TOTeM: Temporal Object Type Model for Object-Centric Process Mining

Lukas Liss, Jan Niklas Adams, and Wil M.P. van der Aalst

RWTH Aachen University, Aachen, Germany
{liss,niklas.adams,wvdaalst}@pads.rwth-aachen.de

Abstract. System behavior emerges from multiple subprocesses operating on interacting objects of different types. The relations between types of subprocesses are essential to understand the system's behavior. This includes temporal relations as well as cardinalities of relationships. Whether a product is produced before or after a customer order, which is essential with respect to lean management, would be an example of a temporal relationship. Conversely, the number of products for each customer order would be an example of a cardinality constraint. Current object-centric process modeling approaches focus on precedence constraints between activities, which is not sufficient to capture temporal and cardinality relationships between types. This paper introduces the temporal object type model (TOTeM) to model and discover process-specific type-level relations. We propose three type-level relations: a temporal relation, an overall log cardinality, and an event cardinality. The contributions of this paper include a definition of the temporal object type model, an algorithm to compute them, a publicly available implementation, and an evaluation.

Keywords: Process model · Object-centric process mining · Relationship model

1 Introduction

Real-world processes are complex orchestrations of interacting subprocesses. Each subprocesses instantiation is identified by an object of the subprocesses' type. The interaction between objects is defined by shared actions between subprocesses. In traditional process mining, a process is assumed to only consist of one single subprocess, i.e., the case identifier. As a result, one can not differentiate the different objects and their interactions that make up a process. To enable this distinction, process mining shifted towards an object-centric perspective on processes [14]. There, individual objects of different types can be tracked throughout the process. Each event has an activity, a timestamp, and a set of objects that are involved in the event via an event-to-object relation. Objects can also be connected without sharing events. These relations can be quantified and captured in explicit object-to-object relations [10]. Current models for object-centric processes have limitations in describing the temporal and cardinality relations between types. Thus, we proposed a new model called TOTeM.

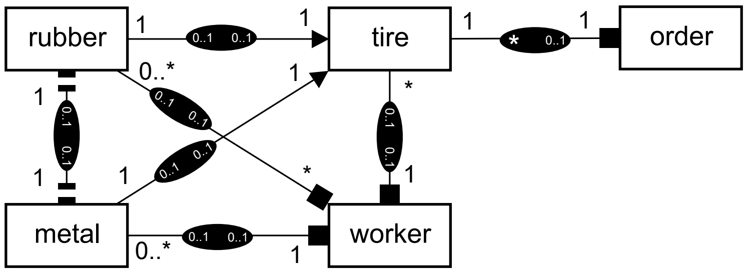


Fig. 1. Temporal object type model for the example of a tire production process.

Let us consider an object-centric example process that produces tires. It involves five object types: orders, tires, rubber, metal, and workers. The workers produce multiple tires per day. A tire is built by exactly one worker using one unit of metal and one unit of rubber. Both the metal and the rubber are prepared before. Orders can be connected to multiple tires, but a tire is always connected to exactly one order. The company follows the lean management pull principle [18], so they only produce tires if there is a demand for them. So, a new tire is only built after its order has been received. All the characteristics mentioned above are modeled in the TOTeM model in Figure 1 for the running example. In Section 4, we introduce TOTeM models and explain how the characteristics are represented. Here, we first show that these characteristics are contained in event logs, but current models are not sufficient to describe all of them.

An example excerpt of the object-centric event log in the OCEL 2.0 format [10] from this process is presented in Table 1. This contains the event-to-object relations as well as explicit object-to-object relations. Some events in the process only involve objects from one type, like *receive order* that only involves an order object. Other involve objects of multiple types, like *build tire*, which involves a worker, metal, rubber, and a tire.

Table 1. Object-centric event log for the example tire production process. The event-to-object relations are on the left and the object-to-object relations are on the right.

eventId	activity	time	object types				
			order	tire	rubber	metal	worker
e1	receive order	1.3.24 14:10	o1				
...							
e12	prepare rubber	1.3.24 15:15			r2		w1
e13	wash rubber	1.3.24 15:25			r2		w2
e14	prepare metal	1.3.24 15:50				m2	w1
e15	form metal	1.3.24 16:10				m2	w1
e16	build tire	1.3.24 16:50		t2	r2	m2	w3
e17	ship order	1.3.24 17:00	o1	t1,t2			
e18	receive payment	5.3.24 13:00	o1				
...							

source object	target object	qualifier
...		
r2	w1	handled by
r2	w2	handled by
m2	w1	handled by
m2	t2	used for
r2	t2	used for
t2	w1	build by
t2	o1	assigned to
w3	w1	supervises
w3	w2	supervises
...		

An object-centric event log in the OCEL 2.0 format [10] contains four types of information about activities and object types. First, the event log contains information about the temporal ordering of activities. Second, the cardinality for how many objects per object type are involved in an activity is contained in the event log. We refer to this as the event cardinality. Third, the information on how object types are related temporally. For example, tires only come into existence after the order object exists. Fourth, the event log contains information about the cardinalities between object types. For example, metal is always handled by exactly one worker, but rubber can be connected to multiple workers. We refer to this as the log-level cardinality. All types of information are valuable in understanding and analyzing a process. Note that an object-centric event log is not limited to these four types of information since there can be, for example, additional attributes, but all object-centric event logs in the OCEL 2.0 format [10] contain these four.

Current approaches model object-centric processes with object-centric Petri nets [5] or class diagrams [8]. Class diagrams model only the fourth type of information, the cardinalities between object types. They use neither temporal information about activities nor object types. On the other hand, object-centric Petri nets focus on the first type of information, the precedence constraints of activities, and the second type. So the object-centric Petri net for the running example in Figure 2 shows that *prepare rubber* has to precede *wash rubber* and that *ship order* can involve multiple tires but just one order. The arcs with a triple arrowhead in the object-centric Petri net are variable arcs that show that the connected activity can involve multiple objects. However, none of the models describe the third type of information, the temporal ordering of object types. It is also impossible to reliably derive one type of information from another, as shown by the example object-centric Petri net in Figure 2.

