dealing with object-centric event logs in which objects are related among them. We acknowledge that the proposed approach may be limited by the need for domain knowledge to identify which objects are causally related and to determine which entities are the triggers and which are the targets. Nevertheless, domain knowledge is crucial for guiding process mining analysis [9], especially in OCPM, where multiple processes are interrelated [13]. In future work, we intend to deepen the study of causal relationships by including additional case studies where more entities are causally related to each other.

References

1. van der Aalst, W.M.P.: Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data. In: Software Engineering and Formal Methods, LNCS, vol. 11724, pp. 3–25. Springer (2019)
2. van der Aalst, W.M.P., Carmona, J.: Process Mining Handbook. Springer (2022)

3. Berti, A., van der Aalst, W.M.P.: Extracting multiple viewpoint models from relational databases. In: Data-Driven Process Discovery and Analysis. LNBIP, vol. 379, pp. 24–51. Springer (2019)
4. Berti, A., van der Aalst, W.M.P.: OC-PM: analyzing object-centric event logs and process models. International Journal on Software Tools for Technology Transfer **25**(1), 1–17 (2023)
5. Berti, A., Montali, M., van der Aalst, W.M.P.: Advancements and challenges in object-centric process mining: A systematic literature review. CoRR **abs/2311.08795** (2023)
6. Conforti, R., Dumas, M., García-Bañuelos, L., Rosa, M.L.: BPMN miner: Automated discovery of BPMN process models with hierarchical structure. Information Systems **56**, 284–303 (2016)
7. Corradini, F., Pettinari, S., Re, B., Rossi, L., Tiezzi, F.: A methodology for the analysis of robotic systems via process mining. In: Enterprise Design, Operations, and Computing. LNCS, vol. 14367, pp. 117–133. Springer (2023)
8. Corradini, F., Pettinari, S., Re, B., Rossi, L., Tiezzi, F.: A technique for discovering BPMN collaboration diagrams. Software and Systems Modeling pp. 1–21 (2024)
9. Dixit, P.M., Buijs, J.C.A.M., van der Aalst, W.M.P., Hompes, B.F.A., Buurman, J.: Using domain knowledge to enhance process mining results. In: Data-Driven Process Discovery and Analysis. pp. 76–104. Springer (2017)
10. van Dongen, B.F., Van der Aalst, W.M.: Multi-phase process mining: Aggregating instance graphs into epcs and petri nets. In: PNCWB workshop. pp. 35–58 (2005)
11. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. Journal on Data Semantics **10**(1-2), 109–141 (2021)
12. Fahland, D.: Process mining over multiple behavioral dimensions with event knowledge graphs. In: Process Mining Handbook, vol. 448, pp. 274–319. Springer (2022)
13. Goossens, A., De Smedt, J., Vanthienen, J., van der Aalst, W.M.P.: Enhancing data-awareness of object-centric event logs. In: Process Mining Workshops. pp. 18–30. Springer (2023)
14. Leemans, S.J.J., van Zelst, S.J., Lu, X.: Partial-order-based process mining: a survey and outlook. Knowledge and Information Systems **65**(1), 1–29 (2023)
15. Liss, L., Adams, J.N., van der Aalst, W.M.P.: Object-centric alignments. In: Conceptual Modeling. LNCS, vol. 14320, pp. 201–219. Springer (2023)
16. Lonchamp, J.: Process model patterns for collaborative work. In: World Computer Congress-Telecooperation'98. p. 12 (1998)
17. Peña, L., Andrade, D., Delgado, A., Calegari, D.: An approach for discovering inter-organizational collaborative business processes in BPMN 2.0. In: Process Mining Workshops. LNBIP, vol. 503, pp. 487–498. Springer (2023)
18. Swevels, A., Fahland, D., Montali, M.: Implementing Object-Centric Event Data Models in Event Knowledge Graphs. In: Process Mining Workshops, LNBIP, vol. 513, pp. 431–443. Springer (2024)
19. Tour, A., Polyvyanyy, A., Kalenkova, A.A., Senderovich, A.: Agent miner: An algorithm for discovering agent systems from event data. In: Business Process Management. LNCS, vol. 14159, pp. 284–302. Springer (2023)
20. Waibel, P., Novak, C., Bala, S., Revoredo, K., Mendling, J.: Analysis of business process batching using causal event models. In: Process Mining Workshops. LNBIP, vol. 406, pp. 17–29. Springer (2020)
21. Waibel, P., Pfahlsberger, L., Revoredo, K., Mendling, J.: Causal process mining from relational databases with domain knowledge. arXiv:2202.08314 (2022)
22. Weber, I., Farshchi, M., Mendling, J., Schneider, J.: Mining processes with multi-instantiation. In: Symposium on Applied Computing. pp. 1231–1237. ACM (2015)