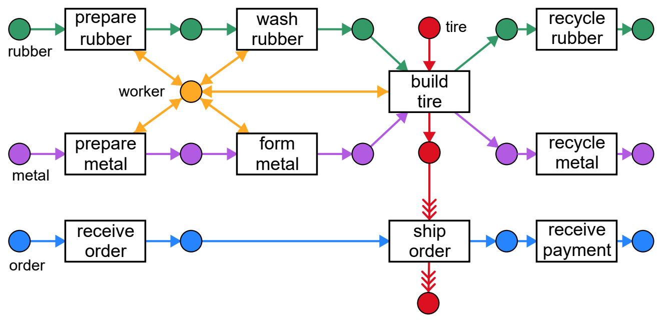


Fig. 2. Object-centric Petri net for the example tire production process.

For example, in the object-centric Petri net for the running example in Figure 2, one can not differentiate whether the pull principle [18] is followed or not. In the context of the running example, this means that tires only come into existence when the order is placed, so the customer demand already exists. This information on the temporal relation between types is contained in the event log but not discovered or modeled by object-centric Petri nets. Since the tire and the order only connect when the order is shipped, there is no temporal relation between *build tire* and *receive order* modeled in the object-centric Petri net.

Another important characteristic of a process is the work distribution. This relates to information about the cardinalities between object types. For example, in the tire process, it makes a huge difference whether there is an assembly line where multiple workers are connected to (work on) a material or just one. In the example process, multiple workers are connected to one *rubber*, but for each *metal* object, there is only one *worker* connected. This difference is not visible in the object-centric Petri net. This example shows the limitations of current modeling approaches for object-centric processes. To close that gap, we propose the temporal object type model (TOTeM).

This paper presents four contributions to enable the data-based discovery of the described object-type relations that characterize object-centric processes.

- We propose a definition for temporal object type models (TOTeM) and the three types of object type relations.
- We describe an algorithm to discover TOTeM models from object-centric event logs [10].
- We implement the algorithm and publish the open-source code.
- We use the implementation to perform an evaluation of our approach on a publicly accessible event log.

The paper is structured in the following way. In Section 2, we present the related work. Afterward, in Section 3, we describe the preliminaries. Section 4 elaborate on the three types of relations and how they are combined to TOTeM models. A definition of the proposed algorithm to compute TOTeM models is given in Section 5. Section 6 evaluates the proposed model and algorithm with a qualitative and quantitative evaluation. In Section 7, we conclude the paper and give directions for future research.

2 Related Work

Process Mining aims to generate insights into business processes [1]. The main three tasks in process mining are discovery, conformance checking, and enhancement. In process discovery, one creates a process model given an event log that describes the process. To better represent real-world processes with multiple objects of different types, object-centric process mining was introduced [14]. This allows distinguishing multiple objects within a process and solves representation problems of traditional process mining like deficiency, convergence, and divergence [2]. Our proposed TOTeM model and the described discovery algorithm, therefore, belong to the field of object-centric process discovery. The goal of TOTeM models is to give insights into the process by understanding the temporal relations and cardinalities between object types. Current discovery approaches focus on mining dependency constraints between events [5].

There exist a variety of process models for traditional event logs, like directly follows graphs, Petri Nets [24], BPMN [12], Causal Nets [3], and Process Trees [21]. Since they are limited to a single case identifier, they do not model any object type relations.

There are also object-centric process models. For example, object-centric Petri nets [5] and object-centric directly follows graphs [9] can be discovered from object-centric event logs [10]. An object-centric directly follows graph shows the directly follows relations between activities per object type [9]. Object-centric Petri nets and a discovery approach for them are introduced by Berti and van der Aalst [5]. Other approaches like Petri nets with identifiers [26] and COA-nets [16] also extend Petri nets to track objects and their types in a process. These models describe process behavior by modeling precedence constraints of activities. As shown in the introduction, one can not always derive clear insights about the type-level behavior from the activity level constraints. Object-centric behavior constrains [4] also model temporal relations between activities and additionally model the cardinality of object types for the activities as well as the cardinality relation between object types. But similar to object-centric Petri nets [5], the temporal ordering of object types is only indirectly modeled via precedence constraints for activities, which is not sufficient, as shown in the introduction. Other approaches like PHILharmonicFlows [20] and the MERODE approach [25] combine multiple models to represent object-centric processes. They use state models to represent the lifecycle of object types and class diagrams to model the relations between them. The temporal relations between object types can be represented indirectly via state change dependencies but must not be explicitly modeled. But they can not be discovered from event data but must be modeled by an domain expert and they distribute the information over multiple models.

Besides the object-centric process models, one can also use class diagrams to describe how object types are related to each other [8]. This modeling is mainly done by hand and is not discovered, which hinders the ability to discover unknown behavior in the process. Many different class diagrams exist, e.g., UML-Class diagrams [8] or Entity-Relationship models [11]. Since they are not specifically designed for processes, they do not support the event-level cardinality and do not model temporal relation between types. There are multiple extensions that try to integrate time into class models [17]. However, those approaches focus on the time that modeled information is valid. Ferg for example, extended ER diagrams to represent the start and end date of the validity of attributes of classes [15]. Elmasri and Wuu extended ER models to model the lifespan of individual classes [13]. However, they do not model how the lifespans of two classes relate to each other, which is relevant for processes.

So far, a process-specific model notation describing the temporal relations and the cardinality relations between types that can be discovered from process data is missing. To close that gap, we propose TOTeM models.

3 Preliminaries

We assume object-centric event data stored in, for example, OCEL 2.0 format [10]. Events are activities that happen at a timestamp for a set of objects of different types. $\mathbb{U}_{event}$ is the universe of event identifiers. The universe $\mathbb{U}_{act}$ contains all visible activities. $\mathbb{U}_{type}$ is the universe of all object types. The universe of objects is $\mathbb{U}_{obj}$. Each object has exactly one type associated with it $\pi_{type} : \mathbb{U}_{obj} \rightarrow \mathbb{U}_{type}$. $\mathbb{U}_{time}$ is the universe of all timestamps.

Definition 1 (Object-Centric Event Log). *$L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ is an event log with:*

- *$E \subseteq \mathbb{U}_{event}$ is a set of events, $O \subseteq \mathbb{U}_{obj}$ is a set of objects,*
- *$OT = \{\pi_{type}(o) | o \in O\}$ is a set of object types,*
- *$O2O \subseteq \{(o_1, o_2) | o_1, o_2 \in O \land o_1 \neq o_2\}$ is the set of object to object relations,*
- *$\pi_{act} : E \rightarrow \mathbb{U}_{act}$ maps each event to an activity,*
- *$\pi_{obj} : E \rightarrow \mathcal{P}(\mathbb{U}_{obj}) \setminus \{\emptyset\}$ maps each event to at least one object,*
- *$\pi_{time} : E \rightarrow \mathbb{U}_{time}$ maps each event to a timestamp.*

Note that we abstract from qualifiers and attribute values, i.e. we use a subset of OCEL 2.0 [10]. The tabular representation of an object-centric event log consists of one table showing the event-to-object relations and another table showing the object-to-object relations. Table 1 shows the event to object relations for the example event log. There, each event is connected to one activity, a timestamp, and some objects of the five types of the example tire production process. Table 1 shows the object-to-object relations for the exemplary tire production process. Often, two objects are connected via an object-to-object relation if they share an event. For example, the objects *r2* and *w1* of the first object-to-object relation in Table 1 are both involved in event *e2*. However, a shared event is not required if there is an explicit object-to-object relationship. For example, there is an object-to-object relation between *w3* and *w2*, although they never share an event because, for example, *w3* supervises *w2*. Older formats of object-centric event logs do not have explicit object-to-object relations. To be able to use these event logs as well and have one source of truth for which objects are connected, we add objects that share events to the potentially empty set of object-to-object relations.

Definition 2 (Object-to-Object Relations from Event-to-Object Relations). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ the event log with all connected objects represented in the object-to-object relations is given by: $L' = (E, O, OT, O2O', \pi_{act}, \pi_{obj}, \pi_{time})$ with $O2O' = O2O \cup \{(o_1, o_2) \in O \times O | o_1 \neq o_2 \land \exists_{e \in E} \{o_1, o_2\} \subseteq \pi_{obj}(e)\}$.*

In the following we assume, that all event logs are event logs that represent all connected objects in the object-to-object relation. We call two objects connected if they are part of an object-to-object relation and we call two types related if there is at least one object of each type that is connected to an object of the other type.

Definition 3 (Connected Objects and Types). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$, the set of connected objects $\overline{O2O} = \{(o_1, o_2) \in O \times O | (o_1, o_2) \in O2O \vee (o_2, o_1) \in O2O\}$ contains all pairs of objects for which at least one direction is part of $O2O$.*
We reduce a set of objects $O' \subseteq O$ to objects of type $t \in OT$ with the following notation: $O'_{\downarrow t} = \{o \in O' | \pi_{type}(o) = t\}$.
We reduce the connected objects to pairs of type $t_1, t_2 \in OT$ with $\overline{O2O}_{\downarrow t_1, t_2} = \{(o_1, o_2) \in \overline{O2O} | o_1 \in O_{\downarrow t_1} \wedge o_2 \in O_{\downarrow t_2}\}$.
Two types $t_1, t_2 \in OT$ are connected iff $\exists_{o_1 \in O_{\downarrow t_1}, o_2 \in O_{\downarrow t_2}} (o_1, o_2) \in \overline{O2O}$.

For all the example objects in Table 1, the reduction to the type worker results in $\{o1, t2, r2, m2, w1, w2, w3\}_{\downarrow worker} = \{w1, w2, w3\}$. The types *rubber* and *worker* are connected because, for example, *r2* and *w1* are connected.

For the temporal type relation, we need to determine the lifespan of objects within a process. We assume that the lifespan of an object, in the context of the given process, is defined by the first and last appearance of that object in the event-to-object table.

Definition 4 (Object Lifespan Interval). *Given an event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ and object $o \in O$. The start of the lifespan of object o is $ots_{L,start}(o) = min(\{t \in \mathbb{U}_{time} | \exists_{e \in E} \pi_{obj}(e) = o \wedge \pi_{time}(e) = t\})$. The end of the lifespan of an object o is $ots_{L,end}(o) = max(\{t \in \mathbb{U}_{time} | \exists_{e \in E} \pi_{obj}(e) = o \wedge \pi_{time}(e) = t\})$. The lifespan of an object is the interval $ols(o) = \{t \in \mathbb{U}_{time} | ots_{L,start}(o) \leq t \leq ots_{L,end}(o)\}$.*

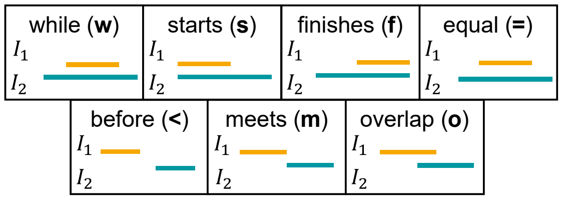


Fig. 3. Allen's interval relations [7].

For example the lifespan for object *r2* starts at 1.3.24 15:15 and ends at 1.3.24 16:50 because event *e12* is the first and event *e16* is the last event that involves *r2*.

We use Allen's interval algebra [7] on the object lifespan intervals. Figure 3 shows all the relations that intervals can have. For example, the first one shows I_1 **w** I_2, which means that the interval for I_1 is *while*(**w**) interval I_2. Each of the relations (except *equal*(=)) has an inverse relation as well. So, for example, if I_1 is *while*(**w**) I_2, then I_2 is *while inverse*($\mathbf{w}^{-1}$) to I_1. In the following, we will focus on non-inverse relations. For example, the lifespan of *r2* is *while*(**w**) the lifespan of *o1* because the lifespan of *o1* starts earlier and ends later. So the *while*(**w**) relation holds: $ols(r2)$ **w** $ols(o1)$. The concrete definitions for the relations are given in [7]. Note that Allen calls the *while*(**w**) relation *during*, but to avoid confusion with the *during*(■) relation proposed in this paper we use another name.

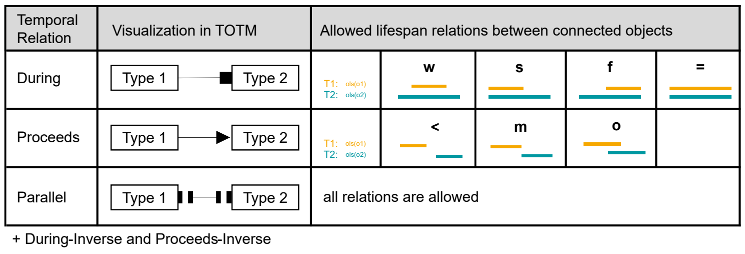
4 Temporal Object Type Model

In this section, we describe the proposed TOTeM model. First, we introduce each of the three relation types separately. Then, we explain how they are combined together in the TOTeM model.

4.1 Temporal Relation

The goal of the temporal relation is to describe how the lifespan of connected objects from two types relate to each other. The relation of one object's lifespan to another can be described using Allen's 13 interval algebra relations. Some of these relations are similar. We group these similar relations together and use these groups to define temporal relations over object types.

For example, the *while*(**w**), *starts*(**s**), *finishes*(**f**), and *equal*(=) interval relations have in common that the first interval only exists if the second one exists. This group of lifespan interval relations belongs to the temporal relation *during*(■). A *during*(■) relation from *type 1* to *type 2* describes that all pairs of connected objects from *type 1* to *type 2* have only lifespan interval relations that belong to the described group. So the lifespan of a *type 1* object either is *while*(**w**), *starts*(**s**), *finishes*(**f**), or is *equal*(=) to the lifespan of a connected *type 2* object. Figure 4 shows the temporal relations and the related groups of interval relations. In the running example, the *during*(■) from the tire to the order means that all tires only exist during their connected order object also exists and never earlier or later. Also, one can differentiate long-term assets like the workers and short-term assets like *rubber*, *metal*, and *tire*. In the TOTeM model, all the connected short-term assets have a *during*(■) relation to the *worker* type.



Temporal Relation	Visualization in TOTM	Allowed lifespan relations between connected objects				
During	Type 1 —■ Type 2	T1: ols(o1) T2: ols(o2)	w	s	f	=
Proceeds	Type 1 —▶ Type 2	T1: ols(o1) T2: ols(o2)	<	m	o	
Parallel	Type 1 ▮▮—▮▮ Type 2	all relations are allowed				

+ During-Inverse and Proceeds-Inverse

Fig. 4. The temporal relation *during*(■), *precedes*(▶), and *parallel*(‖) (based on Allen's interval algebra [7]). For each temporal relation, the allowed lifespan relations for connected objects of *Type 1* (in orange) and *Type 2* (in green) are shown.

We also grouped together the *before*(¡), *meets*(**m**), and *overlaps*(**o**) interval relations because for all of them, the first interval starts before the second one starts and ends before the second one ends. This group relates to the *precedes*(▶) temporal relation. Both the *during*(■) and *precedes*(▶) temporal relation can be inversed by changing the target and source type. In the running example, there are *precedes*(▶) relations from *rubber* and *metal* to the tire. Since these materials are used to produce the tire, they exist before the *tire*.

The parallel(‖) temporal relation allows for all interval relations. This means that there is no clear order in the relationship between lifespans. The parallel relation between metal and rubber in the TOTeM model for the running example shows that the lifespans of objects of these types are not ordered.

Temporal relations are defined in Definition 5 by requiring that all lifespan interval relations for connected objects of the types belong to the described groups for the temporal relation. The definition introduces a scoring function that computes the fraction of object pairs from the two types that fulfill the requirements of the temporal relation. This scoring function is also used in the algorithm in Section 5 together with a noise parameter for filtering undesired infrequent behavior to determine possible temporal relations.

Definition 5 (Temporal Relations). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ we define the score for $(t_1, t_2) \in OT \times OT$ for the temporal relations as:*

$$sc_{\blacksquare}(t_1,t_2) = \frac{|\{(o_1,o_2) \in \overline{O2O}_{\downarrow t1,t2} \mid ols(o_1)\ \boldsymbol{d}\ ols(o_2) \vee ols(o_1)\ \boldsymbol{s}\ ols(o_2) \vee ols(o_1)\ \boldsymbol{f}\ ols(o_2) \vee ols(o_1) = ols(o_2)\}|}{|\overline{O2O}_{\downarrow t1,t2}|}$$

$$sc_{\blacksquare_{inv}}(t_1,t_2) = sc_{\blacksquare}(t_2,t_1)$$

$$sc_{\blacktriangleright}(t_1,t_2) = \frac{|\{(o_1,o_2) \in \overline{O2O}_{\downarrow t1,t2} \mid ols(o_1) < ols(o_2) \vee ols(o_1)\ \boldsymbol{m}\ ols(o_2) \vee ols(o_1)\ \boldsymbol{o}\ ols(o_2)\}|}{|\overline{O2O}_{\downarrow t1,t2}|}$$

$$sc_{\blacktriangleright_{inv}}(t_1,t_2) = sc_{\blacktriangleright}(t_2,t_1)$$

$$sc_{\|}(t_1,t_2) = \frac{|\overline{O2O}_{\downarrow t1,t2}|}{|\overline{O2O}_{\downarrow t1,t2}|} = 1$$

A temporal relation holds for two types if the related score is equal to 1.

4.2 Log Cardinality

The log cardinality describes how many objects of a target type are associated with objects of the source type over the whole lifespan of an object. They are similar to cardinalities described by class diagrams [8]. Types are only connected when there is at least one pair of objects from the two types that are connected. Therefore, we do not need to consider the case that no objects of a type are connected to any object of the other type. The cardinality of connected objects can therefore be *1* (exactly 1), *0..1* (none or 1), *1..** (1 or many) *0..** (none, 1, or many). Figure 5 shows the possible cardinalities from *type 1* to *type 2* and what type of structures this relation allows in the connected object graph. The indicator for how many objects of *type 2* are connected to objects of *type 1* is located on the connection line between the types close to the box that represents *type 2*. Definition 6 defines log cardinalities using a scoring function that represents the fraction of objects of the source type that match a log cardinality by being connected to the right amount of objects from the source type. If all objects of the source type match the cardinality requirement, the log cardinality holds for the given types.

Definition 6 (Log Cardinality). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *and filter parameter* τ*, we detect temporal relation for a pair of connected types* $(t_1, t_2) \in OT \times OT$*:*

$$sc_{l-1}(t_1,t_2) = \frac{|\{o_1 \in O_{\downarrow t_1} \mid |\{o_2 \in O_{\downarrow t_2} | (o_1,o_2) \in \overline{O2O}\}| = 1\}|}{|O_{\downarrow t_1}|}$$

$$sc_{l-0..1}(t_1,t_2) = \frac{|\{o_1 \in O_{\downarrow t_1} \mid |\{o_2 \in O_{\downarrow t_2} | (o_1,o_2) \in \overline{O2O}\}| \in \{0,1\}\}|}{|O_{\downarrow t_1}|}$$

$$sc_{l-1..*}(t_1,t_2) = \frac{|\{o_1 \in O_{\downarrow t_1} \mid |\{o_2 \in O_{\downarrow t_2} | (o_1,o_2) \in \overline{O2O}\}| \geq 1\}|}{|O_{\downarrow t_1}|}$$

$$sc_{l-0..*}(t_1,t_2) = \frac{|\{o_1 \in O_{\downarrow t_1} \mid |\{o_2 \in O_{\downarrow t_2} | (o_1,o_2) \in \overline{O2O}\}| \geq 0\}|}{|O_{\downarrow t_1}|} = 1$$

A log cardinality relation holds for two types if the related score is equal to 1.

The log cardinality can give process owners and analysts insights into how objects of a given type relate to objects of other types. For example, Figure 1 shows that tire objects are always connected to exactly one rubber and one metal object. This shows that these types of materials are required for the tire. If the rubber would be optional for the tire, one would find a *0..1* log cardinality there.

Some log cardinalities are stricter special cases of other cardinalities. For example, all tuples of types that fulfill the *1* log cardinality also fulfill all other log cardinalities because they are all less restrictive. Therefore, one should model the most restrictive log cardinality that applies.

One is interested in selecting the most restrictive cardinality because multiple ones can fit. For example, tires are always connected to only one order. This is allowed for the cardinalities *1*, *0..1*, *1..**, and *0..**. Since cardinality *1* is the most specific, one wants to select that one.

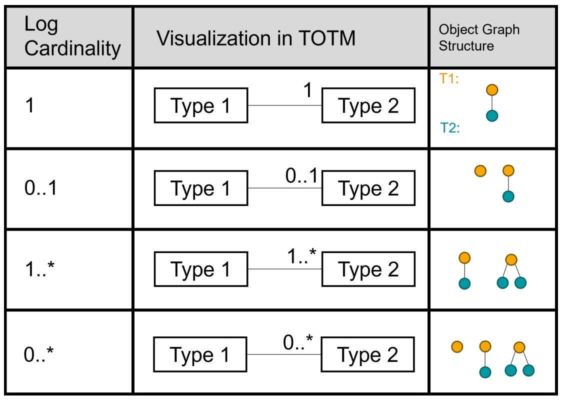


Fig. 5. The four log cardinality relations that describe how many objects of the types can be connected to how many objects of the other type in the log.

4.3 Event Cardinality

The event cardinality describes how many objects of the target type are involved in events that involve at least one object of the source type. So the event cardinality for *type 1* as the source and *type 2* as the target describes how many objects are involved in events that involve at least one object of *type 1.* Since objects that are related via object-to-object relations do not need to share events, there can be either always 0, exactly 1, 0 or 1, multiple (which includes 1), 0 or multiple objects of the target type involved. Figure 6 shows the event cardinalities, how they are visualized, and sketches events with their related objects from the source type (*type 1*) and target type (*type 2*). The cardinality is visualized in a black ellipsis that is on the connection line between two connected types. Since there are two directions between connected types, in a TOTeM model, there are always two event cardinalities in the black ellipses that are separated by a dash. The event cardinality is always placed closer to the target type. So in the first row in Figure 6, the 0 in the black ellipsis is closer located to *type 2.* This shows that it is the cardinality that has *type 1* as the source and *type 2* as the target. Definition 7 defines event cardinalities by introducing scoring functions that compute the fractions of events that belong to a certain cardinality for the two types given. If all events of the given source type follow the cardinality in regards to the target type, the event cardinality holds for them.

Definition 7 (Event Cardinality). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *and filter parameter* τ*, we detect temporal relation for a pair of connected types* $(t_1, t_2) \in OT \times OT$*:*

$$sc_{e-\mathit{0}}(t_1,t_2) = \frac{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset \land |\pi_{obj}(e)_{\downarrow t_2}| = 0\}|}{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset\}|}$$

$$sc_{e-\mathit{1}}(t_1,t_2) = \frac{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset \land |\pi_{obj}(e)_{\downarrow t_2}| = 1\}|}{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset\}|}$$

$$sc_{e-\mathit{0..1}}(t_1,t_2) = \frac{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset \land |\pi_{obj}(e)_{\downarrow t_2}| \in \{0,1\}\}|}{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset\}|}$$

$$sc_{e-\mathit{1..*}}(t_1,t_2) = \frac{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset \land |\pi_{obj}(e)_{\downarrow t_2}| \geq 1\}|}{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset\}|}$$

$$sc_{e-\mathit{0..*}}(t_1,t_2) = \frac{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset \land |\pi_{obj}(e)_{\downarrow t_2}| \geq 0\}|}{|\{e \in E | \pi_{obj}(e)_{\downarrow t_1} \neq \emptyset\}|} = 1$$

Event cardinality relations hold for two types if the related score is equal to 1.

For example the TOTeM model for the tire process in Figure 1 shows an event cardinality of *0..** from *order* to *tire.* This means that there are events involving orders that have no tire involved, but there are also events that involve multiple tires. The event cardinality is especially interesting in combination with the log cardinality. We can see, for example, that the worker is connected to multiple metal objects in total but handles at most one at a time.

4.4 TOTeM: Temporal Object Type Model – Combined Model

The TOTeM model describes the overall behavior-based relations of types in a process. All three relations mentioned above, temporal relations, log cardinality, and event cardinality, are combined in the TOTeM model. In the TOTeM model,

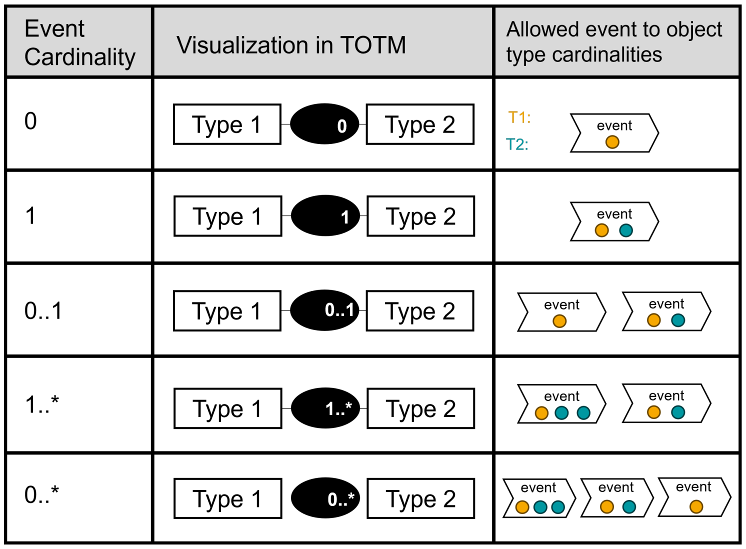


Fig. 6. The three event cardinality relations describe how many objects of the types participate in shared events. Note that we only visualized events with one object of *Type 1*, but there can also be more.

all object types of the object-centric event log are represented as boxes exactly once. Two types are related if there is at least one connection between objects of the two types. For the example tire process, that means that all the five types are represented as boxes in Figure 1. Since there is no object-to-object relation between order objects and rubber objects, these two types are not connected. Whereas for example, metal and worker types are connected because there are workers that are connected with metal objects.

All three type relations are directed between two connected types, assuming a source and target type. For two connected types, we show six relations simultaneously on the connection line (all three relation types for both directions). For example, on the connection between rubber and tire, we show all three relation types, once assuming tire is the source type and once assuming rubber is the source type. So, the log cardinality from tire to rubber is shown, as well as the log cardinality from rubber to tire. The same holds for the event cardinality, which is shown in the black ellipsis on the connection line. Both types of cardinalities are visualized closer to the target type, as explained before, so one can differentiate the cardinalities for both directions. The *during*(■), and *precedes*(▶) temporal relations can be inversed and the *parallel*(‖) relation always holds for both directions. Therefore by visualizing either a *during*(■), *precedes*(▶), or *parallel*(‖) relation between two types, the temporal relations for both directions are defined. The definition of the TOTeM model is given in Definition 8.

Definition 8 (Temporal Object Type Model). *A temporal object type model is a directed graph* $(T, C, \pi_{tr}, \pi_{cc}, \pi_{oc})$ *where:*

- $T \subseteq \mathbb{U}_{type}$ *is a finite set of object types as nodes;*
- $C \subseteq T \times T$ *is a set of directed edges between connected types;*
- $\pi_{tr} : C \to \{\blacksquare, \blacksquare_{inv}, \blacktriangleright, \blacktriangleright_{inv}, \|\}$ *labels each edge with a temporal relation that holds;*
- $\pi_{lc} : C \to \{1, 0..1, 1..*, 0..*\}$ *labels each edge with a log cardinality that holds;*
- $\pi_{ec} : C \to \{0, 1, 0..1, 1..*, 0..*\}$ *labels each edge with an event cardinality that holds.*

5 Mining Algorithm

In this section, we define the algorithm to discover TOTeM models from object-centric event logs. The implementation code is publicly accessible on GitHub[1].

Real-world data, including event data, often contains noise. Process owners want to discover the process, so the influence of noise should be minimized. Also, process variants are often not equally frequent. Sometimes, one is interested in capturing infrequent behavior, but sometimes, one is only interested in mainstream behavior. This motivates the use of a parameter in the model discovery that allows users to adjust how strictly the discovered model should incorporate infrequent behavior. Therefore, we use the parameter $0 \leq \tau \leq 1$, which represents the fraction of conformity, to detect a relation for a tuple of types.

The algorithm starts with importing the object-centric event log. For that purpose, we use the open-source library ocpa [6]. First, all object types and the object-to-object relations are determined. Then, the object-to-object relations are supplemented with connections based on event-to-object relations. Definition 2 defines how the object-to-object relations are computed from the event-to-object relations. After that, we mine the temporal relations, the event cardinality, and the log cardinalities.

We use the scoring function defined in Definition 5, Definition 6, and Definition 7 to mine potential relations for each pair of types. Instead of requiring that the scoring function is equal to one, we require it to be higher or equal to the parameter τ. So for $\tau = 1$, only relations that hold for all objects and events are considered as potential relations. For a τ smaller than 1, we include relations that hold for a fraction of the behavior that is at least as big as τ. For example, if $\tau = 0.8$ then two types $(t1, t2)$ have a *during*($\blacksquare$) relation if at least 80 % of the connected objects of these types fulfill the during requirements.

Out of the potential relations, we select the most specific relation. As mentioned before, it is possible that a tuple is part of multiple relations of each of the three relation types. For example, a type tuple can belong to the *during*($\blacksquare$) relation and also to the *parallel*($\|$) relation. Also, all type tuples that have *1* as a potential log cardinality also have *1..** as a potential log cardinality. We construct the TOTeM model only with the most specific relation for each of the

[1] https://github.com/LukasLiss/TOTeM-temporal-object-type-model

three types of relations for each tuple. Thus, the most precise model given the τ parameter is computed using the relation selection in Definition 9.

Definition 9 (Relations Selection). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ and filter parameter τ, the functions $\pi_{tr,\tau}$, $\pi_{lc,\tau}$, and $\pi_{ec,\tau}$ select the relations for a pair of connected types $(t_1, t_2) \in OT \times OT$:*

$$\pi_{tr,\tau} = \begin{cases} \blacksquare, & \text{if } sc_{\blacksquare}(t_1,t_2) \geq \tau \\ \blacksquare_{inv}, & \text{else if } sc_{\blacksquare_{inv}}(t_1,t_2) \geq \tau \\ \blacktriangleright, & \text{else if } sc_{\blacktriangleright}(t_1,t_2) \geq \tau \\ \blacktriangleright_{inv}, & \text{else if } sc_{\blacktriangleright_{inv}}(t_1,t_2) \geq \tau \\ \|, & \text{else if } sc_{\|}(t_1,t_2) \geq \tau \end{cases}$$

$$\pi_{lc,\tau} = \begin{cases} 1, & \text{if } sc_{l-1}(t_1,t_2) \geq \tau \\ 0..1, & \text{else if } sc_{l-0..1}(t_1,t_2) \geq \tau \\ 1..*, & \text{else if } sc_{l-*}(t_1,t_2) \geq \tau \\ 0..*, & \text{else if } sc_{l-0..*}(t_1,t_2) \geq \tau \end{cases}$$

$$\pi_{ec,\tau} = \begin{cases} 0, & \text{if } sc_{e-0}(t_1,t_2) \geq \tau \\ 1, & \text{else if } sc_{e-1}(t_1,t_2) \geq \tau \\ 0..1, & \text{else if } sc_{e-0..1}(t_1,t_2) \geq \tau \\ 1..*, & \text{else if } sc_{e-*}(t_1,t_2) \geq \tau \\ 0..*, & \text{else if } sc_{e-0..*}(t_1,t_2) \geq \tau \end{cases}$$

6 Evaluation

In this section, we describe the evaluation results. We use a publicly accessible event log [19] to compare the TOTeM model and the object-centric Petri net. This object-centric event log describes the management of customer orders in a company. The log contains 21.008 events and 10.840 objects of 6 types. Figure 7 shows the object-centric Petri net for the order management.

We computed the TOTeM model with a publicly accessible implementation of the algorithm described in Section 5. The computation of the three relations took 3.14 seconds on average (95% confidence interval from 3.09 seconds to 3.19 with a sample size of 100) on an i7 2.8 GHz processor with 16GB RAM. We selected a τ parameter of 0.9 to allow for some noise, infrequent behavior, and unfinished process executions. Figure 8 shows the resulting TOTeM model.

To evaluate whether the TOTeM model provides additional insights compared to existing approaches like object-centric Petri nets [5], we analyzed which insights present in the TOTeM model are missing in the object-centric Petri net.

For example, the TOTeM model shows that all products are ordered at least once because there is a log cardinality of *1..** from products to orders. The object-centric Petri net gives no information about the concrete objects used in the event log. This also leaves it unclear whether items are sold, although the matching product no longer exists. We can not answer that question with

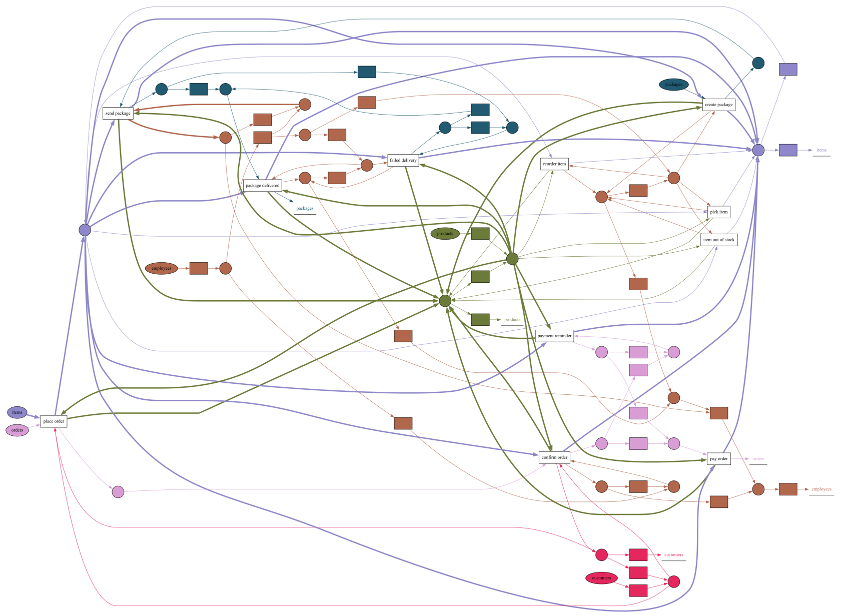


Fig. 7. Object-Centric Petri net for the order management example log [19]. Each color represents an object type. Ellipsis with text are initial places for a certain object type. The final places are represented by underlined text in the color of the object type.

the object-centric Petri net. However, the TOTeM model shows a *during*(■) relation between *items* and *products*, which shows that items only appear while their related product still exists. Moreover, the TOTeM model structures the types into a group with long lifespans (*products*, *customer*, and *employees*) and one with shorter lifespan (*orders*, *items*, and *packages*), such that all with shorter lifespans happen *during*(■) those with longer lifespans. Since most types' start and end activities are separate in the object-centric Petri net, it does not contain this information.

7 Conclusion

This paper presents four contributions to model and mine object-centric processes. First, we defined TOTeM models with temporal relations, log cardinalities, and event cardinalities. Second, we proposed an algorithm to compute TOTeM models from object-centric event logs. The algorithm's τ parameter allows it to handle noise and infrequent behavior. Third, we implemented the algorithm and published it on GitHub. Finally, in the evaluation, we showed that TOTeM models can give new insights into object-centric processes that are not captured in object-centric Petri nets.

This work also has limitations. We assume that the lifespan of objects is defined by their first and last events. Therefore, logging issues that affect these

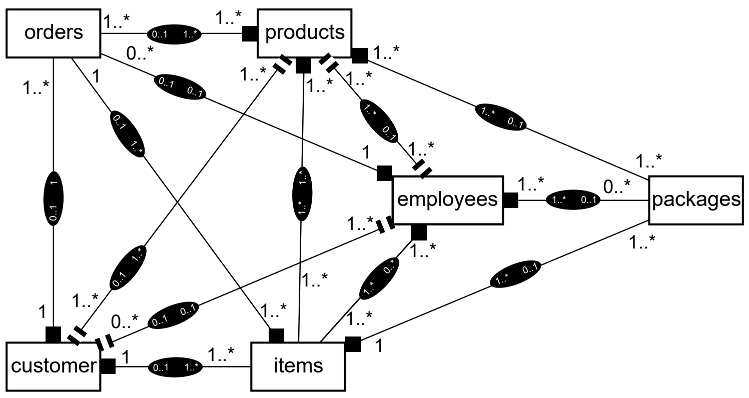


Fig. 8. TOTeM model for the order management process with $\tau = 0.9$.

events can strongly affect the result. The robustness of the presented algorithm for given τ needs to be investigated in more detail. Currently, the TOTeM models algorithm does not consider object-to-object qualifiers, which is additional information that could be used to improve the discovered model. Also, the evaluation is limited to simulated data.

Therefore, one direction for future work is to perform real-world case studies to evaluate the insights analysts can get from TOTeM models. Another direction is to investigate the usage of TOTeM models for other process mining tasks like object-centric conformance alignments [23] and model enhancement [22].

Acknowledgments. The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project Bridging AI (grant no. 16DHBKI023).

SPONSORED BY THE
Federal Ministry
of Education
and Research

References

1. van der Aalst, W.M.P.: Process mining. Commun. ACM **55**(8), 76–83 (2012)
2. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: SEFM. pp. 3–25. Springer (2019)
3. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Causal nets: A modeling language tailored towards process discovery. In: CONCUR. pp. 28–42. Springer (2011)
4. van der Aalst, W.M.P., Artale, A., Montali, M., Tritini, S.: Object-centric behavioral constraints: Integrating data and declarative process modelling. In: Proceedings of the 30th International Workshop on Description Logics. vol. 1879. CEUR-WS.org (2017)

5. van der Aalst, W.M.P., Berti, A.: Discovering object-centric petri nets. Fundam. Informaticae **175**(1-4), 1–40 (2020)
6. Adams, J.N., Park, G., van der Aalst, W.M.P.: ocpa: A Python library for object-centric process analysis. Software Impacts **14**, 100438 (2022)
7. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
8. Ashbacher, C.: The unified modeling language reference manual, Second Edition, by Rumbaugh J. Journal of Object Technol. **3**(10), 193–195 (2004)
9. Berti, A., van der Aalst, W.M.P.: Extracting multiple viewpoint models from relational databases. In: IFIP WG. pp. 24–51. Springer (2019)
10. Berti, A., Koren, I., Adams, J.N., Park, G., Knopp, B., Graves, N., Rafiei, M., Liß, L., Unterberg, L.T.G., Zhang, Y.: OCEL (Object-Centric Event Log) 2.0 Specification (2023)
11. Chen, P.P.: Entity-relationship modeling: Historical events, future trends, and lessons learned. In: Software Pioneers, pp. 296–310. Springer Berlin Heidelberg (2002)
12. Chinosi, M., Trombetta, A.: BPMN: an introduction to the standard. Comput. Stand. Interfaces **34**(1), 124–134 (2012)
13. Elmasri, R., Wuu, G.T.J.: A temporal model and query language for ER databases. In: ICDE. pp. 76–83. IEEE (1990)
14. Fahland, D.: Process mining over multiple behavioral dimensions with event knowledge graphs. In: Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448, pp. 274–319. Springer (2022)
15. Ferg, S.: Modelling the time dimension in an entity-relationship diagram. In: ER. pp. 280–286. IEEE (1985)
16. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Petri net-based object-centric processes with read-only data. Inf. Syst. **107**, 102011 (2022)
17. Gregersen, H., Jensen, C.S.: Temporal entity-relationship models - A survey. IEEE Trans. Knowl. Data Eng. **11**(3), 464–497 (1999)
18. Hopp, W.J., Spearman, M.L.: To pull or not to pull: What is the question? Manuf. Serv. Oper. Manag. **6**(2), 133–148 (2004)
19. Knopp, B., van der Aalst, W.M.P.: Order Management Object-centric Event Log in OCEL 2.0 Standard (Sep 2023)
20. Künzle, V., Reichert, M.: Philharmonicflows: towards a framework for object-aware process management. J. Softw. Maintenance Res. Pract. **23**(4), 205–244 (2011)
21. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - A constructive approach. In: PETRI NETS. pp. 311–329. Springer (2013)
22. de Leoni, M.: Foundations of process enhancement. In: Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448, pp. 243–273. Springer (2022)
23. Liss, L., Adams, J.N., van der Aalst, W.M.P.: Object-centric alignments. In: ER 2023. pp. 201–219. Springer (2023)
24. Peterson, J.L.: Petri nets. ACM Comput. Surv. **9**(3), 223–252 (1977)
25. Snoeck, M.: Enterprise Information Systems Engineering - The MERODE Approach. The Enterprise Engineering Series, Springer (2014)
26. van der Werf, J.M.E.M., Rivkin, A., Polyvyanyy, A., Montali, M.: Data and Process Resonance - Identifier Soundness for Models of Information Systems. In: PETRI NETS 2022. vol. 13288, pp. 369–392. Springer (2022)